Astra Automation

мая 29, 2024

Оглавление

1 Быстрый старт **5**
- 1.1 Начало работы . . . 5
- 1.2 Разработка контента Ansible . . . 12

2 Возможности платформы **13**
- 2.1 Управление облачными ресурсами . . . 13
- 2.2 Установка программного обеспечения . . . 15
- 2.3 Управление безопасностью . . . 16
- 2.4 Управление сетевым оборудованием . . . 16
- 2.5 Рутинные операции по управлению инфраструктурой . . . 16
- 2.6 Разработка собственного инфраструктурного кода . . . 16

3 Обзор архитектуры **19**
- 3.1 Структура платформы . . . 19
- 3.2 Структура кластера . . . 21
- 3.3 Сеть . . . 23
- 3.4 СУБД . . . 23

4 Развертывание платформы **25**
- 4.1 Системные требования . . . 25
- 4.2 Подготовка узлов . . . 27
- 4.3 Подготовка инвентаря . . . 29
- 4.4 Учетные данные системного администратора . . . 34
- 4.5 Сертификаты TLS/SSL . . . 34
- 4.6 Параметры СУБД . . . 35
- 4.7 Установка системы . . . 37
- 4.8 Обновление версии . . . 38
- 4.9 Расчет ресурсов . . . 41
- 4.10 Справочные данные . . . 41

5 Automation Hub **45**
- 5.1 Astra Automation Hub . . . 45
- 5.2 Применение Astra Automation Hub . . . 63
- 5.3 Состав Astra Automation Hub . . . 92
- 5.4 Private Hub . . . 96

6 Контроллер **97**

6.1 Особенности архитектуры . . . 98
6.2 Структура управления . . . 98
6.3 Графический интерфейс . . . 160
6.4 Программный интерфейс (API) . . . 281
6.5 Интерфейс командной строки (CLI) . . . 363
6.6 Настройка контроллера через Ansible . . . 363
6.7 Среда исполнения . . . 363

7 Разработка контента и среды исполнения 367
7.1 Среда исполнения . . . 367
7.2 Создание образа среды исполнения . . . 370
7.3 Разработка коллекций . . . 372
7.4 Проверка стиля и синтаксиса . . . 372
7.5 Тестирование . . . 372
7.6 Поиск неисправностей . . . 373
7.7 Контроль целостности . . . 373

8 Управление событиями 375

9 Средства аналитики 377

10 Примеры решений 379
10.1 Защита информации . . . 379
10.2 Пример организации процесса CI/CD . . . 381
10.3 Применение конструктора кода . . . 390

11 Обеспечение безопасности 391
11.1 Сетевые настройки . . . 391
11.2 Реквизиты в сети Mesh . . . 393
11.3 Управление сертификатами TLS . . . 395

12 Производительность 399
12.1 Концепции . . . 399
12.2 Мониторинг . . . 399
12.3 Управление . . . 399

13 Техническое обслуживание 401
13.1 Журналы . . . 401
13.2 Поиск причин неисправностей . . . 403
13.3 Примеры сценариев . . . 403
13.4 Использование базы знаний . . . 403
13.5 Техническая поддержка . . . 403

14 Термины и определения 405

15 Обновления 409
15.1 Жизненный цикл продукта . . . 409
15.2 Версионирование . . . 410
15.3 Цикл разработки документации . . . 411
15.4 Состав платформы . . . 411
15.5 История обновлений . . . 412
15.6 Стабильные версии . . . 412

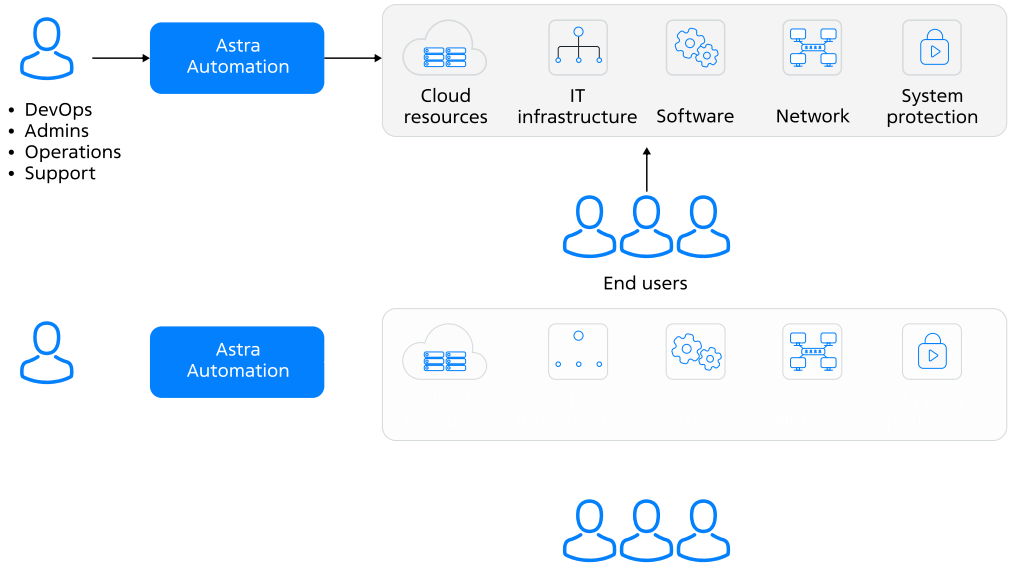
16 Дополнительные материалы 423
16.1 Приложение 1. Устанавливаемые продукты . . . 423
16.2 Приложение 2. Облачные сервисы . . . 431

16.3 Приложение 3. Инструментарий 448

Алфавитный указатель **495**

Платформа для автоматизации множества рутинных операций по управлению информационной инфраструктурой заказчика, позволяющая существенно сократить затраты на ее поддержку и администрирование.

Решения базируются на использовании известного открытого проекта Ansible со свободно распространяемым исходным кодом.



Реализуя принцип инфраструктурного кода *IaC*, платформа позволяет автоматизировать решение следующих задач:

- Управление объектами облачной инфраструктуры на основе различных операционных систем, включая их создание, настройку и обновление.
- Развертывание клиентской инфраструктуры, используя физические серверы, облачное или иное виртуальное окружении на базе различных операционных систем.
- Установка и настройка программного обеспечения, включая одиночные приложения и комплексы.
- Управление сетевым оборудованием от различных производителей.
- Настройка системы защиты информации.

Преимущества

Преимущества использования Astra Automation по отношению к последовательным ручным способам развертывания сложных информационных систем:

- существенное упрощение и ускорение процесса развертывания, обеспечивающие кратное снижение трудозатрат и времени на весь процесс;
- обеспечение безопасности в соответствии с существующими требованиями;
- обеспечение безошибочной непротиворечивой конфигурации устанавливаемых продуктов, значительно снижающее риски возникновения инцидентов и связанных с ними простоев систем;
- развертывание информационной инфрастуктуры любой сложности.

Пользователи

Данная платформа ориентирована на интеграторов сложных систем и крупные компании и обеспечивает необходимые потребительские качества.

Платформа удовлетворяет следующие насущные потребности *интеграторов*:

- обеспечение безопасного развертывания продуктов и отсутствия ошибок в их настройках;
- уменьшение затрат на весь процесс развертывания;
- сокращение времени развертывания комплексной инфрастуктуры.

С применением данной платформы *крупные компании* решают следующие важные проблемы:

- снижение стоимости владения информационной инфрастуктурой;
- высвобождение высококвалифицированных инженеров для решения внутренних сложных задач;
- ускорение развертывания инфраструктуры и снижение рисков.

Знакомство с платформой

Узнайте как это устроено и как работает.

Здесь мы приводим начальные сведения о структуре платформы, включая ее основные компоненты, их взаимодействие между собой и с компонентами развертываемой инфраструктуры. Вы познакомитесь с основами установки и управления платформой.

Быстрый старт Быстрое освоение базовых приемов работы с платформой.

Обзор архитектуры Структура платформы и общие сведения о ее составных частях.

Astra Automation Hub Структура и концепции облачного реестра инфраструктурного кода.

Astra Automation Controller Назначение, структура и настройка контроллера для решения задач автоматизации.

Инструменты для разработки Назначение, состав и применение средств разработки собственного инфраструктурного кода.

Обновления Astra Automation Жизненный цикл продукта и документации, объявления о выпусках новых версий.

Сценарии

Получите практику развертывания информационных систем, следуя пошаговым инструкциям.

Здесь вы освоите на практике основные сценарии установки Astra Automation и развертывание с ее помощью сложных систем в различных средах, используя известные облачные среды или собственное оборудование с применением средств виртуализации.

Развертывание платформы Развертывание платформы с различной топологией, от одиночного узла до распределенной кластерной структуры.

Применение Astra Automation Hub Применение коллекций Ansible и модулей Terraform для развертывания облачной инфраструктуры, установки и настройки программного обеспечения.

Примеры создания комплексной инфраструктуры Примеры применения библиотеки структурного кода для решения различных типовых задач.

Защита информации Применение коллекций Ansible для настройки систем защиты информации в соответствии с требованиями регуляторов.

Справочники

То, что всегда пригодится профессионалу.

Документация содержит справочную информацию, необходимую для профессиональной работы с платформой.

Состав репозитория Содержимое репозитория Astra Automation Hub.

Утилиты управления Утилиты развертывания и управления структурой платформы.

Графический интерфейс Полное описание графического интерфейса контроллера.

REST API Спецификация API контроллера.

Типографский стиль

Для улучшения читаемости в документации используются различные стили для выделения частей, которые по смыслу отличаются от основного текста.

Жирный (Bold) – слова, к которым требуется привлечь внимание, а также названия элементов пользовательского интерфейса (за исключением ссылок и кнопок).

Наклонный (Italic) – термины при первом употреблении и значения, которые необходимо указать в полях формы.

`Моноширинный (Monospace)` – названия параметров, названия ядер и их версии, названия пакетов, названия и значения переменных, названия утилит, имена файлов и каталогов.

`Enter` – названия клавиш и их сочетаний.

File ▸ *Exit* – названия пунктов меню и последовательность их выбора.

Запустить – названия кнопок и ссылок.

Совет: Полезная информация, которая может упростить работу.

Примечание: Описание какой-либо особенности работы с платформой.

Важно: Важная информация, которую необходимо учитывать при использовании платформы.

Предупреждение: Важная информация, игнорирование которой может привести к проблемам в работе платформы, утечке или потере данных и другим неприятным последствиям.

Последнее обновление: 29.05.2024

ГЛАВА 1

Быстрый старт

1.1 Начало работы

В этом руководстве рассматривается развертывание Astra Automation Controller и его использование для настройки управляемого узла.

В качестве стенда используются две ВМ (виртуальная машина) VirtualBox, созданные с помощью *Vagrant*. Они обладают следующими характеристиками:

- ВМ для Astra Automation Controller:
  - количество ядер CPU – 4;
  - объем RAM, ГБ – 8;
  - объем хранилища, ГБ – 30;
  - операционная система – Astra Linux Special Edition 1.7.4 без графического интерфейса;
  - IP-адрес – `192.168.56.11`.
- Управляемый узел:
  - количество ядер CPU – 2;
  - объем RAM, ГБ – 2;
  - объем хранилища, ГБ – 30;
  - операционная система – Astra Linux Special Edition 1.7.4 без графического интерфейса;
  - IP-адрес – `192.168.56.101`.

Для настройки обеих ВМ используется подключение по протоколу SSH, для работы с контроллером – веб-браузер.

1.1.1 Подготовка к работе

Для создания стенда выполните следующие действия:

1. Установите VirtualBox, Vagrant и его расширение для работы с образами Astra Linux согласно *инструкции*.
2. Создайте каталог для хранения файлов проекта. Далее этот каталог называется каталогом проекта.
3. В каталоге проекта создайте `Vagrantfile` со следующим содержимым:

```
# frozen_string_literal: true

Vagrant.configure('2') do |config|
  config.vm.box = 'alse-vanilla-base/1.7.4'
  config.vm.box_url = 'https://dl.astralinux.ru/vagrant/alse-vanilla-base%2F1.7.4'

  # Astra Automation Controller
  config.vm.define 'aac' do |aac|
    aac.vm.hostname = 'aac'

    aac.vm.provider 'virtualbox' do |vb|
      vb.cpus = 4
      vb.memory = 8192 # 8 GB
    end

    aac.vm.network 'private_network',
                   ip: '192.168.56.11'
  end

  # Управляемый узел для развертывания NGINX
  config.vm.define 'node01' do |node01|
    node01.vm.hostname = 'node01'

    node01.vm.provider 'virtualbox' do |vb|
      vb.cpus = 2
      vb.memory = 2248 # 2 GB
    end

    node01.vm.network 'private_network',
                      ip: '192.168.56.101'
  end
end
```

4. Для создания и запуска ВМ с помощью Vagrant в каталоге проекта выполните команду:

```
vagrant up
```

1.1.2 Установка контроллера

Для установки контроллера выполните следующие действия:

1. Чтобы подключиться к ВМ для контроллера, в каталоге проекта выполните команду:

```
vagrant ssh aac
```

2. В каталоге `/etc/apt/sources.list.d/` создайте файл `astra-automation.list` со ссылкой на репозиторий контроллера:

```
deb https://dl.astralinux.ru/aa/aa-debs-for-alse-1.7 <version> main
```

   Доступные версии продукта опубликованы в таблице *История обновлений*. Например, для установки версии 1.0-upd2 вместо <version> подставьте значение `1.0-upd2`.

3. Обновите список доступных пакетов:

```
sudo apt update
```

4. Установите пакет `astra-automation-setup`:

```
sudo apt install astra-automation-setup --yes
```

5. Перейдите в каталог `/opt/rbta/aa/astra-automation-setup/`:

```
cd /opt/rbta/aa/astra-automation-setup/
```

6. Раздел `[automationcontroller]` в файле `inventory` приведите к следующему виду:

```
[automationcontroller]
aac.example.com  ansible_host=192.168.56.11  ansible_connection=local
```

   Здесь:

   - `aac.example.com` – название узла;
   - `192.168.56.11` – IP-адрес настраиваемого узла.

7. Запустите установку:

```
sudo ./aa-setup
```

   Дождитесь завершения выполнения команды, это может занять некоторое время. По окончании установки в терминал выводится строка вида:

```
aac      : ok=233  changed=130  unreachable=0    failed=0    skipped=55   ␣
↪rescued=0    ignored=4
```

   Установка считается успешной, если в поле `failed` указано значение `0`.

8. Для отключения от ВМ выполните команду:

```
exit
```

1.1.3 Настройка управляемого узла

Чтобы к управляемому узлу можно было подключаться по SSH, в настройках сервера следует разрешить парольную аутентификацию. Для этого:

1. Чтобы подключиться к управляемому узлу, в каталоге проекта выполните команду:

   ```
   vagrant ssh node01
   ```

2. В каталоге `/etc/ssh/sshd_config.d/` создайте файл `pass-auth.conf` со следующим содержимым:

   ```
   PasswordAuthentication yes
   ```

3. Перезапустите службу SSH:

   ```
   sudo systemctl restart ssh.service
   ```

4. Для отключения от ВМ выполните команду:

   ```
   exit
   ```

1.1.4 Настройка контроллера

1. Запустите веб-браузер и перейдите по адресу `http://192.168.56.11/`.

   При первом подключении браузер выводит предупреждение о том, что подключение не защищено. Это нормальное поведение, поскольку для защиты подключения контроллер использует самоподписанный сертификат. При использовании браузера на основе Chromium выполните следующие действия:

   a. Нажмите кнопку *Дополнительные*.

   b. Нажмите ссылку *Перейти на сайт 192.168.56.11 (небезопасно)*.

2. Авторизуйтесь в веб-интерфейсе контроллера с учетными данными по умолчанию:

   - **Имя пользователя** (Username): `admin`.
   - **Пароль** (Password): `awx`.

3. Создайте новую запись в инвентаре:

   a. На панели навигации выберите *Ресурсы* ▸ *Инвентарь* (*Resources* ▸ *Inventories*).

   b. Нажмите кнопку *Добавить* (Add) и в открывшемся меню выберите **Добавить инвентарь** (Add inventory).

   c. Заполните форму **Создать новый инвентарь** (Create new inventory):

      - **Название** (Name): *NGINX inventory*.

   d. Нажмите кнопку *Сохранить* (Save).

4. Добавьте управляемый узел в инвентарь:

   a. На странице созданного инвентаря выберите вкладку **Управляемые узлы** (Hosts).

   b. Нажмите кнопку *Добавить* (Add).

   c. Заполните форму **Создать новый узел** (Create new host):

- **Название** (Name): `node01`.
- **Переменные** (Variables): выберите формат YAML и введите строки:

```
---
ansible_host: "192.168.56.101"
ansible_user: "vagrant"
```

d. Нажмите кнопку *Сохранить* (Save).

5. Создайте полномочие для доступа к управляемому узлу:
   a. На панели навигации выберите *Ресурсы* ▸ *Полномочия* (*Resources* ▸ *Credentials*).
   b. Нажмите кнопку *Добавить* (Add).
   c. Заполните форму **Создать новые учетные данные** (Create New Credentials):
      1. **Название** (Name): произвольное название записи, например, `NGINX`.
      2. **Тип полномочия** (Credential Type): **Машина** (Machine).
      3. **Имя пользователя** (Username): `vagrant`.
      4. **Пароль** (Password): `vagrant`.
   d. Нажмите кнопку *Сохранить* (Save).
6. Создайте полномочие для доступа к реестру Astra Automation Hub:
   a. На панели навигации выберите *Ресурсы* ▸ *Полномочия* (*Resources* ▸ *Credentials*).
   b. Нажмите кнопку *Добавить* (Add).
   c. Заполните форму **Создать новые учетные данные** (Create New Credentials):
      1. **Название** (Name): произвольное название полномочия, например, `Astra Automation Hub`.
      2. **Тип полномочия** (Credential Type): **Управление версиями** (Source control).
      3. **Закрытый ключ SCM** (SCM Private Key): содержимое приватного ключа SSH, используемого для доступа к реестру Astra Automation Hub.
      4. **Парольная фраза закрытого ключа** (Private Key Passphrase): если приватный ключ SSH защищен паролем, укажите его в этом поле.
   d. Нажмите кнопку *Сохранить* (Save).
7. Создайте новый проект:
   a. На панели навигации выберите *Ресурсы* ▸ *Проекты* (*Resources* ▸ *Projects*).
   b. Нажмите кнопку *Добавить* (Add).
   c. Заполните форму **Создать новый проект** (Create New Project):
      1. **Название** (Name): укажите произвольное название проекта, например, `Развертывание NGINX`.
      2. **Тип системы управления исходными данными** (Source Control Type): **Git**.
      3. **URL системы управления исходными данными** (Source Control URL):

```
ssh://git@hub.astra-automation.ru:2222/aa-gca/AA/aac-samples.git
```

4. **Полномочия на систему управления исходными данными** (Source Control Credential): выберите созданное ранее полномочие `Astra Automation Hub`.

d. Нажмите кнопку *Сохранить* (Save).

e. Дождитесь окончания создания проекта – это может занять некоторое время.

Об успешном создании проекта свидетельствует значение **Успех** (Successfull) в поле **Статус последнего задания** (Last Job Status).

8. Создайте шаблон задания:

a. На панели навигации выберите *Ресурсы* ▸ *Шаблоны* (*Resources* ▸ *Templates*).

b. Нажмите кнопку *Добавить* (Add) и в открывшемся меню выберите *Добавить шаблон задания* (Add job template).

c. Заполните форму **Создать новый шаблон задания** (Create New Job Template):

1. **Название** (Name): укажите произвольное название проекта, например, `Развертывание NGINX`.
2. **Тип задания** (Job Type): **Выполнение** (Run).
3. **Инвентарь** (Inventory): выберите созданную ранее запись `NGINX inventory`.
4. **Проект** (Project): выберите созданный ранее проект.
5. **Playbook**: выберите playbook `nginx/site.yml`.
6. **Учетные данные** (Credentials): выберите созданное ранее полномочие `NGINX`.

d. Нажмите кнопку *Сохранить* (Save).

9. Запустите задание:

a. На панели навигации выберите *Ресурсы* ▸ *Шаблоны* (*Resources* ▸ *Templates*).

b. В строке с созданным шаблоном задания в колонке **Действия** (Actions) нажмите кнопку запуска.

c. Дождитесь выполнения playbook.

1.1.5 Проверка развернутой инфраструктуры

Чтобы проверить корректность развертывания веб-сервера NGINX, выполните следующие действия:

1. Чтобы подключиться к управляемому узлу, в каталоге проекта выполните команду:

```
vagrant ssh node01
```

2. Проверьте статус службы `nginx`:

```
systemctl status nginx.service
```

При успешном развертывании в терминал выводится сообщение вида:

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:␣
↪enabled)
   Active: active (running) since Mon 2023-12-11 20:47:14 MSK; 40s ago
     Docs: man:nginx(8)
  Process: 12234 ExecReload=/usr/sbin/nginx -g daemon on; master_process on; -s␣
↪reload (code=exited, status=0/SUCCESS)
 Main PID: 11947 (nginx)
    Tasks: 3 (limit: 2546)
   Memory: 3.3M
      CPU: 30ms
   CGroup: /system.slice/nginx.service
           ├─11947 nginx: master process /usr/sbin/nginx -g daemon on; master_
↪process on;
           ├─12235 nginx: worker process
           └─12236 nginx: worker process
```

1.1.6 Освобождение ресурсов

Если созданные ВМ больше не нужны, для их остановки и удаления в каталоге проекта выполните команду:

```
vagrant destroy --force
```

1.1.7 Заключение

В этом сценарии вы познакомились с основными шагами по развертыванию Astra Automation Controller и его использованию для настройки управляемых узлов. Из всей последовательности шагов важно выделить следующие действия:

- Подготовка окружения.
- Указание учетных данных администратора контроллера в файле `inventory`.
- Создание записей в инвентаре.
- Создание полномочий, используемых для доступа к управляемым узлам и реестру Astra Automation Hub.
- Создание проекта.
- Настройка шаблона задания.
- Запуск задания.

1.2 Разработка контента Ansible

Автоматизация управления информационной инфраструктурой основана на использовании инфраструктурного кода со следующими компонентами:

- коллекции, ориентированные на решение определенной группы задач, например, на установку определенного программного обеспечения;
- сборники сценариев (playbooks), определяющие задания по применению коллекций в конкретном окружении.

Приведенное здесь руководство ориентировано на разработку простейшей коллекции и применение ее с помощью playbook.

Важно: Эта часть документации находится в стадии разработки.

Приведенный здесь инструкции предназначены для быстрого освоения базовых приемов работы с платформой:

- установка основных компонентов платформы;
- использование платформы для установки программ;
- разработка простейшего контента Ansible для управления инфраструктурой с помощью платформы.

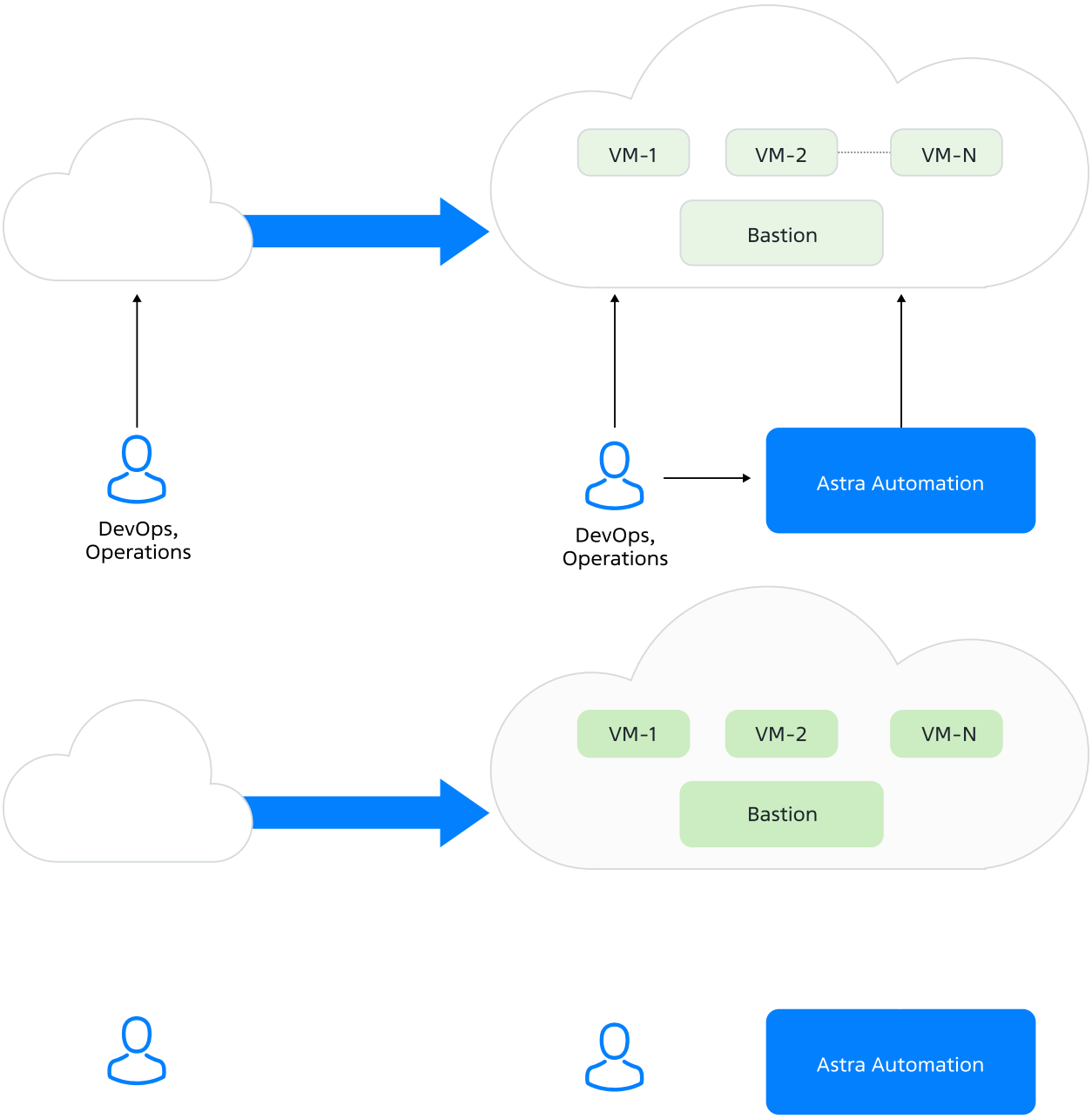
ГЛАВА 2

Возможности платформы

Платформа Astra Automation решает множество задач по управлению IT-инфраструктурой, решение которых описано в следующих секциях. Многие задачи можно решать в произвольной последовательности и в различных сочетаниях. Жизненный цикл инфраструктуры содержит множество повторяющихся операций развертывания, установки, замены и обновления различных ее компонентов.

2.1 Управление облачными ресурсами

Обладая доступом к облачному пространству, публичному или частному, организация может использовать Astra Automation для создания базовой инфраструктуры, используя технологии виртуализации.

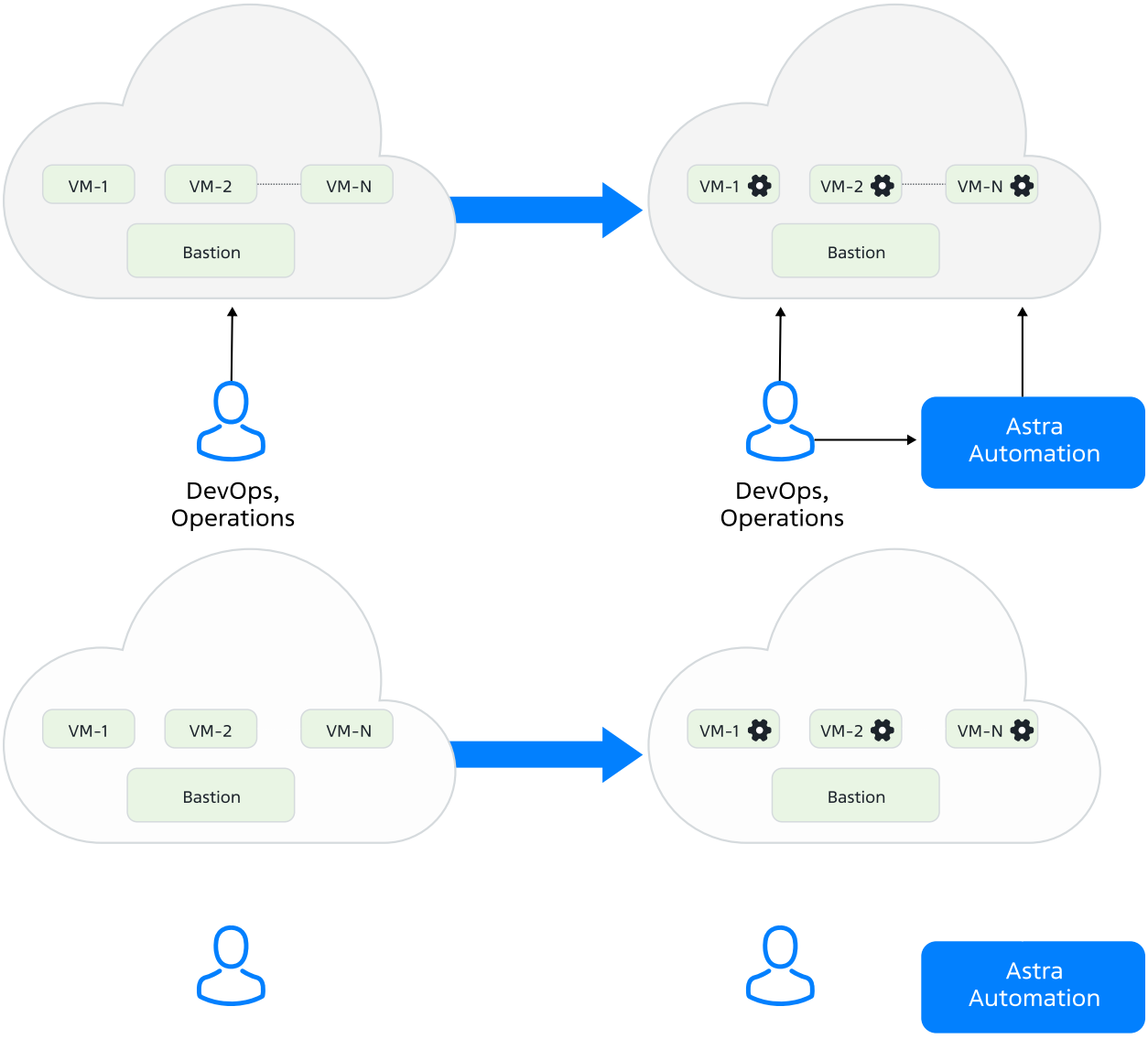


Для создания и управления облачными объектами, в частности, виртуальными машинами (VM, virtual machine) платформа предоставляет различные подходы:

- использование модулей OpenTofu (аналог Terraform);
- использование модулей Ansible.

2.2 Установка программного обеспечения

Astra Automation предоставляет развитый реестр инфраструктурного кода, позволяющего устанавливать разнообразные одиночные системы или комплексы систем на оборудовании организации или в облачном пространстве.



Сертифицированный инфраструктурный код ориентирован на использование операционной системы Astra Linux Special Edition.

2.3 Управление безопасностью

Astra Automation предоставляет специальную коллекцию Ansible для автоматизации применения требуемых мер безопасности в соответствии с рекомендациями CIS[1], ФСТЭК[2] и других сообществ и организаций. Коллекция ориентирована на использование операционной системы Astra Linux Special Edition.

2.4 Управление сетевым оборудованием

Сетевое оборудование (маршрутизаторы и коммутаторы), составляющие основу любой IT-инфраструктуры, в зависимости от сложности сети подлежат достаточно частой настройке под нужды этой инфраструктуры. К таким настройкам, например, относятся следующие составляющие:

- таблицы маршрутизации;
- состав виртуальных сетей;
- входные и выходные фильтры.

Astra Automation предоставляет необходимые модули и коллекции для настройки отечественного и зарубежного оборудования.

2.5 Рутинные операции по управлению инфраструктурой

IT-инфраструктура, подобно живому организму, подлежит частым изменениям, требующим каждодневной настройки, связанной с организационными изменениями, изменениями в ролевых моделях доступа к различным сервисам и другими. Подобные операции требуют значительных трудозатрат, которые можно значительно снизить, используя возможности Astra Automation по автоматизации рутинных операций. Для этого необходимо подготовить шаблоны для запуска типовых заданий или потоков заданий, которые может использовать персонал с минимальными знаниями платформы.

2.6 Разработка собственного инфраструктурного кода

Несмотря на значительный объем инфраструктурного кода, предоставляемого платформой, практически в каждой организации найдутся специфичные задачи, для выполнения которых недостаточно создания сценариев использования готовых решений. При необходимости разработчики организации могут воспользоваться набором инструментов для разработки контента Ansible (CDK, content development kit) от Astra Automation для эффективного решения специфичных задач. CDK позволяет разрабатывать необходимые компоненты:

- контейнерные образы для расширения возможностей среды исполнения, поставляемой в составе Astra Automation;

[1] https://www.cisecurity.org/

[2] https://fstec.ru/

- коллекции Ansible, включающие роли, модули, расширения и другие элементы инфраструктурного кода.

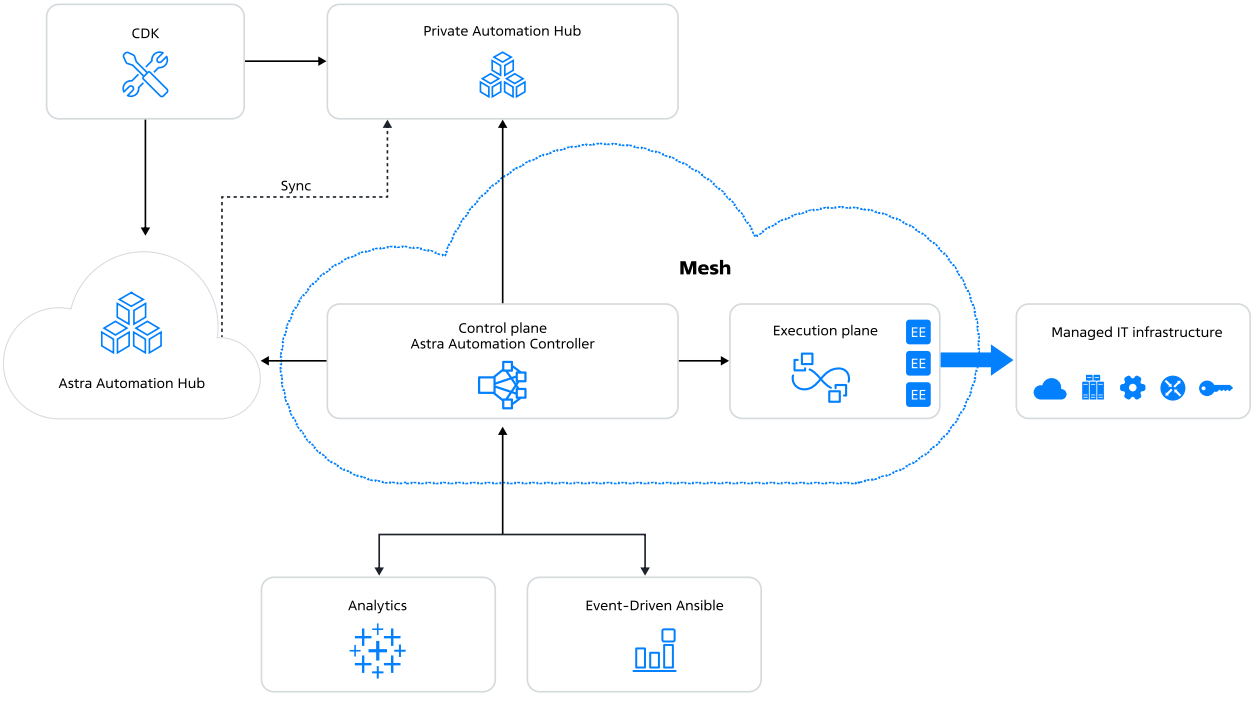
ГЛАВА 3

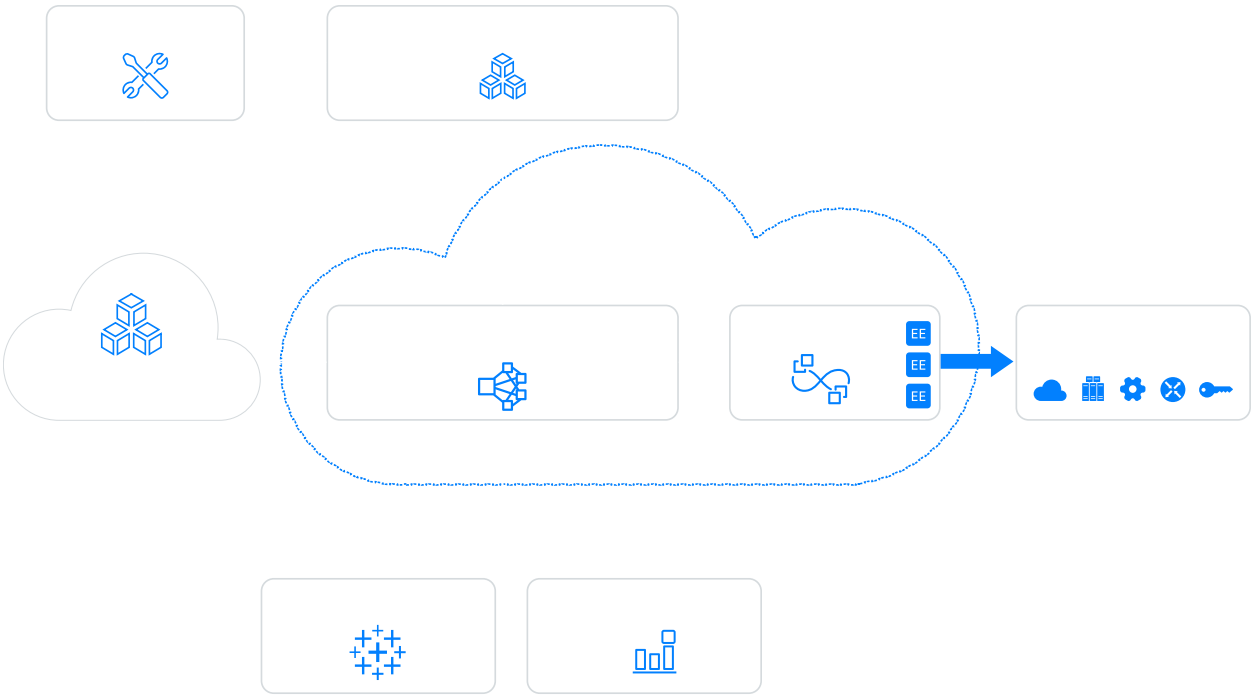
Обзор архитектуры

Astra Automation является гибридной платформой, объединяющей облачный сервис (Astra Automation Hub), доступный всем пользователем платформы, и компоненты управления, устанавливаемые в пространстве организации.

3.1 Структура платформы

Платформа включает ряд компонентов для управления *инфраструктурой заказчика* (Managed IT infrastructure).





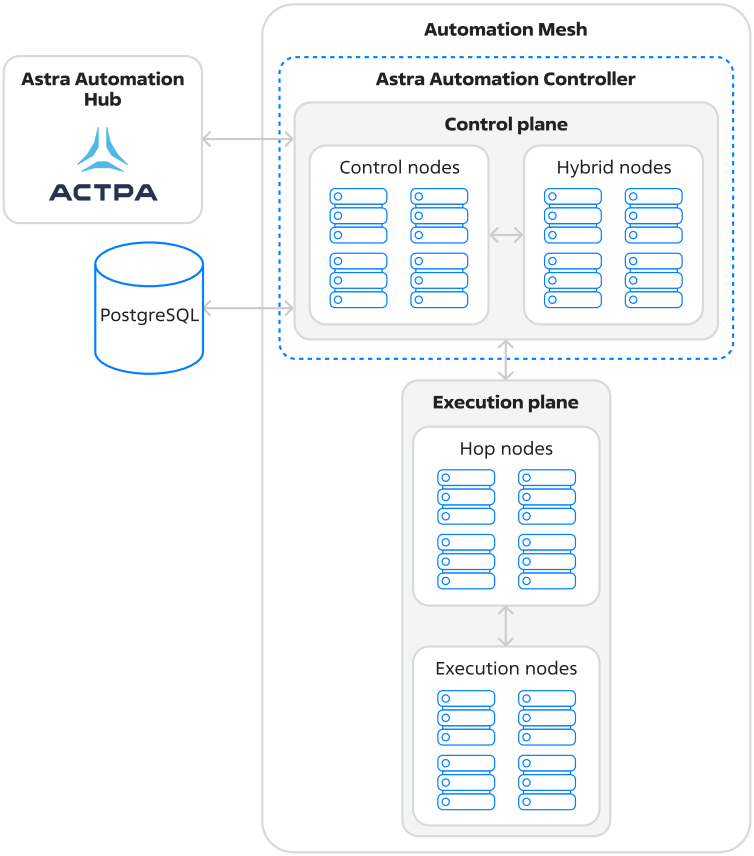
К платформе относятся следующие компоненты:

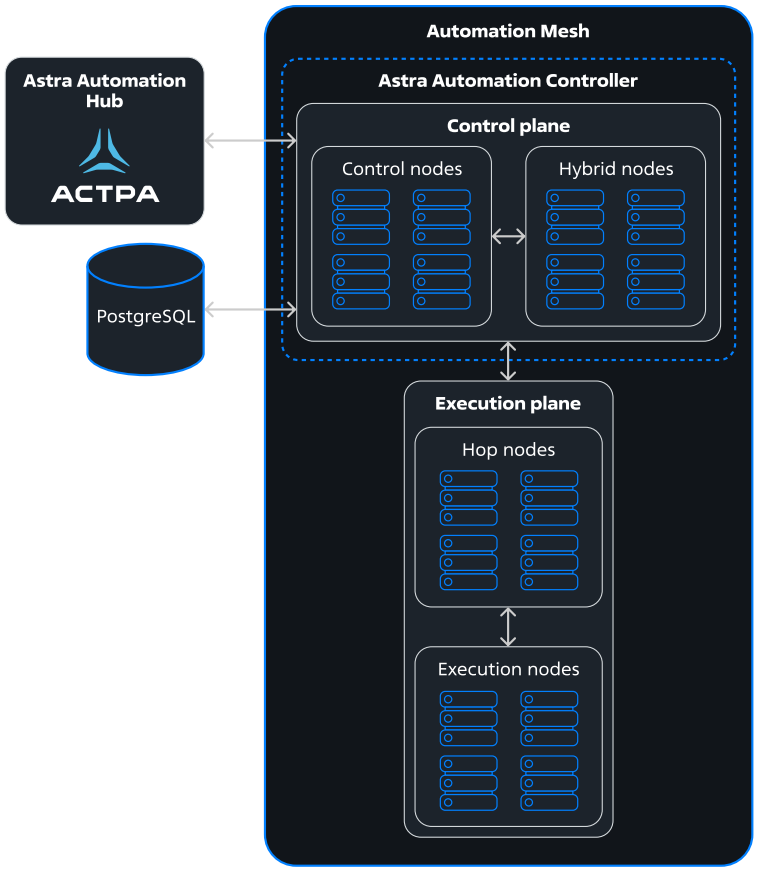
- *Контроллер* (Astra Automation Controller) представляет собой плоскость управления (control plane) на основе кластера. Он предоставляет пользователю удобные средства управления через графический web-интерфейс, вызовы API, командный интерфейс (CLI, command-line interface) и специальную коллекцию Ansible.
- *Плоскость исполнения* (Execution plane) является частью общего кластера, объединяющей узлы, которые выполняют задания контроллера. Особенности плоскости исполнения:
  - Она обеспечивает эффективное управление *распределенной* инфраструктурой путем расположения узлов в одном сетевом окружении с узлами управляемой инфраструктуры.
  - Непосредственным исполнителем заданий в виде playbook является среда исполнения (EE, Execution Environment), выполненная в виде контейнера Podman.
  - Контроллер позволяет назначать различные задания на исполнение определенным группам узлов плоскости исполнения, а также выполнять их с помощью указанных образов EE.
- *Сеть прикладного уровня* (Mesh) объединяет плоскость управления и плоскость исполнения в единый кластер. Настройки платформы позволяют гибко определять связи (топологию) между компонентами этих плоскостей.
- *Astra Automation Hub* является облачным реестром инфраструктурного кода для реализации требуемых задач автоматизации. В состав реестра входят различные типы инфраструктурного кода:
  - набор готовых сценариев;
  - сертифицированные коллекции Ansible для выполнения различных задач автоматизации;
  - набор модулей Terraform для развертывания базовой инфраструктуры в виртуальном окружении.

- *Собственный реестр* инфраструктурного кода (Private Automation Hub) позволяет организации использовать контент, предоставляемый через Astra Automation Hub или другие источники, а также хранить и использовать собственный контент и образы EE. При создании собственного реестра последний получает контент из Astra Automation Hub.
- *Средства разработки* (CDK, content development kit) содержат необходимые инструменты для организации полноценного процесса разработки собственного инфраструктурного кода и образов EE. Созданные артефакты следует сохранять в собственном реестре инфраструктурного кода.
- *Служба управления событиями* (Event-Driven Ansible) позволяет отслеживать события, возникающие в различных компонентах инфраструктуры пользователя. Применение этой службы приводит к существенному снижению объема ручной работы а также к быстрому своевременному реагированию на изменившуюся ситуацию в инфраструктуре.
- Средства аналитики (Analytics) позволяют собирать данные о работе контроллера и формировать отчеты для анализа эффективности организации процессов автоматизации.

3.2 Структура кластера

Центральное место в структуре Astra Automation занимает кластер управления, состоящий из набора узлов, объединенных сетью Mesh.





В Astra Automation используются узлы четырех типов:

- *Управляющие узлы* (control nodes) управляют работой контроллера и формируют задания автоматизации.
- *Исполняющие узлы* (execution nodes) используются для непосредственного запуска заданий автоматизации, сформированных управляющими узлами.
- *Промежуточные* (переходные) узлы (hop nodes) используются в качестве посредников между управляющими и исполняющими узлами. Их назначение – связывать узлы, когда прямой доступ управляющих узлов к исполняющим по какой-либо причине невозможен, например, из-за технических ограничений или соображений безопасности.

  Примечание: Промежуточные узлы нельзя использовать для запуска заданий.

- *Гибридные узлы* (hybrid nodes) выполняют функции управляющих и исполняющих узлов.

Astra Automation Hub – это облачный сервис ПАО Группа Астра, предоставляющий зарегистрированным пользователям доступ к хранилищу инфраструктурного кода.

3.3 Сеть

С помощью полносвязной сети прикладного уровня с автоматическим обнаружением узлов (automation mesh) кластер Astra Automation объединяет узлы разных типов в две плоскости – плоскость управления (control plane) и плоскость исполнения (execution plane). При объединении узлов в сеть происходит настройка связей между узлами плоскостей.

Плоскость управления представляет собой Astra Automation Controller и состоит из управляющих и гибридных узлов, а плоскость исполнения – из промежуточных и исполняющих узлов.

Узлы кластера объединяются в сеть с помощью службы рецепторов, установленной на каждом узле.

3.4 СУБД

СУБД на базе PostgreSQL используется для хранения данных контроллера. Она должна быть доступна для всех узлов из плоскости управления.

По умолчанию СУБД развертывается вместе с контроллером, однако, можно использовать и уже существующую СУБД, развернутую на собственном оборудовании или в одном из облачных сервисов управляемых баз данных.

При развертывании СУБД средствами контроллера необходимо выделить для нее отдельный узел, который не должен использоваться для других целей.

Astra Automation Controller не имеет поддержки кластеризации СУБД. Это значит, что высокую доступность СУБД следует обеспечить самостоятельно.

Использование внешней СУБД накладывает следующие ограничения:

- Контроллер не поддерживает автоматическое переключение на новый мастер в случае переключения мастера в кластере СУБД.
- Не рекомендуется использовать менеджеры соединений типа pgBouncer[3].
- Нельзя использовать менеджер соединений в режиме «Transaction pooling».

[3] https://www.pgbouncer.org/

ГЛАВА 4

Развертывание платформы

Развертывание Astra Automation предусматривает использование одного из двух основных видов окружения:

- виртуальное окружение, предоставляющее необходимые ВМ;
- контейнерное окружение, управляемое с помощью Kubernetes.

Представленная далее процедура ориентирована на использование ВМ.

Для тестового окружения можно создать необходимые ВМ с помощью *Vagrant*.

4.1 Системные требования

В этом разделе представлены системные требования, предъявляемые к рабочим местам администратора и пользователей, узлам платформы и используемой ей СУБД.

4.1.1 Рабочие места администратора и пользователей

Для работы с платформой через веб-интерфейс необходимо наличие на рабочих станциях администратора и пользователей одного из следующих веб-браузеров:

Браузер	Версия, не ниже
Google Chrome	109
Mozilla Firefox	102
Яндекс Браузер	23.7.2.767

4.1.2 Требования к узлам

Типовые системные требования к управляющим, исполняющим, промежуточным и гибридным узлам кластера Astra Automation Controller:

Параметр	Минимальное значение	Рекомендуемое значение
Количество ядер CPU	4	≥ 8
Количество RAM, ГБ	16	≥ 16
Дисковое пространство, ГБ	50	≥ 100
Тип дискового накопителя	SSD	SSD
Тип ОС	Astra Linux Special Edition 1.7.4	Astra Linux Special Edition 1.7.5 или выше
Версия ядра ОС	`5.15.0-generic`	`6.1.50-generic`
Режим защищенности ОС	Базовый («Орел») или усиленный («Воронеж»)	

- Максимальное количество узлов в кластере – 20.

  Предупреждение: Не действует, начиная с версии `1.0-upd2`.

- Все узлы должны иметь постоянные IP-адреса или доменные имена (в зависимости от того, как заполняется файл инвентаря, используемый установщиком).
- Все узлы плоскости управления должны размещаться достаточно близко друг к другу с целью снижения задержек при передаче данных.

Фактический список узлов кластера и их технические характеристики определяются требованиями к отказоустойчивости, обеспечению высокой доступности и быстродействию.

4.1.3 СУБД

Для хранения данных Astra Automation требуются базы данных в СУБД PostgreSQL.

Если вы хотите использовать уже существующий кластер PostgreSQL, к нему предъявляются следующие типовые системные требования:

Параметр	Значение
Версия PostgreSQL	13
Дисковое пространство, ГБ	не менее 20
Производительность, IOPS	не менее 1000

Если отдельного кластера PostgreSQL у вас нет, СУБД следует развернуть на одном из узлов. Типовые системные требования к узлу СУБД:

Параметр	Минимальное значение	Рекомендуемое значение
Количество ядер CPU	4	≥ 8
Количество RAM, ГБ	8	≥ 16
Дисковое пространство, ГБ	20	≥ 150
Тип дискового накопителя	HDD	SSD
Производительность, IOPS	1500	> 1500

Astra Automation не имеет поддержки кластеризации по отношению к СУБД. Это значит, что высокую доступность БД следует обеспечить сторонними средствами.

4.2 Подготовка узлов

Для успешного развертывания платформы необходимо настроить узел, на котором будет запущен установщик, и все узлы платформы. Узел, на котором запускается установщик, называется *установочным узлом*. При необходимости его можно включить в состав платформы.

4.2.1 Подготовка установочного узла

Для подготовки установочного узла подключитесь к нему любым удобным способом и выполните следующие действия:

1. Если в файле `/etc/apt/sources.list` не содержатся ссылки на базовый (base) и расширенный (extended) репозитории Astra Linux Special Edition, добавьте их. Например, для Astra Linux Special Edition 1.7.5 такие ссылки имеют следующий вид:

   ```
   deb http://dl.astralinux.ru/astra/frozen/1.7_x86-64/1.7.5/repository-base     1.7_
   ↪x86-64 main non-free contrib
   deb http://dl.astralinux.ru/astra/frozen/1.7_x86-64/1.7.5/repository-extended 1.7_
   ↪x86-64 main non-free contrib
   ```

   Подробности о сетевых репозиториях Astra Linux Special Edition см. в статье Справочного центра Интернет-репозитории Astra Linux Special Edition x.7[4].

2. В каталоге `/etc/apt/sources.list.d/` создайте файл `astra-automation.list` со ссылкой на репозиторий Astra Automation:

   ```
   deb https://dl.astralinux.ru/aa/aa-debs-for-alse-1.7 <version> main
   ```

   Доступные версии продукта опубликованы в таблице *История обновлений*. Например, для установки версии 1.0-upd2 вместо <version> подставьте значение `1.0-upd2`.

3. Обновите список доступных пакетов:

   ```
   sudo apt update
   ```

4. Установите пакет `astra-automation-setup`:

[4] https://wiki.astralinux.ru/pages/viewpage.action?pageId=158598882

```
sudo apt install astra-automation-setup --yes
```

По окончании установки выводится сообщение следующего вида:

```
---------------------------------
Welcome to Astra Automation Setup!
---------------------------------

Follow next steps to install Astra Automation:
1. Go to /opt/rbta/aa/astra-automation-setup
2. Edit inventory file
3. Run aa-setup
4. Enjoy!

For guidance on installing Astra Automation, consult https://docs.astra-
↪automation.ru/
```

4.2.2 Подготовка узлов

Для подготовки рабочего окружения к развертыванию платформы выполните следующие действия на всех узлах:

1. Обеспечьте сетевую связность всех узлов.
2. Настройте корректное разрешение имен узлов.
3. Настройте доступ к узлам по SSH.

   Инструкции по настройке приведены в секции *Ключи SSH для доступа к узлам*.
4. Включите выполнение команд с использованием sudo без ввода пароля.

   Инструкции по настройке приведены в секции *Использование sudo без ввода пароля*.

4.2.3 Ключи SSH для доступа к узлам

Рекомендуемым способом доступа к узлам является подключение по протоколу SSH.

Для создания пар ключей SSH на установочном узле выполните следующие действия:

1. В каталоге `/opt/rbta/aa/astra-automation-setup/` создайте подкаталог `ssh-keys/`.

   ```
   sudo mkdir -p /opt/rbta/aa/astra-automation-setup/ssh-keys/
   ```
2. Перейдите в созданный каталог:

   ```
   cd /opt/rbta/aa/astra-automation-setup/ssh-keys/
   ```
3. Для каждого узла создайте пару ключей SSH:

   ```
   sudo ssh-keygen -C "<comment>" -f ./<filename> -N "<password>"
   ```

   где:

   - <comment> – комментарий к ключу, например, FQDN или IP-адрес узла, для которого создается ключ.

- <filename> – шаблон имени файлов, в которые следует сохранить пару ключей. Приватный ключ сохраняется в файл <filename>, публичный – <filename>.pub.
- <password> – пароль для защиты приватного ключа. Если пароль пустой, приватный ключ создается без защиты.

Для получения доступа к узлу по SSH выполните на нем следующие действия:

1. В домашнем каталоге пользователя-администратора создайте подкаталог .ssh/, а в нем – файл authorized_keys.
2. Добавьте в файл authorized_keys содержимое соответствующего публичного ключа, например (часть содержимого ключа опущена с целью сокращения):

```
ssh-rsa AAAAB3NzaC1yc2EAd...4bwy3tY2/ node1.example.com
```

Совет: Если узел разрешает подключение по SSH с использованием пароля, для размещения на нем публичной части ключа SSH можно использовать утилиту ssh-copy-id:

```
ssh-copy-id -i <filename> <user>@<host>
```

где:

- <filename> – путь к файлу публичного ключа SSH;
- <user> – имя пользователя узла;
- <host> – IP-адрес или FQDN узла.

4.2.4 Использование sudo без ввода пароля

Для настройки узлов с помощью Ansible необходимо разрешить на них использование команды sudo без ввода пароля.

Чтобы выключить запрос пароля при использовании sudo, выполните команду:

```
sudo astra-sudo-control disable
```

Примечание: В образах, указанных в разделе *Образы виртуальных машин*, использование sudo без ввода пароля включено по умолчанию.

4.3 Подготовка инвентаря

В этом разделе описаны параметры, используемые для развертывания типовых конфигураций Astra Automation.

4.3.1 Инвентарь платформы

Параметры компонентов платформы задаются в файле *инвентаря* `inventory`, находящемся в каталоге `/opt/rbta/aa/astra-automation-setup/`. Для этого используются группы:

- `automationcontroller` – узлы плоскости управления;
- `execution_nodes` – узлы плоскости исполнения.

Важно: Файл инвентаря установщика Astra Automation должен иметь формат INI.

Выберите подходящую конфигурацию платформы и укажите параметры узлов в инвентаре.

4.3.2 Реквизиты доступа к узлам

С целью сокращения в примерах опущены параметры, в которых задаются реквизиты для подключения к узлам:

- `ansible_user` – имя пользователя, используемое для подключения к узлу;
- `ansible_ssh_private_key_file` – путь к файлу приватного ключа SSH.

Если для подключения ко всем узлам используются одни и те же реквизиты, укажите их в секции `[all:vars]`, например:

```
[automationcontroller]
node1.example.com
node2.example.com

[all:vars]
ansible_user=administrator
ansible_ssh_private_key_file=./ssh-keys/ssh_key
```

Если реквизиты для доступа к узлам различаются, укажите их в параметрах соответствующих узлов, например:

```
[automationcontroller]
node1.example.com  ansible_user=alex  ansible_ssh_private_key_file=./ssh-keys/node1_
↪key
node2.example.com  ansible_user=john  ansible_ssh_private_key_file=./ssh-keys/node2_
↪key

[execution_nodes]
node3.example.com  ansible_user=jack  ansible_ssh_private_key_file=./ssh-keys/node3_
↪key
```

4.3.3 Главный узел

При развертывании платформы данные о ее узлах указываются в инвентаре установщика. При первом запуске установщика узел, указанный первым в группе `automationcontroller`, становится главным узлом (primary node).

К главном узлу предъявляются следующие требования:

- При развертывании или обновлении контроллера главный узел всегда должен быть доступен.
- Запись о главном узле в инвентаре установщика должна размещаться на первой позиции в группе `automationcontroller`.

Кластер из одного узла

В самом простом случае кластер и СУБД могут быть развернуты на одном узле. Такая конфигурация не подходит для промышленной эксплуатации, но проста в развертывании и нетребовательна к ресурсам.

Если используется внутренняя СУБД, указывать ее параметры не требуется – она будет развернута на том же узле, что и контроллер.

Укажите сведения об узле в группе `automationcontroller`, например:

```
[automationcontroller]
aac.example.com  ansible_user=alex
```

Если развертывание кластера производится на установочном узле, для подключения не требуется использование SSH. В этом случае в параметре `ansible_connection` укажите тип подключения `local`:

```
[automationcontroller]
aac.example.com  ansible_user=alex  ansible_connection=local
```

Кластер из гибридных узлов

По умолчанию все узлы, входящие в группу `automationcontroller`, считаются гибридными. Допускается создание кластера, состоящего только из гибридных узлов.

Пример заполнения группы `automationcontroller`:

```
[automationcontroller]
hybrid-node-1.example.com
hybrid-node-2.example.com
hybrid-node-3.example.com
```

Если планируется в будущем добавить в кластер узлы других типов, для управляющих узлов рекомендуется создать дополнительную группу, в переменных которой явно задать тип узлов:

```
[automationcontroller]
node-1.example.com
node-2.example.com
node-3.example.com
node-4.example.com
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Отдельная группа для управляющих узлов
[control_group]
node-4.example.com

# Переменные группы control_group
[control_group:vars]
node_type=control
```

Кластер из одного управляющего и одного исполняющего узла

Для создания кластера из одного управляющего и одного исполняющего узла заполните инвентарь следующим образом:

- В группу automationcontroller добавьте запись об управляющем узле.
- Создайте группу execution_nodes и добавьте в нее запись об исполняющем узле.
- Создайте для группы automationcontroller переменные node_type и peers со значениями control и execution_nodes соответственно.

Пример описания кластера из одного управляющего и одного исполняющего узла:

```
# Плоскость управления
[automationcontroller]
control-node-1.example.com

# Переменные группы automationcontroller
[automationcontroller:vars]
node_type=control
peers=execution_nodes

# Плоскость исполнения
[execution_nodes]
execution-node-1.example.ru
```

Примечание: Переменная peers со значением execution_nodes указывает, что управляющие узлы должны связываться с исполняющими узлами напрямую, без промежуточных узлов.

Кластер из управляющих и исполняющих узлов

Отказоустойчивая конфигурация в минимальной комплектации требует наличия в кластере хотя бы двух управляющих и двух исполняющих узлов. Для получения такой конфигурации заполните инвентарь следующим образом:

1. Добавьте в группу automationcontroller записи об управляющих узлах.
2. Создайте группу execution_nodes и добавьте в нее записи об исполняющих узлах.
3. Создайте для группы automationcontroller переменные node_type и peers со значениями control и execution_nodes соответственно.

Пример описания кластера из двух управляющих и двух исполняющих узлов:

```
# Плоскость управления
[automationcontroller]
control-node-1.example.com
control-node-2.example.com

# Переменные группы automationcontroller
[automationcontroller:vars]
node_type=control
peers=execution_nodes

# Плоскость исполнения
[execution_nodes]
execution-node-1.example.com
execution-node-2.example.com
```

Все управляющие узлы здесь имеют прямой доступ ко всем исполняющим узлам.

Кластер с удаленным исполнением

Конфигурация с промежуточными узлами позволяет подключаться через них к удаленным исполняющим узлам.

Пример отказоустойчивого кластера с промежуточным узлом `hop-node-1`, через который выполняется подключение к исполняющему узлу `execution-node-2`:

```
# Плоскость управления
[automationcontroller]
controller-node-1.example.com
controller-node-2.example.com

[automationcontroller:vars]
node_type=control
peers=instance_group_local

# Плоскость исполнения
[execution_nodes]
execution-node-1.example.com
execution-node-2.example.com
execution-node-3.example.com
hop-node-1.example.com

# Исполняющие узлы с прямым подключением к управляющим узлам
[instance_group_local]
execution-node-1.example.com
execution-node-2.example.com

# Промежуточный узел
[hop]
hop-node-1.example.com

[hop:vars]
# Прямое подключение к управляющим узлам
node_type=hop
peers=automationcontroller

# Удаленный исполняющий узел
```

(continues on next page)

(продолжение с предыдущей страницы)

```
[instance_group_remote]
execution-node-3.example.com

# Прямое подключение к узлам из группы hop
[instance_group_remote:vars]
peers=hop
```

Примечание: Переменная peers со значением instance_group_local указывает, что управляющие узлы должны связываться с исполняющими узлами из группы instance_group_local напрямую, без промежуточных узлов.

4.4 Учетные данные системного администратора

Укажите учетные данные системного администратора Astra Automation Controller в конфигурационном файле /opt/rbta/aa/astra-automation-setup/inventory в глобальных переменных (раздел [all:vars]), например:

```
[all:vars]
admin_username=admin
admin_email=admin@example.com
awx_install_admin_password=p@ssw0rd!
```

Назначение переменных:

- admin_username – имя учетной записи;
- admin_email – адрес электронной почты;
- awx_install_admin_password – пароль.

4.5 Сертификаты TLS/SSL

По умолчанию Astra Automation использует для защиты подключения сапомодписанный сертификат TLS.

Чтобы использовать сертификат, выданный удостоверяющим центром, выполните следующие действия:

1. Скопируйте файлы сертификата и соответствующего ему ключа в каталог /opt/rbta/aa/astra-automation-setup/ или один из его подкаталогов.
2. В файле inventory в секцию [all:vars] добавьте переменные web_servercertificates_cert и web_servercertificates_key, в значении которых укажите полные пути к файлам сертификата и ключа соответственно, например:

   ```
   [all:vars]
   web_servercertificates_cert = /opt/rbta/aa/astra-automation-setup/cert.pem
   web_servercertificates_key = /opt/rbta/aa/astra-automation-setup/cert.key
   ```

Подробности об управлении сертификатами см. в документе *Управление сертификатами TLS*.

4.6 Параметры СУБД

Astra Automation позволяет развернуть СУБД PostgreSQL на одном из узлов или использовать подключение к уже существующей базе данных (далее – БД) в отдельном кластере, например, в одном из облачных сервисов управляемых баз данных.

Примечание: СУБД PostgreSQL используется платформой для хранения данных, но не является его частью. Настройка отказоустойчивой конфигурации PostgreSQL в этом руководстве не рассматривается. Для получения соответствующих инструкций обращайтесь к документации PostgreSQL[5].

4.6.1 Внутренняя СУБД

Чтобы на одном из узлов развернуть СУБД и создать в ней БД для контроллера, выполните следующие действия:

1. Создайте в инвентаре группу узлов database и добавьте в нее сведения о нужном узле, например:

```
[database]
dbms.example.com
```

Важно: Узел, используемый для развертывания PostgreSQL, не должен использоваться для других целей, например, запуска заданий или управления контроллером.

---

2. Укажите значения глобальных переменных с настройками подключения к узлу СУБД.

4.6.2 Внешняя СУБД

Если у вас уже есть кластер PostgreSQL и в нем существует БД для контроллера, чтобы использовать ее, выполните следующие действия:

1. Если в инвентаре есть группа database, закомментируйте ее:

```
#[database]
#dbms.example.com
```

2. Укажите значения глобальных переменных с настройками подключения к кластеру PostgreSQL.

[5] https://www.postgresql.org/docs/current/high-availability.html

4.6.3 Параметры PostgreSQL

В глобальных переменных (раздел `[all:vars]`) можно задать параметры подключения к СУБД:

```
[all:vars]
pg_host = ''
pg_port = 5432
pg_database = 'aac'
pg_username = 'aac'
pg_password = ''
pg_sslmode = 'prefer'
postgres_use_ssl = True
postgres_ssl_cert = <path_to_SSL_cert>
postgres_ssl_key = <path_to_SSL_key>
```

Здесь:

- `pg_host` – IP-адрес или FQDN узла для подключения к СУБД. Значение по умолчанию – пустая строка (используется локальное подключение).
- `pg_port` – порт для подключения к СУБД. Значение по умолчанию – 5432.
- `pg_database` – название БД.
- `pg_username` – имя пользователя БД.
- `pg_password` – пароль пользователя БД. При использовании локального подключения пароль указывать не требуется.
- `pg_sslmode` – режим использования SSL.

  Поддерживаются следующие значения:

  - `disable` – проверка SSL выключена;
  - `allow` – подключение к БД будет защищено с помощью SSL, если использование шифрования требуют настройки сервера;
  - `prefer` – подключение к БД будет защищено с помощью SSL, если использование шифрования поддерживается настройками сервера;
  - `require` – защита подключения с помощью SSL необходима, верификация сервера не производится;
  - `verify-ca` – защита подключения с помощью SSL необходима, верификация сервера производится;
  - `verify-full` – защита подключения с помощью SSL необходима, производится строгая верификация сервера.
- `postgres_use_ssl` – использование SSL:
  - `True` – включено;
  - `False` – выключено.
- `postgresql_ssl_cert` – полный путь к файлу сертификата SSL.
- `postgresql_ssl_key` – полный путь к файлу ключа для сертификата SSL.

Примечание: Файлы сертификата и его ключа должны размещаться в каталоге `/opt/rbta/aa/astra-automation-setup/` или одном из его подкаталогов.

Подробности о поддерживаемых параметрах шифрования см. в документации PostgreSQL[6].

4.7 Установка системы

После указания в инвентаре учетных данных системного администратора Astra Automation Controller, СУБД и параметров узлов запустите развертывание платформы. Ее работоспособность проверьте простейшим тестированием.

4.7.1 Процедура установки

В каталоге `/opt/rbta/aa/astra-automation-setup/` выполните команду:

```
./aa-setup --log-path=<log_path>
```

В значении параметра `--log-path` укажите путь к каталогу для сохранения журнала установки. Этот каталог должен быть доступен для записи от имени текущего пользователя.

Чтобы использовать путь к журналу установки по умолчанию, выполните команду с привилегиями суперпользователя:

```
sudo ./aa-setup
```

По умолчанию необходимые пакеты устанавливаются из репозитория `dl.astralinux.ru`.

Дождитесь завершения выполнения команды, это может занять некоторое время. По окончании установки в терминал выводится набор строк вида (названия узлов и их количество должны совпадать со значениями, указанными в файле инвентаря):

```
INFO: PLAY RECAP *********************************************************************
aac01                      : ok=233  changed=118  unreachable=0    failed=0    ␣
↪skipped=74   rescued=0    ignored=4
aac02                      : ok=191  changed=107  unreachable=0    failed=0    ␣
↪skipped=72   rescued=0    ignored=2
db                         : ok=41   changed=20   unreachable=0    failed=0    ␣
↪skipped=9    rescued=0    ignored=0
ex01                       : ok=82   changed=46   unreachable=0    failed=0    ␣
↪skipped=47   rescued=0    ignored=1
ex02                       : ok=82   changed=46   unreachable=0    failed=0    ␣
↪skipped=47   rescued=0    ignored=1
INFO: Astra Automation Setup has been finished

You can now login to Astra Automation Controller using following credentials:

Login: admin
Password: p@ssw0rd!
```

[6] https://www.postgresql.org/docs/current/libpq-ssl.html

Развертывание считается успешным, если для всех узлов в поле `failed` указано значение `0`.

Примечание: Синтаксис и полный список параметров утилиты `aa-setup` приведены в *справочных данных*.

4.7.2 Проверка работоспособности контроллера

Чтобы проверить корректность развертывания платформы, выполните следующие действия:

1. Запустите браузер и введите в адресной строке IP-адрес или FQDN управляющего узла Astra Automation Controller.

   Если управляющих узлов несколько – выберите любой из них и подключитесь к нему.

Примечание: Если используется самоподписанный сертификат, в окне браузера может выводиться предупреждение о вероятной угрозе безопасности. В данном случае подтвердите исключение безопасности.

---

1. В форму для ввода учетных данных введите имя и пароль администратора контроллера.
2. На панели навигации выберите раздел *Администрирование* ▸ *Узлы контроллера* (*Administration* ▸ *Instances*).
3. Убедитесь, что все узлы, указанные ранее в файле `inventory`, находятся в статусе **Готов** (Ready).

4.8 Обновление версии

Обновление версии Astra Automation осуществляется с помощью утилиты `aa-setup`. Она может обновить Astra Automation до определенной или последней версии.

Примечание: Так как возможность обновления версии с помощью утилиты `aa-setup` добавлена в версии `1.0-upd1`, использовать ее для обновления можно будет начиная с версии `1.0-upd1`. Для обновления с версии `1.0` до `1.0-upd1` воспользуйтесь *универсальным способом обновления*.

4.8.1 Подготовка к обновлению

Необходимо обеспечить выполнение следующих условий:

- Для обновления используется тот же установочный узел, с помощью которого происходило первоначальное развертывание платформы. Начиная с версии `1.0-upd2`, вместо такого узла можно воспользоваться другим, на котором установлен пакет для развертывания Astra Automation и восстановлены необходимые настройки установочного узла.
- На этом узле остался в неизменном виде каталог `/opt/rbta/aa/astra-automation-setup/`, в частности файл инвентаря `/opt/rbta/aa/astra-automation-setup/inventory`, описывающий состав платформы.

4.8.2 Обновление до последней версии

Запустите процесс обновления одним из следующих способов:

- Запуск с использованием привилегий суперпользователя:

```
sudo ./aa-setup --upgrade
```

- Запуск с привилегиями обычного пользователя:

```
./aa-setup --upgrade --log-path=<log_path>
```

  В значении параметра `--log-path` укажите путь к каталогу для сохранения журнала установки. Этот каталог должен быть доступен активному пользователю для записи.

При запуске с параметром `--upgrade` утилита `aa-setup` выполняет следующие действия:

1. Соединяется с репозиторием, из которого был установлен Astra Automation, и считывает из него список доступных версий.
2. Определяет версию пакета, использованного для установки или обновления Astra Automation в предыдущий раз:
   - При обновлении с версии `1.0-upd2` и более поздней текущая версия хранится в файле `/etc/astra_automation.version`.
   - При обновлении с более ранних версий текущая версия хранится в `/etc/apt/sources.list.d/astra-automation.list`.
3. Выполняет обновление при необходимости:
   - если последняя версия в репозитории совпадает с версией, установленной на узле, то обновление не производится, `aa-setup` завершает свою работу;
   - если версия, установленная на узле, ниже самой последней в репозитории, то автоматически производится обновление утилиты `aa-setup` на самую новую версию, утилита перезапускается и обновляет Astra Automation.

Необходимо дождаться завершения выполнения команды, это может занять некоторое время. По окончании обновления в терминал выводится строка вида:

```
aac      : ok=233  changed=130  unreachable=0    failed=0    skipped=55    rescued=0 ␣
↪  ignored=4
```

Обновление считается успешным, если в поле `failed` указано значение `0`.

4.8.3 Обновление до определенной версии

Для обновления Astra Automation до определенной версии используйте параметр `--upgrade` в сочетании со следующими параметрами:

- `--repo-url` – когда необходимо указать репозиторий, отличный от используемого по умолчанию (`dl.astralinux.ru`).

  Например, для использования репозитория `https://example.com/artifactory/automation-debs-for-alse` аналогично тому, как тот же репозиторий использовался бы при начальной установке Astra Automation, воспользуйтесь командой:

  ```
  sudo ./aa-setup --upgrade --repo-url https://example.com/artifactory/automation-
  ↪debs-for-alse
  ```

- `--product-version` – когда необходимо обновить платформу до определенной версии.

  Если необходимо обновить платформу до определенной версии, например, `1.0-upd3`, воспользуйтесь командой:

  ```
  sudo ./aa-setup --upgrade --product-version 1.0-upd3
  ```

Если необходимо использовать другой репозиторий и указать определенную версию воспользуйтесь командой:

```
sudo ./aa-setup --upgrade --repo-url https://some.company.ru/artifactory/automation-
↪debs-for-alse --product-version 1.0-upd3
```

4.8.4 Универсальный способ обновления

Примечание: Данный способ обновления не является рекомендуемым. Его следует использовать в крайнем случае. Например, когда невозможно удовлетворить требованиям по подготовке.

---

1. Убедитесь, что в списке репозиториев APT в каталоге `/etc/apt/` указана ссылка на репозиторий, содержащий требуемую версию установочного пакета Astra Automation:

   ```
   deb https://dl.astralinux.ru/aa/aa-debs-for-alse-1.7 1.0-upd1 main
   ```

2. Обновите индекс пакетов:

   ```
   sudo apt update
   ```

3. Обновите пакет `astra-automation-setup`:

   ```
   sudo apt install --only-upgrade astra-automation-setup
   ```

4. Запустите процесс обновления таким же образом, как вы изначально развертывали платформу:

   ```
   sudo ./aa-setup
   ```

4.9 Расчет ресурсов

Важно: Эта часть документации находится в стадии разработки.

4.10 Справочные данные

Для развертывания платформы и управления ею в процессе эксплуатации используются специальные утилиты. Здесь приведены справочные данные по их назначению и использованию.

4.10.1 aa-setup

В этом документе описаны назначение и синтаксис утилиты Astra Automation Setup (`aa-setup`).

Назначение

Утилита `aa-setup` выполняет следующие действия:

- развертывает платформу в виде нескольких связанных компонентов;
- обновляет версии и структуру компонентов платформы;
- выполняет резервное копирование и восстановление базы данных;
- генерирует и распространяет секретный ключ;
- проверяет версии Astra Automation, доступные в репозитории;
- удаляет Astra Automation.

Установка и запуск

Для установки утилиты `aa-setup` выполните следующие действия:

1. В каталоге `/etc/apt/sources.list.d/` создайте файл `astra-automation.list` со ссылкой на репозиторий контроллера:

   ```
   deb https://dl.astralinux.ru/aa/aa-debs-for-alse-1.7 <version> main
   ```

   Доступные версии продукта опубликованы в таблице *История обновлений*. Например, для установки версии `1.0-upd2` вместо <version> подставьте значение `1.0-upd2`.

2. Обновите список доступных пакетов:

   ```
   sudo apt update
   ```

3. Выполните команду установки:

   ```
   sudo apt install astra-automation-setup --yes
   ```

Описание аргументов

Запустить утилиту можно в каталоге `/opt/rbta/aa/astra-automation-setup/` со следующими аргументами:

- **`-h`**, **`--help`**

  Выводит информацию о доступных опциях и соответствующих аргументах.

- **`-i`** `<inventory_path>`, **`--inventory`** `<inventory_path>`

  Путь к файлу инвентаризации. Значение по умолчанию – `./inventory`.

- **`-p`** `<log_path>`, **`--log-path`** `<log_path>`

  Путь к альтернативному каталогу для сохранения журнала установки. Текущий пользователь должен иметь привилегию на создание файлов и запись в этом каталоге. Значение по умолчанию – `./`.

- **`-k`**, **`--generate-key`**

  Генерирует и распространяет новый секретный ключ, используемый для выполнения следующих задач:

  - шифрование данных пользовательских сессий;
  - генерация токенов для паролей;
  - защита данных контроллера.

- **`-b`**, **`--backup`**

  Сохраняет в виде *резервной копии* данные, необходимые для полного восстановления платформы на узлах, описанных в инвентаре установщика.

- **`-r`**, **`--restore`**

  Использует резервную копию данных для восстановления платформы на узлах, описанных в инвентаре установщика.

- **`-d`**, **`--debug`**

  Включает режим отладки для более подробного вывода информации о процессе.

- **`-u`**, **`--uninstall`**

  Запускает процесс удаления компонентов Astra Automation, перечисленных в файле инвентаризации.

  Added in version 1.0-upd2.

- **`--force-postgres-removal`**

  Запускает процесс удаления PostgreSQL.

  Added in version 1.0-upd1.

- **`--plain`**

  Отключает расширенный вариант текстового пользовательского интерфейса.

- **`--repo-url`** `<repo_url>`

  Указывает URL альтернативного репозитория для установки из него Astra Automation. Например, для использования репозитория `https://example.com/artifactory/automation-debs-for-alse` воспользуйтесь командой:

  ```
  sudo ./aa-setup --repo-url https://example.com/artifactory/automation-debs-
  ↪for-alse
  ```

- **--default-job-ee** <ee_url>

  Указывает на образ контейнейнера, который будет использован в качестве *среды исполнения* по умолчанию (Default execution environment). Например, для использования образа `registry.astralinux.ru/aa/aa-base-ee:0.5.1` воспользуйтесь командой:

  ```
  sudo ./aa-setup --default-job-ee registry.astralinux.ru/aa/aa-base-ee:0.5.1
  ```

  Added in version 1.0-upd2.

- **--component** <component_name>

  Указывает альтернативный компонент репозитория. Например, если необходимо использовать пакеты, размещенные в компоненте `testing`, воспользуйтесь командой:

  ```
  sudo ./aa-setup --component testing
  ```

- **--product-version** <product_version>

  Указывает определенную версию Astra Automation. По умолчанию устанавливается новейшая версия. Например, для установки версии `1.0-upd2` воспользуйтесь командой:

  ```
  sudo ./aa-setup --product-version 1.0-upd2
  ```

- **-v**, **--version**

  Показывает версию утилиты `aa-setup`.

- **--upgrade**

  Обновляет компоненты Astra Automation до новейшей версии. Может быть использован совместно с аргументом `--product-version` для обновления до определенной версии.

  Added in version 1.0-upd1.

  Примечание: Аргумент `--product-version` обязателен при обновлении с версии `1.0-upd1` до `1.0-upd2`.

  Например, для обновления до версии `1.0-up2` используйте команду:

  ```
  sudo ./aa-setup --upgrade --product-version 1.0-up2
  ```

- **--check-releases**

  Выводит на экран терминала версии Astra Automation, доступные в репозитории.

  Added in version 1.0-upd1.

Также вы можете передать дополнительные аргументы для команды `ansible-playbook` после `--`, например:

```
sudo ./aa-setup -b -- -vvv --ask-pass
```

4.10.2 awx-manage

Важно: Эта часть документации находится в стадии разработки.

ГЛАВА 5

Automation Hub

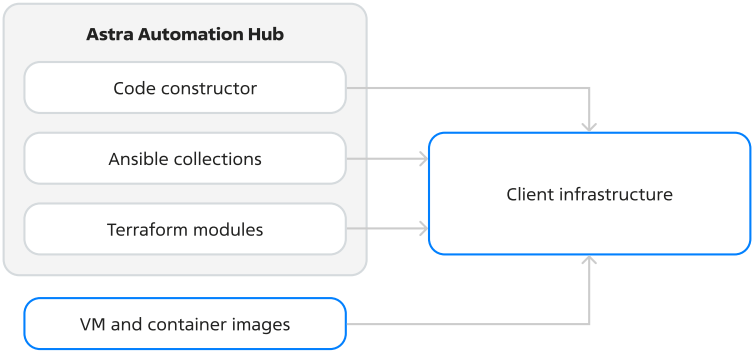
Astra Automation позволяет использовать различные источники инфраструктурного кода и реестры образов контейнеров (для среды исполнения). Основными источниками являются следующие:

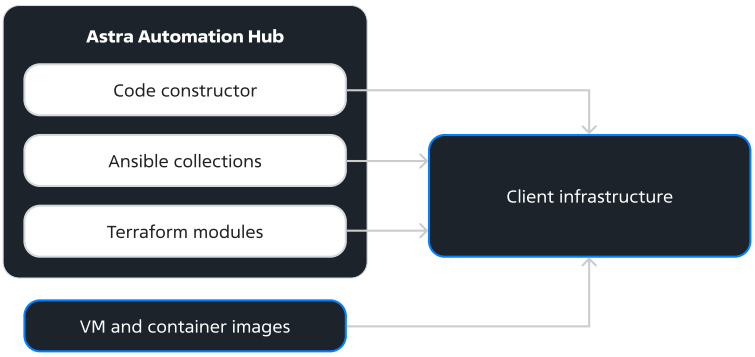
- Astra Automation Hub предоставляет сертифицированный контент для всех потребителей платформы.
- Private Automation Hub содержит контент, выбранный владельцем платформы, в том числе из Astra Automation Hub, а также контент и образы для среды исполнения, разработанные собственными силами.

Для понимания контента и умения работать с ним необходимо ознакомиться с концепциями и способами применения содержимого реестра.

5.1 Astra Automation Hub

Astra Automation Hub представляет собой облачный реестр для хранения инфраструктурного кода для всех потребителей продукта. Владельцы платформы могут использовать его напрямую или через собственный реестр инфраструктурного кода (*private hub*), предварительно поместив содержимое Astra Automation Hub в этот реестр.





В состав Astra Automation Hub входят несколько групп компонентов инфраструктурного кода:

- готовые решения на основе конструктора кода[7];
- коллекции Ansible[8];
- модули Terraform[9];
- простейшие примеры[10].

Возможно как прямое применение библиотеки кода для решения простых задач автоматизации, так и через контроллер.

Astra Automation Hub использует образы ВМ (VM images) на базе операционной системы Astra Linux.

5.1.1 Модули Terraform

Использование модулей позволяет упростить управление ресурсами с помощью Terraform. Модули Terraform, разрабатываемые командой Astra Automation, доступны в реестре[11].

Назначение

Каждому модулю из реестра Astra Automation Hub соответствует отдельный репозиторий. Модули ориентированы на определенное рабочее окружение для развертывания (ПК СВ «Брест», Yandex Cloud, Cloud.ru и другие) и выполняют определенную функцию. Например, модуль yandex-cloud-compute-instance[12] используется для создания одного или нескольких экземпляров (instance) виртуальных машин в облаке Yandex Cloud.

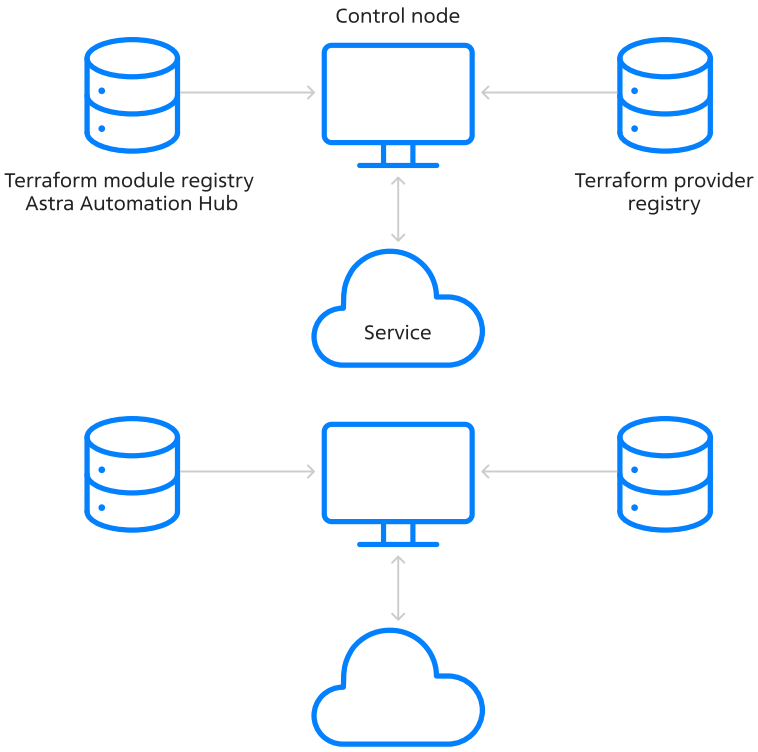
Примечание: Terraform можно использовать для управления инфраструктурой в окружениях, для которых существуют соответствующие провайдеры.

---

[7] https://hub.astra-automation.ru/aa-gca/AAC
[8] https://hub.astra-automation.ru/aa-gca/ARFA
[9] https://hub.astra-automation.ru/aa-gca/AMFT
[10] https://hub.astra-automation.ru/aa-gca/AA
[11] https://hub.astra-automation.ru/aa-gca/AMFT
[12] https://hub.astra-automation.ru/aa-gca/AMFT/yandex-cloud-compute-instance

Структура рабочего окружения

При использовании модулей из реестра Astra Automation Hub схема работы Terraform имеет вид:



1. Terraform загружает на управляющий узел необходимые провайдеры из реестра провайдеров HashiCorp (или другого источника) и модули из реестра Astra Automation Hub.
2. Используя провайдеры и модули, Terraform изменяет ресурсы таким образом, чтобы их фактическое состояние соответствовало описанию в файлах проекта.

Содержание реестра

Название модулей, размещенных в реестре Astra Automation Hub, зависит от используемого провайдера и типа ресурсов. Список существующих модулей представлен в разделе *Модули Terraform*.

Структура модуля

В составе модуля могут быть следующие файлы и каталоги:

- `datasources.tf`

  Источники данных, используемые модулем, например:

```
data "vsphere_datacenter" "datacenter" {
  name = var.vsphere_datacenter_name
}
```

(continues on next page)

(продолжение с предыдущей страницы)

```
data "vsphere_network" "project_network" {
  name          = "${var.name}-port-group"
  datacenter_id = data.vsphere_datacenter.datacenter.id
  depends_on    = [vsphere_host_port_group.port_group]
}
```

Здесь datacenter и project_network – источники данных типов vsphere_datacenter и vsphere_network соответственно. При этом доступ к нужному ресурсу типа vsphere_datacenter производится с помощью поиска по названию (атрибут name), а доступ к ресурсу типа vsphere_network – по названию сети и идентификатору ресурса типа vsphere_datacenter.

- main.tf

  Основной код модуля. Содержит описание создаваемых ресурсов и зависимостей между ними.

- outputs.tf

  Описание значений, возвращаемых модулем.

  Возвращаемые значения имеют следующие особенности:

  - выводятся в терминал при выполнении команд Terraform;
  - в отличие от локальных значений могут быть использованы в других модулях.

  Пример описания возвращаемых значений:

```
output "id" {
  description = "The VPC ID in UUID format."
  value       = sbercloud_vpc.this.id
}

output "name" {
  description = "Specifies the name of the VPC."
  value       = try(sbercloud_vpc.this.name, "")
}

output "subnet_ids" {
  description = "Map of subnet name => subnet ID pairs"
  value       = { for sb in var.subnets : sb.name => sbercloud_vpc_subnet.
↪subnet[sb.name].id if sbercloud_vpc_subnet.subnet[sb.name].id != null }
}
```

  Здесь:

  - id – идентификатор облачной сети;
  - name – название облачной сети;
  - subnet_ids – список идентификаторов подсетей.

- variables.tf

  Описание переменных модуля: названия, типы, ограничения, способы проверки, значения по умолчанию и так далее.

  Например, в модуле sbercloud-vpc[13] имеется такое описание переменной subnets:

[13] https://hub.astra-automation.ru/aa-gca/AMFT/sbercloud-vpc

```
variable "subnets" {
  description = "A list of subnets inside the VPC"
  type = list(object({
    name                              = string,
    cidr                              = string,
    availability_zone                 = string,
    gateway_ip                        = optional(string, null),
    create_nat_gateway                = optional(bool, false),
    nat_gateway_spec                  = optional(string, "1"),
    nat_gateway_description           = optional(string, null),
    nat_gateway_enterprise_project_id = optional(string, null)
  }))
  default = [{ name = "vpc-default-subnet1", cidr = "10.0.0.0/24", gateway_ip =
↪"10.0.0.1", create_nat_gateway = false, availability_zone = "ru-moscow-1a" }]
}
```

Здесь:

- `subnets` – название переменной.
- `description` – краткое описание переменной.
- `type` – тип переменной. В данном случае – список объектов, каждый из которых содержит следующие поля:
  - `name` – название подсети;
  - `cidr` – диапазон IP-адресов подсети в формате *CIDR*;
  - `availability_zone` – зона доступности, в которой размещается подсеть;
  - `gateway_ip` – IP-адрес шлюза, используемого подсетью;
  - `create_nat_gateway` – использование технологии NAT;
  - `nat_gateway_spec` – тип шлюза, определяющий максимальное количество подключений SNAT;
  - `nat_gateway_description` – описание шлюза;
  - `nat_gateway_enterprise_project_id` – идентификатор проекта на платформе Enterprise.

  Примечание: Поля `gateway_ip`, `create_nat_gateway`, `nat_gateway_spec`, `nat_gateway_description` и `nat_gateway_enterprise_project_id` не обязательны для заполнения.

- `default` – значение переменной `subnets` по умолчанию. Для указанных выше полей используются следующие значения:

Поле	Значение
`name`	`vpc-default-subnet1`
`cidr`	`10.0.0.0/24`
`gateway_ip`	`10.0.0.1`
`create_nat_gateway`	`false`
`availability_zone`	`ru-moscow-1a`

- `versions.tf`

Требования к версиям Terraform и провайдера, например:

```
terraform {
  required_version = ">= 1.3.0"

  required_providers {
    yandex = {
      source  = "yandex-cloud/yandex"
      version = ">= 0.84"
    }
  }
}
```

Здесь:

- `terraform.required_version` – для работы с модулем следует использовать версию Terraform не ниже 1.3.0;
- `terraform.required_providers.yandex` – модуль зависит от провайдера Yandex Cloud;
- `terraform.required_providers.yandex.source` – ссылка для загрузки файлов провайдера Yandex Cloud (указанное здесь значение будет использовано, если источник не задан в настройках проекта);
- `terraform.required_providers.yandex.version` – для работы с модулем следует использовать версию провайдера Yandex Cloud не ниже 0.84.

Применение

Несмотря на то, что Terraform не имеет строгих требований к структуре проекта, описанные в документации Astra Automation проекты Terraform устроены одинаково.

Типовая структура проекта

Проекты, описанные в разделе *Модули Terraform*, имеют одинаковую структуру и могут включать в себя следующие каталоги и файлы:

- `.terraform/modules/`

  Каталог с кодом сторонних модулей. В него загружаются модули из реестра Astra Automation Hub.

- `.terraform/providers/`

  Каталог с файлами провайдеров.

Примечание: Каталог `.terraform/` и его подкаталоги создаются в каталоге проекта автоматически в результате выполнения команды `terraform init`.

- `.terraformrc`

  Настройки, влияющие на поведение Terraform. Как правило, в этом файле указываются альтернативные источники для загрузки файлов провайдеров.

- `provider.tf`

  Список используемых провайдеров и их настройки.

- `variables.tf`

  Декларация переменных, используемых в проекте. Для каждой переменной указываются:

  - обязательные параметры:
    * название;
    * тип;
  - опциональные параметры:
    * описание;
    * правила проверки;
    * значение по умолчанию.

- `terraform.tfvars`

  Присваивание фактических значений переменным, декларированным в файле `variables.tf`.

- `metadata.yml`

  Настройки ВМ для сервисов, поддерживающих технологию *Cloud-Init*.

- `main`

  Здесь размещается описание инфраструктуры. В этом же файле подключаются и используются модули из реестра Astra Automation Hub.

- `deployment` и `deployment.pub`

  Пара ключей SSH, которая используется для подключения к создаваемым ВМ.

  Пошаговая инструкция по созданию ключей приведена в разделе *Ключи SSH для доступа к стендам*.

Использование модулей

Использование модулей Terraform из реестра Astra Automation Hub имеет следующие особенности:

- Необходим доступ к реестру Astra Automation Hub по SSH, настроенный согласно *инструкции*.
- Для подключения нужных модулей используется конструкция следующего вида:

```
module "module_name" {
  source = "git::ssh://git@hub.astra-automation.ru:2222/aa-gca/AMFT/module_name.
↪git"
  name   = "resource_name"
  # ...
}
```

  где

  - module_name – название модуля;
  - resource_name – название ресурса.

Например, для модуля brest-virtual-machine[14] указанная запись должна иметь вид:

```
module "brest-virtual-machine" {
  source               = "git::ssh://git@git.astralinux.ru:7999/amft/brest-
↪virtual-machine.git"
  name                 = "brest-virtual-machine-module-test"
  group                = "brest-mng-group"
  system_disk_image_id = 204
  networks             = [{ id = 1 }]
}
```

Примечание: Для всех модулей из реестра Astra Automation Hub в поле `source` следует указывать ссылку на соответствующий репозиторий.

- Перед первым использованием модули необходимо загрузить в рабочий каталог с помощью команды:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume ~/.ssh/hub.astra-automation.ru:/root/.ssh/id_rsa \
   --volume "$(pwd):/app" \
   registry.astralinux.ru/aa/aa-base-ee \
   bash -c 'terraform init'
```

Подробности об использовании EE (Execution Environment, среда исполнения) см. в разделе *Среда исполнения*.

5.1.2 Коллекции Ansible

Версионирование коллекций

При описании зависимостей Ansible рекомендуется всегда явно указывать номер версии используемой коллекции. Это позволит избежать незапланированных изменений в существующем окружении и несовместимости новой версии коллекции с ним.

Коллекции версионируются согласно правилу семантического версионирования[15]. Версия коллекции выглядит как:

```
<major_version>.<minor_version>.<patch>[-rc.<rc_version>]
```

где:

- `<major_version>` – номер мажорной версии. Версия `0` используется для начальной разработки. Номер мажорной версии увеличивается, если в код коллекции добавляются изменения API.
- `<minor_version>` – номер минорной версии. Увеличивается при добавлении новой функциональности или существенных изменениях в уже существующей функциональности без нарушения совместимости с более ранним API. Минорная версия обнуляется при увеличении мажорной версии.

[14] https://hub.astra-automation.ru/aa-gca/AMFT/brest-virtual-machine
[15] https://semver.org/

- <patch> – номер срочного обновления при исправлении ошибок. Увеличивается после исправления ошибок в имеющейся функциональность без потери обратной совместимости. Новая функциональность не вносится, существенные изменения в уже существующей не производятся. Это поле обнуляется при увеличении мажорной и минорной версий.
- (опционально) -rc.<rc_version> – версия Release Candidate (RC) указывает на то, что это еще не финальная версия, но функциональность почти готова к выпуску. Номер версии RC увеличивается по мере подготовки коллекции к выпуску.

Получение списка версий коллекции

Для получения списка доступных версий коллекции выполните следующие действия:

1. Перейдите на страницу коллекции в соответствующем разделе Astra Automation Hub[16].
2. Нажмите ссылку *Tags* в описании репозитория.

Указание версии коллекции в файле зависимостей

Чтобы зафиксировать номер версии используемой коллекции, добавьте к соответствующей записи в *файле зависимостей* строку с параметром version, например:

```
---
collections:
  - name: astra.ald_pro
    type: git
    source: ssh://git@hub.astra-automation.ru:2222/aa-gca/ARFA/cups.git
    version: "<version>"
```

Если это поле отсутствует, будет загружена самая свежая версия коллекции из доступных. В значении поля version можно задать условия, определяющие допустимый диапазон номеров устанавливаемой версии коллекции:

- * – самая свежая (latest) версия. Это значение используется по умолчанию.
- != – использовать любую версию, кроме указанной.
- == – использовать строго указанную версию.
- >= – использовать указанную версию или более новую.
- > – использовать более новую версию, чем указанная.
- <= – использовать указанную версию или более старую.
- < – использовать версию более старую, чем указанная.

Смотрите более подробную информацию о версионировании коллекций в документации Ansible[17].

[16] https://hub.astra-automation.ru/aa-gca/ARFA
[17] https://docs.ansible.com/ansible/latest/collections_guide/collections_installing.html#installing-an-older-version-of-a-collection

Примеры

- Строгое указание версии

```
version: "==0.1.1"
```

- Указанная или более новая версия

```
version: ">=0.1"
```

Сертифицированные и проверенные коллекции

Форматы распространения кода Ansible

Код для автоматизации с помощью Ansible может включать следующие элементы:

- сценарии;
- определения переменных;
- модули и другие.

Код может иметь произвольную структуру каталогов в проекте. Для обеспечения модульности и переиспользования Astra Automation предлагает контент в виде коллекций Ansible, которые объединяют элементы кода в стандартную структуру. Смотрите более подробную информацию в *описании коллекций Ansible*.

Все коллекции Astra Automation доступны в Astra Automation Hub[18] и являются *проверенными*.

API

Коллекции Ansible используются посредством подключения их к другому коду Ansible (например, к playbook или другим коллекциям) в качестве зависимостей и «библиотек» – подобно тому, как аналогичный подход применяется в высокоуровневых языках программирования. У пользователя коллекции нет необходимости редактировать ее код, чтобы использовать необходимую функциональность в своем окружении.

Пользователь взаимодействует с коллекцией с помощью следующих элементов интерфейса:

- переменные ролей и playbook, объявленные в коллекции;
- инвентарь в формате, определенном коллекцией;
- параметры модулей, содержащихся в коллекции;
- названия ролей, модулей и playbook.

Совокупность перечисленных интерфейсов можно воспринимать как API коллекции, который для каждой коллекции описан в ее документации. API определяется функциональными возможностями коллекции, но также на него может влиять состав *Execution Environment*.

[18] https://hub.astra-automation.ru/aa-gca/ARFA

Проверенный контент

В условиях динамического развития продуктов ПАО Группа Астра, а также продуктов других компаний, потенциальные пользователи Astra Automation хотели бы получать код автоматизации на тестирование как можно раньше, даже если он не полностью охватывает все запланированные возможности и не стабилизирован. К такому коду со стороны Astra Automation предъявляется ограниченный набор требований:

- Код организован в соответствии со структурой коллекции Ansible.
- Коллекция версионируется в соответствии с SemVer (Semantic Versioning).
- Код не требует интерактивного взаимодействия с пользователем.
- Коллекция содержит документацию на все компоненты (роли, модули, общий `README.md`).

После прохождения проверки инженерами Astra Automation коллекция, отвечающая данным требованиям, получает статус проверенной.

Проверенный контент имеет следующие ограничения:

- Отсутствует фиксированный жизненный цикл и выпуски с длительным сроком поддержки.
- Код может быть неидемпотентным[19].

Сертифицированный контент

Коллекции Ansible могут быть сертифицированы в рамках определенной версии Astra Automation. К сертифицированным коллекциям предъявляются следующие требования (помимо требований, предъявляемых к проверенным коллекциям):

- Код не должен дублироваться. Если коллекция collection1 разработана для настройки сервиса service1, которому для функционирования необходим сервис service2, код настройки которого уже есть в составе отдельной сертифицированной коллекции collection2, то запрещается дублировать его в коллекцию collection1.
- Коллекция настраивает только тот объект (сервис, устройство), для которого разрабатывается. Коллекция не должна содержать реализации функций настройки каких-либо других объектов – такие функции должны быть добавлены в соответствующие сертифицированные коллекции.
- Код коллекции должен быть идемпотентным.
- Возможности коллекции должны быть покрыты тестами.

[19] https://en.wikipedia.org/wiki/Idempotence

Жизненный цикл

Для сертифицированных коллекций поддерживаются следующие типы выпусков:

- **FR (feature release)** – выпуски с новыми возможностями коллекции. Периодичность выпуска составляет 4 недели, если есть готовые к публикации изменения.
- **LTS (long-term support)** – выпуски с длительным сроком поддержки в течение 24 месяцев. В таких выпусках зафиксированы функциональность коллекции и версии поддерживаемых продуктов. Выпуски получают только обновления безопасности и устранение ошибок, а также совместимость с новой версией Astra Automation Controller, если подходит к концу срок поддержки текущей версии контроллера.

Порядок выпуска FR

В рамках выпуска FR в сертифицированных коллекциях могут происходить следующие изменения, отражающиеся в соответствующем изменении версии коллекции согласно семантическому версионированию:

- **Исправления** – исправления обнаруженных ошибок, в том числе исправление и дополнение документации.
- **Минорные обновления** – добавление новых функций или новой версии коллекции, не затрагивающие обратную совместимость с текущей мажорной версией API. Это означает, что playbook и другие коллекции, использующие данную коллекцию, не потребуют адаптации для уже задействованных в них возможностей коллекции.
- **Мажорные обновления** – значительные изменения в коллекции, которые меняют формат входных/выходных параметров ролей, playbook и других компонентов коллекции (API). Для таких версий теряется обратная совместимость с предыдущими мажорными версиями коллекции. Изменения могут быть связаны с добавлением поддержки новой версии продукта, исправлениями критических ошибок или удалением компонентов коллекции.

При внесении изменений в служебные файлы (`taskfile.yml`, конфигурации linter и другие) и тесты версия коллекции не меняется.

Версия FR выпускается каждые 4 недели или чаще в случае обнаружения значительных ошибок. Если изменений в коллекции за цикл выпуска не произошло, новая версия не выпускается.

В рамках выпусков FR может добавляться поддержка новых версий продуктов. Для сертифицированных коллекций это должно происходить при очередном выпуске FR после даты релиза продукта или ранее.

Порядок выпуска LTS

Использование долговременно-поддерживаемых версий (LTS, long-term support) позволяет сопровождать инфраструктуру с помощью коллекции с зафиксированным API в течение длительного времени. Новый выпуск LTS формируется каждые 18 месяцев на основе текущего выпуска FR, что позволяет плавно мигрировать с предыдущего LTS-выпуска.

Выпуски LTS поддерживаются в течение 24 месяцев со дня выпуска. В течение этого периода коллекция получает следующие обновления:

- исправления обнаруженных ошибок;

- исправления безопасности;
- новые функции, если это требуется для исправлений.

Мажорная и минорные значения версии коллекции всегда зафиксированы для выпуска LTS. В версии коллекции изменяется только поле patch.

Поддержка

Для сертифицированных и проверенных коллекций клиент может направлять запросы о нововведениях или исправлениях ошибок в поддержку Astra Automation. Запросы будут отрабатываться инженерами Astra Automation и партнерами, сопровождающими сертифицированную коллекцию.

Удаление коллекций

Коллекции могут терять свой статус сертифицированной или проверенной, а также быть удалены с Astra Automation Hub[20].

Коллекция теряет статус сертифицированной в следующих случаях:

- Если сопровождающий коллекцию не обеспечивает устранение ошибок для выпусков LTS.
- Если сопровождающий коллекцию не обеспечил совместимость с новой версией Astra Automation.
- Если закончилась поддержка версии Astra Automation, для которой сертифицировалась коллекция.

Коллекция теряет статус проверенной, когда версия Astra Automation Controller, для которой она предназначена, перестает поддерживаться.

Использование коллекций позволяет значительно упростить создание playbook для решения самых разных задач по развертыванию и настройке ПО с помощью Ansible.

Назначение

Коллекции Ansible, доступные в реестре Astra Automation Hub[21], предназначены для развертывания и настройки продуктов ПАО Группа Астра и вспомогательного стороннего ПО.

От коллекций, имеющихся в свободном доступе, они отличаются следующим:

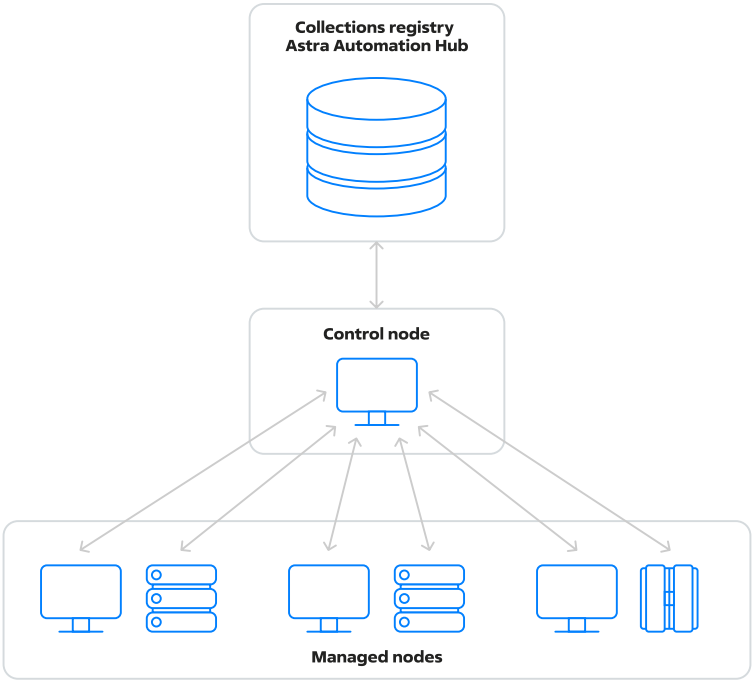
- Все роли распространяются в составе коллекций Ansible, в которые включаются также модули, специфичные для продуктов ПАО Группа Астра.
- При разработке коллекций учитываются особенности продуктов ПАО Группа Астра.
- Длительный срок поддержки.
- Доработка функциональности исходя из потребностей клиентов ПАО Группа Астра.
- Оперативное исправление всех найденных ошибок.
- Многократная проверка кода, в том числе в реальных условиях.

[20] https://hub.astra-automation.ru/explore/groups
[21] https://hub.astra-automation.ru/aa-gca/ARFA

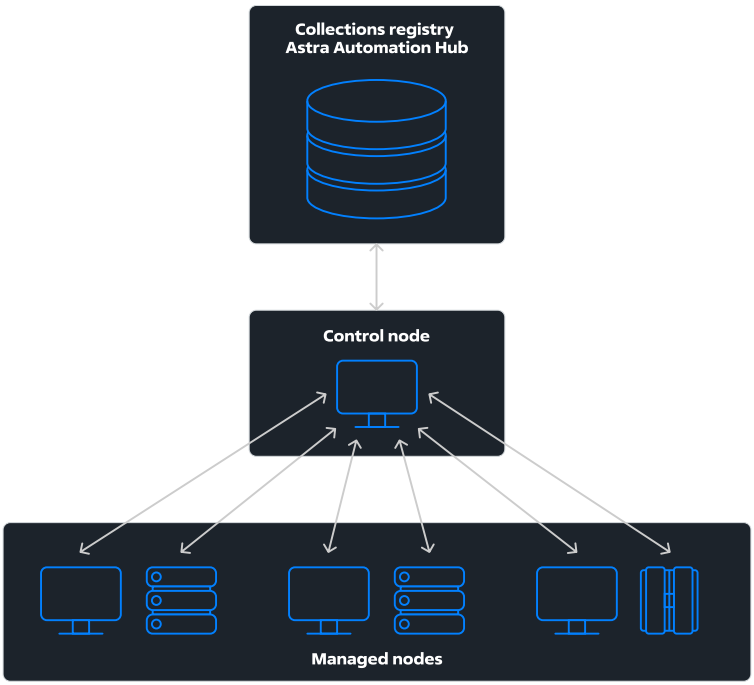
- Единая логика и стандартизированные подходы при написании кода, повышающие качество продукта.
- Названия всех коллекций используют пространство имен `astra`.
- Версии коллекций формируются с использованием семантического версионирования[22] (semantic versioning).
- Каждая коллекция хранится в отдельном репозитории Git.

Структура рабочего окружения

При использовании коллекций из реестра Astra Automation Hub схема работы Ansible имеет вид:



[22] https://semver.org/



Перед выполнением playbook следует установить на управляющий узел все необходимые коллекции. Инструкции по установке коллекций приведены ниже в разделе *Применение*.

Структура коллекции

В составе коллекции могут быть следующие файлы и каталоги:

- `meta/`

  Каталог с файлами, содержащими служебную информацию о коллекции, используемую утилитой `ansible-galaxy`.

- `plugins/`

  Опциональный каталог, содержащий расширения, необходимые для работы коллекции.

- `roles/`

  Каталог с подкаталогами ролей. Подробности см. в разделе *Структура роли*.

- `tests/`

  Каталог с файлами автоматических тестов для коллекции. Файлы этого каталога могут быть полезны при изучении особенностей использования коллекции и подготовке окружения для работы с ней.

- `CHANGELOG.md`

  Файл с историей изменений между версиями.

- `LICENSE`

  Файл лицензии, под которой распространяется код коллекции.

- `README.md`

Описание коллекции, включающее в себя:

- Предназначение коллекции.
- Требования к окружению.
- Список поддерживаемых операционных систем.
- Порядок описания коллекции в файле зависимостей Ansible и способ установки.
- Список ролей.
- Порядок запуска автоматических тестов.
- Тип лицензии, под которой распространяется код коллекции.
- Информация об авторских правах.

- `galaxy.yml`

  Сведения о коллекции, используемые утилитой `ansible-galaxy`, в том числе номер актуальной версии коллекции.

- `requirements_ansible.yml`

  Зависимости Ansible, необходимые для использования коллекции.

Структура роли

Все роли, размещенные в реестре Astra Automation Hub, распространяются только в составе коллекций. Каждая роль используется для установки и настройки только определенной функциональности. Например, роль `astra.ald_pro.client` используется только для настройки клиентов домена ALD Pro. Для настройки контроллера или его реплики следует использовать роли `astra.ald_pro.controller` и `astra.ald_pro.replica` соответственно.

В составе роли могут быть следующие файлы и каталоги:

- `defaults/`

  Каталог с файлами, содержащими значения по умолчанию для переменных роли.

- `files/`

  Каталог с вспомогательными файлами, используемыми ролью, например, шаблоны HTML-страниц, изображения и т. д. При выполнении роли файлы из этого каталога копируются на управляемые узлы.

- `handlers/`

  Каталог с файлами обработчиков, выполняемых при использовании роли.

- `meta/`

  Каталог с файлами, содержащими служебную информацию о роли, используемую утилитой `ansible-galaxy`.

- `tasks/`

  Каталог с файлами, выполняющимися при использовании роли.

- `templates/`

  Каталог с шаблонами формата Jinja 2.

- `vars/`

  Каталог с файлами, содержащими список переменных роли. Каждая роль содержит список переменных, позволяющих управлять настройками соответствующего ПО.

- `LICENSE`

  Файл лицензии, под которой распространяется код роли.

- `README.md`

  Описание роли, включающее в себя:

  - Предназначение роли.
  - Требования к окружению.
  - Список переменных роли.

    Переменные роли делятся на обязательные и опциональные. Для опциональных переменных, значения которых не заданы, будут использованы значения по умолчанию.

    > **Предупреждение:** Значения обязательных переменных должны быть явно заданы в playbook или используемом им файле с переменными.

  - Примеры использования.
  - Порядок запуска автоматических тестов.
  - Тип лицензии, под которой распространяется код роли.
  - Информация об авторских правах.
  - Список поддерживаемых операционных систем.

Настройки Ansible

Во время работы Ansible использует настройки, переданные в аргументах командной строки или указанные в конфигурационном файле `ansible.cfg`. Поиск файла настроек `ansible.cfg` выполняется в следующем порядке:

1. Каталог, указанный в значении переменной окружения `ANSIBLE_CONFIG`.
2. Текущий каталог.
3. Домашний каталог активного пользователя.
4. Каталог `/etc/ansible/`.

Поиск прекращается как только файл `ansible.cfg` будет найден в любом из указанных расположений.

> **Предупреждение:** При использовании для работы с Ansible *среды исполнения* файл `ansible.cfg` следует размещать в каталоге проекта.

Подробности о файле настроек см. в документации Ansible[23].

[23] https://docs.ansible.com/ansible/latest/reference_appendices/config.html

Применение

Использование коллекций Ansible из реестра Astra Automation Hub имеет следующие особенности:

- Необходим доступ к реестру Astra Automation Hub по SSH, настроенный согласно инструкций из раздела *Доступ к реестрам Astra Automation Hub*.
- Для описания зависимостей создайте в каталоге проекта файл `requirements.yml` с записями следующего вида:

```
---
collections:
  - name: astra.<collection_name>
    type: git
    source: ssh://git@hub.astra-automation.ru:2222/aa-gca/ARFA/<collection_name>.
↪git
# ...
```

  где <collection_name> – название коллекции. Например, для коллекции astra.ceph[24] указанная запись должна иметь вид:

```
---
collections:
  - name: astra.ceph
    type: git
    source: ssh://git@hub.astra-automation.ru:2222/aa-gca/ARFA/ceph.git
```

Примечание: Для всех коллекций из реестра Astra Automation Hub в поле `type` следует указывать значение `git`, а в поле `source` – ссылку на соответствующий репозиторий.

- При использовании роли в playbook указывайте *FQCN*, например:

```
---
- name: Set up ALD Pro domain controller
  hosts: dc01
  # ...
  roles:
    - role: astra.ald_pro.controller
      vars:
        aldpro_domain: aldpro.example.com
        aldpro_pdc_ip: 192.168.56.11
        aldpro_pdc_name: dc01
        aldpro_admin_password: p@ssW0rD!
```

- При использовании EE используйте следующие команды:

  1. Установка зависимостей:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume ~/.ssh/hub.astra-automation.ru:/root/.ssh/id_rsa \
```

(continues on next page)

[24] https://hub.astra-automation.ru/aa-gca/ARFA/ceph

(продолжение с предыдущей страницы)

```
   --volume "$(pwd):/app" \
   registry.astralinux.ru/aa/aa-base-ee \
   bash -c 'ansible-galaxy install -r requirements.yml'
```

2. Запуск playbook:

```
docker run \
   --rm \
   --tty \
   --volume "$(pwd):/app/" \
   registry.astralinux.ru/aa/aa-base-ee \
   bash -c 'ansible-playbook playbook.yml'
```

Подробности об использовании EE см. в разделе *Среда исполнения*.

5.1.3 Библиотека инфраструктурного кода

Важно: Эта часть документации находится в стадии разработки.

Библиотека состоит из репозиториев с кодом (в парадигме *IAC*) для развертывания различных комплексных систем, некоторые из которых приведены в следующем списке:

- ALD Pro – система управления доменами;
- RuPost – система управления электронной почтой.

5.2 Применение Astra Automation Hub

Инструкции этого раздела знакомят пользователя с основными приемами работы с контентом, представленным в реестре инфраструктурного кода, без использования контроллера:

- настройка рабочего окружения;
- использование образов виртуальных машин;
- развертывание облачной инфраструктуры, используя модули Terraform;
- развертывание комплексной инфраструктуры, используя коллекции Ansible;
- развертывание комплексной инфраструктуры, используя генератор кода.

5.2.1 Общая подготовка системы

Подготовьте вашу рабочую машину и рабочее окружение, где вы будете развертывать инфраструктуру, к выполнению сценариев. Все сценарии разбиты на группы, каждая из которых требует специальной подготовки.

Здесь приведены инструкции по настройкам, общим для всех сценариев.

Доступ к реестрам Astra Automation Hub

Для доступа к реестрам Astra Automation Hub следует использовать протокол SSH. Чтобы настроить его, выполните следующие действия:

1. Убедитесь, что у вас есть пара ключей SSH:

   ```
   ls ~/.ssh/
   ```

   Пример вывода на экран:

   ```
   id_ed25519  id_ed25519.pub  known_hosts
   ```

   Если ключей нет, для их создания используйте команду `ssh-keygen`, например:

   ```
   ssh-keygen -C "<comment>" -f ~/.ssh/hub.astra-automation.ru -N "<password>"
   ```

   где

   - `<comment>` – произвольный комментарий, позволяющий отличать один ключ SSH от другого.
   - `~/.ssh/hub.astra-automation.ru` – путь к файлу, в который следует сохранить пару ключей SSH. При этом приватный ключ сохраняется в файл с указанным именем, а к имени файла для публичного ключа добавляется расширение `.pub`. Например, в данном случае публичный ключ сохраняется в файл `~/.ssh/hub.astra-automation.ru.pub`.
   - `<password>` – опциональный пароль для защиты приватного ключа.

   Примечание: Подробные инструкции по использованию утилиты `ssh-keygen` доступны во встроенной справке:

   ```
   man ssh-keygen
   ```

   ---

2. Скопируйте содержимое файла с публичным ключом (`~/.ssh/hub.astra-automation.ru.pub` для примера выше) в буфер обмена.
3. В персональных настройках[25] Astra Automation Hub выполните следующие действия:
   1. В меню *User Settings* выберите пункт **SSH Keys**.
   2. В поле **Key** вставьте публичный ключ из буфера обмена.
   3. (Опционально) В поле **Title** введите название ключа.
   4. Нажмите кнопку *Add key*.

[25] https://hub.astra-automation.ru/-/profile/preferences

Ключи SSH для доступа к стендам

Рекомендуемым способом доступа к *управляемым узлам*, из которых состоят настраиваемые с помощью Astra Automation стенды, является подключение по протоколу SSH. Для каждого стенда, описанного в приведенных далее руководствах, рекомендуется создавать отдельную пару ключей SSH.

Для создания такой пары ключей используйте команду `ssh-keygen` со следующими параметрами:

```
ssh-keygen -C "<comment>" -f ~/.ssh/<file_name> -N ""
```

Например:

```
ssh-keygen -C "Deployment key" -f ~/.ssh/deployment -N ""
```

Примечание: Далее ключи `~/.ssh/deployment` и `~/.ssh/deployment.pub` используются для создания стендов и подключения к ним.

Использование sudo без ввода пароля

Для настройки управляемых узлов с помощью Ansible необходимо разрешить на них использование команды `sudo` без ввода пароля.

Чтобы выключить запрос пароля при использовании команды `sudo`, выполните команду:

```
sudo astra-sudo-control disable
```

Примечание: В образах, указанных в разделе *Образы виртуальных машин*, использование `sudo` без ввода пароля включено по умолчанию.

Служебные программы

Для управления проектом необходимы следующие вспомогательные программы на вашей рабочей машине:

- Git – для работы с репозиториями Git.
- Podman (рекомендуется) или Docker – для создания вспомогательного контейнера, в котором запускаются Ansible, Terraform и другие утилиты.

  Инструкции по установке приведены в соответствующих разделах:

  - *Podman*;
  - *Docker*.

  Подробности об использовании среды исполнения см. в разделе *Среда исполнения*.
- Task – для запуска задач, описанных в файлах `Taskfile.yml`.

  Подробности см. в разделе *Task*.

Установка Git

Для установки Git выполните команду:

```
sudo apt update && sudo apt install git --yes
```

5.2.2 Модули Terraform

Рассмотрим применение модулей Terraform из реестра Astra Automation Hub для развертывания базовой инфраструктуры в Yandex Cloud.

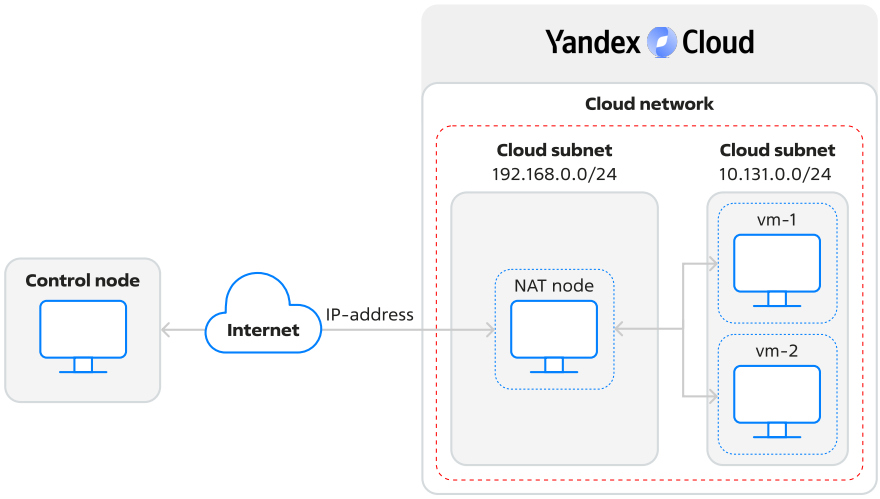
Описание сценария

Рассматриваемый сценарий приводит к созданию инфраструктуры из следующих компонентов:

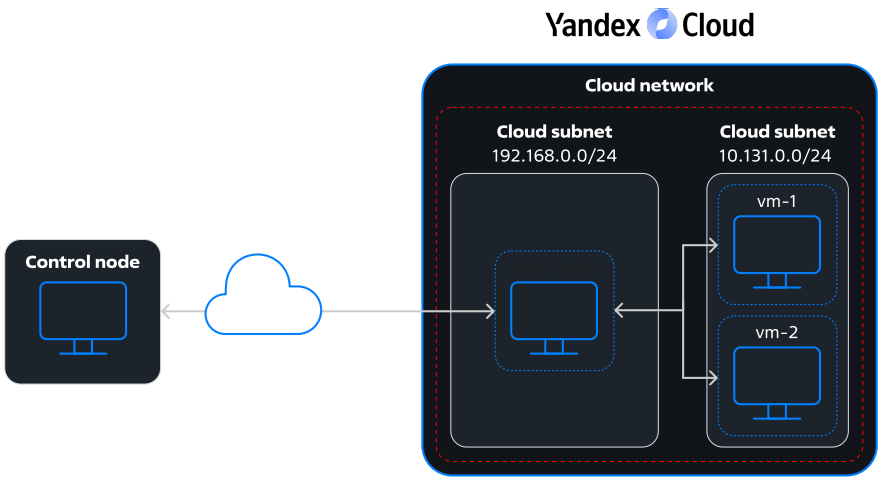
- облачную сеть;
- две ВМ под управлением Astra Linux Special Edition;
- узел NAT.

  В данном примере для создания узла NAT используется образ NAT-инстанс[26] из Yandex Cloud Marketplace.

Схема развертываемой инфраструктуры показана на рисунке:



[26] https://yandex.cloud/ru/marketplace/products/yc/nat-instance-ubuntu-18-04-lts



В этом сценарии выполните следующие операции:

- Создайте одну облачную сеть с двумя облачными подсетями, размещенными в одной зоне доступности.

  Так как обе подсети принадлежат одной сети, трафик между ними может передаваться без использования публичных IP-адресов. Подробнее об устройстве облачных сетей и подсетей см. в документации Yandex Cloud[27].

- В одной облачной подсети разместите ВМ `vm-1` и `vm-2`, а в другой – узел NAT.

  Примечание: Размещение узла NAT в отдельной подсети является требованием Yandex Cloud.

- Настройте ВМ с помощью *Cloud-Init*, чтобы к ним можно было подключаться по SSH.

Процесс подготовки инфраструктуры состоит из следующих этапов:

1. Создание конфигурационных файлов Terraform.

   Назначение файлов:

   - `.terraformrc` – настройки, позволяющие использовать копию (зеркало) репозитория Terraform, предоставляемую компанией Яндекс.
   - `main.tf` – описание создаваемых ресурсов;
   - `metadata.yml` – настройки ОС ВМ в формате Cloud-Init;
   - `provider.tf` – подключение и настройка провайдера Terraform для Yandex Cloud;
   - `sa_key.json` – пара авторизованных ключей сервисного аккаунта;
   - `variables.tf` – описания переменных;
   - `terraform.tfvars` – значения переменных.

   Подробности о типовой структуре проекта Terraform, используемой в документации Astra Automation, см. в разделе *Модули Terraform*.

2. Создание необходимых ресурсов в Yandex Cloud с помощью Terraform.

[27] https://yandex.cloud/ru/docs/vpc/concepts/network

Для запуска Terraform используется контейнер, описанный в разделе *Среда исполнения*.

Сетевые ресурсы:

- Сеть `example-network`, состоящая из двух подсетей:
  - `example-nat-subnet` – подсеть `192.168.0.0/24` для размещения узла NAT;
  - `example-ru-central1-a-0` – подсеть `10.131.0.0/24` для размещения ВМ.

  Обе подсети должны находиться в одной зоне доступности `ru-central1-a`.
- Таблица маршрутизации, обеспечивающая прохождение трафика из интернета в облачные подсети.
- Служебные DNS-зоны, необходимые для корректной работы облачной сети и маршрутизации.

Виртуальные машины:

Название ВМ	Количество vCPU	Доля vCPU	Объем RAM, ГБ	Размер диска, ГБ	ОС
`example-nat-ir`	2	100 %	2	12	Ubuntu 18.04 LTS
`vm-1`	2	50 %	4	30	Astra Linux Special Edition 1.7.3
`vm-2`	2	50 %	4	30	Astra Linux Special Edition 1.7.3

На каждой ВМ создается учетная запись `administrator`, от имени которой к ВМ можно будет подключиться по SSH.

Подготовка к работе

Подготовьте рабочее окружение:

1. Изучите описание модулей Terraform из реестра Astra Automation Hub:
   - yandex-cloud-compute-instance[28];
   - yandex-cloud-vpc[29].
2. Подготовьте управляющий узел к работе с реестром Astra Automation Hub и стендом согласно *инструкции*.
3. Создайте каталог для хранения файлов проекта, например:

   ```
   mkdir aa-tf-yc/
   ```

   Примечание: Далее этот каталог будет называться каталогом проекта. Все упомянутые ниже файлы следует создавать в этом каталоге, если явно не указано иное.

4. Согласно *инструкции* создайте пару ключей SSH, которая будет использоваться для подключения к ВМ. Сохраните их в каталоге проекта под именами `deployment` и `deployment.pub` соответственно.

[28] https://hub.astra-automation.ru/aa-gca/AMFT/yandex-cloud-compute-instance
[29] https://hub.astra-automation.ru/aa-gca/AMFT/yandex-cloud-vpc

5. Получите идентификаторы образов Astra Linux Special Edition, доступных в Yandex Cloud Marketplace[30]:

```
yc compute image list --folder-id standard-images | grep astra
```

   В терминал выводится таблица следующего вида:

```
| fd81pgsokk5vtjm1qgie | astralinux-alse-v20221221 | astralinux-alse |␣
↪f2eni8mvsjeqmflq62qq | READY  |
| fd87qct47l4isk56gucq | astralinux-alse-v20230215 | astralinux-alse |␣
↪f2esbkvqjcg1kk3i29ra | READY  |
```

   Идентификаторы образов указаны в первой колонке.

6. Получите идентификатор используемого облака:

```
yc resource-manager cloud list
```

   В терминал выводится таблица следующего вида:

```
+----------------------+------------+----------------------+
|          ID          |    NAME    |   ORGANIZATION ID    |
+----------------------+------------+----------------------+
| b1.................f4 | rbta-cloud | bp.................f4 |
+----------------------+------------+----------------------+
```

   Идентификатор облака указан в колонке `ID`.

7. Получите идентификатор используемого каталога:

```
yc resource-manager folder list
```

   В терминал выводится таблица следующего вида:

```
+----------------------+--------------------------+--------+--------+
|          ID          |           NAME           | LABELS | STATUS |
+----------------------+--------------------------+--------+--------+
| b1.................bv | td.....................it |        | ACTIVE |
+----------------------+--------------------------+--------+--------+
```

   Идентификатор каталога указан в колонке `ID`.

8. В каталоге проекта разместите файл `sa_key.json`, содержащий пару авторизованных ключей сервисного аккаунта.

   Если у вас нет пары авторизованных ключей сервисного аккаунта, для их создания выполните инструкции из раздела *Создание пары авторизованных ключей доступа*.

9. Создайте в каталоге проекта файл `metadata.yml` с описанием настроек ОС на создаваемых ВМ.

```
#cloud-config
---
users:
  - name: administrator
    groups: sudo
    shell: /bin/bash
```

(continues on next page)

[30] https://yandex.cloud/ru/marketplace

(продолжение с предыдущей страницы)

```
    sudo: ["ALL=(ALL) NOPASSWD:ALL"]
    ssh-authorized-keys:
      - INSERT_PUBLIC_KEY_HERE
```

Здесь:

Параметр	Значение	Описание
name	admini	Имя учетной записи
group	sudo	Список дополнительных групп, в которые должна быть включена созданная учетная запись
shell	/bin/bash	Оболочка пользователя по умолчанию
sudo	["ALL=(NOPASSW	Разрешение всем пользователям на выполнение с sudo любых команд без ввода паролям
ssh-a	—	Список публичных ключей SSH, используемых для подключения к ВМ. Укажите в значении этого поля содержимое файла deployment.pub. Указанный публичный ключ будет автоматически добавлен в файл /home/administrator/.ssh/authorized_keys, что обеспечит возможность подключения к ВМ по SSH.

Подготовка файлов Terraform

Для подготовки файлов Terraform выполните следующие действия:

1. Создайте файл variables.tf со следующим содержимым:

```
variable "service_account_key_file" {
  description = "Service account key file"
  type        = string
  default     = "/app/sa_key.json"
}

variable "cloud_id" {
  description = "Cloud ID"
  type        = string
}

variable "folder_id" {
  description = "ID of the folder"
  type        = string
}

variable "image_id" {
  description = "Image ID"
  type        = string
}

variable "name" {
  description = "Name to be used on all the resources as identifier"
  type        = string
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  default     = "yc-tf"
  validation {
    condition     = length(var.name) <= 32
    error_message = "Must be a 32 or less character long string."
  }
}

variable "subnets" {
  description = "Subnets in folder"
  type = list(object({
    name           = optional(string)
    zone           = optional(string, "ru-central1-a")
    v4_cidr_blocks = list(string)
  }))
}
```

В этом файле находятся описания переменных, которые используются в плане инфраструктуры.

Предупреждение: Не изменяйте файл `variables.tf`! Для присвоения переменным фактических значений укажите их в файле `terraform.tfvars`.

Список переменных в файле `variables.tf`:

Переменная	Тип	Значение по умолчанию	Описание
service_a	stri	/app/sa_key.json	Путь к файлу с парой авторизованных ключей сервисного аккаунта. Поскольку используется контейнер EE, путь указывается относительно каталога /app внутри него
cloud_id	stri	—	Идентификатор облака Yandex Cloud
folder_id	stri	—	Идентификатор облачного каталога
image_id	stri	—	Идентификатор образа, используемого при создании ВМ
name	stri	yc-tf	Префикс, добавляемый к именам создаваемых ресурсов. Требования – длина не более 32 знаков
subnets	Запись	—	Параметры создаваемых подсетей

Поля записи, используемой для переменной `subnets`:

Имя поля	Тип	Описание
name	string	Имя подсети, опциональный параметр
zone	string	Зона доступности
v4_cidr_blocks	string	Строка, задающая диапазон IP-адресов подсети.

Подробное описание переменных см. в документации Terraform[31].

[31] https://developer.hashicorp.com/terraform/language/values/variables

2. Создайте файл `terraform.tfvars` со следующим содержимым:

```
cloud_id  = "" # Set your cloud ID
folder_id = "" # Set your folder ID
image_id  = "" # Set Astra Linux Image ID
subnets = [
  {
    zone           = "ru-central1-a"
    v4_cidr_blocks = ["10.131.0.0/24"]
  }
]
```

3. Создайте файл `.terraformrc` со следующим содержимым:

```
provider_installation {
  network_mirror {
    url     = "https://terraform-mirror.yandexcloud.net/"
    include = ["registry.terraform.io/*/*"]
  }
  direct {
    exclude = ["registry.terraform.io/*/*"]
  }
}
```

Здесь указаны настройки Terraform, позволяющие использовать для загрузки провайдера Yandex Cloud реестр компании Яндекс вместо реестра компании HashiCorp.

4. Создайте файл `provider.tf` со следующим содержимым:

```
terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
    }
  }
  required_version = ">= 0.13"
}

provider "yandex" {
  service_account_key_file = pathexpand(var.service_account_key_file)
  cloud_id                 = var.cloud_id
  folder_id                = var.folder_id
}
```

Этот файл содержит описание провайдера Terraform, необходимого для работы с Yandex Cloud. Идентификаторы облака и облачного каталога, а также путь к файлу с авторизованным ключом сервисного аккаунта указаны в переменных `cloud_id`, `folder_id` и `service_account_key_file` соответственно.

5. Создайте файл `main.tf` со следующим содержимым:

```
module "vpc" {
  source              = "git::ssh://git@hub.astra-automation.ru:2222/aa-gca/AMFT/
↪yandex-cloud-vpc.git"
  name                = "example"
  folder_id           = var.folder_id
  create_nat_instance = true
  subnets             = var.subnets
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  metadata            = file("./metadata.yml")
}

module "vm" {
  count = 2

  source                  = "git::ssh://git@hub.astra-automation.ru:2222/aa-gca/
↪AMFT/yandex-cloud-compute-instance.git"
  folder_id               = var.folder_id
  name                    = "vm-${count.index + 1}"
  image_id                = var.image_id
  resources_cpu           = 2
  resources_core_fraction = 50
  resources_ram           = 4
  boot_disk_size          = 30
  boot_disk_type          = "network-ssd"
  nat                     = false
  nat_ip_address          = null
  description             = "Created with Terraform"
  labels                  = {}
  metadata                = file("./metadata.yml")
  network_interfaces = [{
    subnet_id = module.vpc.subnet_ids[0]
  }]
}
```

Этот файл содержит описание инфраструктуры, создаваемой в Yandex Cloud с использованием модулей `yandex-cloud-vpc` и `yandex-cloud-compute-instance` из реестра Astra Automation Hub.

Первым идет описание модуля vpc, задающего настройки создаваемых сети, подсетей и узла NAT:

```
module "vpc" {
  source              = "git::ssh://git@hub.astra-automation.ru:2222/aa-gca/AMFT/
↪yandex-cloud-vpc.git"
  name                = "vpc"
  folder_id           = var.folder_id
  create_nat_instance = true
  subnets             = var.subnets
  metadata            = file("./metadata.yml")
}
```

Здесь:

- `source` – источник, используемый для загрузки файлов модуля. В данном случае указана ссылка для загрузки из Git-репозитория Astra Automation Hub с использованием протокола SSH.
- `name` – префикс, используемый при создании ресурсов с помощью этого модуля.
- `folder_id` – идентификатор облачного каталога. Используется значение переменной `folder_id`.
- `create_nat_instance` – создание узла NAT.

  В данном случае указано значение `true`, а значит, будут автоматически созданы узел NAT и необходимые для его работы ресурсы:

  - отдельная подсеть;
  - таблица маршрутизации, перенаправляющая трафик из `vm-1` и `vm-2` через узел NAT в интернет.
- `subnets` – параметры создаваемых облачных подсетей.

  Используется значение переменной `subnets`.
- `metadata` – при создании ВМ для узла NAT к ней будут применены настройки, указанные в файле `metadata.yml`. Это необходимо для того, чтобы к узлу NAT можно было подключаться из интернета, используя ключи SSH.

> **Предупреждение:** Настройки из этого файла будут применены только один раз – при первом запуске ВМ. Если файл содержит ошибки, и ВМ создана не с теми настройками, которые необходимы, удалите ее и создайте заново.

В этом блоке идет описание модуля `vm`, задающего настройки ВМ `vm-1` и `vm-2`:

```
module "vm" {
  count = 2

  source                    = "git::ssh://git@hub.astra-automation.ru:2222/aa-gca/
↪AMFT/yandex-cloud-compute-instance.git"
  folder_id                 = var.folder_id
  name                      = "vm-${count.index + 1}"
  image_id                  = var.image_id
  resources_cpu             = 2
  resources_core_fraction = 50
  resources_ram             = 4
  boot_disk_size            = 30
  boot_disk_type            = "network-ssd"
  nat                       = false
  nat_ip_address            = null
  description               = "Created with Terraform"
  labels                    = {}
  metadata                  = file("./metadata.yml")
  network_interfaces = [{
    subnet_id = module.vpc.subnet_ids[0]
  }]
}
```

Здесь:

- `count` – мета-аргумент, задающий количество создаваемых ресурсов данного типа.
- `source` – источник, используемый для загрузки файлов модуля. В данном случае указана ссылка для загрузки из Git-репозитория Astra Automation Hub с использованием протокола SSH.
- `folder_id` – идентификатор облачного каталога. Используется значение переменной `folder_id`.
- `name` – название ВМ. Так как значения мета-аргумента `count` начинаются с 0, для создания ВМ с именами `vm-1` и `vm-2` к значению `count` нужно прибавить единицу.

- `image_id` – идентификатор образа, используемого для создания ВМ. Используется значение, указанное в переменной `image_id`.
- `resources_cpu` – количество ядер vCPU.
- `resources_core_fraction` – доля доступности vCPU, в процентах.
- `resources_ram` – количество RAM, ГБ.
- `boot_disk_size` – размер загрузочного диска ВМ, ГБ.
- `boot_disk_type` – тип загрузочного диска ВМ.

  Подробности о типах дисков см. в документации Yandex Cloud[32].
- `nat` – использование NAT для предоставления ВМ доступа в интернет.
- `nat_ip_address` – IP-адрес для NAT-шлюза. Поскольку обе ВМ используют для доступа к интернету узел NAT, для этого параметра задано значение `null`.
- `description` – описание ВМ.
- `labels` – метки ресурса. В данном случае обеим ВМ не присваивается никаких меток.
- `metadata` – путь к файлу с настройками ОС ВМ в формате Cloud-Init. Указан путь к созданному ранее файлу `metadata.yml`.
- `network_interfaces` – параметры сетевых интерфейсов создаваемой ВМ. Каждая ВМ создается с одним сетевым интерфейсом в первой подсети, созданной модулем `vpc`.

Развертывание инфраструктуры

Для выполнения описанных далее шагов используется *Execution Environment*.

1. Выполните команду инициализации проекта Terraform:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume ~/.ssh/hub.astra-linux.ru:/root/.ssh/id_rsa \
   --volume "$(pwd)/.terraformrc:/root/.terraformrc" \
   --volume "$(pwd):/app/" \
   registry.astralinux.ru/aa-base-ee:latest \
   bash -c "terraform init"
```

   Здесь:

   - `~/.ssh/hub.astra-linux.ru` – путь к файлу приватного ключа SSH, используемого для доступа к Astra Automation Hub.
   - `$(pwd)` – команда для определения абсолютного пути к каталогу проекта.

   В терминал выводится предупреждение о подключении к узлу `hub.astra-automation.ru`.

[32] https://yandex.cloud/ru/docs/compute/concepts/disk#disks-types

Пример сообщения

```
Initializing modules...
Downloading git::ssh://git@hub.astra-automation.ru:2222/aa-gca/AMFT/yandex-cloud-
↪compute-instance.git for vm...
The authenticity of host '[hub.astra-automation.ru]:2222 ([84.201.175.30]:2222)'␣
↪can't be established.
ECDSA key fingerprint is SHA256:bvbETX8oC4ZwFtiE30SCN6yOQxRyOjgGO+lq2U2Dy00.
Are you sure you want to continue connecting (yes/no)?
```

Введите значение yes и нажмите Enter.

Если приватный ключ для доступа к реестру Astra Automation Hub защищен паролем, будет выведен запрос на его ввод:

```
Enter passphrase for key '/root/.ssh/id_rsa':
```

Введите пароль и нажмите Enter.

2. Дождитесь загрузки модулей и файлов провайдера.

   В процессе загрузки в каталоге проекта создается подкаталог `.terraform/`, а в терминал выводится сообщение.

Пример сообщения

```
- vm in .terraform/modules/vm
- vpc in .terraform/modules/vpc

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of yandex-cloud/yandex from the dependency lock file
- Installing yandex-cloud/yandex v0.97.0...
- Installed yandex-cloud/yandex v0.97.0 (unauthenticated)

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

3. Для проверки конфигурационных файлов Terraform выполните команду:

```
docker run \
   --rm \
   --tty \
   --volume "$(pwd):/app/" \
   registry.astralinux.ru/aa-base-ee:latest \
   bash -c "terraform validate"
```

Если в конфигурации нет ошибок, выводится сообщение:

```
Success! The configuration is valid.
```

При наличии ошибок в терминал выводится сообщение с описанием первой из них и указанием на место ее расположения.

4. Для создания ресурсов в Yandex Cloud выполните команду:

```
docker run \
   --rm \
   --tty \
   --volume ~/.ssh/hub.astra-linux.ru:/root/.ssh/id_rsa \
   --volume "$(pwd)/.terraformrc:/root/.terraformrc" \
   --volume "$(pwd):/app/" \
   registry.astralinux.ru/aa-base-ee:latest \
   bash -c "terraform apply -auto-approve"
```

Примечание: Дополнительный параметр `-auto-approve` избавляет от необходимости подтверждать выполняемые действия.

Дождитесь завершения создания ресурсов. О завершении создания ресурсов свидетельствует строка вида:

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
```

5. Для завершения работы контейнера EE выполните команду:

```
exit
```

Проверка работоспособности развернутой инфраструктуры

Чтобы проверить корректность развертывания инфраструктуры, выполните следующие действия:

1. Получите список ВМ в своем каталоге:

```
yc compute instance list
```

В терминал будет выведена таблица вида:

```
+----------------------+----------------------+--------------+---------+---------
↪-------+-------------+
|          ID          |         NAME         |    ZONE ID    | STATUS  | ␣
↪EXTERNAL IP   | INTERNAL IP  |
+----------------------+----------------------+--------------+---------+---------
↪-------+-------------+
| fhmaruuj14ljbe8duobg | example-nat-instance | ru-central1-a | RUNNING | 158.160.
↪38.156 | 192.168.0.3  |
| fhmoj4t896ei2ti10o7e | vm-1                 | ru-central1-a | RUNNING |         ␣
↪       | 10.131.0.101 |
| fhma2ollfugoob6j9m2a | vm-2                 | ru-central1-a | RUNNING |         ␣
↪       | 10.131.0.102 |
+----------------------+----------------------+--------------+---------+---------
↪-------+-------------+
```

где:

- `ID` – идентификатор ВМ.
- `NAME` – название ВМ.
- `ZONE ID` – идентификатор зоны доступности.
- `STATUS` – состояние ВМ. Значение `RUNNING` указывает на то, что ВМ запущена и работает.
- `EXTERNAL IP` – публичный IP-адрес ВМ. Присвоен только узлу NAT.
- `INTERNAL IP` – IP-адрес ВМ из диапазона соответствующей подсети.

2. Чтобы подключиться к ВМ `vm-1` или `vm-2` выполните команду:

```
ssh -J <nat_user>@<nat_IP> <vm_user>@<vm_IP>
```

Где:

- `-J` – аргумент, указывающий на то, что первый узел будет использоваться в качестве промежуточного (Jump host);
- <nat_user> – имя пользователя узла NAT;
- <nat_IP> – публичный IP-адрес узла NAT;
- <vm_user> – имя пользователя ВМ;
- <vm_IP> – внутренний IP-адрес ВМ.

Например, чтобы подключиться к `vm-1` из таблицы выше, выполните команду:

```
ssh -J administrator@158.160.38.156 administrator@10.131.0.101
```

При первом подключении к каждой ВМ выводится сообщение о неизвестном отпечатке ключа.

Подтверждение подключения по SSH

```
The authenticity of host '158.160.47.217 (158.160.47.217)' can't be established.
ED25519 key fingerprint is SHA256:AJiqY0YTT9c60qsiSJX3bbLRXWAlto//aat94tB9gg4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Введите `yes` и нажмите `Enter`.

При успешном подключении к нужной ВМ меняется приглашение командной строки:

```
administrator@vm-1:~$
```

3. Чтобы убедиться, что ВМ работает под управлением ОС Astra Linux Special Edition, выполните команду:

```
cat /etc/astra_version
```

Она выводит в терминал номер версии ОС.

4. Для отключения от ВМ выполните команду:

```
exit
```

Освобождение ресурсов

Если созданные ресурсы больше не нужны, для их удаления выполните команду:

```
docker run \
   --rm \
   --volume "$(pwd)/.terraformrc:/root/.terraformrc" \
   --volume "$(pwd):/app/" \
   registry.astralinux.ru/aa-base-ee:latest \
   bash -c "terraform destroy -auto-approve"
```

Особенности проекта

Обратите внимание на следующие особенности проекта:

- Использование конфигурационного файла `.terraformrc` для загрузки провайдера Terraform для Yandex Cloud с использованием ресурсов компании Яндекс.
- Порядок подключения и использования модулей из реестра Astra Automation Hub.
- Параметры, передаваемые в EE при запуске Terraform.
- Настройка ОС создаваемых ВМ с помощью конфигурационного файла `metadata.yml`.
- Автоматическое создание подсети и таблицы маршрутизации, необходимых для корректной работы узла NAT.

Заключение

В этом сценарии вы познакомились с основными шагами по использованию модулей Terraform из реестра Astra Automation Hub. Из всей последовательности шагов важно выделить следующие действия:

- Подготовка окружения.
- Описание и использование модулей в конфигурационных файлах Terraform.
- Использование файла с переменными.
- Запуск Terraform с помощью EE.
- Проверка корректности развертывания инфраструктуры.

5.2.3 Коллекции Ansible

В данном разделе приведена серия сценариев для развертывания систем различной сложности, используя реестр коллекций Ansible.

Веб-сервер NGINX

Развертывание веб-сервера NGINX с использованием коллекции astra.nginx[33] из реестра Astra Automation Hub позволяет автоматизировать процесс установки и настройки на необходимом количестве узлов.

Описание сценария

Роль `astra.nginx.nginx` из реестра коллекций Astra Automation Hub позволяет за несколько минут в автоматическом режиме развернуть веб-сервер NGINX.

Процесс установки и настройки веб-сервера NGINX состоит из следующих этапов:

1. Создание стенда.

   В качестве стенда в этом руководстве используется ВМ VirtualBox, созданная с помощью *Vagrant*.

   Она обладает следующими характеристиками:

   - количество ядер CPU – 2;
   - объем RAM, ГБ – 2;
   - объем хранилища, ГБ – 30;
   - операционная система – Astra Linux Special Edition 1.7.4 без графического интерфейса;
   - уровень защищенности ОС – максимальный («Смоленск»).

2. Подготовка файлов Ansible:

   - `requirements.yml` – зависимости;
   - `ansible.cfg` – настройки Ansible;
   - `inventory` – инвентарь;
   - `playbook.yml` – playbook, содержащий сценарии настройки управляемых узлов.

3. Настройка стенда с помощью *Ansible*.

   Для запуска Ansible используется образ, описанный в разделе *Среда исполнения*.

Настройки разворачиваемого веб-сервера NGINX отличаются от значений по умолчанию:

- `worker_connections` – 2048;
- `keepalive_timeout` – 30.

Для подключения к стенду используется протокол SSH.

[33] https://hub.astra-automation.ru/aa-gca/ARFA/nginx/

Подготовка к работе

Подготовьте рабочее окружение:

1. Изучите описание коллекции astra.nginx[34].
2. Подготовьте управляющий узел к работе с реестром Astra Automation Hub и стендом согласно *инструкции*.
3. Установите VirtualBox, Vagrant и его расширение для работы с образами Astra Linux согласно *инструкции*.
4. Создайте каталог для хранения файлов проекта, например:

```
mkdir ~/nginx/
```

Примечание: Далее этот каталог будет называться каталогом проекта. Все упомянутые ниже файлы следует создавать в этом каталоге, если явно не указано иное.

5. Согласно *инструкции* создайте пару ключей SSH, которая будет использоваться для подключения к ВМ.

Развертывание ВМ

Для подготовки ВМ выполните следующие действия:

1. Создайте файл `Vagrantfile` со следующим содержимым:

```
# frozen_string_literal: true

Vagrant.configure('2') do |config|
  # Настройки Vagrant Box
  config.vm.box = 'alse-vanilla-max/1.7.4' # Имя бокса
  config.vm.box_url = 'https://dl.astralinux.ru/vagrant/alse-vanilla-max%2F1.7.4'

  # Настройки SSH
  config.vm.provision 'file',
                      source: '~/.ssh/deployment.pub',
                      destination: '/home/vagrant/.ssh/deployment.pub'

  config.vm.provision 'shell',
                      inline: 'cat /home/vagrant/.ssh/deployment.pub >> /home/
↪vagrant/.ssh/authorized_keys'

  # Настройка параметров ВМ
  config.vm.define 'host01' do |node|
    node.vm.hostname = 'host01' # Короткое имя хоста

    # Настройки, специфичные для VirtualBox
    node.vm.provider 'virtualbox' do |vb|
      vb.cpus = 2 # Кол-во ядер CPU
      vb.memory = 2296 # Объем RAM, МБ
    end
```

(continues on next page)

[34] https://hub.astra-automation.ru/aa-gca/ARFA/nginx/

(продолжение с предыдущей страницы)

```
    # Параметры сети
    node.vm.network 'private_network',
                    ip: '192.168.56.11', # Статический IP-адрес
                    virtualbox__host_ip: '192.168.56.1', # Шлюз
                    virtualbox__netmask: '255.255.255.0', # Маска подсети
                    virtualbox__dhcp_enabled: false # Запрет использования DHCP
  end
end
```

2. Для создания и запуска ВМ с помощью Vagrant в каталоге проекта выполните команду:

```
vagrant up
```

Подготовка Ansible

Подготовьте ресурсы, необходимые для использования Ansible.

1. Создайте файл `ansible.cfg`, содержащий настройки Ansible для проекта:

```
[defaults]
collections_path = ./collections
ansible_python_interpreter = /usr/bin/python3
inventory = inventory
host_key_checking = false
```

   Здесь:

   - `collections_path` – путь к каталогу, в который будут загружены файлы коллекции `astra.nginx`.
   - `host_key_checking` – проверка ключей SSH при подключении к управляемым узлам. В данном случае она отключена, так как playbook запускается внутри контейнера.
   - `ansible_python_interpreter` – версия Python, которую следует использовать на управляемых узлах.
   - `inventory` – путь к файлу инвентаря.

2. Создайте файл инвентаря `inventory`:

```
[all]
host01 ansible_host=192.168.56.11 ansible_user=vagrant ansible_ssh_private_key_
↪file=/root/.ssh/deployment
```

   В этом файле содержатся настройки подключения к управляемому узлу `host01`:

   - Имя пользователя – `vagrant`.
   - IP-адрес – `192.168.56.11`.

     Должен совпадать с IP-адресом, указанным в `Vagrantfile`.

   - Путь к файлу приватного ключа SSH, используемого для подключения к ВМ.

3. Создайте файл playbook `playbook-nginx.yml`:

```
---
- name: Install NGINX server
  hosts: host01
  become: true
  gather_facts: true

  roles:
    - role: astra.nginx.nginx
      vars:
        nginx_worker_connections: 2048
        nginx_keepalive_timeout: 30
```

В файле `playbook-nginx.yml` настройки NGINX заданы через значения переменных роли `astra.nginx.nginx`.

4. Создайте файл зависимостей Ansible `requirements.yml`:

```
---
collections:
  - name: astra.nginx
    type: git
    version: 1.6.0
    source: ssh://git@hub.astra-automation.ru:2222/aa-gca/ARFA/nginx.git
```

В этом файле указаны параметры подключаемой коллекции:

- `name` – название коллекции из реестра Astra Automation Hub, в данном случае – `astra.nginx`.
- `type` – способ установки. Для установки коллекций из реестра Astra Automation Hub используется Git.
- `version` – версия используемой коллекции, `1.6.0`.
- `source` – ссылка на репозиторий Git, в котором хранится коллекция.

Запуск playbook

Чтобы развернуть веб-сервер NGINX с параметрами, описанными в конфигурационном файле `playbook-nginx.yml`, выполните в каталоге проекта следующие действия:

1. Запустите Docker-контейнер с EE:

```
docker run \
  --rm \
  --interactive \
  --tty \
  --volume ~/.ssh/deployment:/root/.ssh/deployment \
  --volume ~/.ssh/hub.astra-automation.ru:/root/.ssh/id_rsa \
  --volume "$(pwd):/app/" \
  registry.astralinux.ru/aa/aa-base-ee:0.2.1 \
  bash -c 'ansible-galaxy collection install -r requirements.yml && `
          `ansible-playbook playbook-nginx.yml'
```

Подробное описание EE см. в разделе *Среда исполнения*.

2. В терминал выводится сообщение о подключении к Astra Automation Hub:

```
Cloning into '/root/.ansible/tmp/ansible-local-*********/************/
↪nginx********'...
The authenticity of host '[hub.astra-automation.ru]:2222 ([84.201.175.30]:2222)'␣
↪can't be established.
ECDSA key fingerprint is SHA256:bvbETX8oC4ZwFtiE30SCN6yOQxRyOjgGO+lq2U2Dy00.
Are you sure you want to continue connecting (yes/no)?
```

Введите значение yes и нажмите Enter.

3. Если приватный ключ для доступа к реестру Astra Automation Hub защищен паролем, будет выведен запрос на его ввод:

```
Enter passphrase for key '/root/.ssh/id_rsa':
```

Введите пароль и нажмите Enter. Дождитесь выполнения playbook, это может занять некоторое время. По окончании выполнения в терминал выводится строка вида:

```
host01       : ok=8   changed=6   unreachable=0   failed=0   skipped=8   ␣
↪rescued=0   ignored=0
```

Проверка работоспособности развернутой инфраструктуры

Чтобы проверить корректность развертывания веб-сервера NGINX, выполните следующие действия:

1. Подключитесь к ВМ по SSH:

```
vagrant ssh
```

2. Проверьте статус службы nginx:

```
systemctl status nginx
```

При успешном развертывании в терминал выводится сообщение вида:

```
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendoer preset:␣
↪enabled)
     Active: active (running) since Tue 2023-04-18 08:57:44 MSK; 11s ago
       Docs: man:nginx(8)
   Main PID: 7970 (nginx)
      Tasks: 5 (limit: 4621)
     Memory: 4,9M
        CPU: 953ms
     CGroup: /system.slice/nginx.service
             ├─7970 nginx: master process /usr/sbin/nginx -g daemon on; master_
↪process on;
             ├─7973 nginx: worker process
             ├─7974 nginx: worker process
             ├─7975 nginx: worker process
             └─7976 nginx: worker process
```

3. Убедитесь, что в файле /etc/nginx/nginx.conf значения параметров worker_connections и keepalive_timeout соответствуют заданным в playbook.

4. Для отключения от ВМ выполните команду:

```
exit
```

Освобождение ресурсов

Если созданная ВМ больше не нужна, для ее остановки и удаления выполните в каталоге проекта команду:

```
vagrant destroy --force
```

Особенности проекта

Обратите внимание на следующие особенности проекта:

- Порядок подключения коллекции.
- Настройка параметров роли.
- Настройка параметров Ansible.
- Параметры, передаваемые в EE при запуске Ansible.

Заключение

В этом сценарии вы познакомились с основными шагами по развертыванию веб-сервера NGINX с использованием коллекции из реестра Astra Automation Hub. Из всей последовательности шагов важно выделить следующие действия:

- Подготовка окружения.
- Описание коллекции в файле зависимостей.
- Описание параметров доступа к управляемому узлу в файле инвентаря.
- Связывание управляемого узла и роли в playbook.
- Запуск playbook с помощью EE.
- Проверка корректности развертывания NGINX.

Контроллер домена ALD Pro

Развертывание контроллера домена ALD Pro с использованием коллекции astra.ald_pro[35] из реестра Astra Automation Hub позволяет автоматизировать большую часть рутинных операций.

[35] https://hub.astra-automation.ru/aa-gca/ARFA/ald_pro/

Описание сценария

Роль `astra.ald_pro.controller` из реестра коллекций Astra Automation Hub позволяет за несколько минут в автоматическом режиме развернуть контроллер домена ALD Pro.

Процесс развертывания контроллера домена ALD Pro состоит из следующих этапов:

1. Создание стенда.

   В качестве стенда в этом руководстве используется ВМ VirtualBox, созданная с помощью *Vagrant*.

   Она обладает следующими характеристиками:

   - количество ядер CPU – 4;
   - объем RAM, ГБ – 4;
   - объем хранилища, ГБ – 30;
   - операционная система – Astra Linux Special Edition 1.7.3 без графического интерфейса;
   - уровень защищенности ОС – максимальный («Смоленск»);
   - выключено разрешение имен с помощью служб хостовой машины.

2. Подготовка файлов Ansible:

   - `requirements.yml` – зависимости;
   - `ansible.cfg` – настройки Ansible;
   - `inventory` – инвентарь;
   - `playbook.yml` – playbook, содержащий сценарии настройки управляемого узла;
   - `vars.yml` – значения переменных.

3. Настройка стенда с помощью *Ansible*.

   Для запуска Ansible используется образ, описанный в разделе *Среда исполнения*.

Развертываемый контроллер домена ALD Pro обладает следующими характеристиками:

- Имя домена – `aldpro.example.com`.
- FQDN контроллера домена – `dc01.aldpro.example.com`.
- Учетные данные администратора домена:
  - имя пользователя – `admin`;
  - пароль – `p@ssW0a!`.

Для подключения к стенду используется протокол SSH.

Подготовка к работе

Подготовьте рабочее окружение:

1. Изучите описание коллекции astra.ald_pro[36]. Обратите внимание на обязательные и опциональные переменные роли `astra.ald_pro.controller`.
2. Подготовьте управляющий узел к работе с реестром Astra Automation Hub и стендом согласно *инструкции*.
3. Установите Vagrant и его расширение для работы с образами Astra Linux согласно *инструкции*.
4. Создайте каталог для хранения файлов проекта, например:

```
mkdir ~/aa-ald-pro/
```

> **Примечание:** Далее этот каталог называется каталогом проекта. Все упомянутые ниже файлы следует создавать в этом каталоге, если явно не указано иное.

5. Согласно *инструкции* создайте пару ключей SSH, которая будет использоваться для подключения к ВМ.

Развертывание ВМ

Для подготовки ВМ выполните следующие действия:

1. Создайте файл `Vagrantfile` со следующим содержимым:

```
# frozen_string_literal: true

Vagrant.configure('2') do |config|
  config.vm.box = 'alse-vanilla-max/1.7.3' # Имя бокса

  # Ссылка на образ
  config.vm.box_url = 'https://dl.astralinux.ru/vagrant/alse-vanilla-max%2F1.7.3'

  config.vm.provision 'file',
                      source: '~/.ssh/deployment.pub',
                      destination: '/home/vagrant/.ssh/deployment.pub'

  config.vm.provision 'shell',
                      inline: 'cat /home/vagrant/.ssh/deployment.pub >> /home/
↪vagrant/.ssh/authorized_keys'

  # Настройка параметров ВМ
  config.vm.define 'dc01' do |node|
    node.vm.hostname = 'dc01' # Короткое имя хоста

    # Настройки, специфичные для VirtualBox
    node.vm.provider 'virtualbox' do |vb|
      vb.cpus = 4 # Кол-во ядер CPU
      vb.memory = 4296 # Объем RAM, МБ
      vb.auto_nat_dns_proxy = false # Не использовать DNS-прокси
```

(continues on next page)

[36] https://hub.astra-automation.ru/aa-gca/ARFA/ald_pro/

(продолжение с предыдущей страницы)

```
    vb.customize ['modifyvm', :id, '--natdnshostresolver1', 'off']
    vb.customize ['modifyvm', :id, '--natdnsproxy1', 'off']
  end

  # Параметры сети
  node.vm.network 'private_network',
                  ip: '192.168.56.11', # Статический IP-адрес
                  netmask: '255.255.255.0', # Маска подсети
                  dhcp_enabled: false # Запрет использования DHCP

  node.vm.provision 'shell', inline: <<~SCRIPT
    cat <<EOF > /etc/dhcp/dhclient-enter-hooks.d/resolv
    make_resolv_conf () {
      true
    }
    EOF
  SCRIPT
 end
end
```

В ALD Pro используется собственный DNS-сервер, поэтому необходимо:

1. Отключить разрешение имен через проксирование DNS-запросов в хостовую ОС:

   ```
   vb.customize ['modifyvm', :id, '--natdnshostresolver1', 'off']
   vb.customize ['modifyvm', :id, '--natdnsproxy1', 'off']
   ```

2. Изменить настройки клиента DHCP таким образом, чтобы он не вносил изменений в конфигурационный файл `/etc/resolv.conf`.

   С этой целью в конце настройки ВМ выполняется сценарий, создающий в каталоге `/etc/dhcp/dhclient-enter-hooks.d/` файл `resolv`. В этом файле содержится функция, выполняемая при формировании конфигурационного файла `/etc/resolv.conf`. В данном случае она возвращает значение `true`, не выполняя никаких дополнительных действий.

   ```
   node.vm.provision 'shell', inline: <<~SCRIPT
     cat <<EOF > /etc/dhcp/dhclient-enter-hooks.d/resolv
     make_resolv_conf () {
       true
     }
     EOF
   SCRIPT
   ```

2. Для создания и запуска ВМ с помощью Vagrant в каталоге проекта выполните команду:

   ```
   vagrant up
   ```

Подготовка Ansible

Подготовьте ресурсы, необходимые для использования Ansible.

1. Создайте файл `ansible.cfg`, содержащий настройки Ansible для проекта:

   ```
   [defaults]
   collections_path = collections
   host_key_checking = false
   ansible_python_interpreter = /usr/bin/python3
   inventory = inventory
   ```

   где

   - `collections_path` – путь к каталогу с файлами коллекций.
   - `host_key_checking` – проверка ключей SSH при подключении к управляемым узлам. В данном случае она отключена, так как playbook будет запускаться внутри контейнера.
   - `ansible_python_interpreter` – версия Python, которую следует использовать на управляемых узлах.
   - `inventory` – путь к файлу инвентаря.

2. Создайте файл инвентаря `inventory`:

   ```
   [all]
   dc01 ansible_host=192.168.56.11 ansible_user=vagrant ansible_ssh_private_key_
   ↪file=/root/.ssh/deployment
   ```

   В этом файле содержатся настройки подключения к управляемому узлу `dc01`:

   - Имя пользователя – `vagrant`.
   - IP-адрес – `192.168.56.11`.

     Должен совпадать с IP-адресом, указанным в `Vagrantfile`.
   - Путь к файлу приватного ключа SSH, используемого для подключения к ВМ.

3. Создайте файл `vars.yml`:

   ```
   ---
   aldpro_domain: aldpro.example.com
   aldpro_pdc_ip: 192.168.56.11
   aldpro_pdc_name: dc01
   aldpro_admin_password: p@ssW0rD!
   ```

   При развертывании контроллера ALD Pro обязательно должны быть заданы:

   - имя домена;
   - IP-адрес и FQDN контроллера домена;
   - пароль администратора домена.

   В файле `vars.yml` указанные параметры заданы через значения переменных роли `astra.ald_pro.controller`:

   - `aldpro_domain` – имя домена.
   - `aldpro_pdc_ip` – IP-адрес контроллера домена.

> **Предупреждение:** IP-адрес контроллера домена должен совпадать с IP-адресом ВМ, указанным в `Vagrantfile`.

- `aldpro_pdc_name` – сетевое имя контроллера домена (при выполнении playbook автоматически расширяется до FQDN путем присоединения имени домена).
- `aldpro_admin_password` – пароль администратора домена.

4. Создайте файл playbook `playbook.yml`:

```
---
- name: Set up ALD Pro domain controller
  hosts: dc01
  become: true
  vars_files:
    - vars.yml
  roles:
    - role: astra.ald_pro.controller
```

5. Создайте файл зависимостей Ansible `requirements.yml`:

```
---
collections:
  - name: astra.ald_pro
    type: git
    version: 0.2.0
    source: ssh://git@hub.astra-automation.ru:2222/aa-gca/ARFA/ald_pro.git
```

В этом файле указаны параметры подключаемой коллекции:

- `name` – название коллекции из реестра Astra Automation Hub, в данном случае – `astra.ald_pro`.
- `type` – способ установки. Для установки коллекций из реестра Astra Automation Hub используется Git.
- `version` – версия используемой коллекции, `0.2.0`.
- `source` – ссылка на репозиторий Git, в котором хранится коллекция.

Запуск playbook

Чтобы развернуть контроллер домена ALD Pro с параметрами, описанными в конфигурационном файле `playbook.yml`, выполните в каталоге проекта следующие действия:

1. Запустите Docker-контейнер с EE:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume ~/.ssh/deployment:/root/.ssh/deployment \
   --volume ~/.ssh/hub.astra-automation.ru:/root/.ssh/id_rsa \
   --volume "$(pwd):/app" \
   registry.astralinux.ru/aa/aa-base-ee \
   bash -c 'ansible-galaxy install -r requirements.yml && `
           `ansible-playbook playbook.yml'
```

Подробное описание EE см. в разделе *Среда исполнения*.

2. В терминал выводится сообщение о подключении к Astra Automation Hub:

```
Cloning into '/root/.asible/tmp/ansible-local-******'...
The authenticity of host '[hub.astra-automation.ru]:2222 ([84.201.175.30]:2222)'␣
↪can't be established.
ECDSA key fingerprint is SHA256:*******.
Are you sure you want to continue connecting (yes/no)?
```

Введите значение yes и нажмите Enter.

3. Если приватный ключ SSH для доступа к реестру Astra Automation Hub защищен паролем, будет выведен запрос на его ввод:

```
Enter passphrase for key '/root/.ssh/id_rsa':
```

Введите пароль и нажмите Enter. Дождитесь выполнения playbook. По окончании выполнения в терминал выводится строка вида:

```
dc01         : ok=30   changed=10   unreachable=0   failed=0   skipped=4   ␣
↪rescued=0   ignored=0
```

Проверка работоспособности развернутой инфраструктуры

Чтобы проверить корректность развертывания контроллера домена ALD Pro, выполните следующие действия:

1. Подключитесь к ВМ по SSH:

```
vagrant ssh
```

2. Проверьте статус служб контроллера домена:

```
sudo ipactl status
```

При успешном развертывании в терминал выводится сообщение вида:

```
Directory Service: RUNNING
krb5kdc Service: RUNNING
kadmin Service: RUNNING
named Service: RUNNING
httpd Service: RUNNING
ipa-custodia Service: RUNNING
smb Service: RUNNING
winbind Service: RUNNING
ipa-otpd Service: RUNNING
ipa-dnskeysyncd Service: RUNNING
ipa: INFO: The ipactl command was successfull
```

3. Для отключения от ВМ выполните команду:

```
exit
```

Освобождение ресурсов

Если созданная ВМ больше не нужна, для ее остановки и удаления в каталоге проекта выполните команду:

```
vagrant destroy --force
```

Особенности проекта

Обратите внимание на следующие особенности проекта:

- Опциональные и обязательные параметры роли `astra.ald_pro.controller`.
- Описание переменных роли в отдельном файле.

Заключение

В этом сценарии вы познакомились с основными шагами по развертыванию контроллера домена ALD Pro с использованием коллекции из реестра Astra Automation Hub. Из всей последовательности шагов, которые вы прошли, важно выделить следующие действия:

- Подключение коллекции Ansible из реестра Astra Automation Hub.
- Настройка параметров роли в отдельном файле и подключение его в playbook.

5.2.4 Комплексные решения

Важно: Эта часть документации находится в стадии разработки.

5.3 Состав Astra Automation Hub

Здесь представлены актуальные данные по содержимому Astra Automation Hub.

5.3.1 Модули Terraform

Следующая таблица предоставляет краткую информацию о реестре[37], содержащем модули Terraform:

[37] https://hub.astra-automation.ru/aa-gca/AMFT

Модуль	Сервис		Описание
brest-image[38]	ПК «Брест»	СВ	Образ виртуальной машины
brest-virtual-machine[39]	ПК «Брест»	СВ	Виртуальная машина
sbercloud-compute-instance[40]	Cloud.ru		Виртуальная машина
sbercloud-vpc[41]	Cloud.ru		Виртуальная сеть
vsphere-virtual-machine[42]	VMware vSphere		Виртуальная машина
vsphere-vpc[43]	VMware vSphere		Виртуальная сеть
yandex-cloud-compute-instance[44]	Yandex Cloud		Виртуальная машина
yandex-cloud-vpc[45]	Yandex Cloud		Облачная сеть

5.3.2 Коллекции Ansible

Коллекции Ansible доступны в реестре Astra Automation Hub по адресу https://hub.astra-automation.ru/aa-gca/ARFA.

[38] https://hub.astra-automation.ru/aa-gca/AMFT/brest-image
[39] https://hub.astra-automation.ru/aa-gca/AMFT/brest-virtual-machine
[40] https://hub.astra-automation.ru/aa-gca/AMFT/sbercloud-compute-instance
[41] https://hub.astra-automation.ru/aa-gca/AMFT/sbercloud-vpc
[42] https://hub.astra-automation.ru/aa-gca/AMFT/vsphere-virtual-machine
[43] https://hub.astra-automation.ru/aa-gca/AMFT/vsphere-vpc
[44] https://hub.astra-automation.ru/aa-gca/AMFT/yandex-cloud-compute-instance
[45] https://hub.astra-automation.ru/aa-gca/AMFT/yandex-cloud-vpc

Коллекция	Поддерживаемые ОС	Название ПО	Комментарии
astra.ald_p	Astra Linux Special Edition 1.7.4 («Smolensk») Astra Linux Special Edition 1.7.3 («Smolensk») Astra Linux Special Edition 1.7.2 («Smolensk»)	ALD Pro[47]	Домен, управляемый заданным количеством контроллеров, на базе ALD Pro.
astra.brest	Astra Linux Special Edition 1.7.2 («Smolensk») Astra Linux Special Edition 1.7.2 («Voronezh») Astra Linux Special Edition 1.7.2 («Orel») Astra Linux Special Edition 1.6.12 Astra Linux Special Edition 1.6.10 Astra Linux Special Edition 1.6.9	ПК СВ «Брест»[49]	
astra.ceph	Astra Linux Special Edition 1.7.2 Astra Linux Special Edition 1.7.1 Astra Linux Special Edition 1.6.10	Ceph[51]	Система хранения данных Ceph с вариациями: - распределенная - на одном сервере
astra.cups	Astra Linux Special Edition 1.7.2 («Orel») Astra Linux Special Edition 1.7.1 («Orel») Astra Linux Special Edition 1.6.10 Astra Linux Common Edition 2.12.45 Astra Linux Common Edition 2.12.29	CUPS[53]	Служба печати
astra.dcim	Astra Linux Special Edition 1.7.3 («Orel»)	DCImanager[55]	

5.3.3 Библиотека инфраструктурного кода

Реестр содержит репозиторий инфраструктурного кода[90]. В нем хранятся шаблоны конфигураций, позволяющие за короткое время развернуть сложную инфраструктуру, состоящую из тесно интегрированных между собой программных продуктов.

Важно: Эта часть документации находится в стадии разработки.

[46] https://hub.astra-automation.ru/aa-gca/ARFA/ald_pro/
[47] https://www.aldpro.ru/
[48] https://hub.astra-automation.ru/aa-gca/ARFA/brest
[49] https://astragroup.ru/software-services/application-software-astra-group/brest/
[50] https://hub.astra-automation.ru/aa-gca/ARFA/ceph
[51] https://ceph.io/
[52] https://hub.astra-automation.ru/aa-gca/ARFA/cups
[53] https://www.cups.org/
[54] https://hub.astra-automation.ru/aa-gca/ARFA/dcimanager
[55] https://www.ispsystem.ru/dcimanager
[56] https://hub.astra-automation.ru/aa-gca/ARFA/dhcp
[57] https://www.isc.org/dhcp/
[58] https://hub.astra-automation.ru/aa-gca/ARFA/docker
[59] https://www.docker.com/
[60] https://hub.astra-automation.ru/aa-gca/ARFA/freeipa
[61] https://www.freeipa.org/
[62] https://hub.astra-automation.ru/aa-gca/ARFA/grafana
[63] https://grafana.com/
[64] https://hub.astra-automation.ru/aa-gca/ARFA/iscsi
[65] https://www.open-iscsi.com/
[66] https://hub.astra-automation.ru/aa-gca/ARFA/keycloak
[67] https://www.keycloak.org/
[68] https://hub.astra-automation.ru/aa-gca/ARFA/memcached
[69] https://memcached.org/
[70] https://hub.astra-automation.ru/aa-gca/ARFA/nginx/
[71] https://nginx.org/
[72] https://hub.astra-automation.ru/aa-gca/ARFA/ocfs2
[73] https://ocfs2.wiki.kernel.org/
[74] https://hub.astra-automation.ru/aa-gca/ARFA/postgresql
[75] https://www.postgresql.org/
[76] https://hub.astra-automation.ru/aa-gca/ARFA/postgresql
[77] https://prometheus.io/
[78] https://hub.astra-automation.ru/aa-gca/ARFA/rabbitmq
[79] https://www.rabbitmq.com/
[80] https://hub.astra-automation.ru/aa-gca/ARFA/rubackup
[81] https://www.rubackup.ru/
[82] https://hub.astra-automation.ru/aa-gca/ARFA/rupost
[83] https://www.rupost.ru/
[84] https://hub.astra-automation.ru/aa-gca/ARFA/tantor
[85] https://tantorlabs.ru/
[86] https://hub.astra-automation.ru/aa-gca/ARFA/termidesk
[87] https://termidesk.ru/
[88] https://hub.astra-automation.ru/aa-gca/ARFA/vmmanager
[89] https://www.ispsystem.ru/vmmanager
[90] https://hub.astra-automation.ru/aa-gca/AAC

5.4 Private Hub

Важно: Эта часть документации находится в стадии разработки.

ГЛАВА 6

Контроллер

Astra Automation Controller (контроллер) – центральный компонент платформы Astra Automation, который управляет автоматизацией различных процессов с помощью следующих встроенных механизмов:

- Управление через графический интерфейс пользователя, CLI, API и Ansible.
- Наблюдение за различными аспектами процессов выполнения playbook с использованием средств фильтрации.
- Упрощение запуска процессов с минимальным количеством действий.
- Развитая система безопасности, базирующаяся на ролевой модели и аудите.
- Масштабирование системы управления.
- Обеспечение надежности с помощью кластерных решений и встроенной системы резервного копирования и восстановления.
- Расширение возможностей за счет интеграции с различными репозиториями коллекций Ansible.
- Статическая и динамическая инвентаризация.
- Отправка уведомлений по электронной почте и в различные средства обработки, анализа и распространения сообщений (Grafana, IRC, Mattermost, Slack, Twilio и другие).
- Автоматическое распределение заданий по группам управляемых узлов для снижения пиковой нагрузки.
- И многое другое.

6.1 Особенности архитектуры

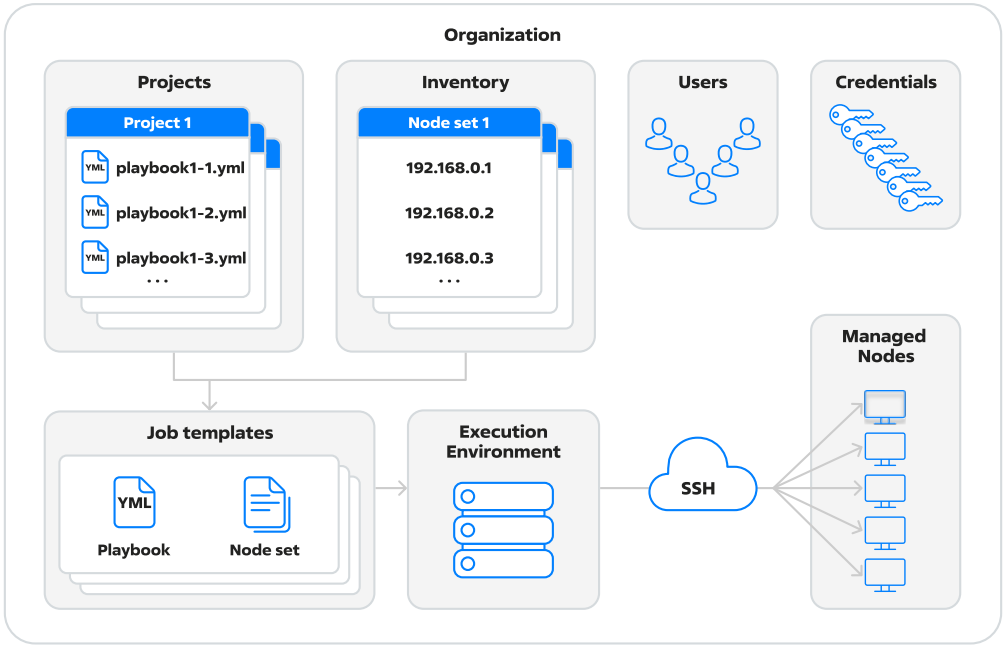
Важно: Эта часть документации находится в стадии разработки.

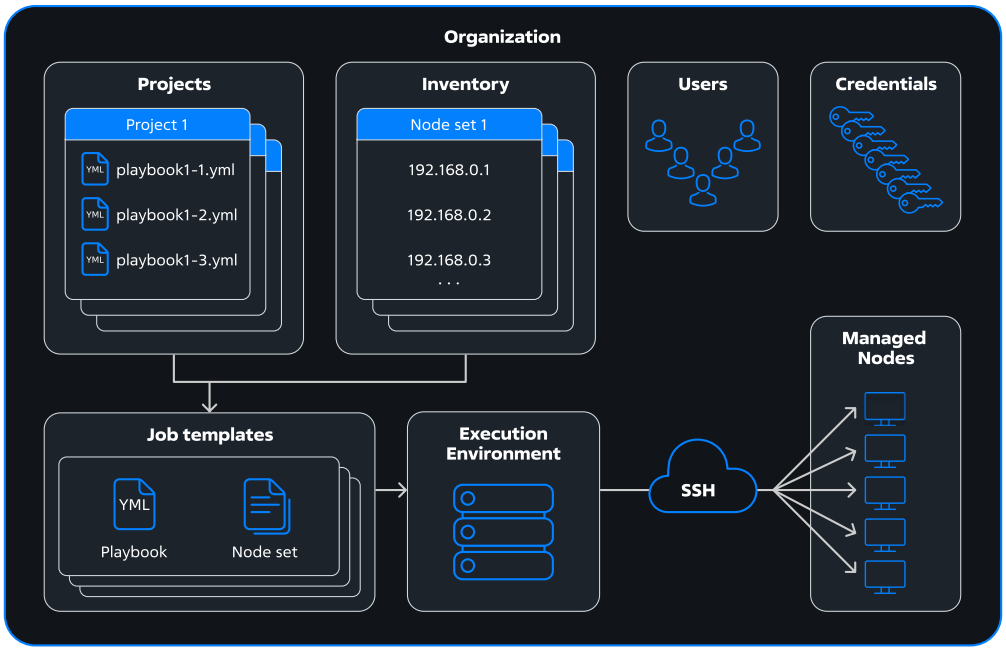
6.2 Структура управления

Для управления инфраструктурой контроллер использует следующие компоненты:

- playbooks;
- инвентарь (inventory);
- проекты (projects);
- шаблоны (templates);
- среды исполнения (execution environments);
- полномочия (credentials).

Взаимосвязи между компонентами показаны на схеме:





6.2.1 Проект

Проект состоит из одного или нескольких Ansible playbook. Контроллер позволяет импортировать существующие наборы playbook из различных источников:

- репозиторий Git;
- репозиторий Subversion;
- архивы форматов `.zip` и `.tar.gz`.

Также существующие наборы playbook можно вручную разместить в одном из каталогов локальной файловой системы контроллера, после чего указать путь к нужному каталогу в настройках проекта.

Подробности см. в разделе *Проекты*.

6.2.2 Playbook

Для настройки управляемых узлов контроллер использует *Ansible playbook*.

6.2.3 Инвентарь

Контроллер предоставляет следующие возможности для управления *инвентарем Ansible*:

- заполнение списка управляемых узлов вручную или путем импорта из стороннего источника;
- использование статического и динамического инвентаря.

Подробности см. в разделе *Инвентарь*.

6.2.4 Шаблон

Шаблоны бывают двух типов:

- шаблоны заданий;
- шаблоны потоков заданий.

Шаблон задания связывает между собой набор управляемых узлов из инвентаря и один из playbook, доступных в проекте.

Шаблон потока заданий связывает между собой шаблоны заданий, а также предоставляет дополнительные функции, позволяющие настроить выполнение отдельных заданий при наступлении определенных условий.

Подробности о шаблонах см. в разделе *Шаблоны*.

6.2.5 Среда исполнения

Среда исполнения содержит все необходимое для запуска playbook – фиксированные версии ОС, приложений и библиотек, в том числе Ansible, Python и других. Использование среды исполнения позволяет сделать выполнение playbook предсказуемым и не зависящим от ОС, в которой развернут контроллер.

6.2.6 Организации, пользователи и команды

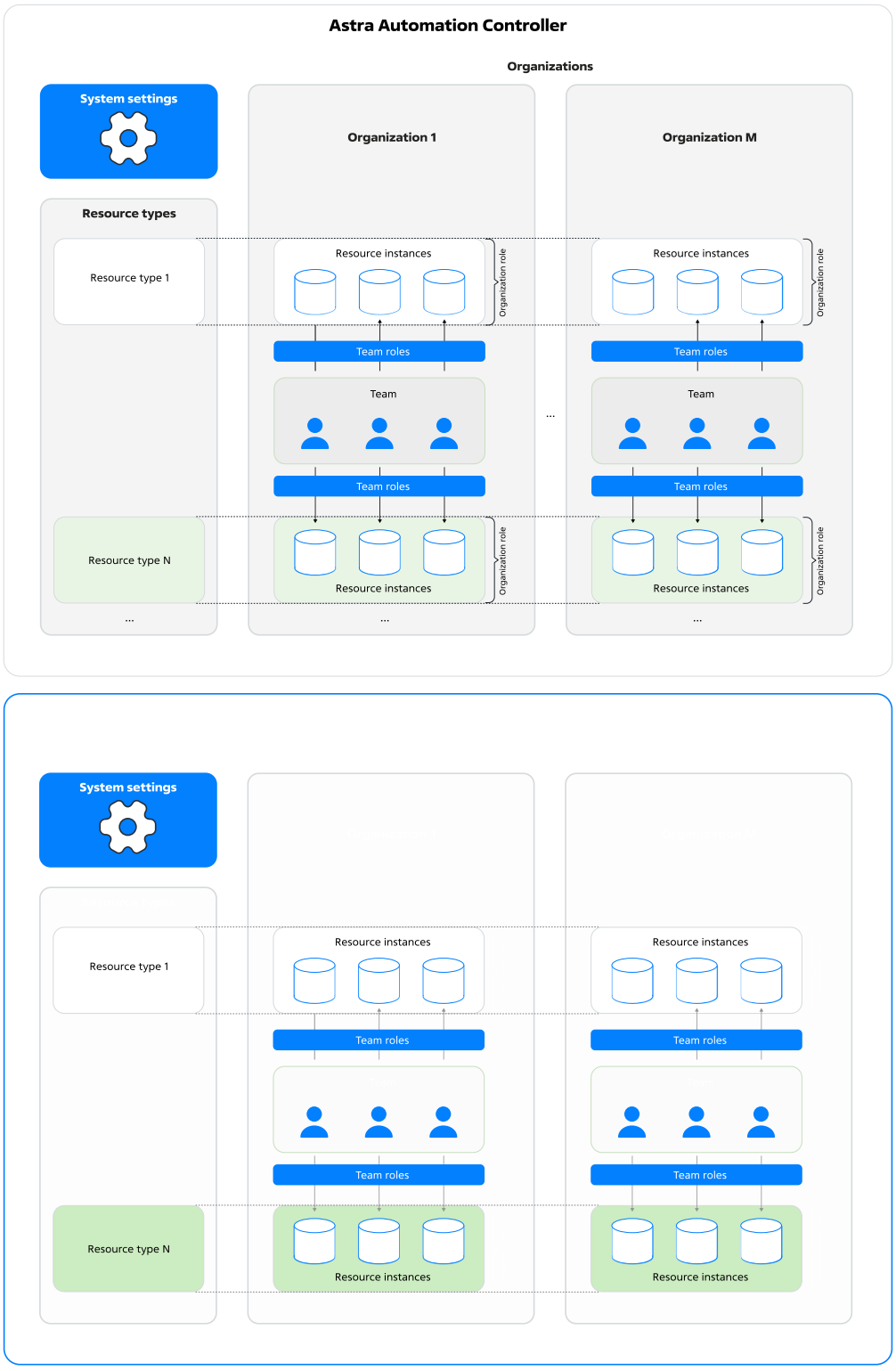
В Astra Automation Controller используется управление доступом на основе ролей (*RBAC*), реализованное с помощью организаций, пользователей и команд.

Организационная структура

В Astra Automation Controller используется разграничение доступа на основе ролей (*RBAC*). Набор привилегий пользователя на доступ к ресурсам определяется следующими параметрами:

- тип пользователя;
- назначенные роли.

Структура управления привилегиями показана на схеме:



Область действия предоставляемых ролью привилегий определяется уровнем, на котором она назначена:

1. Astra Automation Controller – изменение настроек контроллера и доступ ко всем ресурсам и компонентам;

2. организация – все ресурсы организации;
3. команда – отдельные экземпляры ресурсов.

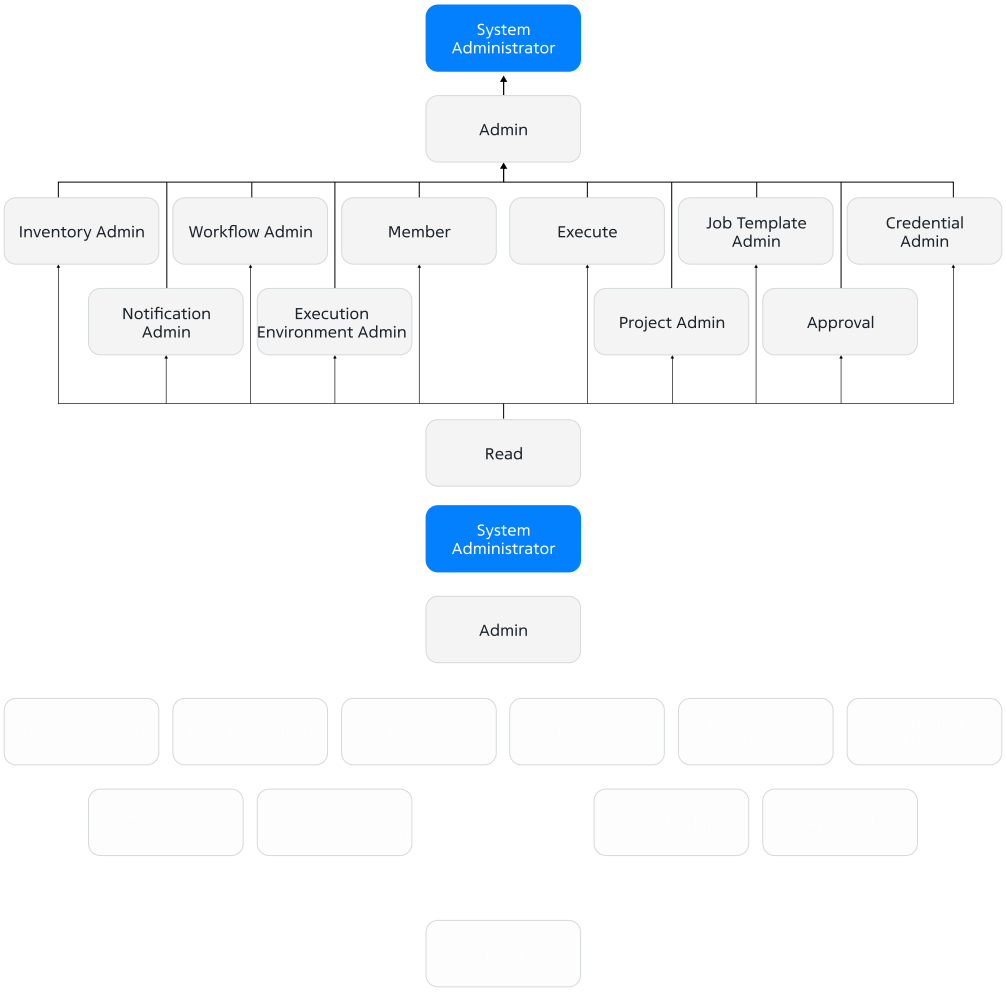
Роли

Роль – это набор привилегий на доступ к определенному ресурсу или типу ресурсов.

В Astra Automation Controller роли имеют следующие особенности:

- Набор ролей фиксирован, создание дополнительных ролей не допускается.
- Роли имеют иерархическую структуру – при назначении пользователю родительской роли он получает все привилегии дочерних ролей.

Пример иерархии наследования ролей на уровне организации показан на схеме:



Здесь от роли «Read» наследуют свои привилегии следующие роли:

- Execute;
- Project Admin;
- Inventory Admin;
- Credential Admin;

- Workflow Admin;
- Notification Admin;
- Job Template Admin;
- Execution Environment Admin;
- Approval;
- Member.

Например, при назначении роли «Approval» пользователь получает также привилегии роли «Read», назначать ее отдельно нет необходимости.

Роль «Admin» наследует привилегии от всех ролей, указанных в списке выше – при ее назначении пользователь получает все привилегии дочерних ролей, в том числе «Read».

Роль «System Administrator» наследует привилегии от роли «Admin»: пользователь с ролью «System Administrator» получает все привилегии дочерних ролей.

Пользователи

Пользователю можно назначить различные роли, в том числе в разных организациях и командах.

Примечание: Чтобы пользователь считался участником организации или команды, ему должна быть назначена роль «Участник» (Member) на уровне организации или команды соответственно. Для получения привилегий на ресурсы назначение роли «Участник» не обязательно.

Типы пользователей

В Astra Automation Controller доступны три типа пользователей (user types):

- системный администратор (System Administrator);
- системный аудитор (System Auditor);
- обычный пользователь (Normal User).

При необходимости тип пользователя можно изменить.

Системный администратор

При создании учетных записей пользователей этого типа им автоматически назначается системная роль «Системный администратор».

Системные администраторы имеют полный доступ ко всем компонентам и ресурсам контроллера, в том числе они имеют следующие привилегии:

- изменение системных настроек контроллера;
- создание, изменение и удаление следующих компонентов:
  - организаций;
  - команд;

  - пользователей, в том числе других системных администраторов;
- создание, изменение и удаление любых ресурсов.

При развертывании контроллера автоматически создается один пользователь типа «Системный администратор», который не входит ни в одну организацию. Имя учетной записи этого пользователя задается в файле `inventory` на этапе установки.

Системный аудитор

При создании учетных записей пользователей этого типа им автоматически назначается системная роль «Системный аудитор».

Системные аудиторы по умолчанию имеют привилегию на получение информации о любых ресурсах контроллера, а также создание полномочий.

Привилегии пользователей этого типа могут быть расширены путем назначения дополнительных ролей на уровне организаций и команд.

Обычный пользователь

При создании учетных записей пользователей этого типа им автоматически назначаются две роли:

- системная роль «Обычный пользователь» (Normal User);
- роль «Участник» (Member) на уровне указанной организации.

Привилегии пользователей этого типа расширяются путем назначения им дополнительных ролей на уровне организации и команды.

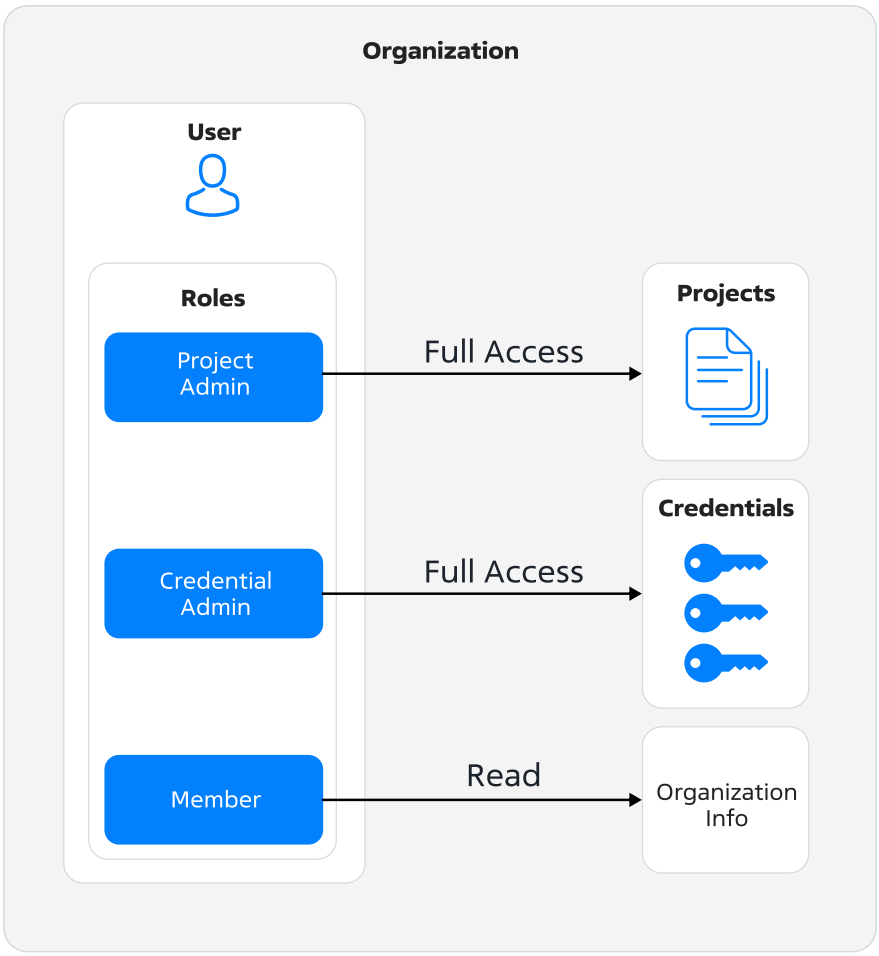
Организации

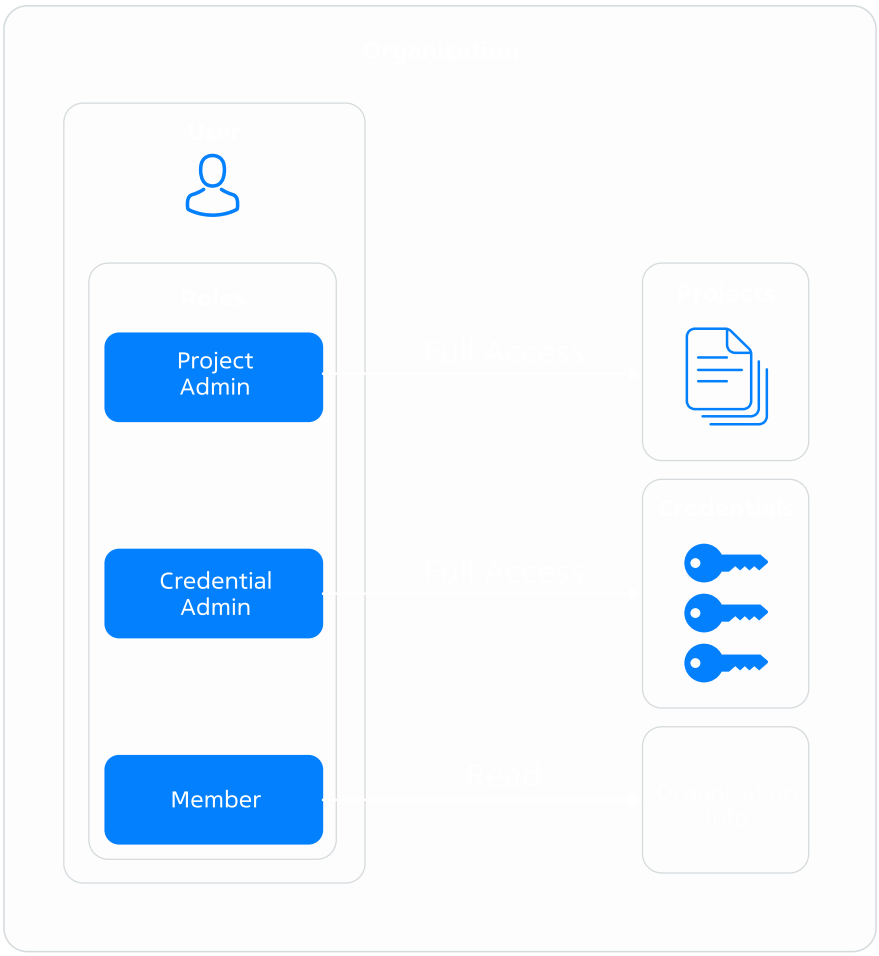
Организации используются для разграничения доступа к *типам ресурсов* контроллера.

На уровне организации роли могут быть назначены отдельным пользователям и командам.

Если на уровне организации роль назначена команде, то привилегии распространяются на всех членов команды, в том числе добавленных позднее.

Пусть пользователю на уровне организации назначены роли «Администратор проекта» (Project Admin), «Администратор учетных данных» (Credential Admin) и «Участник» (Member):


Organization
User
Roles
Project Admin
Credential Admin
Member
Full Access
Full Access
Read
Projects
Credentials
Organization Info



Назначенные роли предоставляют пользователю следующие привилегии на ресурсы организации:

- полный доступ ко всем проектам;
- полный доступ ко всем полномочиям;
- просмотр сведений об организации.

При развертывании контроллера в нем создается организация с названием «Default». Если другие организации отсутствуют, то все создаваемые ресурсы, пользователи и команды будут принадлежать этой организации.

Организационные роли

В таблице представлены привилегии, предоставляемые каждой организационной ролью на ресурсы организации.

Название роли	Может быть назначена пользователю	Может быть назначена команде	Описание
Администратор (Administrator)	Да	Нет	Полный доступ к ресурсам и свойствам организации
Администратор инвентаря (Inventory Admin)	Да	Да	Полный доступ к *инвентарю* и управляемым узлам
Администратор потока заданий (Workflow Admin)	Да	Да	Полный доступ к потокам заданий
Администратор проекта (Project Admin)	Да	Да	Полный доступ к *проектам*
Администратор среды исполнения (Execution Environment Admin)	Да	Да	Полный доступ к средам исполнения
Администратор уведомлений (Notification Admin)	Да	Да	Полный доступ к уведомлениям
Администратор учетных данных (Credential Admin)	Да	Да	Полный доступ к полномочиям
Администратор шаблона задания (Job Template Admin)	Да	Да	Полный доступ к *шаблонам заданий*
Аудитор (Auditor)	Да	Да	Получение сведений об организации и принадлежащих ей ресурсах
Выполнять (Execute)	Нет	Да	Запуск любых исполняемых ресурсов
Одобрить (Approve)	Да	Да	Подтверждение переходов между узлами потоков заданий
Участник (Member)	Да	Нет	Пользователь является членом организации
Чтение (Read)	Да	Да	Получение кратких сведений об организации

Связи с другими компонентами

Для некоторых компонентов связь с организацией обязательна:

Компонент	Связь с организацией
Шаблон задания	Обязательна Принадлежит организации через проект
Шаблон потока заданий	Не обязательна
Полномочия	Не обязательна
Проект	Обязательна
Инвентарь	Обязательна
Управляемый узел	Обязательна Принадлежит организации через инвентарь
Уведомления	Обязательна
Среда исполнения	Не обязательна
Задание	Не обязательна Задания связаны с организациями через шаблоны заданий
Команда	Обязательна
Пользователь	Обязательна при создании учетной записи При удалении организации связанные с ней пользователи сохраняются

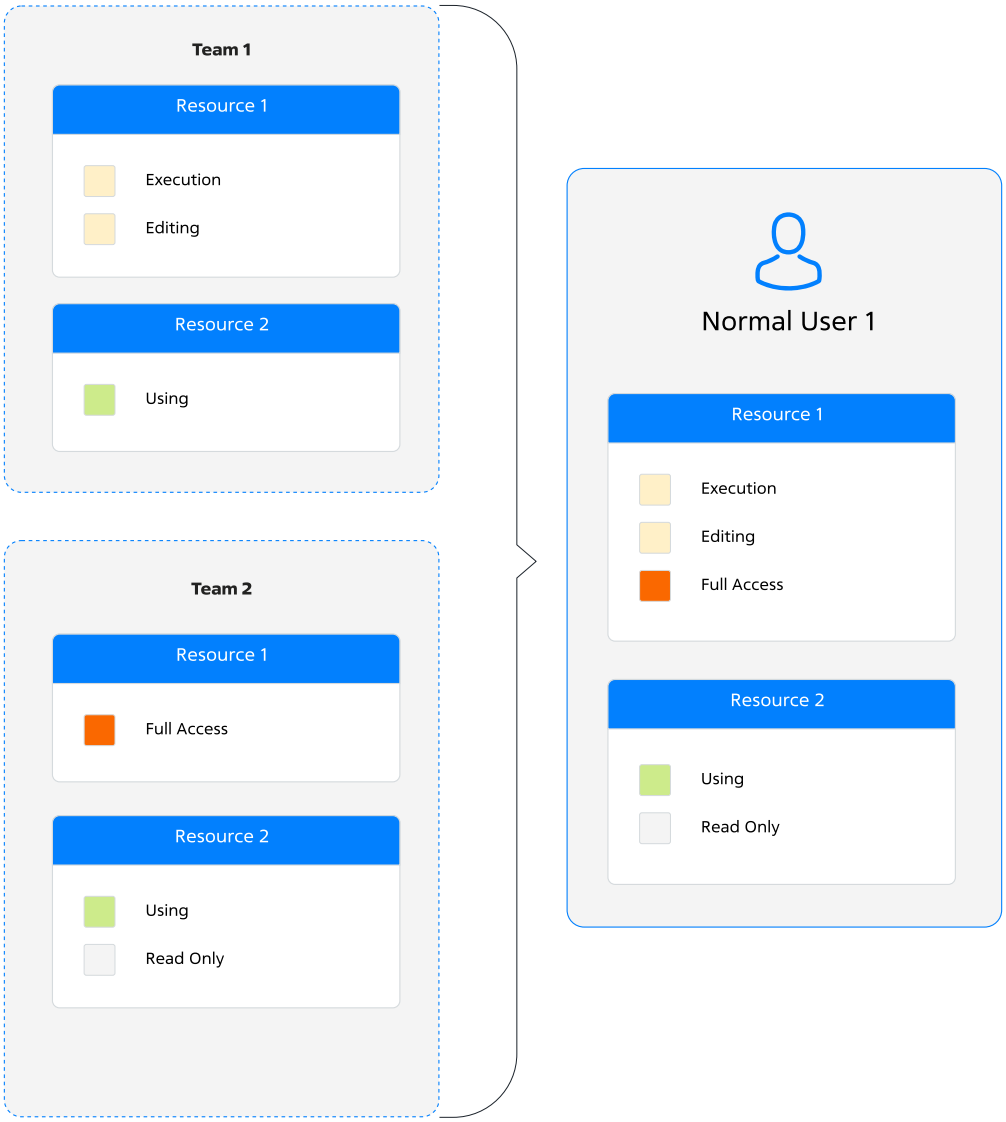
Команды

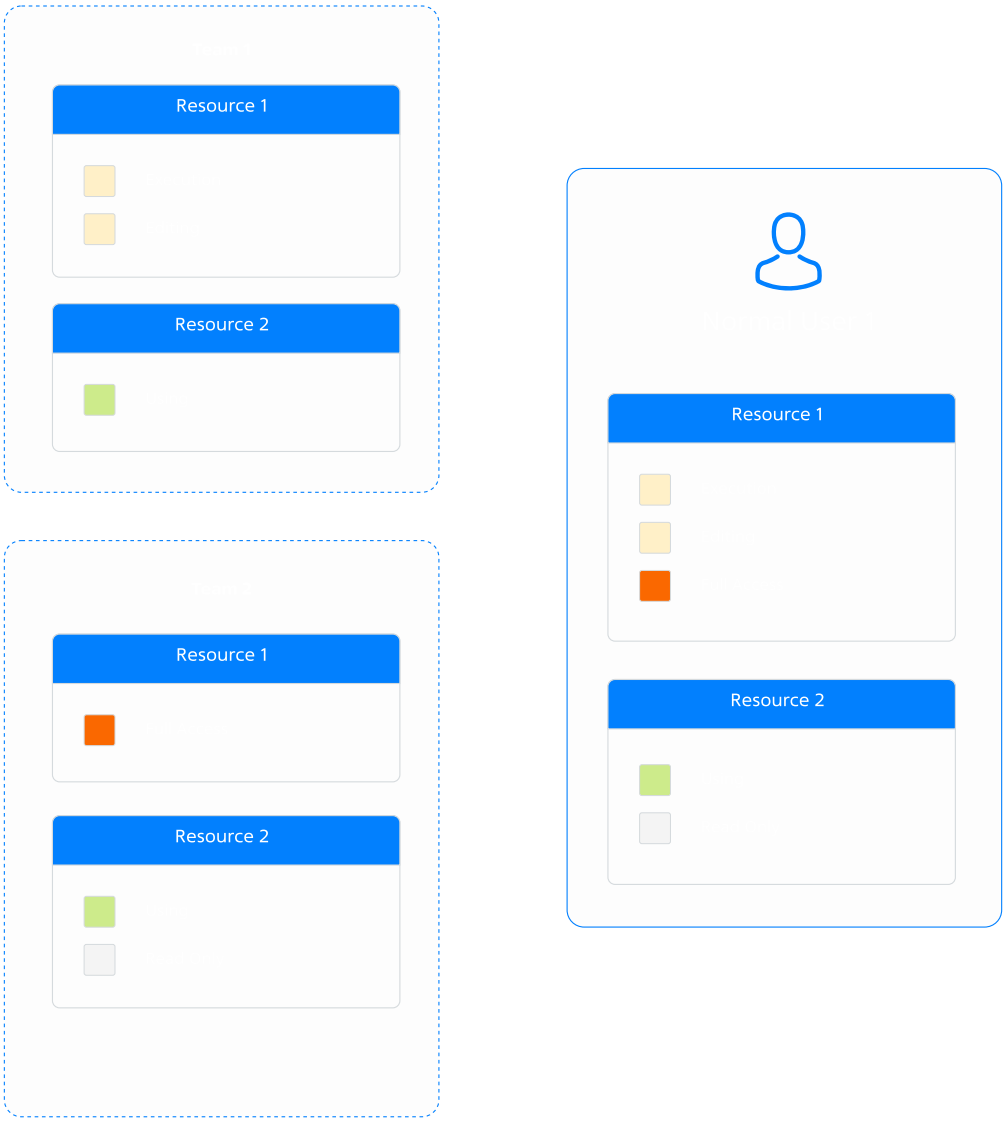
Команды существуют в рамках организаций и используются для разграничения доступа к *экземплярам ресурсов*. Это значит, что участники команды получают доступ не ко всем принадлежащим организации ресурсам определенного типа, а только к отдельным их экземплярам.

Например, если пользователю назначена командная роль «Администратор среды исполнения», то он может выполнять любые действия только с указанной средой исполнения: изменять, удалять и использовать ее.

Участниками команды могут быть пользователи из разных организаций.

Привилегии доступа к конкретному экземпляру ресурса определяются типом пользователя, а также объединением множеств привилегий, предоставляемых его пользовательскими и командными ролями во всех организациях и командах. Принцип объединения привилегий, предоставляемых ролями, показан на схеме:


Team 1
Resource 1
Execution
Editing
Resource 2
Using
Team 2
Resource 1
Full Access
Resource 2
Using
Read Only
Normal User 1
Resource 1
Execution
Editing
Full Access
Resource 2
Using
Read Only



Командные роли команды «Team 1» предоставляют к ресурсу «Resource 1» привилегии на исполнение (execution) и изменение (editing), а к ресурсу «Resource 2» – на использование (using). Командные роли команды «Team 2» предоставляют к ресурсу «Resource 1» привилегии полного доступа (full access), а к ресурсу «Resource 2» – на использование и только чтение (read only).

Все пользователи, входящие одновременно в команды «Team 1» и «Team 2», получают следующие привилегии на ресурсы:

- «Resource 1» – исполнение, изменение и полный доступ;
- «Resource 2» – использование и только чтение.

Командные роли

Командные роли определяют режим доступа **всех** участников команды к существующим ресурсам организации.

- шаблоны заданий;
- шаблоны потоков заданий;
- полномочия;
- инвентарь;
- проекты;
- организации;
- группы узлов контроллера.

На каждый ресурс можно назначить произвольную комбинацию доступных командных ролей.

Перечень доступных командных ролей определяется типом ресурса.

Шаблоны заданий (Job Templates)

Для предоставления привилегий на шаблоны заданий пользователям могут быть назначены следующие роли:

- Администратор (Admin) – полный доступ к шаблонам заданий.
- Выполнять (Execute) – запуск заданий из шаблона.
- Чтение (Read) – получение сведений о шаблонах заданий.

Шаблоны потоков заданий (Workflow Templates)

Для предоставления привилегий на шаблоны потоков заданий пользователям могут быть назначены следующие роли:

- Администратор (Admin) – полный доступ к шаблонам потоков заданий.
- Выполнять (Execute) – запуск шаблонов потоков заданий.
- Одобрить (Approval) – подтверждение перехода между заданиями в выбранных потоках заданий.
- Чтение (Read) – получение информации о шаблонах потоков заданий.

Полномочия (Credentials)

Для предоставления привилегий на полномочия пользователям могут быть назначены следующие роли:

- Администратор (Admin) – полный доступ к полномочиям.
- Использовать (Use) – использование указанных полномочий при создании ресурсов, например, проектов или шаблонов заданий.
- Чтение (Read) – получение сведений о полномочиях.

Инвентарь (Inventories)

Для предоставления привилегий на инвентарь пользователям могут быть назначены следующие роли:

- Администратор (Admin) – полный доступ к инвентарю.
- Использовать (Use) – использование инвентаря при создании шаблонов заданий и шаблонов потоков заданий.
- Обновление (Update) – обновление инвентарных списков, получаемых из внешних источников.
- Специальный (Ad Hoc) – запуск специальных (ad-hoc) команд на узлах из указанных инвентарных списков.
- Чтение (Read) – получение сведений об инвентарных списках.

Проекты (Projects)

Для предоставления привилегий на проекты пользователям могут быть назначены следующие роли:

- Администратор (Admin) – полный доступ к проектам.
- Использовать (Use) – использование проектов при создании шаблонов заданий.
- Обновление (Update) – изменение свойство проектов.
- Чтение (Read) – получение сведений о проектах.

Организации (Organizations)

Для предоставления привилегий на ресурсы организации пользователям могут быть назначены следующие роли:

- Администратор инвентаря (Inventory Admin) – полный доступ к инвентарю.
- Администратор потока заданий (Workflow Admin) – полный доступ к потокам заданий.
- Администратор проекта (Project Admin) – полный доступ к проектам.
- Администратор среды исполнения (Execution Environment Admin) – полный доступ к средам исполнения.
- Администратор уведомлений (Notification Admin) – полный доступ к уведомлениям.

- Администратор учетных данных (Credential Admin) – полный доступ к полномочиям.
- Администратор шаблона задания (Job Template Admin) – полный доступ к шаблонам заданий.
- Аудитор (Auditor) – получение сведений обо всех ресурсах организации.
- Выполнять (Execute) – запуск любых исполняемых ресурсов.
- Одобрить (Approve) – подтверждение перехода между заданиями потоков.
- Чтение (Read) – получение сведений об организации.

Группы узлов контроллера (Instance Groups)

Для предоставления привилегий на группы узлов контроллера пользователям могут быть назначены следующие роли:

- Администратор (Admin) – полный доступ к группам узлов.
- Использовать (Use) – использование групп узлов контроллера.
- Чтение (Read) – получение сведений о группах узлов контроллера.

Пользовательские роли

На уровне команды пользователю могут быть назначены индивидуальные роли, влияющие на доступ ко всем ресурсам команды:

- Администратор (Administrator) – полный доступ ко всем ресурсам команды.
- Участник (Member) – получение расширенных сведений об организации. Привилегии доступа к отдельным ресурсам определяются назначенными командными ролями.
- Чтение (Read) – получение сведений о команде.

Провайдеры аутентификации

Astra Automation Controller поддерживает использование внешних провайдеров аутентификации (в алфавитном порядке):

- Azure AD;
- GitHub;
- Google OAuth2;
- LDAP;
- OIDC;
- RADIUS;
- SAML;
- TACACS+.

Предупреждение: Провайдеры RADIUS и TACACS+ считаются устаревшими и не поддерживаются. В одной из следующий версий Astra Automation Controller эти провайдеры станут недоступными для применения.

Поддерживается автоматическая ассоциация пользователей из внешней системы аутентификации с организациями и командами контроллера.

Ассоциация с организациями

Настройки ассоциации внешних пользователей с организациями контроллера задаются в виде словаря, где ключ – название организации в контроллере, а значение – словарь, который может содержать следующие ключи:

- `admins` – администраторы организации;
- `remove_admins`;
- `users` – пользователи организации;
- `remove_users`.

Если организация с указанным названием не существует, она будет создана.

Обработка значений параметров `admins` и `users` выполняется следующим образом:

- Если значение не задано, список администраторов или пользователей организации не обновляется.
- Если значение параметра равно `true`, все аутентифицированные внешние пользователи автоматически получают привилегии администраторов организации или пользователей организации соответственно.
- Если значение параметра равно `false`, внешние пользователи получают привилегии администраторов организации или пользователей организации соответственно независимо от того, выполняли ли они аутентификацию.
- Если в значении параметра указана строка или массив строк, то интерпретация этих строк зависит от типа провайдера аутентификации.
  - Azure AD, GitHub, Google OAuth2

    Строки, в начале и конце которых стоит символ `/`, обрабатываются как регулярные выражения. Для регулярных выражений поддерживаются модификаторы, которые ставятся после закрывающей косой черты `/`:

    * `i` – игнорирование регистра символов;
    * `m` – поддержка многострочного текста.
  - LDAP

    В строках указываются параметры отбора пользователей LDAP, например:

```
"admins": "CN=Administrators,OU=groups,DC=example,DC=com"
```

Значение параметров `remove_admins` и `remove_users` определяет порядок обработки записей, не соответствующих условиям отбора. Если значение параметра равно `true`, внешние пользователи, не удовлетворяющие критериям отбора, удаляются из списка администраторов организации и пользователей организации соответственно.

Ассоциация с командами

Настройки ассоциации внешних пользователей с командами контроллера задаются в виде словаря, где ключ – название команды в контроллере, а значение – словарь, который может содержать следующие ключи:

- `organization` – название организации, которой принадлежит команда.

  Если организация с указанным названием не существует, она будет создана.

  Если в организации не существует команда с указанным названием, она будет создана.

- `users` – параметр, определяющий обработку пользователей команды.

  Если значение параметра не задано, список участников команды не меняется.

  Если значение параметра равно `true`, пользователи добавляются как участники команды.

  Если значение параметра равно `false`, пользователи удаляются из участников команды.

  Если в значении параметра указана строка или массив строк, то интерпретация этих строк зависит от типа провайдера аутентификации.

  - Azure AD, GitHub, Google OAuth2

    Строки, в начале и конце которых стоит символ `/`, обрабатываются как регулярные выражения. Для регулярных выражений поддерживаются модификаторы, которые ставятся после закрывающей косой черты `/`:

    * `i` – игнорирование регистра символов;
    * `m` – поддержка многострочного текста.

  - LDAP

    В строках указываются параметры отбора пользователей LDAP, например:

    ```
    "admins": "CN=Administrators,OU=groups,DC=example,DC=com"
    ```

- `remove` – если значение этого параметра равно `true`, внешние пользователи, не удовлетворяющие критериям отбора, удаляются из команды.

LDAP

Контроллер позволяет использовать одновременно до шести различных конфигураций подключения к серверам аутентификации LDAP.

По умолчанию используются настройки для работы со службой каталогов Microsoft Active Directory.

Идентификация пользователей, групп и организаций производится по *отличительному наименованию* (DN, Distinguished Name).

В каждой конфигурации можно задать URI одного или нескольких серверов LDAP. В качестве разделителя URI используется символ пробела или запятой.

Настройки поиска пользователей

Настройки поиска пользователей задаются в записи следующего вида:

```
[
  "<search_query>",
  "<scope>",
  "<key>"
]
```

Здесь:

- <search_query> – область поиска пользователей, например, `OU=Users,DC=example,DC=com`.
- <scope> – ограничение области поиска.

  Поддерживаются следующие значения:

  - `SCOPE_BASE` – поиск только по базовому отличительному наименованию;
  - `SCOPE_ONELEVEL` – поиск на один уровень ниже базового отличительного наименования, но не в самом базовом отличительном наименовании и не на более низких уровнях;
  - `SCOPE_SUBTREE` – поиск по базовому отличительному наименованию и по всем нижележащим уровням.
- <key> – указание того, какие данные о записи в LDAP следует считать названием учетной записи пользователя, например, `(cn=%(user)s)`.

Поддерживаются множественные параметры поиска (LDAPSearchUnion), например:

```
[
  [
    "OU=Users,DC=west,DC=example,DC=com",
    "SCOPE_SUBTREE",
    "(sAMAAccountName=%(user)s)"
  ], [
    "OU=Users,DC=north,DC=example,DC=com",
    "SCOPE_SUBTREE",
    "(sAMAAccountName=%(user)s)"
  ]
]
```

Настройки поиска групп

Настройки поиска групп задаются в записи следующего вида:

```
[
  "<search_query>",
  "<scope>",
  "key"
]
```

Здесь:

- <search_query> – область поиска групп, например, `DC=example,DC=com`.

- <scope> – ограничение области поиска.

  Поддерживаются следующие значения:

  - SCOPE_BASE – поиск только по базовому отличительному наименованию;
  - SCOPE_ONELEVEL – поиск на один уровень ниже базового отличительного наименования, но не в самом базовом отличительном наименовании и не на более низких уровнях;
  - SCOPE_SUBTREE – поиск по базовому отличительному наименованию и по всем нижележащим уровням.

- <key> – класс объекта LDAP, считающегося группой, например, `(objectClass=group)`.

Полномочия

Полномочия используются для доступа к различным ресурсам за пределами контроллера.

При создании любых полномочий обязательно должны быть указаны их название и тип. Если при создании полномочий указать организацию, то они будут доступны только в пределах этой организации. В противном случае полномочия доступны для использования всеми организациями.

Поддерживаемые типы полномочий

В Astra Automation Controller поддерживаются следующие типы полномочий:

- API-токен Ansible Galaxy/Automation Hub (Ansible Galaxy/Automation Hub API Token);
- Centrify Vault Credential Provider Lookup;
- CyberArk Central Credential Provider Lookup;
- CyberArk Conjur Secrets Manager Lookup;
- Google Compute Engine;
- GPG Public Key;
- HashiCorp Vault Secret Lookup;
- HashiCorp Vault Signed SSH;
- Insights;
- Microsoft Azure Key Vault;
- OpenStack;
- Red Hat Satellite 6;
- Thycotic DevOps Secrets Vault;
- Thycotic Secret Server;
- Vault;
- VMware vCenter;
- Web-сервисы Amazon (Amazon Web Services);

- Виртуализация Red Hat (Red Hat Virtualization);
- Диспетчер ресурсов Microsoft Azure (Microsoft Azure Resource Manager);
- Личный токен доступа к GitHub (GitHub Personal Access Token);
- Машина (Machine);
- Платформа автоматизации Red Hat Ansible (Red Hat Ansible Automation Platform);
- Реестр контейнеров (Container Registry);
- Сеть (Network);
- Токен персонального доступа GitLab (GitLab Personal Access Token);
- Токен доступа OpenShift или Kubernetes API (OpenShift or Kubernetes API Bearer Token);
- Управление версиями (Source Control).

Примечание: По умолчанию при развертывании контроллера в нем создаются полномочия для доступа к Ansible Galaxy[91] типа «Ansible Galaxy/Automation Hub». Контроллер позволяет добавлять собственные типы полномочий.

Также Astra Automation Controller позволяет создавать собственные типы полномочий.

API-токен Ansible Galaxy/Automation Hub

Полномочия этого типа используются для доступа к Ansible Galaxy или коллекциям, опубликованным в частном Automation Hub.

Для создания полномочий этого типа необходимо предоставить следующие данные:

- токен доступа, создаваемый пользователем на сервере Galaxy;
- адрес сервера Keykcloak для выдачи токена (при использовании SSO).

Centrify Vault Credential Provider Lookup

Полномочия этого типа используются для доступа к системе управления секретами Centrify Vault Credential Provider[92].

CyberArk Central Credential Provider Lookup

Полномочия этого типа используются для доступа к системе управления секретами CyberArk Central Credential Provider[93]. Для использования данной интеграции требуется запущенный веб-сервис хранения секретов CyberArk Central Credential Provider.

Поддерживается защита подключения с помощью SSL.

[91] https://galaxy.ansible.com/
[92] https://developer.delinea.com/docs/vault-functionality
[93] https://docs.cyberark.com/credential-providers/Latest/en/Content/CCP/The-Central%20-Credential-Provider.htm

CyberArk Conjur Secrets Manager Lookup

Полномочия этого типа используются для доступа к системе управления секретами CyberArk Conjur Secrets Manager[94].

Google Compute Engine

Полномочия этого типа используются для доступа к списку управляемых узлов в Google Compute Engine[95] с реквизитами сервисной учетной записи.

Для создания полномочий этого типа необходимо предоставить следующие данные о сервисной учетной записи Google Compute Cloud:

- адрес электронной почты;
- закрытый ключ RSA, связанный с указанным адресом электронной почты.

Совет: Для автоматического заполнения реквизитов сервисной учетной записи Google Compute Cloud можно использовать файл в формате JSON. Пошаговые инструкции по созданию такого файла см. в документации Google Compute Cloud[96].

GPG Public Key

Полномочия этого типа используются для проверки подписи содержимого, полученного из *внешних источников*.

HashiCorp Vault Secret Lookup

Полномочия этого типа используются для доступа к системе управления секретами HashiCorp Vault[97].

HashiCorp Vault Signed SSH

Полномочия этого типа используются для доступа к подписанным сертификатам HashiCorp Vault SSH[98].

[94] https://www.conjur.org/get-started/why-conjur/how-conjur-works/
[95] https://cloud.google.com/compute/docs/instances/os-inventory-management
[96] https://cloud.google.com/iam/docs/keys-create-delete
[97] https://developer.hashicorp.com/vault/docs/what-is-vault
[98] https://developer.hashicorp.com/vault/docs/secrets/ssh/signed-ssh-certificates

Insights

Полномочия этого типа используются для доступа к списку управляемых узлов в Red Hat Insights[99].

Microsoft Azure Key Vault

Полномочия этого типа используются для доступа к списку управляемых узлов в Microsoft Azure Key Vault[100].

OpenStack

Полномочия этого типа используются для доступа к списку управляемых узлов в OpenStack[101].

Поддерживается защита подключения с помощью SSL.

Red Hat Satellite 6

Полномочия этого типа используются для доступа к списку управляемых узлов в Red Hat Satellite 6[102].

Thycotic DevOps Secrets Vault

Полномочия этого типа используются для доступа к системе управления секретами Thycotic DevOps Secrets Vault.

Thycotic Secret Server

Полномочия этого типа используются для доступа к серверу секретов Thycotic Secret Server.

Vault

Полномочия этого типа используются для доступа к списку управляемых узлов в Ansible Vault[103].

Если необходим доступ к нескольким хранилищам, необходимо предоставить соответствующие им идентификаторы Vault.

[99] https://access.redhat.com/documentation/en-us/red_hat_insights/1-latest
[100] https://azure.microsoft.com/ru-ru/products/key-vault
[101] https://docs.openstack.org/2024.1/
[102] https://access.redhat.com/documentation/ru-ru/red_hat_satellite/6.0/html/installation_guide/chap-introduction
[103] https://docs.ansible.com/ansible/latest/cli/ansible-vault.html

VMware vCenter

Полномочия этого типа используются для доступа к списку управляемых узлов в VMware vCenter[104].

Web-сервисы Amazon

Полномочия этого типа используются для доступа к списку управляемых узлов в Web-сервисах Amazon[105].

Для создания полномочий этого типа необходимо предоставить следующие данные Web-сервисов Amazon:

- публичный ключ доступа;
- секретный ключ.

Виртуализация Red Hat

Полномочия этого типа используются для доступа к расширению (plug-in) инвентаря `oVirt4.py`, которым управляют с помощью Виртуализации Red Hat[106].

Диспетчер ресурсов Microsoft Azure

Полномочия этого типа используются для доступа к списку управляемых узлов в Диспетчере ресурсов Microsoft Azure[107].

Личный токен доступа к GitHub

Полномочия этого типа используются для доступа к GitHub[108] с помощью личного токена доступа. Данный тип полномочий позволяет устанавливать API-соединение с GitHub для использования в заданиях и публикации обновлений состояния.

Пошаговые инструкции по созданию токена см. в документации GitHub[109].

[104] https://docs.vmware.com/en/VMware-vSphere/index.html
[105] https://aws.amazon.com/documentation-overview/?nc2=h_ql_doc_do
[106] https://access.redhat.com/products/red-hat-virtualization
[107] https://learn.microsoft.com/ru-ru/azure/azure-resource-manager/management/overview
[108] https://github.com/
[109] https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens

Машина

Полномочия этого типа используются для доступа к управляемым узлам по SSH. Поддерживается аутентификация по паролю и ключам SSH, созданным с применением различных алгоритмов.

При настройке полномочия можно выбрать один из следующих способов повышения привилегий:

- sudo;
- su;
- pbrun;
- pfexec;
- dzdo;
- pmrun;
- runas;
- enable;
- doas;
- ksu;
- machinectl;
- sesu.

Использование этой настройки эквивалентно выполнению команды `ansible-playbook` с параметром `--become-method=<method>`.

Платформа автоматизации Red Hat Ansible

Полномочия этого типа используются для доступа к Платформе автоматизации Red Hat Ansible[110].

Поддерживается защита подключения с помощью SSL.

Реестр контейнеров

Полномочия этого типа используются для доступа к различным хранилищам образов, например:

- Docker[111];
- Quay[112];
- GitLab[113].

Поддерживается защита подключения с помощью SSL.

[110] https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.4
[111] https://www.docker.com/
[112] https://quay.io/
[113] https://registry.gitlab.com/

Сеть

Полномочия этого типа используются для управления сетевым оборудованием. К настоящему времени, согласно документации Ansible[114], он устарел и не рекомендован к использованию. Вместо него рекомендуется использовать тип «Машина».

Токен персонального доступа GitLab

Полномочия этого типа используются для доступа к GitLab[115] с помощью личного токена доступа. Данный тип полномочий позволяет устанавливать API-соединение с GitLab для использования в заданиях и публикации обновлений состояния с помощью webhook.

Пошаговые инструкции по созданию токена см. в документации GitLab[116].

Токен доступа для OpenShift или Kubernetes API

Полномочия этого типа используются для создания групп с доступом в контейнеры OpenShift или Kubernetes.

Для обеспечения доступа необходимо предоставить следующие данные:

- адрес точки доступа OpenShift или Kubernetes API для аутентификации;
- токен.

Поддерживается защита подключения с помощью SSL.

Управление источниками кода

Полномочия этого типа используются для доступа к системам контроля версий на основе Git и Subversion.

Поддерживается доступ с помощью ключа SSH или с использованием имени пользователя и пароля.

Создание типа полномочий

Astra Automation Controller поддерживает создание дополнительных типов полномочий, которые можно использовать в заданиях на основе шаблонов и заданиях обновления инвентаря.

Поддерживаются следующие способы передачи данных идентификации в среду исполнения:

- переменные окружения;
- дополнительные переменные Ansible;
- шаблонизация на основе файлов (генерация файлов в форматах CONF и INI).

[114] https://docs.ansible.com/ansible/devel/network/getting_started/network_differences.html#multiple-communication-protocols
[115] https://about.gitlab.com/
[116] https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html

Создание типа полномочий имеет следующие особенности:

- Контроллер не проверяет наличие коллизий названий переменных окружения, дополнительных переменных Ansible и пространств имен.
- Следует избегать использования названий переменных, начинающихся с `ANSIBLE_`, поскольку они зарезервированы.

Настройка входных данных

Входные данные используются при создании экземпляра полномочий указанного типа.

Типовая конфигурация входных данных состоит из обязательного блока `fields` и опционального блока `required`.

В блоке `fields` содержится список записей с описанием полей ввода данных полномочия. Запись может состоять из следующих полей:

- `id` – идентификатор поля. Значение, указанное в этом поле, используется для передачи данных в инжектор.
- `type` – тип поля.

  Поддерживаются следующие значения:

  - `string` – строка;
  - `boolean` – логическое.
- `default` – значение по умолчанию.
- `format` – дополнительная проверка корректности значения, указанного в поле.

  Поддерживаются следующие значения:

  - `ssh_private_key` – приватный ключ SSH;
  - `url` – URI.
- `label` – название поля при заполнении сведений о полномочии через графический интерфейс.
- `help_text` – подсказка, которая выводится рядом с полем при заполнении сведений о полномочии через графический интерфейс.
- `secret` – если равно `true`, вводимое значение является секретом.

  Для защиты ввода используется маскировка символов, для защиты значения в таблице базы данных контроллера – защитное преобразование.
- `multiline` – логическое значение для полей типа `string`. При значении `true` поле позволяет вводить многострочный текст.
- `choices` – массив доступных значений для полей типа `string`. Если массив не пуст, при заполнении поля через графический интерфейс значение нужно выбрать из списка, а не вводить вручную.

В блоке `required` содержится список идентификаторов полей, обязательных для заполнения.

Конфигурация инжектора

Настройки инжектора определяют порядок передачи данных полномочия в среду исполнения.

Для доступа к значению, полученному из входных данных, следует в двойных фигурных скобках указать идентификатор входных данных, например:

```
---
file:
  template: '[credentials]\nusername={{ username }}\npassword={{ password }}'
env:
  API_URL: '{{ api_url }}'
  AUTH_DATA_FILE: '{{ tower.filename }}'
extra_vars:
  auth_token: '{{ auth_token }}'
```

Типовая конфигурация инжектора может содержать следующие блоки:

- `file` – шаблон, на основе которого генерируется содержимое временного файла.

  Имя файла генерируется случайным образом. Оно хранится в переменной Ansible `tower.filename`.

  Если используется несколько полей для загрузки файлов, их следует описать в отдельных переменных, названия которых начинаются со слова `template.`. Для получения имени файла следует обратиться к соответствующему значению словаря `tower.filename`, например:

  ```
  ---
  file:
    template.cert_file: '{{ tls_cert }}'
    template.key_file: '{{ tls_key }}'
  env:
    TLS_CERT_FILE_NAME: '{{ tower.filename.cert_file }}'
    TLS_KEY_FILE_NAME: '{{ tower.filename.key_file }}'
  ```

  Здесь для хранения сертификата и соответствующего ему ключа создаются два временных файла, имена которых сохраняются в переменных `tower.filename.cert_file` и `tower.filename.key_file` соответственно.

- `env` – список названий переменных окружения и соответствующих им значений.
- `extra_vars` – список названий дополнительных переменных Ansible и соответствующих им значений.

  Подробности о порядке разрешения значений переменных см. в документе *Переменные*.

Примеры

Пусть для работы с некоторым сервисом необходимы следующие учетные данные:

- имя пользователя;
- токен;
- уровень доступа (может принимать значения `low`, `medium` и `high`);
- приватный ключ SSH для защиты соединения.

Пусть для получения данных используется форма со следующими полями:

Параметр	Идентификатор	Тип	Обязательное
Имя пользователя	`username`	`string`	Да
Токен	`token`	`string`	Да
Уровень доступа	`access_level`	`string`	Да
Ключ SSH	`ssh_key`	`string`	Нет

В этом случае настройки входных данных могут иметь следующий вид:

```
---
fields:
  - id: username
    type: string
    label: Username

  - id: token
    type: string
    label: Token
    help_text: A string of 32 charactes long.

  - id: access_level
    type: string
    label: Access level
    choices:
      - low
      - medium
      - high
    default: low

required:
  - username
  - token
  - access_level
```

Пусть в среду исполнения данные передаются следующими способами:

- Имя пользователя и токен – в переменных окружения `API_USERNAME` и `API_TOKEN` соответственно. При этом строка для передачи токена должна иметь следующий вид:

  ```
  Bearer-Token: <token>
  ```

- Уровень доступа – в дополнительной переменной Ansible `api_access_level`.
- Приватный ключ SSH – в виде файла.

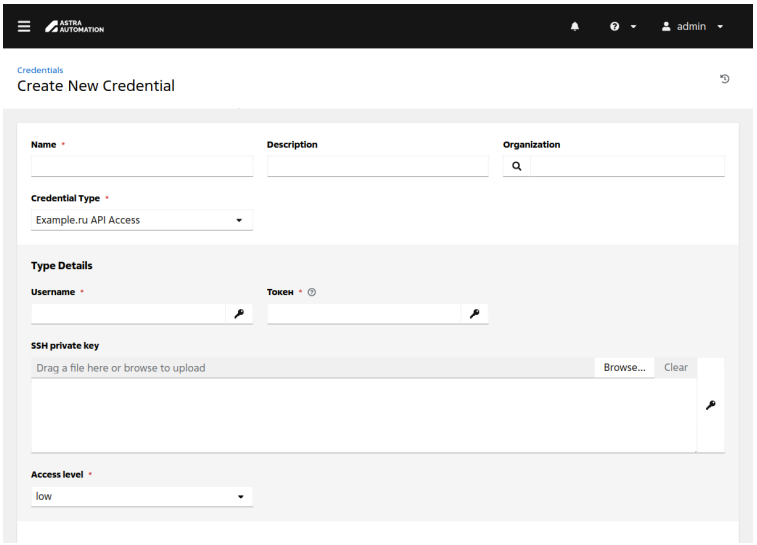
Имя файла формируется автоматически и передается через переменную окружения API_PRIVATE_SSH_KEY, а содержимое формируется на основе входных данных с идентификатором ssh_key.

В этом случае конфигурация инжектора выглядит следующим образом:

```
---
env:
  API_USERNAME: '{{ username }}'
  API_TOKEN: 'Bearer-Token:: {{ token }}'
  API_PRIVATE_SSH_KEY: '{{ tower.filename.private_ssh_key }}'
extra_vars:
  api_access_level: '{{ access_level }}'
file:
  template.private_ssh_key: '{{ ssh_key }}'
```

Примечание: Два двоеточия в строке с параметром API_TOKEN необходимы для соблюдения корректности синтаксиса YAML. При передаче данных строка будет преобразована в указанный выше формат с одним двоеточием.

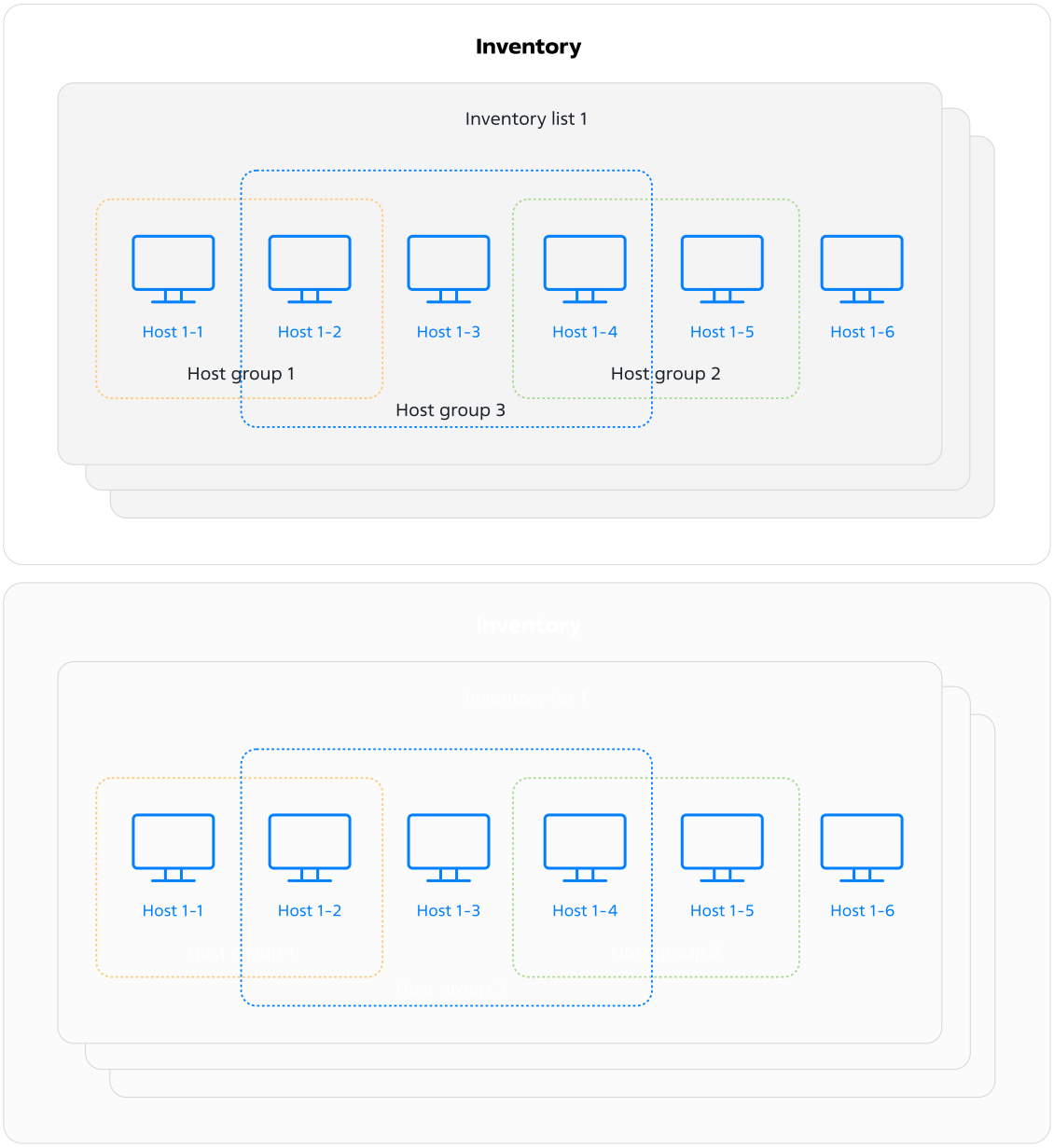
Форма для создания полномочий этого типа имеет следующий вид:



Инвентарь

Инвентарь (inventory) в Astra Automation Controller используется для хранения сведений об управляемых узлах. При создании шаблона задания управляемые узлы из выбранного инвентаря связываются с Ansible playbook из проекта.

Структура инвентаря показана на схеме:



Инвентарь состоит из инвентарных списков различного типа. Каждый инвентарный список содержит перечень управляемых узлов (hosts). Один и тот же узел может входить в несколько инвентарных списков.

Примечание: Далее для краткости инвентарные списки будут называться инвентарями.

Группы узлов

Группы узлов (inventory groups) используются для логического объединения узлов, входящих в один инвентарь. Один и тот же управляемый узел может входить в несколько групп одновременно.

Группы узлов обладают следующими свойствами:

- Для каждой группы узлов можно задать свои значения переменных Ansible[117].
- При импорте инвентаря из внешнего источника, содержащего группы узлов, одноименные группы в инвентаре контроллера создаются автоматически.
- Astra Automation Controller позволяет выполнять специальные (ad-hoc) команды на узлах выбранной группы без создания шаблонов.

Типы инвентаря

В Astra Automation Controller поддерживаются три типа инвентаря:

- обычный (standard);
- умный (smart);
- сборный (constructed).

Тип инвентаря влияет на способ добавления в него сведений об управляемых узлах. Подробности о каждом типе инвентаря приведены далее.

Особенности удаления

Удаление инвентарных списков имеет следующие особенности:

- Вместе с обычным инвентарным списком из контроллера удаляются данные обо всех связанных с ним управляемых узлах и группах узлов.
- При удалении обычных и сборных инвентарных списков состав связанных с ними сборных инвентарных списков **не** меняется. Однако, он может измениться, если запустить синхронизацию сборного инвентарного списка.
- Связанные с инвентарным списком шаблоны заданий не удаляются и могут быть связаны с другим инвентарным списком.

Состояния

Инвентарь может находиться в одном из состояний:

- **Успех** (Success) – синхронизация инвентаря прошла успешно.
- **Запрещен** (Disabled) – в инвентарь не добавлено ни одного источника сведений об управляемых узлах.
- **Ошибка** (Error) – последняя попытка синхронизации инвентаря с источником была неудачной.

[117] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html

Статусы управляемых узлов

Контроллер позволяет исключить использование в заданиях управляемых узлов без необходимости изменения существующих или создания новых инвентарных списков, а также внесения изменений в шаблоны заданий. Для этого каждой записи об управляемом узле присваивается *статус*, который указывает, может ли управляемый узел быть объектом выполняемых заданий Ansible:

- **Вкл** (On) – может. Этот статус используется по умолчанию при создании и импорте записей об управляемых узлах.
- **Выкл** (Off) – не может. Управляемые узлы с этим статусом исключаются из инвентаря при запуске заданий.

Обычный инвентарь

Обычный инвентарь, также называемый просто инвентарем (inventory), имеет следующие особенности:

- Сведения об управляемых узлах могут быть добавлены вручную или импортированы из внешнего источника, в том числе из файлов инвентаря Ansible, добавленных в проект.
- Поддерживает использование переменных Ansible и групп узлов.

Источники сведений об управляемых узлах

Импорт сведений об управляемых узлах из внешних источников имеет следующие особенности:

- Если в качестве источника выбран отдельный файл, импортируются узлы только из него.
- Если в качестве источника выбран каталог, импортируются узлы из всех размещенных в нем файлов, включая дочерние каталоги.
- Поддерживается фильтрация узлов по названию с помощью регулярных выражений. Этот фильтр применяется в последнюю очередь, после всех остальных фильтров, предоставляемых используемым расширением (plug-in) инвентаря.

  Пусть файл инвентаря содержит записи о тридцати управляемых узлах (диапазон IP-адресов от `10.1.0.1` до `10.1.0.30`):

```
---
all:
  hosts:
    # Контроллеры домена
    dc1.example.com:
    dc2.example.com:
    # Сервер печати
    cups.example.com:
    # Клиенты домена
    client01.example.com:
    client02.example.com:
    client03.example.com:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    # ...
    client30.example.com:
```

Для импорта только первых десяти клиентов домена (с `client01.example.com` по `client10.example.com` включительно) следует использовать регулярное выражение следующего вида:

```
client([0][1-9]|10)\.example\.com
```

- В описании импортированных узлов указывается значение **imported**.

  Это значение можно изменить, добавив в файл инвентаря поле `_awx_description` для нужных узлов, например:

```
---
domain-controllers:
  hosts:
    dc1.example.com:
      _awx_description: Primary domain controller
    dc2.example.com:
      _awx_description: First replica of domain controller
```

- Если в файле инвентаря узел принадлежит одной или нескольким группам, в контроллере эти группы будут автоматически созданы, а узел включен в каждую из них.

  Пусть файл инвентаря имеет следующий вид:

```
---
domain-controllers:
  # Контроллеры домена
  hosts:
    dc1.example.com:
    dc2.example.com:
clients:
  # Клиенты домена
  hosts:
    client-1.example.com:
    client-2.example.com:
    client-3.example.com:
    client-4.example.com:
```

  При импорте инвентаря в нем будут автоматически созданы группы `domain-controllers` и `clients`, содержащие контроллеры домена и клиенты домена соответственно.

- Если в источнике группы узлов организованы иерархически, структура групп будет воссоздана при импорте.

Важно: Узлы дочерних групп не связываются с родительской группой автоматически – их нужно добавлять вручную.

Например, пусть есть инвентарь следующего вида:

```
---
all:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  hosts:
    node-[01:10].example.com:
controllers:
  hosts:
    node-01.example.com:
    node-02.example.com:
clients:
  hosts:
    node-[03:10].example.com:
production:
  children:
    controllers:
    clients:
```

При импорте такого инвентаря будут созданы три группы – `production`, `clients` и `controllers`. Управляемые узлы будут связаны только с группами `clients` и `controllers`.

- По умолчанию при повторной синхронизации инвентарного списка с источником импортированные ранее управляемые узлы, группы узлов и переменные не удаляются.
- Поддерживается использование сценариев динамического создания инвентаря.

  Подробности о динамическом создании инвентаря см. в документации Ansible[118].
- Поддерживается синхронизация с источником по расписанию.

Поддержка расширений

Astra Automation Controller поддерживает следующие расширения инвентаря:

Тип источника	Ссылка на руководство
Amazon EC2	aws_ec2_inventory[119]
Google Compute Engine	gcp_compute_inventory[120]
Диспетчер ресурсов Microsoft Azure	azure_rm_inventory[121]
VMware vCenter	vmware_vm_inventory_inventory[122]
Red Hat Satellite 6	foreman_inventory[123]
OpenStack	oprenstack_inventory[124]
Виртуализация Red Hat	ovirt_inventory[125]
Платформа автоматизации Red Hat Tower	tower_inventory[126]
Red Hat Insights	insights_inventory[127]

[118] https://docs.ansible.com/ansible/latest/inventory_guide/intro_dynamic_inventory.html

Умный инвентарь

Умный инвентарь представляет собой список узлов, сформированный путем фильтрации всех управляемых узлов организации. При изменении списка управляемых узлов организации содержимое умного инвентарного списка обновляется автоматически, так как он хранит не копии сведений об управляемых узлах, а лишь ссылки на них.

Предупреждение: Умный инвентарь считается устаревшим и не поддерживается. В одной из следующих версий Astra Automation Controller этот тип инвентаря станет недоступным для применения. В качестве замены следует использовать *сборный инвентарь*.

Умный инвентарь имеет следующие особенности:

- Использование переменных Ansible не поддерживается.
- Группы узлов импортируются из источников, но доступны только для просмотра – создать новые группы или изменить существующие нельзя.

Базовые настройки фильтрации узлов

В самом простом случае запрос на выборку узлов формируется путем одновременного применения одного или нескольких фильтров из указанного списка:

- `enabled` (Enabled / Подключен) – фильтр по статусу управляемого узла.

  Если значение фильтра равно `true`, отбираются только узлы со статусом **Вкл** (On).

- `group__name__icontains` (Group / Группа) – фильтр по названию группы узлов.

  Отбираются узлы, которые принадлежат группе, в названии которой есть указанная подстрока. Поиск группы по названию производится без учета регистра.

- `instance_id` (Instance ID / Идентификатор узла контроллера) – фильтр по идентификаторам узлов контроллера.

- `inventory_id` (Inventory ID / Идентификатор инвентаря) – фильтр по идентификаторам инвентарных списков.

  Отбираются узлы, входящие в инвентарные списки с указанными идентификаторами.

- `last_job` (Last Job / Последнее задание) – фильтр по идентификатору последнего выполненного на узле задания.

  Отбираются узлы, последнее выполненное на которых задание имеет идентификатор из указанного списка.

[119] https://docs.ansible.com/ansible/latest/collections/amazon/aws/aws_ec2_inventory.html
[120] https://docs.ansible.com/ansible/latest/collections/google/cloud/gcp_compute_inventory.html
[121] https://docs.ansible.com/ansible/latest/collections/azure/azcollection/azure_rm_inventory.html
[122] https://docs.ansible.com/ansible/latest/collections/community/vmware/vmware_vm_inventory_inventory.html
[123] https://docs.ansible.com/ansible/latest/collections/theforeman/foreman/foreman_inventory.html
[124] https://docs.ansible.com/ansible/latest/collections/openstack/cloud/openstack_inventory.html
[125] https://docs.ansible.com/ansible/latest/collections/ovirt/ovirt/ovirt_inventory.html
[126] https://docs.ansible.com/ansible/latest/collections/awx/awx/tower_inventory.html
[127] https://github.com/RedHatInsights/insights-host-inventory

- `name__icontains` (Name / Название) – фильтр по названию узла.

  Отбираются узлы, в названии которых есть указанная подстрока. Поиск подстроки производится без учета регистра.

Пусть имеется следующий список управляемых узлов:

```
---
controllers:
  hosts:
    dc1.example.com:
    dc2.example.com:
clients:
  hosts:
    client-1.example.com:
    client-2.example.com:
    client-3.example.com:
    client-4.example.com:
```

Необходимо создать умный инвентарь, в который входят только узлы клиентов домена. С помощью умного инвентаря выполнить эту задачу можно различными способами, например:

- Создать умный инвентарь с фильтром `name__icontains` со значением `client-`.
- Создать умный инвентарь с фильтром `group__name__icontains` со значением `clients`.
- Создать умный инвентарь сразу с двумя указанными выше фильтрами.

Расширенные настройки фильтрации узлов

Умный инвентарь поддерживает расширенные настройки фильтрации узлов. В этом случае выражение поиска строится по следующей схеме:

```
<type>__<key>__<search_type> <value>
```

Здесь:

- <type> – способ обработки нескольких условий фильтрации.

  Поддерживаются следующие значения:

  - `or` – это необязательное условие;
  - and – это обязательное условие.

- <key> – название поля в свойствах узла или одного из связанных с ним ресурсов.

  Поддерживаются следующие значения:

  - `ad_hoc_command_events` – название событий, произошедших на управляемых узлах при выполнении специальных (ad-hoc) команд.
  - `ad_hoc_commands` – название специальной команды.
  - `ansible_facts_modified` – дата и время последнего обновления фактов Ansible.

    Совет: Как правило, факты Ansible об управляемом узле обновляются при выполнении заданий.

  - `ansible_facts` – факт Ansible.
  - `created_by` – имена пользователей, создавших записи об управляемых узлах.
  - `created` – дата и время создания записи об управляемом узле.
  - `description` – описание управляемого узла.
  - `enabled` – выбор узлов по статусу в Astra Automation Controller.
  - `groups` – названия групп узлов.
  - `id` – идентификатор управляемого узла.
  - `inventory_sources` – источник инвентаря.
  - `inventory` – идентификатор инвентаря, из которого должны быть выбраны управляемые узла.
  - `jobs` – список идентификаторов заданий, выполненных на управляемом узле.
  - `last_job_host_summary` – статус выполнения последнего задания, выполненного на управляемом узле.
  - `last_job` – идентификатор последнего задания, выполненного на управляемом узле.
  - `modified_by` – имена пользователей, последними изменявших данные об управляемом узле.
  - `modified` – дата и время последнего изменения записи об управляемом узле.
  - `name` – название управляемого узла.
  - `smart_inventories` – названия умных инвентарных списков.
  - `variables` – фильтр узлов по названиям указанных в их данных переменных.
- <search_type> – обработка значения, используемого в качестве фильтра.

  Поддерживаются следующие способы обработки указанного значения:

  - `exact` – точное совпадение с учетом регистра.

    Это значение используется по умолчанию.
  - `gt` – проверка условия «больше чем», >.
  - `gte` – проверка условия «больше или равно», ≥.
  - `in` – проверка на присутствие в указанном списке.
  - `iregex` – поиск по регулярному выражению без учета регистра.
  - `isnull` – проверка на пустое значение.
  - `lt` – проверка условия «меньше чем», <.
  - `lte` – проверка условия «меньше или равно», ≤.
  - `regex` – поиск по регулярному выражению с учетом регистра.
- <value> – значение, используемое в качестве фильтра.

  Особенности ввода значений фильтрации:

  - Для ввода дробных чисел в качестве разделителя следует использовать символ точки, например, `3.1415926`.
  - Для ввода дат следует использовать формат `YYYY-MM-DD`, где:

    * YYYY – четырехзначное значение года, например, 2024.
    * MM – двузначеное значение месяца. Отсчет месяцев начинается с 01.
    * DD – двузначное значение числа. Отсчет чисел начинается с 01.
  - Для ввода времени следует использовать формат HH:MM:ssuuuuuuTZ, где:
    * HH – двузначное значение часа в 24-часовом формате, например, 07.
    * MM – двузначное значение минут, от 00 до 59.
    * (опционально) ss – двузначное значение секунд, от 00 до 59.
    * (опционально) uuuuuu – микросекунды.
    * (опционально) TZ – название временной зоны.

Сборный инвентарь

Сборный инвентарь представляет собой список записей об управляемых узлах, сведения о которых получены путем копирования из обычных инвентарных списков. В этом документе обычные инвентарные списки, из которых копируются данные, называются источниками.

Сборный инвентарь имеет следующие особенности:

- Для формирования сборного инвентаря требуется хотя бы один источник.
- Сборный инвентарь хранит **копии** записей об управляемых узлах и группах управляемых узлов, а не ссылки на них. Это значит, что при обновлении источников сборный инвентарь не обновляется автоматически.
- Существующие группы узлов импортируются из источника независимо от того, будет ли в них хотя бы один узел после выполнения условий отбора узлов.
- Новые группы узлов создаются и заполняются автоматически в соответствии с параметрами, заданными в настройках расширения Ansible для работы с инвентарем.
- Сведения об управляемых узлах копируются в сборный инвентарь при синхронизации. Для первичного заполнения сборного инвентаря ее необходимо запустить вручную.

Фильтрация узлов

Параметры отбора управляемых узлов задаются в переменных source_vars и limit.

- source_vars – параметры, определяющие название используемого расширения Ansible и правила его использования.

  Примечание: Для создания сборного инвентаря в Astra Automation Controller используется расширение Ansible ansible.builtin.constructed[128].

- limit – правила выборки записей из существующих или временных групп управляемых узлов, сформированных при обработке значения переменной source_vars.

[128] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/constructed_inventory.html

Важно: Для использования параметра `limit` требуется версия контроллера не ниже `1.0-upd2` и версия среды исполнения Control Plane Execution Environment не ниже `0.5.1`.

В самом простом случае сборный инвентарь формируется путем копирования сведений обо всех управляемых узлах из источников. Значение переменной `source_vars` в этом случае задается следующим образом:

```
---
plugin: ansible.builtin.constructed
```

Здесь `plugin` – переменная, в которой указывается название расширения Ansible, используемого для работы с инвентарем. Это обязательная запись, значение которой должно быть равно `ansible.builtin.constructed` или `constructed`.

Примечание: Рекомендуется использовать полное название расширения – `ansible.builtin.constructed`.

В настройки фильтрации может быть добавлен логический параметр `strict`, значения которого интерпретируются следующим образом:

- `true` – считать любую ошибку обработки сведений об управляемых узлах критической и останавливать создание или обновление сборного инвентарного списка;
- `false` – пропускать управляемые узлы, при обработке сведений о которых возникли ошибки.

Примеры

В рассматриваемых далее примерах в качестве источника используются следующий инвентарь:

```
[controllers]
dc1.example.com
dc2.example.com      timezone=Novosibirsk
dc3.example.com      timezone=Chita
dc4.example.com      timezone=Chita        env=testing

[rubackup]
rb1.example.com
rb2.example.com
rb3.example.com      timezone=Chita
rb4.example.com                            env=testing

[clients]
client1.example.com
client2.example.com
client3.example.com  timezone=Novosibirsk
client4.example.com                        env=testing
client5.example.com  timezone=Novosibirsk  env=testing
client6.example.com  timezone=Chita
client7.example.com  timezone=Chita        env=testing
```

Здесь:

- controllers – группа узлов контроллера домена;
- rubackup – узлы кластера RuBackup;
- clients – клиенты домена;
- timezone – часовой пояс, в котором физически находится управляемый узел;
- env – окружение, к которому относится управляемый узел.

Пример 1

Пусть необходимо сформировать сборный инвентарь, узлы которого отбираются по следующим признакам:

- часовой пояс не задан или равен Moscow;
- окружение тестовое.

В этом случае параметры сборного инвентаря можно задать следующим образом:

- source_vars:

```
---
plugin: ansible.builtin.constructed
strict: true
groups:
  timezone_is_moscow: timezone | default("Moscow") == "Moscow"
  env_is_testing: env | default("production") == "testing"
```

- limit – timezone_is_moscow:&env_is_testing.

Пример 2

Пусть необходимо сформировать сборный инвентарь, узлы которого отбираются по следующим признакам:

- окружение не тестовое;
- не входит в группу rubackup.

В этом случае параметры сборного инвентаря можно задать следующим образом:

- source_vars:

```
---
plugin: ansible.builtin.constructed
strict: true
groups:
  env_is_not_testing: env | default("production") != "testing"
```

- limit – env_is_not_testing:!rubackup

Источники кода

Astra Automation Controller позволяет при создании проектов и заполнении инвентаря использовать код из внешних источников. В этом разделе описаны доступные типы источников кода и особенности работы с ними.

При создании проекта может быть использован один из указанных источников:

- ручное управление (локальный каталог);
- репозиторий Git;
- репозиторий Subversion;
- внешний архив.

Для хранения локальной копии кода каждого проекта Astra Automation Controller создает подкаталог в каталоге, используемом для хранения проектов (далее – каталог проектов). Путь к каталогу для хранения проектов хранится в параметре контроллера PROJECTS_ROOT. Его значение задается на этапе развертывания контроллера и по умолчанию равно `/var/lib/awx/projects/`.

При создании или удалении проекта с любым типом источника, кроме «Ручное управление», подкаталог проекта в каталоге проектов соответственно создается или удаляется автоматически.

Ручное управление

Для использования локального каталога в качестве источника кода проекта необходимо соблюдение следующих условий:

- код проекта размещен в одном из подкаталогов каталога проектов;
- владельцами каталога проекта являются пользователь и группа awx;
- на файлы и каталоги проекта установлены следующие режимы доступа:
  - файлы – `0744`;
  - каталоги – `0755`.

Репозиторий Git

Astra Automation Controller поддерживает использование репозиториев Git в качестве источников кода проектов. Основной параметр, который при этом должен быть указан – URL репозитория в *VCS*. В каталоге для хранения проектов создается подкаталог с локальной копией указанного репозитория.

URL репозитория в зависимости от используемого протокола подключения может иметь следующий вид:

- HTTPS:

```
https://example.org/project.git
```

- SSH

```
ssh://git@example.org/project.git
```

- GIT

```
git://example.org/project.git
```

Примечание: Поддержка протоколов и особенности их использования определяются возможностями сервиса, в котором размещен репозиторий. Для Astra Automation Hub доступны протоколы HTTPS и SSH.

Дополнительные настройки

Для репозитория Git можно указать дополнительные параметры системы управления исходными данными.

Название ветки, название тега или хэш коммита

По умолчанию используется последний коммит из основной ветки репозитория. Эта настройка позволяет использовать другие версии кода:

- Название ветки.

  Используется версия кода из последнего коммита в указанной ветке.

  Например, если в репозитории основная ветка называется `main`, а ветка с экспериментальными возможностями называется `development`, для использования кода из ветки `development` необходимо указать ее название в свойствах источника.

- Название тега.

  Используется версия кода, отмеченная указанным тегом.

  Номера версий коллекций Ansible в Astra Automation Hub указываются в названиях тегов. Чтобы использовать коллекцию `astra.nginx` версии `1.7.0`, соответствующее значение следует указать в настройках источника.

- Хэш коммита.

  Используется версия кода из коммита с указанным хэшем. Хэш коммита может быть указан полностью или частично (первые несколько символов).

  Для использования коммита с хэшем `d7e82ab6ebff598b928edd356803758a3f3435ce` допускается указать значение `d7e82ab6`.

- Произвольная ссылка на версию кода в локальной копии репозитория.

Важно: Некоторые хэши коммитов и ссылки могут быть недоступны без указания спецификации (refspec) для измененной ссылки.

Спецификация (refspec)

Эта настройка используется для управления ссылками в локальной копии репозитория на версии кода в удаленном репозитории.

Например, для получения из внешнего репозитория всех возможных ссылок в настройках спецификации необходимо указать следующую строку:

```
refs/*:refs/remotes/origin/*
```

Примечание: Для удаленного источника используется название `origin`.

Подробности использования спецификаций ссылок см. в документации Git[129].

Полномочия

Эта настройка позволяет указать полномочия, которые следует использовать для доступа к удаленному репозиторию. Для выбора доступны полномочия типа «Управление версиями».

Настройки управления версиями кода

В настройках Astra Automation Controller можно указать дополнительные параметры, управляющие поведением при работе с репозиторием.

- **Очистить**

  Перед выполнением обновления удаляются все сделанные изменения в локальной копии репозитория.

- **Удалить**

  При обновлении локальная копия репозитория удаляется полностью, после чего загружается заново.

  Примечание: В зависимости от размера репозитория и скорости соединения процесс загрузки может занять длительное время.

- **Отслеживание подмодулей**

  Если эта настройка включена, отслеживается состояние не только основного репозитория, но и имеющихся подмодулей. Использование этой настройки эквивалентно выполнению команд `git fetch` и `git clone` с параметром `--recurse-submodules`.

  Подробности об использовании подмодулей см. в документации Git[130].

- **Обновить версию при запуске**

  Если эта настройка включена, перед запуском любого задания, использующего код из этого источника, он будет автоматически обновлен.

[129] https://git-scm.com/book/ru/v2/Git-изнутри-Спецификации-ссылок
[130] https://git-scm.com/book/ru/v2/Инструменты-Git-Подмодули

- **Разрешить переопределение ветки**

  Если эта настройка включена, в шаблоне задания разрешается выбрать ветку или версию кода, отличную от заданной в свойствах проекта.

Репозиторий Subversion

Astra Automation Controller поддерживает использование репозиториев Subversion в качестве источников кода проектов. Основной параметр, который при этом должен быть указан – URL репозитория в *VCS*.

URL репозитория в зависимости от используемого протокола подключения может иметь следующий вид:

- HTTP(S):

```
https://example.org/project
```

- SVN

```
svn@example.org/project
```

- SVN+SSH

```
svn+ssh://example.org/project
```

Примечание: Поддержка протоколов и особенности их использования определяются возможностями сервиса, в котором размещен репозиторий.

Дополнительные настройки

Для репозитория Subversion можно указать дополнительные параметры:

- номер ревизии;
- полномочия на систему управления исходными данными.

Номер версии

По умолчанию используется самая свежая ревизия кода из основной ветки репозитория. Эта настройка позволяет использовать другие ревизии кода:

- Название ветки.

  Используется последняя ревизия кода в указанной ветке.

- Номер ревизии.

  Используется код из указанной ревизии.

Полномочия на систему управления исходными данными

Эта настройка позволяет указать полномочия типа «Управление версиями», которые следует использовать для доступа к удаленному репозиторию.

Настройки управления ревизиями кода

В настройках Astra Automation Controller можно указать дополнительные параметры, управляющие поведением при работе с репозиторием.

- **Очистить**

  Перед выполнением обновления удаляются все сделанные изменения в локальной копии репозитория.

- **Удалить**

  При обновлении локальная копия репозитория удаляется полностью, после чего загружается заново.

  Примечание: В зависимости от размера репозитория и скорости соединения процесс загрузки может занять длительное время.

- **Обновить версию при запуске**

  Если эта настройка включена, перед запуском любого задания, использующего код из этого источника, он будет автоматически обновлен.

- **Разрешить переопределение ветки**

  Если эта настройка включена, в шаблоне задания разрешается выбрать ветку или версию кода, отличную от заданной в свойствах проекта.

Внешний архив

В качестве источника кода Astra Automation Controller может использовать архивы, содержащие код самого проекта и его зависимостей. Основной параметр, который при этом должен быть указан – URL архива в источнике. Поддерживаются архивы форматов ZIP и TAR.GZ.

Доступ к архивам возможен по протоколам HTTP и HTTPS.

В каталоге для хранения проектов создается подкаталог, в который распаковывается содержимое загруженного архива.

Дополнительные настройки

Для внешнего архива доступны следующие дополнительные настройки:

- Полномочия

  Эта настройка позволяет указать полномочия, которые следует использовать для доступа к архиву в источнике. Для выбора доступны полномочия типа «Управление версиями».

- **Очистить**

  Перед выполнением обновления удаляются все сделанные изменения в локальной копии репозитория.

- **Удалить**

  При обновлении локальная копия репозитория удаляется полностью, после чего загружается заново.

  Примечание: В зависимости от размера репозитория и скорости соединения процесс загрузки может занять длительное время.

- **Обновить версию при запуске**

  Если эта настройка включена, перед запуском любого задания, использующего код из этого источника, он будет автоматически обновлен.

- **Разрешить переопределение ветки**

  Если эта настройка включена, в шаблоне задания разрешается выбрать ветку или версию кода, отличную от заданной в свойствах проекта.

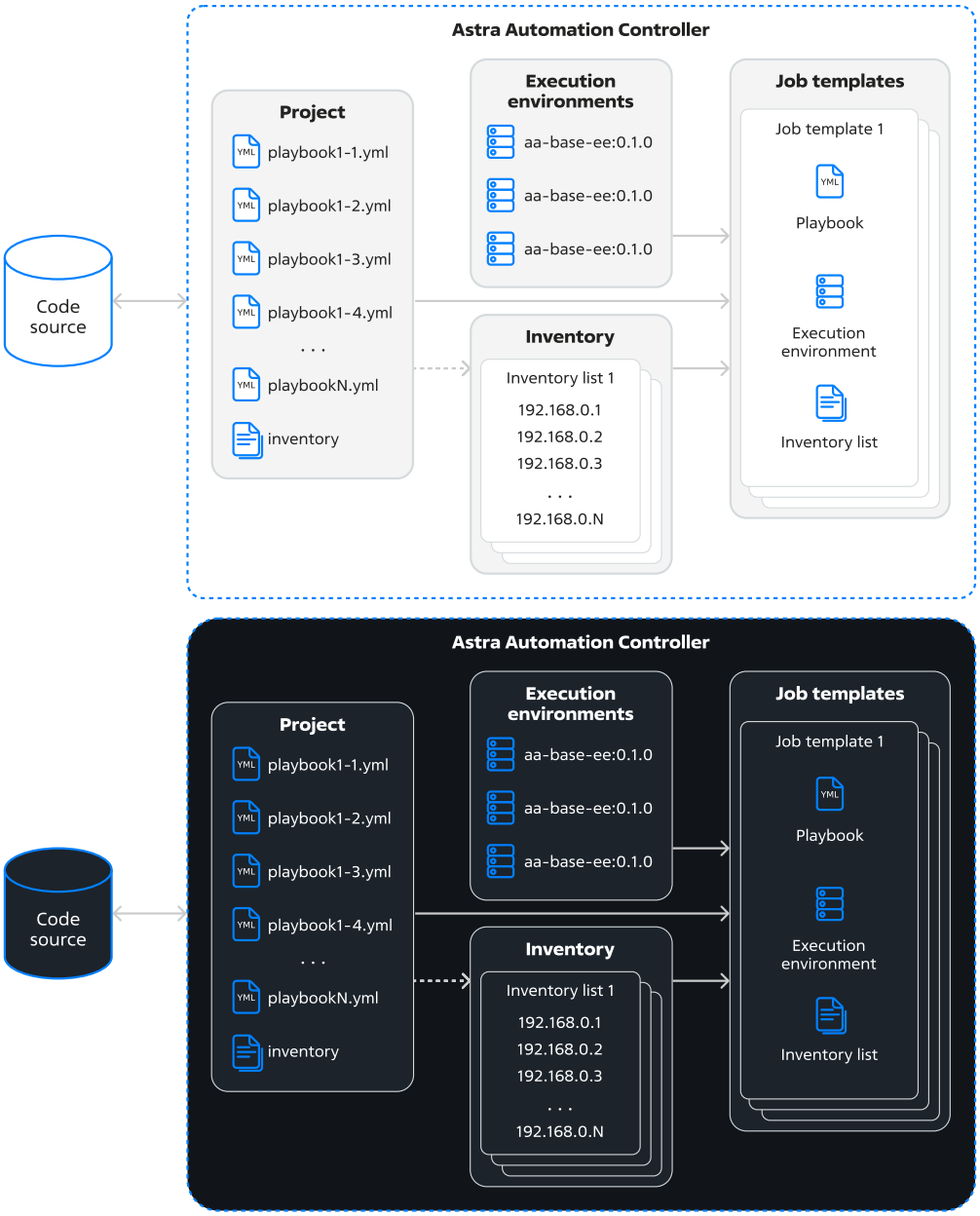
Проекты

В Astra Automation Controller проект – это набор Ansible playbook. Также проект может содержать инвентарные списки, зависимости Ansible и другие файлы.

Ansible playbook из проектов используются при создании шаблонов заданий.

Файлы инвентаря из проектов можно использовать в качестве источников при заполнении *обычных инвентарных списков*.

Взаимодействие проектов с другими компонентами контроллера показано на схеме:



Проект как источник файлов инвентаря

Astra Automation Controller позволяет использовать проекты для заполнения обычных инвентарных списков.

Поддерживается импорт инвентарных списков из отдельных файлов и каталогов.

Дополнительные параметры проекта

В настройках проекта можно указать дополнительные параметры:

- Среда исполнения.

  Если при создании шаблона не будет выбрана другая среда исполнения, то вместо среды исполнения по умолчанию будет использоваться значение, указанное в настройках проекта.

- Учетные данные для проверки подписи содержимого.

  Указанное полномочие типа «*GPG Public Key*» используется для проверки аутентичности исходного кода проекта.

  Если хотя бы один из подписанных файлов не пройдет проверку, обновление кода проекта будет завершено со статусом **Отказ** (Fail). Также контроллер заблокирует запуск всех заданий на основе шаблонов, использующих код проекта.

- URL системы управления исходными данными.

  Эта настройка доступна для проектов со следующими источниками управления исходными данными:

  - Git
  - Subversion
  - Внешний архив

  Для источников типа «Git» и «Subversion» необходимо указать URL репозитория.

Шаблоны

В этом разделе рассматриваются шаблоны заданий (Job Template) и шаблоны потоков заданий (Workflow Template).

Шаблоны заданий

Шаблон задания (Job Template) связывает между собой *инвентарь* и Ansible playbook из выбранного *проекта*.

Для шаблонов заданий поддерживаются следующие дополнительные параметры:

- тип задания (job type);
- среда исполнения (execution environment);
- полномочия (credentials);
- метки (labels);
- переменные (variables);
- ответвления (forks);
- лимит на управляемые узлы (limit);
- степень подробности вывода (verbosity);
- деление на срезы (job slicing);
- таймаут (timeout);

- показ изменений (show changes);
- теги задания (job tags) и пропуск тегов (skip tags);
- повышение привилегий (privilege escalation);
- параллельные задания (concurrent jobs);
- обратные вызовы процесса обеспечения сервиса (provisioning callbacks);
- хранилище фактов (fact storage);
- запрет отката группы узлов управления (prevent instance group fallback);
- настройки webhook (webhook).

Для некоторых параметров доступна опция **Запрос при запуске**. Если она включена, значение для связанного параметра можно задать позже:

- Если задание на основе шаблона запускается вне потока заданий – в момент запуска задания.
- Если шаблон задания используется в шаблоне потока заданий – в момент добавления шаблона задания в шаблон потока заданий.

Тип задания

Поддерживаются задания двух типов:

- Исполнение (Run) – на управляемых узлах будут выполнены необходимые задания автоматизации. Этот тип заданий используется по умолчанию.
- Проверка (Check) – выполняется проверка синтаксиса playbook, настроек среды исполнения и окружения, в котором будут выполняться задания автоматизации. Изменения в конфигурации управляемых узлов не производятся.

  Использование этого значения эквивалентно запуску команды `ansible-playbook` с параметром `--check`.

Среда исполнения

В настройках шаблона задания можно выбрать *среду исполнения*, используемую для запуска заданий. Эта среда исполнения имеет более высокий приоритет, чем заданная на уровне контроллера, организации и проекта.

Полномочия

Полномочия, используемые для доступа к управляемым узлам.

Поддерживается одновременное использование полномочий разных типов, но не более одной записи каждого типа.

Метки

С помощью меток можно группировать задания, созданные на основе шаблона. Метки также удобно использовать для поиска нужных записей журнала выполнения заданий или их логической группировки.

Переменные

Переменные шаблона задания позволяют задать значения переменных во время выполнения заданий.

Использование этого параметра эквивалентно запуску команды `ansible playbook` с одним или несколькими аргументами `--extra-vars` (`-e`).

В переменных шаблона задания используется формат `ключ:значение`.

Поддерживаются форматы YAML и JSON.

Подробности о порядке разрешения значений переменных см. в разделе *Переменные*.

Ответвления

Количество параллельных процессов, используемых для выполнения playbook.

Если для этого параметра задано значение меньше 1, используется значение по умолчанию, равное 5.

Максимальное значение этого параметра задается в значении *системной настройки контроллера* **Максимальное количество ответвлений задания** (Maximum number of forks per job) и по умолчанию равно 200.

Лимит на управляемые узлы

Фильтр управляемых узлов по их названию.

Если значение не указано, либо равно `all` или `*`, действие playbook распространяется на все узлы из выбранного инвентарного списка.

Значение по умолчанию – пустая строка (фильтрация узлов не используется).

Подробности о шаблонах названий узлов см. в документации Ansible[131].

Степень подробности

Детализация журнала выполнения задания:

- `0` – нормальный (normal);
- 1 – подробный (verbose);
- 2 – более подробный (more verbose);
- 3 – отладка (debug);
- 4 – отладка подключения (connection debug);

[131] https://docs.ansible.com/ansible/latest/inventory_guide/intro_patterns.html

- 5 – отладка WinRM (WinRM debug).

Значение по умолчанию – `0`.

Деление на срезы

Деление на срезы используется для параллельного запуска одного и того же задания на основе шаблона на нескольких исполняющих узлах.

Исходный инвентарь делится на несколько частей. Каждый исполняющий узел запускает playbook только для своей части исполняемых узлов.

По умолчанию значение этого параметра равно 1 – все задания из playbook запускаются на одном исполняющем узле.

Важно: Рекомендуется использовать значения, не превышающие количество исполняющих узлов.

Таймаут

Время в секундах на выполнение задания. Если в течение указанного времени задание не выполняется, оно отменяется автоматически.

При значении `0` используется таймаут, заданный в системных настройках контроллера (значение по умолчанию – `0`, время выполнения заданий не ограничивается).

Отрицательное значение разрешает неограниченно долгое выполнение задания.

Значение по умолчанию – `0` (используется значение, заданное в глобальных настройках контроллера).

Показ изменений

Отображение изменений в конфигурации управляемых узлов, сделанных в результате выполнения заданий автоматизации.

Использование этого параметра эквивалентно запуску команды `ansible-playbook` с параметром `--diff`.

Примечание: Для отображения списка изменений необходима поддержка со стороны используемых модулей Ansible.

Теги

Эта настройка позволяет запускать или пропускать описанные в playbook задачи, отмеченные определенными тегами.

По умолчанию фильтрация по тегам не используется – выполняются все задачи, описанные в playbook.

Особенности настройки и применения:

- Названия тегов задаются одной строкой через запятую.
- Если заданы теги задания, то при запуске задания будут выполнены только задачи, отмеченные указанными тегами.

  Использование этого параметра эквивалентно запуску команды `ansible-playbook` с параметром `--tags`.
- Если заданы пропускаемые теги, то при запуске задания задачи с указанными тегами будут пропущены.

  Использование этого параметра эквивалентно запуску команды `ansible-playbook` с параметром `--skip-tags`.

Подробности о тегах см. в документации Ansible[132].

Повышение привилегий

Если эта настройка включена, playbook запускается с повышенными привилегиями.

Использование этого параметра эквивалентно запуску команды `ansible-playbook` с параметром `--become`.

Важно: Запрос пароля суперпользователя не производится. Убедитесь, что на управляемых узлах разрешено выполнение команды `sudo` без ввода пароля.

Инструкции по настройке приведены в разделе *Использование sudo без ввода пароля*.

Параллельные задания

Если эта настройка включена, на основе шаблона можно запустить одновременное выполнение нескольких заданий с одним и тем же инвентарным списком.

[132] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_tags.html

Обратные вызовы процесса обеспечения сервиса

Если эта настройка включена, необходимо указать ключ конфигурации узла. Это URL, по которому узел может связаться с Astra Automation Controller и запросить обновление конфигурации, используя шаблон задания.

Хранилище фактов

Если эта настройка включена, при запуске задания собранные об узлах *факты Ansible* сохраняются в кэше фактов. Затем собранные факты можно использовать, например, для создания *сборного инвентаря*.

Настройки webhook

Если эта настройка включена, запуск задания на основе шаблона будет происходить при наступлении определенного события в сервисе хранения исходного кода.

При включении этой настройки необходимо задать значения дополнительных параметров:

- Сервис – сервис хранения исходного кода из списка: - Bitbucket Data Center; - GitHub; - GitLab.
- Webhook URL – адрес, по которому доступен нужный webhook. Поле заполняется автоматически.
- Ключ webhook – ключ, используемый для подписи данных, отправляемых в Astra Automation Controller. Значение из этого поля следует указать в параметрах webhook используемого сервиса хранения исходного кода.
- Полномочия webhook – полномочия, используемые для передачи данных обратно в webhook.

Шаблоны потоков заданий

Шаблоны потоков заданий (workflow templates) используются для реализации сложной логики управления выполнением заданий. Они поддерживают следующую функциональность:

- Запуск заданий при выполнении определенных условий[133] без необходимости их описания в playbook.
- Обновление кода проектов и инвентарных списков.
- Запуск заданий и потоков заданий только после одобрения пользователей (согласование).
- Параллельный запуск нескольких потоков заданий из одного потока.
- Синхронизация заданий, выполняемых параллельно.
- Запуск потоков заданий по расписанию.

[133] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html

- Рекурсивное использование шаблонов потоков заданий.

  По отношению к родительскому шаблону потока заданий такие шаблоны потоков заданий называются вспомогательными. Артефакты, собранные во время выполнения заданий вспомогательного шаблона, могут быть переданы в следующие узлы.

Шаблон потока заданий представляет собой направленный ациклический граф (*DAG*), начинающийся с корневого узла «НАЧАЛО» (START). Этот узел нельзя изменить или удалить.

Узлами графа могут выступать следующие компоненты и процессы:

- Согласование (approval).

  При переходе к узлу этого типа пользователь с ролью «Одобрить» (Approval) должен подтвердить или отклонить выполнение дальнейших действий.

- Обновление *инвентаря* (inventory update).

  Если инвентарь использует внешние источники, при переходе к узлу этого типа выполняется обновление сведений об управляемых узлах.

- *Шаблон задания* (job template).

  При переходе к узлу этого типа запускается задание на основе указанного шаблона.

- Обновление *проекта* (project update).

  При переходе к узлу этого типа выполняется обновление проекта, если его источником являются репозиторий Git, репозиторий Subversion или внешний архив.

- Шаблон потока заданий (workflow template).

  В роли узлов этого типа могут выступать уже существующие шаблоны потоков заданий.

- Служебное задание (management job).

  Служебные задания используются для выполнения действий, связанных с внутренним функционированием контроллера. Для узлов этого типа поддерживаются следующие служебные задания:

  - Cleanup Activity Stream – удаление записей в ленте активности;
  - Cleanup Expired OAuth 2 Tokens – очистка токенов OAuth 2 с истекшим сроком действия;
  - Cleanup Expired Sessions – очистка истекших сессий пользователей контроллера;
  - Cleanup Job Details – очистка сведений о выполнении задания.

Ветви графа могут быть одного из трех типов:

- успешное выполнение задания или согласование;
- ошибка при выполнении задания, его отмена или отказ в согласовании;
- безусловный переход, когда результат выполнения предыдущих этапов не имеет значения.

Особенности работы с инвентарем

Инвентарь может быть задан во время создания шаблона потока заданий или во время запуска потока заданий.

Если в настройках шаблона потока заданий включен выбор инвентаря при запуске, инвентарь может быть задан в настройках планировщика или узлов.

Если инвентарь указан в настройка шаблона задания, то значение из настроек шаблона потока заданий игнорируется.

Если в настройках шаблона задания включен выбор инвентаря при запуске, то используется инвентарь, заданный на уровне шаблона потока заданий.

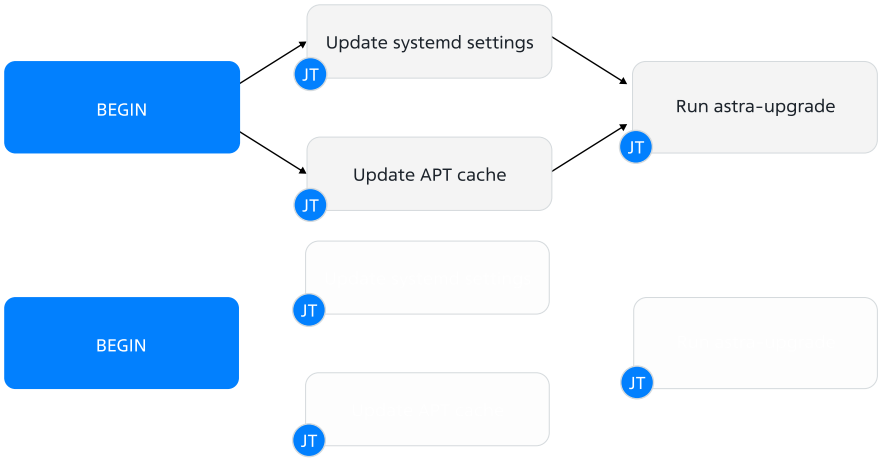
Конвергенция

Если узел имеет несколько входящих ветвей, настройки конвергенции определяют количество выполненных условий, необходимых для запуска задания узла. Поддерживаются следующие значения:

- **Любой** – достаточно успешного выполнения заданий на любом связанном узле;
- **Все** – должны быть успешно выполнены все условия на всех связанных узлах.

Пример

Пусть имеется шаблон потока заданий, показанный на схеме:



Узлы «Update systemd settings» и «Update APT cache» выполняют задания по обновлению настроек systemd и обновлению кэша APT соответственно. При этом указанные задания выполняются параллельно.

Задание «Run astra-upgrade» следует запускать только при успешном выполнении обоих предшествующих заданий. Для этого в настройках конвергенции следует выбрать значение «Все».

Важно: Порядок вычисления значения переменной `set_stats`, хранящей сведения о результатах выполнения задания, не определен. По этой причине для хранения сведений

о результатах выполнения заданий рекомендуется использовать уникальные названия ключей.

Дополнительные переменные (extra_vars)

Значения переменных могут быть заданы следующими способами:

- Определение переменных и их значений в свойствах шаблона потока заданий.
- Интерактивный опрос пользователя при запуске потока заданий.
- Дополнительные переменные, переданные в момент запуска потока заданий.

Подробности о порядке разрешения значений переменных см. в разделе *Переменные*.

Состояния

Шаблоны потоков заданий могут находиться в одном из следующих состояний:

- Ожидание (Waiting) – для запуска или продолжения выполнения заданий необходимо согласование.
- Выполнение (Running) – выполняются задания.
- Успех (Success) – выполнение задания завершено без ошибок.
- Отмена (Cancel) – выполнение задания отменено.
- Ошибка (Error) – при выполнении заданий произошли ошибки. Для обработки ошибок запущены соответствующие задания.
- Сбой (Failed) – при выполнении заданий произошли ошибки. Продолжение выполнения заданий невозможно.

Опросы

Опросом называется процесс получения значений переменных Ansible путем интерактивного взаимодействия с пользователем. Для этого пользователю, запустившему задание или поток заданий, предлагается ответить на ряд заранее подготовленных вопросов.

Состав вопросов и тип ожидаемых ответов определяется автором опроса.

Опросы обладают следующими свойствами:

- С каждым шаблоном может быть связан только один опрос.
- Каждый вопрос предусматривает получение ответа определенного типа.

  Поддерживаются следующие типы ответов:

  - **Текст** (Text) – строка символов.
  - **Текстовая область** (Textarea) – текст, состоящий из одной или более строк.
  - **Пароль** (Password) – защищенная строка символов.

    Для защиты ввода используется маскировка символов, для защиты значения в таблице базы данных контроллера – защитное преобразование.

  - **Множественный выбор (одиночное выделение)** (Multiple Choice (single select)) – выбор одного значения из предложенного списка.
  - **Множественный выбор (множественное выделение)** (Multiple Choice (multiple select)) – набор записей, выбранных из предложенного списка.
  - **Целое число** (Integer).
  - **Плавающее** (Float) – число с плавающей точкой, например, 3.1415926.
- Для каждого вопроса можно задать значение ответа по умолчанию.

Подробности о порядке разрешения значений переменных см. в разделе *Переменные*.

Задания и потоки заданий

В этом разделе рассматриваются задания и потоки заданий.

Задания

Задания (jobs) бывают следующих типов:

- запуск playbook (run playbook) – выполнение playbook, указанного в *шаблоне задания*;
- служебное (management jobs) – используется для обслуживания самого контроллера;
- обновление проекта (source control update);
- синхронизация инвентаря (inventory sync);
- команда (command) – выполнение специальных команд (ad-hoc) на выбранных управляемых узлах.

Все задания контроллер ставит в очередь. Задания из очереди выполняются в том порядке, в котором они были в нее добавлены.

Если в шаблоне заданий включено ветвление, для выполнения задания будет использоваться указанное количество параллельных процессов.

Если в шаблоне задания разрешено деление на срезы, связанный с заданием инвентарь делится на части, и задания для управляемых узлов в каждой из них запускаются параллельно.

Запуск заданий может быть привязан к определенной группе узлов контроллера из плоскости исполнения. Для этого в настройках организации, шаблона заданий, шаблона потока заданий или инвентаря необходимо выбрать группу узлов контроллера.

Потоки заданий

Потоки заданий создаются на основе *шаблонов потоков заданий*. При запуске потока заданий обход направленного ациклического графа начинается с точки «НАЧАЛО» (START).

Служебные задания

В контроллере доступны следующие служебные задания:

- Cleanup Activity Stream – очистка истории активности;
- Cleanup Expired OAuth 2 Tokens – очистка токенов OAuth с истекшим сроком действия;
- Cleanup Expired Sessions – очистка данных истекших сессий;
- Cleanup Job Details – очистка истории выполнения заданий.

При выполнении потока заданий некоторые служебные задания вызываются неявно.

Задания синхронизации

Задания синхронизации используются для обновления локальной копии playbook и инвентаря и называются, соответственно, заданиями обновления проекта и заданиями обновления инвентаря.

При выполнении задания обновления проекта контроллер получает версию кода, указанную в настройках *источника*:

- если не указаны ветка, тег, ревизия или хэш коммита – последний коммит в основной ветке проекта;
- если указана ветка – последний коммит в указанной ветке;
- если указан тег – коммит, к которому привязан тег;
- если указана ревизия – указанную ревизию кода.

Если инвентарь использует один из файлов проекта в качестве источника сведений об управляемых узлах, то его автоматическое обновление при обновлении кода проекта не происходит. Для обновления сведений об управляемых узлах необходимо запустить задание обновления инвентаря.

Задания обновления могут находиться в одном из следующих статусов:

- **Назначено** (Pending) – задание создано, но еще не поставлено в очередь и не запущено;
- **Ожидание** (Waiting) – задание поставлено в очередь, но еще не запущено;
- **Выполнение** (Running) – задание выполняется;
- **Успех** (Successful) – задание выполнено без ошибок либо с ошибками, для которых предусмотрен обработчик;
- **Сбой** (Failed) – во время выполнения задания произошли ошибки, обработка которых не предусмотрена.

Журналирование

Контроллер позволяет просматривать вывод выполнения заданий и специальных команд в реальном времени. Журнал выполнения каждого задания сохраняется в контроллере.

Управление узлами

Группы узлов

Узлы можно объединять в группы узлов (instance groups). Один и тот же узел может входить в несколько групп узлов.

При развертывании контроллера все гибридные и исполняющие узлы автоматически включаются в группу узлов по умолчанию с названием `default`. Название этой группы нельзя изменить.

Выбор группы узлов для запуска заданий

Группы узлов, доступные на уровне контроллера, называются глобальными. По умолчанию задания запускаются на наиболее свободной глобальной группе узлов. Если все глобальные группы узлов заняты, задание ставится в очередь и запускается на узлах первой освободившейся глобальной группы.

Группы узлов могут быть *ассоциированы* с *организациями* (через проекты), *инвентарными списками* и *шаблонами заданий*. В этом случае для запуска заданий используются только ассоциированные группы узлов. Если они будут заняты, контроллер не будет использовать для запуска заданий глобальные группы узлов, а поставит задание в очередь и запустит его на первой освободившейся группе узлов из ассоциированных.

Если группы узлов ассоциированы сразу на нескольких уровнях, выбор групп узлов для запуска заданий выполняется в следующем порядке:

1. шаблон задания;
2. инвентарный список;
3. организация.

Например, пусть в контроллере существуют следующие группы узлов:

- `default`;
- `south`;
- `north`;
- `east`;
- `west`.

Пусть эти группы ассоциированы следующим образом:

1. шаблон задания – `north`;
2. инвентарный список – `east` и `west`;
3. организация – `south`.

В этом случае выбор группы узлов для запуска задания выполняется в следующем порядке:

1. Если свободна хотя бы одна группа, ассоциированная с шаблоном задания, задания запускаются на ее узлах.

   В данном случае проверяется статус группы `north`. Если она занята, выполняется проверка статуса групп узлов, ассоциированных с инвентарным списком.

2. Если свободна хотя бы одна группа, связанная с инвентарным списком, задания запускаются на ее узлах.

   В данном случае проверяется статус групп `east` и `west`. Если обе группы заняты, выполняется проверка статуса групп узлов, связанных с организацией.

3. Если свободна хотя бы одна группа, связанная с организацией, задания запускаются на ее узлах.

   В данном случае проверяется статус группы `south`. Если группа `south` тоже занята, задание ставится в очередь и выполняется на первой освободившейся группе узлов, ассоциированной с шаблоном задания, инвентарным списком или организацией.

Запрет отката группы узлов управления

Запрет отката группы узлов изменяет порядок выбора групп узлов для запуска заданий. В зависимости от уровня, на котором он включен, выбор групп узлов выполняется следующим образом:

- Шаблон задания.

  Если шаблон задания ассоциирован с группами узлов, контроллер будет ожидать их освобождения. Если список ассоциированных групп узлов пуст, используются глобальные группы узлов. При этом игнорируются группы, ассоциированные с инвентарным списком и организацией.

- Инвентарный список.

  Если инвентарный список ассоциирован с группами узлов, контроллер будет ожидать их освобождения. Если список ассоциированных групп узлов пуст, используются глобальные группы узлов. При этом игнорируются группы, ассоциированные с организацией.

Переменные

В этом документе описывается порядок разрешения значений переменных Ansible в Astra Automation Controller.

Правило присвоения значения

При присвоении значений переменным используется следующий порядок (значения, указанные на более поздних этапах, заменяют значения, указанные ранее):

1. Значения по умолчанию, заданные в роли (role defaults).
2. Переменные динамического инвентаря (dynamic inventory variables).
3. Переменные инвентаря (inventory variables).
4. Переменные группы узлов (group variables).
5. Переменные управляемого узла (node variables).
6. `group_vars` в playbook.
7. `host_vars` в playbook.
8. Факты Ansible (facts).
9. Зарегистрированные переменные (registered variables).
10. Значения, заданные с помощью `set_facts`.
11. Переменные сценария (play variables)
12. Переменные из файлов переменных сценария (play `vars_files`).
13. Переменные роли (role variables) и подключенных (include) файлов.
14. Переменные блока (block variables).
15. Переменные задачи (task variables).
16. Дополнительные переменные из свойств шаблона задания (Job Template extra variables).
17. Переменные, полученные путем интерактивного опроса пользователя при запуске задания на основе шаблона (Job Template Survey). Если опрос пользователя выключен, используются значения по умолчанию.
18. Дополнительные переменные, заданные при запуске задания (Job Launch extra variables).
19. Артефакты задания.
20. Дополнительные переменные из свойств шаблона потока заданий (Workflow Job Template extra variables).
21. Переменные, полученные путем интерактивного опроса пользователя при запуске потока заданий на основе шаблона (Workflow Job Template Survey). Если опрос пользователя выключен, используются значения по умолчанию.
22. Дополнительные переменные, заданные при запуске потока заданий (Workflow Job Launch extra variables).

Пример

Пусть значение переменной `astra_linux_version` задано в нескольких местах:

- инвентарь – `1.7.1`;
- playbook – `1.7.5`;
- шаблон задания – `1.7.3`;
- шаблон потока заданий – `1.7.4`.

Шаблон потока заданий связывает вместе указанные инвентарь, playbook и шаблон задания. Согласно описанному выше порядку, для переменной `astra_linux_version` принимается значение `1.7.4`, заданное последним, на уровне шаблона потока заданий. Это же значение передается на уровни всех остальных компонентов, связанных с шаблоном потока заданий. Таким образом, при запуске задания и выполнении сценария для переменной `astra_linux_version` также будет использоваться значение `1.7.4`.

Уведомления

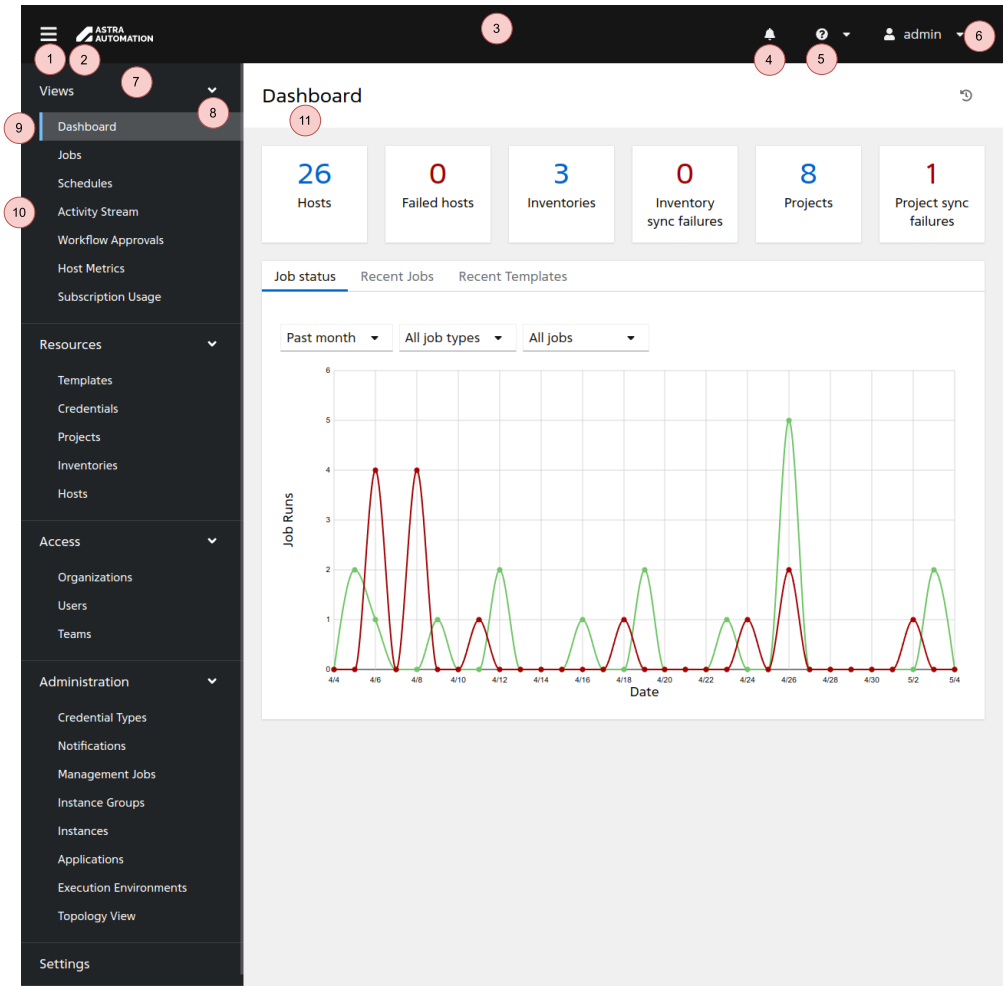
Важно: Эта часть документации находится в стадии разработки.

6.3 Графический интерфейс

В этом разделе приводятся инструкции по управлению Astra Automation Controller через веб-интерфейс. Порядок размещения разделов документации соответствует размещению разделов на панели навигации контроллера.

6.3.1 Основные компоненты

На рисунке представлены основные компоненты пользовательского интерфейса контроллера:



Здесь:

1. Кнопка переключения видимости панели навигации.
2. Логотип.
3. Панель инструментов.
4. Кнопка-индикатор, показывающая количество уведомлений.
5. Кнопка вызова справки.
6. Пользовательское меню.
7. Панель навигации.
8. Кнопка переключения видимости раздела навигации.
9. Активный раздел панели навигации.
10. Неактивные разделы панели навигации.
11. Заголовок окна.

6.3.2 Режимы просмотра

Раздел панели навигации **Режимы просмотра** (Views) содержит ссылки на окна, содержащие различные сведения о работе контроллера и выполнении заданий:

- **Информационная панель** (Dashboard) содержит краткую сводку о состоянии самых важных ресурсов контроллера – проектах, инвентарных списках и управляемых узлах.

  В верхней части окна размещаются панели:

  - **Управляемые узлы** (Hosts) – общее количество управляемых узлов во всех инвентарных списках.
  - **Узлы с ошибками** (Failed Hosts) – количество узлов, при выполнении заданий на которых произошли ошибки.
  - **Инвентарь** (Inventories) – количество существующих инвентарных списков.
  - **Ошибки синхронизации инвентаря** (Inventory Sync failures) – количество ошибок при выполнении заданий синхронизации инвентарных списков.
  - **Проекты** (Projects) – количество проектов.
  - **Проекты с ошибками синхронизации** (Project sync failures) – количество существующих проектов.

  Нажатие на панель приводит к переходу к соответствующему разделу на панели навигации.

  Нижнюю часть окна занимает панель с вкладками:

  - **Статус задания** (Job status) – график, отображающий количество выполненных заданий и заданий с ошибками.
  - **Последние задания** (Recent Jobs) – таблица со списком заданий, статусом их выполнения, временем начала и завершения.
  - **Последние шаблоны** (Recent Templates) – таблица со списком последних использованных *шаблонов заданий*.
- **Задания** (Jobs) содержит таблицу со списком запущенных заданий. Поддерживается фильтрация записей по различным критериям, запуск заданий напрямую из таблицы и переход к журналам выполнения заданий.
- **Расписание** (Schedules) содержит таблицу заданий, выполняемых по расписанию. Поддерживается фильтрация записей по различным критериям, управление запуском заданий и переход к редактированию их свойств.
- **Лента активности** (Activity Stream) содержит таблицу со списком событий, произошедших в контроллере. По умолчанию отображаются все произошедшие события. Поддерживается фильтрация записей по различным критериям и просмотр подробной информации о каждом событии.
- **Согласования потоков заданий** (Workflow Approvals) содержит таблицу согласований и кнопки для управления ими.
- **Метрики узлов** (Host Metrics) используется для отображения сведений об управляемых узлах и связанных событиях автоматизации.
- **Расход ресурсов подписки** (Subscription Usage) содержит график использования ресурсов подписки.

6.3.3 Ресурсы

Раздел панели навигации *Ресурсы* (Resources) содержит ссылки на разделы, используемые для управления следующими ресурсами:

- **Шаблоны** (Templates) – *шаблоны заданий* и *шаблоны потоков заданий*;
- **Полномочия** (Credentials) – полномочия, используемые для доступа к различным ресурсам, в том числе управляемым узлам, источникам инфраструктурного кода и облачным сервисам;
- **Проекты** (Projects) – *проекты*;
- **Инвентарь** (Inventories) – *инвентарные списки*;
- **Управляемые узлы** (Hosts) – сведения об управляемых узлах.

6.3.4 Доступ

Раздел панели навигации *Доступ* (Access) содержит ссылки на разделы, используемые для настройки разграничения доступа пользователей к ресурсам:

- **Организации** (Organizations)

  Управление *организациями*:

  - создание;
  - удаление;
  - назначение и отзыв ролей пользователям и командам.

- **Пользователи** (Users)

  Управление пользователями контроллера, в том числе создание учетных записей *системных администраторов и системных аудиторов*.

- **Команды** (Teams)

  Управление *командами*:

  - создание;
  - удаление;
  - назначение командных и индивидуальных ролей.

6.3.5 Администрирование

Раздел панели навигации *Администрирование* содержит ссылки на разделы, используемые для управления расширенными функциями контроллера:

- **Типы полномочий** (Credential Types) – управление нестандартными типами *полномочий*.
- **Уведомления** (Notifications) – настройка способов отправки оповещений о различных событиях.
- **Служебные задания** (Management Jobs) – настройка и запуск заданий по обслуживанию самого контроллера: очистка истории, истекших токенов, пользовательских сессий и истории выполнения заданий.

- **Группы узлов контроллера** (Instance Groups) – управление группами узлов контроллера.
- **Узлы контроллера** (Instances) – просмотр сведений о состоянии отдельных узлов контроллера и запуск диагностических тестов.
- **Приложения** (Applications) – управление приложениями.
- **Среды исполнения** (Execution Environments) – управление *средами исполнения*.
- **Топология** – отображение в реальном времени состояния узлов, используемых контроллером.

6.3.6 Настройки

Раздел *Настройки* используется для управления общими настройками контроллера:

- Аутентификация с помощью внешних провайдеров.
- Общие настройки выполнения заданий.
- Общесистемные настройки.
- Настройки пользовательского интерфейса.
- Отладка работы контроллера.

6.3.7 Запрос при запуске

Для некоторых полей доступна опция **Запрос при запуске** (Prompt on launch). Если соответствующий флаг включен, связанное поле меняет свое поведение следующим образом:

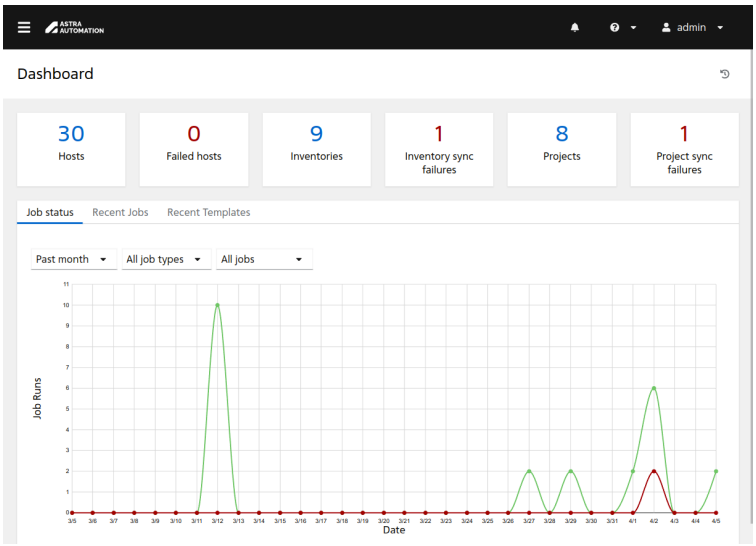
- Обязательное поле становится не обязательным для заполнения прямо сейчас. Однако, его все равно нужно будет заполнить позже.
- Если поле заполнено, указанное значение используется как значение по умолчанию.

Режимы просмотра

В этом разделе описывается просмотр сведений о состоянии заданий.

Информационная панель

Окно **Информационная панель** (Dashboard) используется для вывода сводной информации об управляемых узлах, статусах запущенных заданий, заданий синхронизации инвентаря и заданий синхронизации проектов.



Окно состоит из панелей и группы вкладок.

Панели, слева направо:

- **Управляемые узлы** (Hosts) – общее количество управляемых узлов.

  При расчете значения учитываются в том числе управляемые узлы, находящиеся в *статусе* «Выкл» (Off).

- **Узлы с ошибками** (Failed Hosts) – количество управляемых узлов, для которых статус выполнения последнего связанного задания – «Отказ» (Failed).

- **Инвентарь** (Inventories) – количество *инвентарных списков* всех типов.

- **Ошибки синхронизации инвентаря** (Inventory sync failures) – количество инвентарных списков, в которых выполнение задания синхронизации списка узлов с источником завершилось ошибкой.

- **Проекты** (Projects) – количество *проектов*.

- **Проекты с ошибками синхронизации** (Project sync failures) – количество проектов, задание обновления которых завершилось ошибкой.

Нажатие на любую панель приводит к переходу в соответствующее окно.

Во вкладках под панелями отображается следующая информация:

- **Статус задания** (Job Status) – график запуска заданий и статус их выполнения за последний месяц.

  График отображает количество запущенных заданий за каждые сутки. Сведения о количестве успешно выполненных заданий отображаются зеленым цветом, а о количестве заданий с ошибками или отказами – красным.

  Поддерживаются следующие настройки отображения:

  - Размер периода:
    - **Последний месяц** (Past month);
    - **Последние две недели** (Past two weeks);
    - **Последняя неделя** (Past week);
    - **Последние 24 часа** (Past 24 hours).

  - Тип заданий:
    - **Все типы заданий** (All job types);
    - **Синхронизация инвентаря** (Inventory sync);
    - **Обновлением SCM** (SCM update);
    - **Запуск playbook** (Playbook run).
  - Статус заданий – может принимать следующие значения:
    - **Все задания** (All jobs);
    - **Успешные задания** (Successful jobs);
    - **Задания с ошибками** (Failed jobs).

  При изменении значения любого из параметров график обновляется автоматически.

- **Последние задания** (Recent Jobs) – информация о последних запущенных заданиях.

  Таблица состоит из следующих колонок:

  - Переключатель подробности вывода основных сведений о задании.

    Набор отображаемых полей зависит от типа задания.
  - Флаги для выбора нескольких записей.
  - **Название** (Name) – ссылка для перехода в окно просмотра журнала выполнения задания или графа потока заданий.
  - **Статус** (Status)
    - **Успех** (Successful) – не было ошибок;
    - **Ошибка** (Error) – возникли ошибки, которые были успешно обработаны;
    - **Сбой** (Failed) – возникли необрабатываемые ошибки, выполнение задания прервано;
    - **Отменено** (Canceled) – задание отменено пользователем;
    - **Ожидание** (Pending) – задание находится в очереди;
    - **Выполнение** (Running) – задание выполняется.
  - **Время начала** (Start Time) – время запуска задания.
  - **Время завершения** (Finish Time) – время завершения, отмены или прерывания задания.
  - **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
    - отмена задания;
    - перезапуск задания.

- **Последние шаблоны** (Recent Templates) – информация о последних использованных шаблонах заданий и шаблонах потоков заданий.

  Таблица состоит из следующих колонок:

  - Переключатель подробности вывода основных сведений о шаблоне.

    Набор отображаемых полей зависит от типа шаблона.

- Флаги для выбора нескольких записей.
- **Название** (Name) – ссылка для перехода в окно просмотра подробных сведений о шаблоне задания или шаблоне потока заданий.
- **Тип** (Type) – тип шаблона.
- **Организация** (Organization) – ссылка для перехода в окно просмотра подробных сведений об организации, которой принадлежит шаблон.
- **Последний запуск** (Last Ran) – дата и время последнего запуска заданий на основе шаблона заданий или потока заданий.
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - переход в окно редактирования графа шаблона потока заданий (эта кнопка доступна только для шаблонов потоков заданий);
  - запуск задания на основе шаблона задания или потока заданий на основе шаблона потока заданий;
  - переход в окно редактирования свойств шаблона;
  - копирование шаблона.

Задания

Окно **Задания** (Jobs) используется для просмотра сведений о запущенных и выполненных заданиях. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Режимы просмотра* ▸ *Задания* (*Views* ▸ *Jobs*).

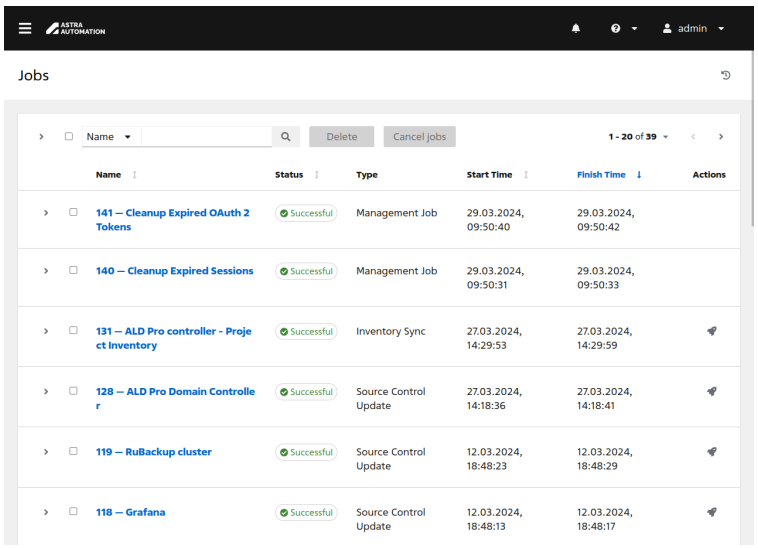


Таблица заданий состоит из следующих колонок:

- Переключатель подробности вывода основных сведений о задании.

  При нажатии на переключатель выводится дополнительная информация о задании:

Примечание: Набор полей зависит от типа задания.

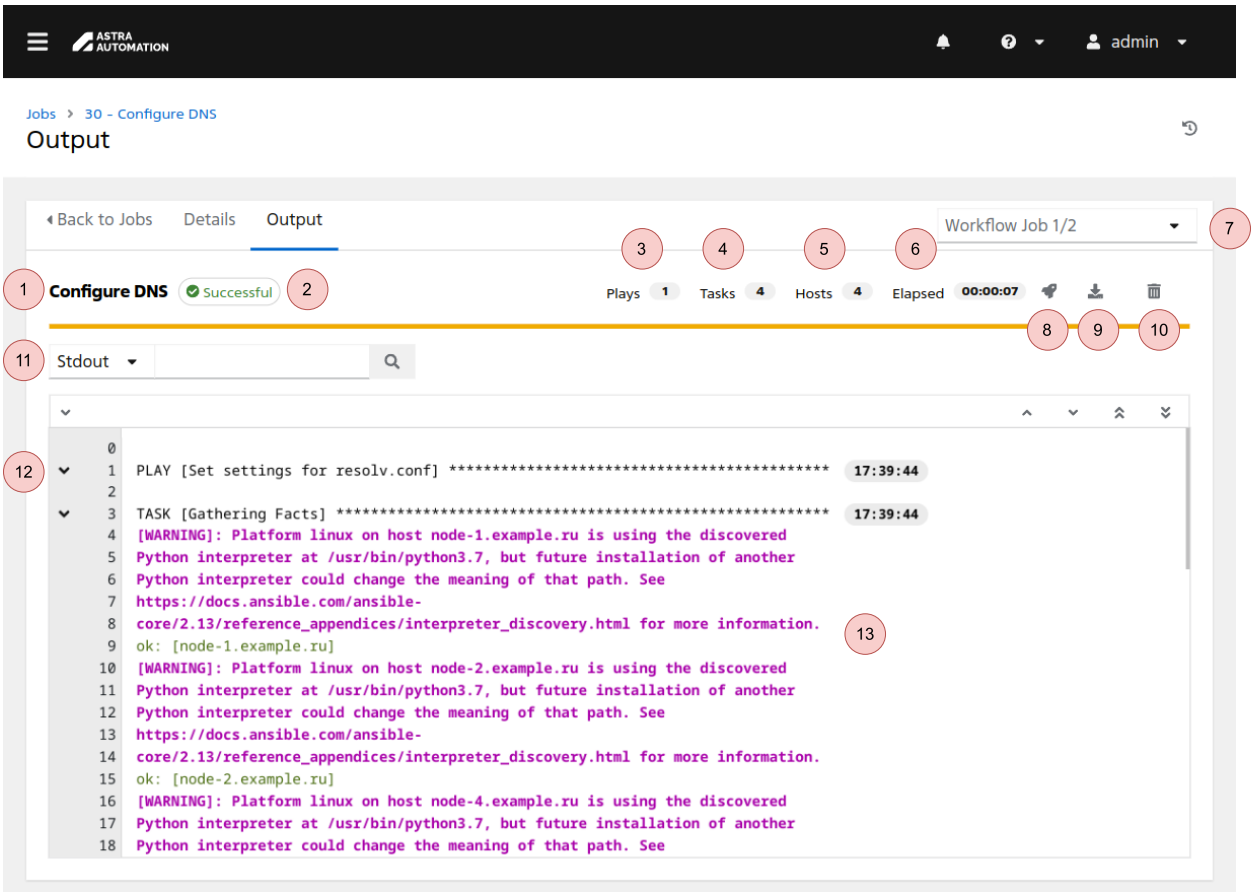
  - **Запущено** (Launched By) – ссылка на профиль пользователя, запустившего задание;
  - **Проект** (Project) – ссылка на *проект*;
  - **Шаблон задания** (Job Template) – ссылка на *шаблон задания*;
  - **Среда исполнения** (Execution Environment) – ссылка на среду исполнения, используемую для выполнения задания;
  - **Инвентарь** (Inventory) – ссылка на инвентарь;
  - **Срез задания** (Job Slice) – номер среза и общее количество срезов, используемых для выполнения задания;
  - **Расписание** (Schedule) – ссылка на расписание, по которому выполняется запуск задания;
  - **Источник** (Source) – тип *источника*, на основе которого создано задание синхронизации проекта или инвентарного списка.
- Флаги для выбора нескольких записей.
- **Название** (Name) – ссылка для перехода в окно просмотра журнала выполнения задания или окно просмотра графа потока заданий.
- **Статус** (Status) – текущий статус выполнения задания:
  - **Успех** (Successful) – не было ошибок;
  - **Ошибка** (Error) – возникли ошибки, которые были успешно обработаны;
  - **Сбой** (Failed) – возникли необрабатываемые ошибки, выполнение задания прервано;
  - **Отменено** (Canceled) – задание отменено пользователем;
  - **Ожидание** (Pending) – задание находится в очереди;
  - **Выполнение** (Running) – задание выполняется.
- **Тип** (Type) – тип задания.
- **Время начала** (Start Time) – время запуска задания.
- **Время завершения** (Finish Time) – время завершения, отмены или прерывания задания.
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - отмена задания;
  - перезапуск задания.

Просмотр вывода Ansible

Для просмотра вывода Ansible нажмите на ссылку с названием задания в таблице заданий. Для заданий в статусе «Выполнение» (Running) используется вывод в реальном времени.

На снимке экрана представлены основные компоненты вкладки **Вывод** (Output).

Примечание: Состав компонентов зависит от типа задания.



Здесь:

1. Название задания.
2. Статус выполнения задания.
3. Количество выполненных сценариев.
4. Общее количество выполненных задач (на всех узлах).
5. Количество управляемых узлов.
6. Общее время выполнения задания.
7. Поле выбора заданий потока.

   Чтобы открыть вывод одного из заданий потока, выберите его в этом списке.
8. Кнопка перезапуска задания.
9. Кнопка сохранения на локальный компьютер файла с выводом Ansible.
10. Кнопка удаления задания из истории.
11. Поле для настройки фильтров.

    Поле настройки фильтров позволяет отфильтровать строки в выводе Ansible по различным критериям.
12. Кнопки сворачивания / разворачивания части вывода. Позволяют скрыть часть вывода для более удобного просмотра оставшихся строк.
13. Вывод Ansible.

Удаление

Чтобы удалить сведения о задании, выполните следующие действия:

1. В таблице заданий нажмите на ссылку с названием удаляемого задания.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Расписание

Окно **Расписание** (Schedules) используется для просмотра сведений о заданиях, выполняемых по расписанию, и управления ими. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Режимы просмотра* ▸ *Расписание* (*Views* ▸ *Schedules*).

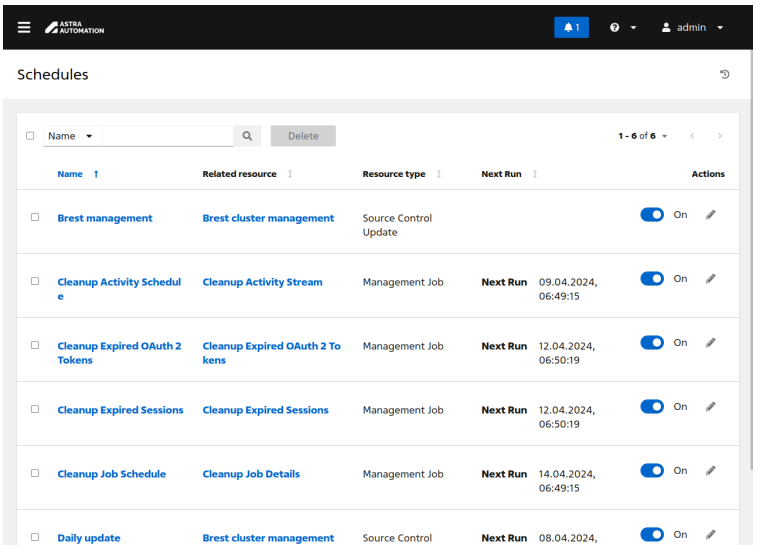
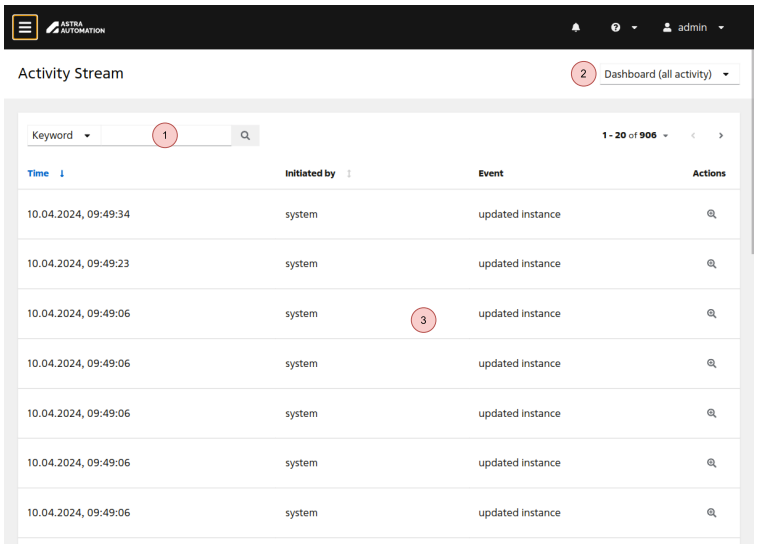


Таблица с расписанием запуска заданий состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name) – ссылка на расписание, по которому выполняется запуск задания;
- **Связанный ресурс** (Related resource) – ссылка на ресурс, связанный с заданием;
- **Тип ресурса** (Resource type) – тип задания;
- **Следующий запуск** (Next Run) – дата и время ближайшего запуска задания;
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых с расписанием действий:
  - переключение статуса;
  - редактирование.

Лента активности

Окно **Лента активности** (Activity Stream) используется для просмотра информации о действиях, выполненных внутри Astra Automation Controller. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Режимы просмотра* ▸ *Лента активности* (*Views* ▸ *Activity Stream*).



Окно состоит из следующих элементов:

1. Поле поиска по строкам.

   Используется для фильтрации записей в таблице событий на основе вхождения в значение одного из полей указанной строки. Поддерживается создание сложных выражений фильтрации с использованием логических условий.

2. Поле фильтрации записей по типу связанных ресурсов или компонентов.

3. Таблица событий.

   Таблица событий состоит из следующих колонок:

   - **Время** (Time) – дата и время события.
   - **Инициировано** (Initiated by) – инициатор события. Если инициатором события был один из пользователей контроллера, в этой колонке выводится ссылка на его профиль.
   - **Событие** (Event) – название события.

     Если событие связано с одним из ресурсов или компонентов контроллера (проект, шаблон, поток заданий и так далее), в этом поле выводится ссылка на него.
   - **Действия** (Actions) – кнопка просмотра подробной информации о событии.

     При нажатии на кнопку открывается диалоговое окно **Подробная информация о событии** (Event detail), в котором в дополнение ко времени, инициатору и типу события выводится список сделанных изменений.

Согласования потоков заданий

Окно **Согласования потоков заданий** (Workflow Approvals) используется для подтверждения или отмены действий, связанных с узлами типа «Согласование» (Approval) *шаблонов потоков заданий*. Для просмотра согласований и принятия решений по ним необходимо перейти в соответствующий раздел с помощью панели навигации: *Режимы просмотра* ▸ *Согласование потоков заданий* (*Views* ▸ *Workflow Approvals*).

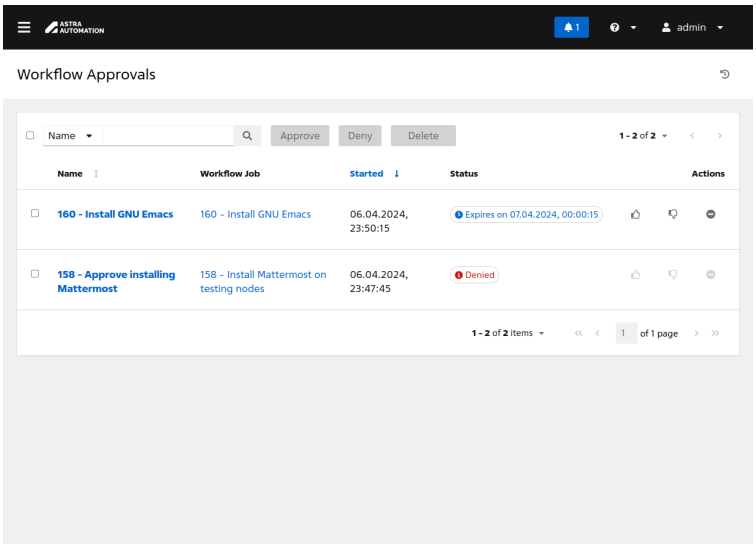


Таблица согласований состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name) – ссылка на окно просмотра сведений о согласовании.
- **Задание потока** (Workflow Job) – ссылка на окно просмотра сведений о задании потока, которое будет запущено при согласовании.
- **Начато** (Started) – дата и время запуска согласования.
- **Статус** (Status) – текущий статус согласования.

  Если время на согласование ограничено, для нерешенных согласований в этой колонке выводятся дата и время, до наступления которых нужно принять решение.

  Для согласований, по которым решение принято, в этой колонке выводится их статус – «Согласовано» (Approved) и «Отказано» (Denied) соответственно.
- **Действия** (Actions) – кнопки управления согласованиями:
  - принятие положительного решения по согласованию;
  - отказ в согласовании;
  - остановка выполнения потока заданий.

Просмотр

Для просмотра подробных сведений о согласовании нажмите на ссылку с его названием в таблице согласований.

Окно подробных сведений о согласовании состоит из вкладки **Подробности** (Details), на которой выводится следующая информация:

- **Название** (Name) – название узла в графе потока заданий.
- **Создано** (Created) – дата и время создания согласования, а также ссылка на профиль связанного пользователя.
- **Истекает** (Expires) – если время на согласование ограничено, в этом поле выводятся дата и время, до наступления которых нужно принять решение.
- **Последнее изменение** (Last Modified) – дата и время последнего изменения согласования.
- **Шаблон потока заданий** (Workflow Job Template) – ссылка на окно просмотра подробных сведений о шаблоне потока заданий.
- **Время выполнения** (Elapsed) – время, прошедшее с момента запуска потока заданий.
- **Сведения о задании потока** (Workflow job details) – набор полей, связанных с заданием:
  - **Задание потока** (Workflow Job) – ссылка на задание, которое будет запущено при положительном решении о согласовании.
  - **Инвентарь** (Inventory) – ссылка на инвентарный список, связанный с заданием.
  - **Переменные** (Variables) – значения переменных Ansible, заданные на уровне потока заданий.

    Подробности о порядке присвоения значений переменным см. в разделе *Переменные*.

Метрики узлов

Окно **Метрики узлов** (Host Metrics) используется для вывода статистической информации о запуске заданий автоматизации на управляемых узлах.

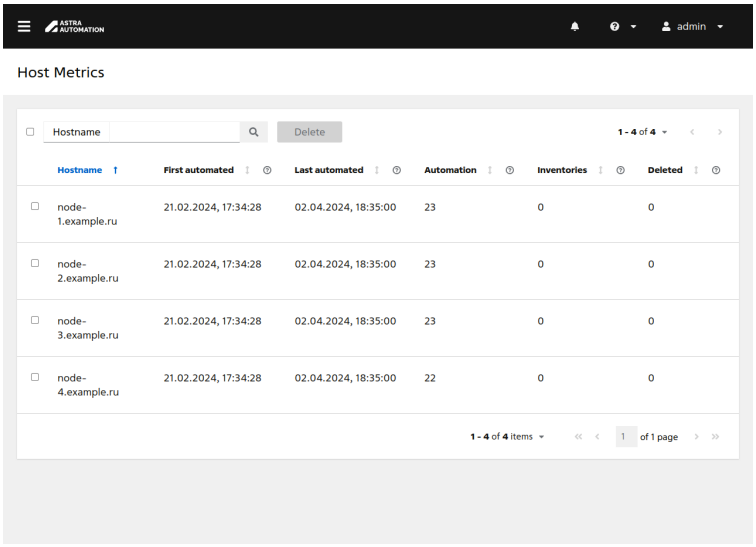


Таблица с информацией о выполнении заданий автоматизации на управляемых узлах состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Наименование узла** (Hostname) – название управляемого узла;
- **Первый автоматизированный** (First automated) – дата и время первого запуска задания автоматизации;
- **Последняя автоматизация** (Last automated) – дата и время последнего запуска задания автоматизации;
- **Автоматизация** (Automation) – общее количество выполненных заданий автоматизации;
- **Инвентарь** (Inventories) – количество инвентарных списков, в которые входит управляемый узел;
- **Удалено** (Deleted) – количество раз, которые управляемый узел был удален из списка управляемых узлов.

Поддерживается фильтрация и сортировка записей по различным параметрам.

Удаление записей

Чтобы удалить сводную информацию об определенных управляемых узлах, выполните следующие действия:

1. В таблице со статистической информацией включите флаги в строках с управляемыми узлами, сведения о которых следует удалить.
2. Нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление записей.

Расход ресурсов подписки

Важно: Эта часть документации находится в стадии разработки.

Ресурсы

В этом разделе описывается управление ресурсами контроллера, в которые входят:

- шаблоны заданий и шаблоны потоков заданий;
- полномочия;
- проекты;
- инвентарь;
- управляемые узлы.

Шаблоны

Окно **Шаблоны** (Templates) используется для управления *шаблонами заданий* и *шаблонами потоков заданий*. Для создания шаблонов и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Ресурсы* ▸ *Шаблоны* (*Resources* ▸ *Templates*).

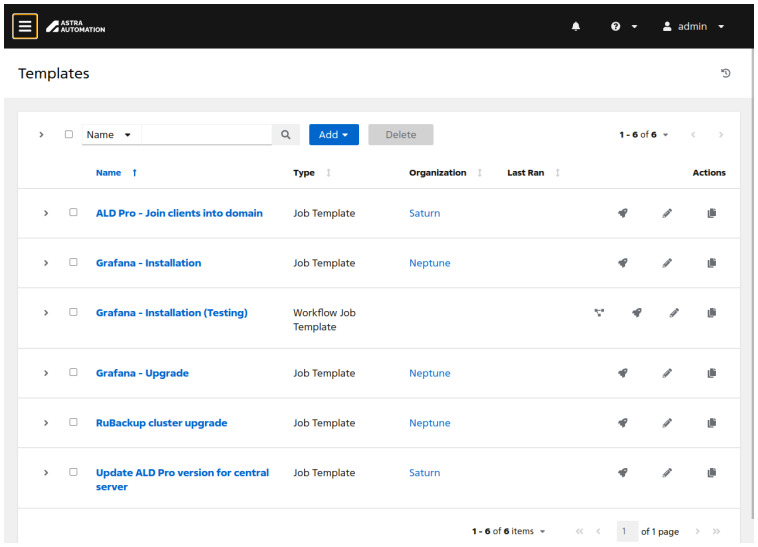


Таблица шаблонов состоит из следующих колонок:

- Флаги для выбора нескольких записей.
- **Название** (Name).
- **Тип** (Type).
- **Организация** (Organization) – ссылка на окно просмотра свойств *организации*, которой принадлежит *проект*, связанный с шаблоном.

- **Последний запуск** (Last Ran).

  Дата и время последнего запуска задания или шаблона потока заданий.
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - переход в окно визуализатора графа шаблона потока заданий;
  - запуск задания или потока заданий на основе шаблона;
  - переход в окно редактирования свойств шаблона;
  - создание копии шаблона потока заданий.

Просмотр

Для получения подробных сведений о шаблоне нажмите на ссылку с его названием в таблице шаблонов. Окно просмотра сведений о шаблоне состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о шаблоне.
- **Доступ** (Access) – таблица пользователей и назначенных им пользовательских и командных *ролей*.
- **Уведомления** (Notifications) – таблица уведомлений.
- **Расписание** (Schedules) – расписание запуска заданий на основе шаблона.
- **Визуализатор** (Vizualizer) – графический редактор узлов потока заданий.

  Примечание: Эта вкладка доступна только для шаблонов потоков заданий.

- **Задания** (Jobs) – таблица заданий, созданных на основе шаблона.
- **Опрос** (Survey) – таблица параметров, которые должны быть заданы при запуске заданий на основе шаблона.

Настройка доступа

Во вкладке **Доступ** (Access) выводится список пользователей и назначенных им *ролей*. Нажатие на ссылку с именем пользователя приводит к переходу в окно просмотра сведений о пользователе.

Назначение ролей отдельным пользователям

Чтобы настроить доступ к шаблону для отдельных пользователей, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Пользователи* (Users) и нажмите кнопку *Продолжить* (Next).
3. Выберите пользователей, которым хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли, которые хотите назначить выбранным на предыдущем шаге пользователям.

5. Нажмите кнопку *Сохранить* (Save).

Назначение командных ролей

Чтобы настроить доступ к шаблону для всех участников одной или нескольких команд, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Команды* (Teams) и нажмите кнопку *Продолжить* (Next).
3. Выберите команды, участникам которых хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли и нажмите кнопку *Сохранить* (Save).

Отзыв ролей

Чтобы отозвать пользовательскую или командную роль, выполните следующие действия:

1. Во вкладке **Доступ** (Access) в строке с нужным пользователем нажмите кнопку × рядом с названием отзываемой роли.
2. Подтвердите отзыв роли.

Примечание: Роли «Системный администратор» (System Administrator) и «Системный аудитор» (System Auditor) нельзя отозвать через настройку доступа на уровне проекта, однако, можно *изменить тип пользователя*.

Запуск

Для запуска задания или потока заданий выполните следующие действия:

1. В таблице шаблонов нажмите на ссылку с названием нужного шаблона.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Запустить* (Run).
3. Если в шаблоне настроен и включен опрос – в открывшемся окне **Запуск** (Run) заполните форму во вкладке **Опрос** (Survey) и нажмите кнопку *Продолжить* (Next).

   Заданные значения передаются в задание или поток заданий как переменные.

   Подробности о порядке разрешения значений переменных см. в документе *Переменные*.
4. Во вкладке **Предварительный просмотр** (Preview) проверьте корректность введенных данных. Если все значения указаны верно, нажмите кнопку *Запустить* (Run).

Удаление

Чтобы удалить шаблон, выполните следующие действия:

1. В таблице шаблонов нажмите на ссылку с названием удаляемого шаблона.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Шаблоны заданий

В этой секции рассматривается создание *шаблонов заданий* и запуск заданий на их основе через веб-интерфейс.

Создание шаблона заданий

Для создания шаблона задания выполните следующие действия:

1. В окне **Шаблоны** (Templates) нажмите кнопку *Добавить* (Add) и в открывшемся меню выберите **Добавить шаблон задания** (Add job template).
2. Заполните форму **Создать новый шаблон задания** (Create New Job Template):
   - **Название** (Name).

     Особенности заполнения поля:

     - В рамках одной организации не допускается создание шаблонов с одним и тем же названием.
     - Название шаблона не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использования символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите описание шаблона, например, его назначение или особенности использования.
   - **Тип задания** (Job Type) – выберите, будет ли выполнение задания вносить фактические изменения в конфигурацию управляемых узлов:
     - **Выполнение** (Run) – да;
     - **Проверка** (Check) – нет.

     Подробности см. в секции *Тип задания*.
   - **Инвентарь** (Inventory) – выберите *инвентарный список*, управляемые узлы из которого настраиваются.
   - **Проект** (Project) – выберите *проект*.
   - **Среда исполнения** (Execution Environment) – выберите среду исполнения для запуска заданий.

     Примечание: Это поле становится доступным после заполнения поля **Проект** (Project).

- **Playbook** – выберите файл playbook из состава проекта.

  Примечание: Это поле становится доступным после заполнения поля **Проект** (Project).

- **Полномочия** (Credentials) – укажите полномочия, которые следует использовать для доступа к управляемым узлам.

  Подробности см. в секции *Полномочия*.

- **Метки** (Labels).

  С помощью меток можно группировать задания, созданные на основе шаблона. Их удобно использовать для поиска нужных записей журнала выполнения заданий или их логической группировки.

- **Переменные** (Variables) – укажите дополнительные переменные, которые следует использовать при запуске заданий на основе шаблона.

  Подробности см. в секции *Переменные*.

- **Ответвления** (Forks) – укажите количество параллельных процессов, используемых для запуска заданий на основе шаблона.

  Подробности см. в секции *Ответвления*.

- **Лимит** (Limit) – укажите дополнительные фильтры названий управляемых узлов.

  Подробности см. в секции *Лимит на управляемые узлы*.

- **Степень подробности** (Verbosity) – укажите степень детализации журнала выполнения заданий на основе шаблона.

  Подробности см. в секции *Степень подробности*.

- **Деление заданий на срезы** (Job Slicing) – укажите количество частей, на которые следует разделить итоговый инвентарный список.

  Подробности см. в секции *Деление на срезы*.

- **Таймаут** (Timeout) – укажите ограничение по времени (в секундах) на выполнение задания на основе шаблона.

  Подробности см. в секции *Таймаут*.

- **Показать изменения** (Show Changes).

  Если эта настройка включена, при запуске задания на основе шаблона в журнал выводится список изменений, выполненных на управляемых узлах.

  Значение по умолчанию – Выкл (`Off`).

  Подробности см. в секции *Показ изменений*.

- **Группы узлов контроллера** (Instance Groups) – укажите группы узлов контроллера, которые следует использовать для запуска заданий на основе шаблона.

- **Теги задания** (Job Tags) – укажите одной строкой через запятую названия тегов, задачи с которыми следует выполнять при запуске заданий на основе шаблона. Прочие задачи будут пропущены.

  Подробности см. в секции *Теги*.

- **Пропускать теги** (Skip Tags) – укажите одной строкой через запятую названия тегов, задачи с которыми следует пропустить при запуске заданий на основе шаблона. Прочие задачи будут выполнены.

  Подробности см. в секции *Теги*.

- **Повышение привилегий** (Privilege Escalation) – укажите, следует ли использовать повышение привилегий при выполнении заданий на основе шаблона.

  Подробности см. в секции *Повышение привилегий*.

- **Обратные вызовы процесса обеспечения сервиса** (Provisioning Callbacks).

  Если эта настройка включена:

  - при выполнении заданий на основе шаблона управляемые узлы могут связаться с контроллером и запросить обновление конфигурации;
  - становится доступным для заполнения поле **Ключ конфигурации узла** (Host Config Key).

- **Ключ конфигурации узла** (Host Config Key).

  URL, используемый управляемыми узлами для запроса в контроллере обновления конфигурации.

- **Включить webhook** (Enable Webhook).

  Если эта настройка включена, разрешено создание заданий на основе шаблона путем запроса к Webhook.

- **Сервис webhook** (Webhook Service).

  Выберите сервис хранения исходного кода, используемый для работы с Webhook. Поддерживаются следующие значения:

  - BitBucket Data Center;
  - GitHub;
  - GitLab.

Примечание: Это поле отображается, если включен флаг **Включить webhook** (Enable Webhook).

- **Webhook URL**.

  URL для работы с Webhook.

  Значение в этом поле формируется автоматически при создании шаблона.

Примечание: Это поле отображается, если включен флаг **Включить webhook** (Enable Webhook).

- **Ключ webhook** (Webhook Key).

  Ключ для работы с Webhook.

  Значение в этом поле формируется автоматически при создании шаблона.

Примечание: Это поле отображается, если включен флаг **Включить webhook** (Enable Webhook).

- **Полномочия webhook** (Webhook Credential).

  Полномочия, используемые для доступа к выбранному сервису хранения исходного кода.

Примечание: Это поле отображается, если выбрано значение в поле **Сервис webhook** (Webhook Service).

- **Параллельные задания** (Concurrent Jobs).

  Если эта настройка включена, на основе шаблона можно запустить одновременное выполнение нескольких заданий с одним и тем же инвентарным списком.

- **Включить хранилище фактов** (Enable Fact Storage) – укажите, следует ли сохранить в кэш Astra Automation Controller факты, полученные при выполнении заданий на основе шаблона.

  Подробности см. в секции *Хранилище фактов*.

- **Запретить откат группы узлов управления** (Prevent Instance Group Fallback).

  Если эта настройка включена, для запуска заданий будут использоваться группы узлов, указанные в поле **Группы узлов контроллера** (Instance Groups). Если группы узлов контроллера не выбраны, используются глобальные группы узлов.

  Подробности см. в секции *Запрет отката группы узлов управления*.

3. Нажмите кнопку *Сохранить* (Save).

Запуск задания на основе шаблона

Чтобы запустить задание на основе шаблона, выполните следующие действия:

1. В окне **Шаблоны** (Templates) нажмите на ссылку с названием нужного шаблона задания.
2. Во вкладке **Подробности** (Details) нажмите на кнопку *Запустить* (Run).
3. Если в настройках шаблона включен опрос, откроется диалоговое окно **Запуск** (Run).
   1. Во вкладке **Опрос** (Survey) заполните значения полей.
   2. Нажмите кнопку *Продолжить* (Next).
   3. Во вкладке **Предварительный просмотр** (Preview) проверьте настройки запуска задания. Если все значения заданы верно, нажмите кнопку *Запустить* (Run).

Шаблоны потоков заданий

В этом документе рассматриваются следующие операции, связанные с *шаблонами потоков заданий*:

- создание;
- запуск.

Создание шаблонов потоков заданий

Для создания шаблона потока заданий выполните следующие действия:

1. В окне **Шаблоны** (Templates) нажмите кнопку *Добавить* (Add) и в открывшемся меню выберите **Добавить шаблон потока заданий** (Add workflow template).
2. Заполните форму **Создать новый шаблон потока заданий** (Create New Workflow Template):
   - **Название** (Name).

     Особенности заполнения поля:

     - В рамках одной организации не допускается создание шаблонов с одним и тем же названием.
     - Название шаблона не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использование символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите описание шаблона, например, его назначение или особенности использования.
   - **Организация** (Organization) – укажите организацию, которой принадлежит шаблон потока заданий.
   - **Инвентарь** (Inventory) – укажите инвентарный список.

     Важно: Значение в этом поле переопределяет значения, заданные на уровне шаблонов заданий.

   - **Лимит** (Limit) – укажите дополнительные фильтры названий управляемых узлов.

     Подробности см. в подразделе *Лимит на управляемые узлы*.
   - **Ветка системы управления исходными данными** (Source control branch) – если при запуске заданий следует использовать ветку, отличную от основной ветки проекта, укажите ее название в этом поле.

     Важно: Название ветки, указанное в этом поле, применяется ко всем шаблонам заданий, используемым в потоке.

   - **Метки** (Labels).

С помощью меток можно группировать задания, созданные при выполнении потока. Их удобно использовать для поиска нужных записей журнала выполнения заданий или их логической группировки.

- **Переменные** (Variables) – укажите дополнительные переменные, которые следует использовать при запуске заданий потока.

  Подробности о порядке разрешения переменных см. в документе *aac_management_workflow_templates_extra_vars*.

- **Теги задания** (Job Tags) – укажите одной строкой через запятую названия тегов, задачи с которыми следует выполнять при запуске заданий на основе шаблона. Прочие задачи будут пропущены.

  Подробности см. в подразделе *Теги*.

- **Пропускать теги** (Skip Tags) – укажите одной строкой через запятую названия тегов, задачи с которыми следует пропустить при запуске заданий на основе шаблона. Прочие задачи будут выполнены.

  Подробности см. в подразделе *Теги*.

- **Включить webhook** (Enable Webhook).

  Если эта настройка включена, разрешено создание потока заданий на основе шаблона путем запроса к Webhook.

- **Разрешить одновременные задания** (Enable Concurrent Jobs).

  Если эта настройка включена, на основе шаблона можно запустить одновременно несколько потоков заданий с одним и тем же инвентарным списком.

- **Сервис Webhook** (Webhok Service).

  Выберите сервис хранения исходного кода, используемый для работы с Webhook. Поддерживаются следующие значения:

  - BitBucket Data Center;
  - GitHub;
  - GitLab.

Примечание: Это поле отображается, если включен флаг **Включить webhook** (Enable Webhook).

- **Webhook URL** (Webhook URL)

  URL для работы с Webhook.

  Значение в этом поле формируется автоматически при создании шаблона.

Примечание: Это поле отображается, если включен флаг **Включить webhook** (Enable Webhook).

- **Ключ webhook** (Webhook Key)

  Ключ для работы с Webhook.

  Значение в этом поле формируется автоматически при создании шаблона.

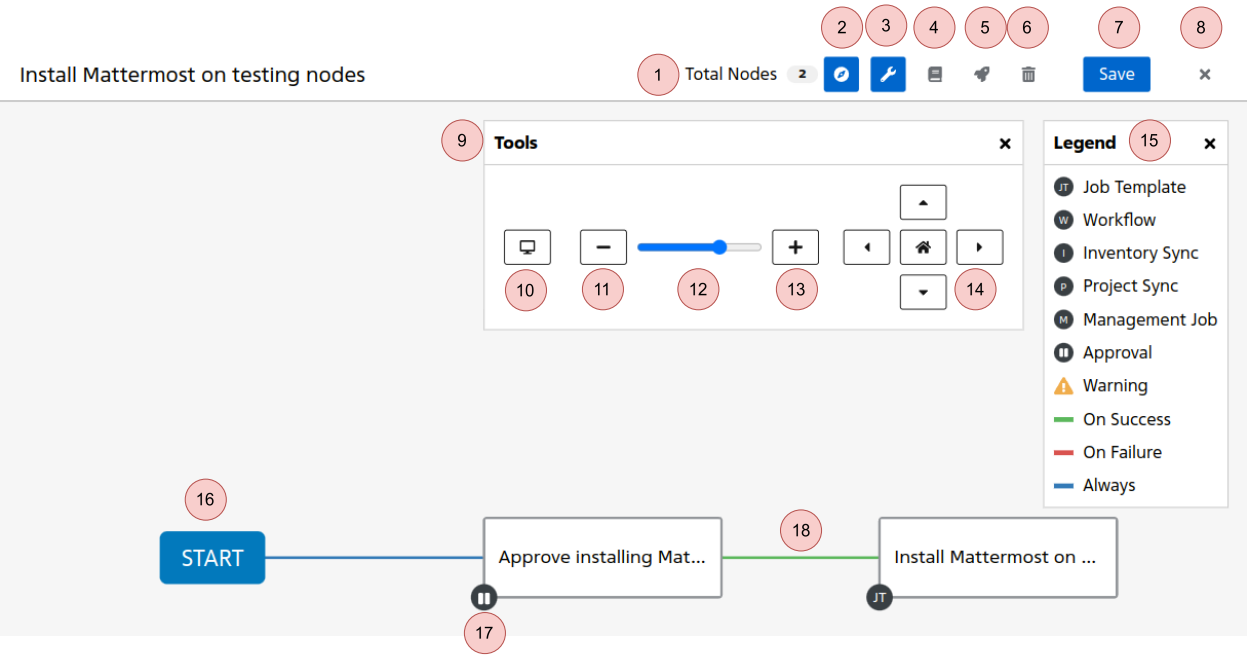
Примечание: Это поле отображается, если включен флаг **Включить webhook** (Enable Webhook).

3. Нажмите кнопку *Сохранить* (Save).

Визуализатор шаблонов потоков заданий

После создания шаблона потока заданий визуализатор запускается автоматически. Для его запуска для существующего шаблона потока заданий выполните следующие действия:

1. В таблице шаблонов нажмите на ссылку с названием нужного шаблона потока заданий.
2. Выберите вкладку **Визуализатор** (Visializer).



Окно визуализатора содержит следующие элементы:

1. Общее количество узлов. При расчете значения узел «НАЧАЛО» (START) не учитывается.
2. Переключатель отображения легенды.
3. Переключатель отображения кнопок управления масштабом.
4. Кнопка вызова справочной информации.
5. Кнопка запуска потока заданий.
6. Кнопка удаления всех узлов шаблона потока заданий.
7. Кнопка сохранения изменений в шаблоне.
8. Кнопка закрытия визуализатора без сохранения внесенных изменений.
9. Панель управления масштабом.

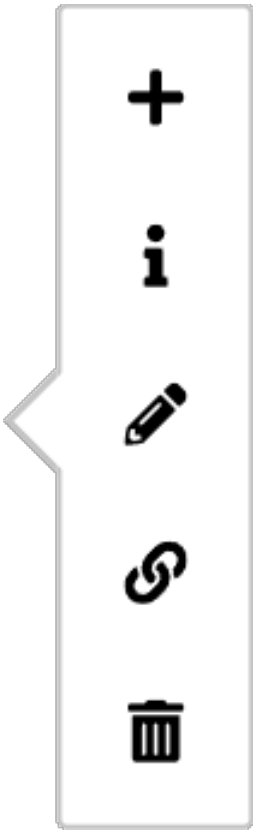
10. Кнопка изменения масштаба под размер экрана.
11. Кнопка увеличения масштаба.
12. Ползунок изменения масштаба.
13. Кнопка уменьшения масштаба.
14. Кнопки управления положением графа на экране.
15. Панель легенды.
16. Узел «НАЧАЛО» (START).
17. Индикатор типа узла. В данном случае – «Согласование» (Approval).
18. Ссылка на следующий узел.

    Цвет ссылки указывает на условие, при котором выполняется переход:

    - красный – ошибки при выполнении задания или отказ в согласовании;
    - зеленый – успешное выполнение задания или согласование;
    - синий – безусловный переход.

При нажатии на узел или ссылку появляется контекстное меню.

Контекстное меню узла:

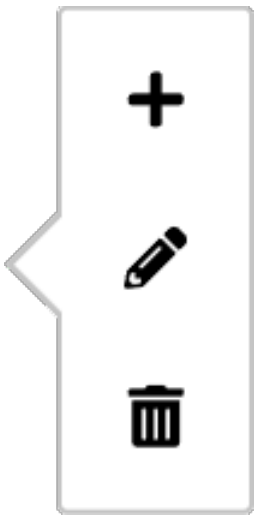


Контекстное меню узла содержит кнопки для выполнения следующих действий:

- создание нового узла и связь с ним;
- просмотр сведений об узле;

- редактирование узла;
- создание ссылки на существующий узел;
- удаление узла.

Контекстное меню ссылки:



Контекстное меню ссылки содержит кнопки для выполнения следующих действий:

- вставка узла между двумя существующими узлами;
- редактирование ссылки;
- удаление ссылки.

Начало работы

Чтобы начать работу с графом нового шаблона потока заданий, выполните следующие действия:

1. Нажмите кнопку **Пуск** (Start).
2. В диалоговом окне **Добавить узел** (Add Node) выберите тип узла.
3. Заполните поля, специфичные для выбранного типа узла. В зависимости от типа узла могут быть доступны дополнительные настройки.
4. Нажмите кнопку *Сохранить* (Save).

Первый добавленный в граф узел связывается с узлом «НАЧАЛО» (START), который имеет следующие особенности:

- его нельзя изменить или удалить;
- тип ссылки на другие узлы – только «Всегда» (Always).

Создание узлов

Чтобы добавить в граф новый узел, выполните следующие действия:

1. Вызовите контекстное меню узла или ссылки.
2. Нажмите кнопку создания нового узла. Это приводит к открытию диалогового окна **Добавить узел** (Add Node).
3. Во вкладке **Тип выполнения** (Run type) выберите условие перехода к создаваемому узлу:
   - **При успехе** (On Success) – согласование или успешное выполнение задания;
   - **При неудаче** (On Failure) – отказ в согласовании или ошибки при выполнении задания;
   - **Всегда** (Always) – переход выполняется в любом случае.
4. Нажмите кнопку *Продолжить* (Next).
5. Во вкладке **Тип узла** (Node type) выберите тип создаваемого узла.
6. Заполните поля, соответствующие выбранному типу узла:
   - **Согласование** (Approval).

     Узлы этого типа используются для создания согласований – запросов к пользователю на подтверждение перехода к следующему заданию в потоке.

     Укажите параметры согласования.

     – **Название** (Name) – название узла.

       Указанное название выводится в таблице согласований.

     – **Описание** (Description) – описание узла.

     – **Таймаут** (Timeout) – время на согласование.

       Если в полях для минут и секунд указаны значения `0`, время на согласование не ограничено.

   - **Источники синхронизации инвентаря** (Inventory Source Sync).

     Узлы этого типа создают задание синхронизации инвентарного списка с источником.

     – **Название** (Name) – выберите источник сведений об управляемых узлах.

   - **Шаблон задания** (Job Template).

     Узлы этого типа используются для запуска заданий на основе выбранного шаблона задания.

     Если в настройках шаблона задания включен опрос, вместо кнопки *Сохранить* (Save) в форме выводится кнопка *Продолжить* (Next). При ее нажатии происходит переход к шагу опроса.

     Подробности об опросах см. в разделе *Опросы*.

     – **Название** (Name) – выберите шаблон задания.

   - **Синхронизация проекта** (Project Sync).

     Узлы этого типа создают задание синхронизации кода проекта с источником.

     – **Название** (Name) – выберите проект.

- **Шаблон потока заданий** (Workflow Job Template).

  Узлы этого типа запускают поток заданий на основе выбранного шаблона потока заданий.

  - **Название** (Name) – выберите шаблон потока заданий.

- **Служебное задание** (Management Job).

  Узлы этого типа создают служебное задание.

  При выборе заданий «Cleanup Activity Stream» и «Cleanup Job Details» вместо кнопки *Сохранить* (Save) в форме выводится кнопка *Продолжить* (Next). При ее нажатии происходит переход к шагу заполнения поля **Количество дней, в течение которых будут храниться данные** (Days of data to be retained). Все записи старше указанного периода будут удалены.

  Значение по умолчанию – `30`.

  Подробности о служебных заданиях см. в разделе *Служебные задания*.

7. Заполните поля **Конвергенция** (Convergence) и **Псевдоним узла** (Alias):

   - **Конвергенция** (Convergence) – укажите количество условий, которые должны быть выполнены для перехода к этому узлу потока заданий.

     Подробности см. в подразделе *Конвергенция*.

   - **Псевдоним узла** (Node Alias) – если в этом поле указано какое-либо значение, при просмотре графа потока заданий оно выводится вместо значения, указанного в поле **Название** (Name).

8. Нажмите кнопку *Сохранить* (Save).

Удаление узлов и ссылок

Чтобы удалить узел или ссылку, выполните следующие действия:

1. Вызовите контекстное меню узла или ссылку.
2. Нажмите кнопку удаления.
3. Подтвердите удаление.

Запуск потока заданий

Чтобы запустить поток заданий, выполните следующие действия:

1. В таблице шаблонов нажмите на ссылку с названием нужного шаблона потоков заданий.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Запустить* (Launch).

Опросы

Для управления *опросами* Astra Automation Controller предоставляет интуитивно понятный пользовательский интерфейс.

При просмотре подробных сведений о шаблоне задания или шаблоне потока заданий во вкладке **Опрос** (Survey) выводится список вопросов, задаваемых пользователю при запуске заданий или потока заданий.

Примечание: В этом документе для краткости шаблоны заданий и шаблоны потоков заданий именуются общим термином «шаблоны».

Таблица вопросов состоит из следующих колонок:

- Флаги для выбора нескольких записей.
- **Название** (Name) – текст вопроса, задаваемого пользователю.
- **Тип** (Type).
- **По умолчанию** (Default).

  Значение по умолчанию. Если ответ имеет тип «Пароль» (Password), в этой колонке выводится значение ЗАШИФРОВАНО (ENCRYPTED).
- **Действия** (Actions) – кнопка для быстрого перехода в окно редактирования вопроса.

Переключение состояния

Чтобы при запуске задания или потока заданий пользователю были заданы вопросы, связанные с шаблоном, необходимо включить соответствующий опрос. Для изменения статуса опроса выполните следующие действия:

1. В таблице шаблонов нажмите на ссылку с названием нужного шаблона.
2. Выберите вкладку **Опрос** (Survey).
3. Переведите переключатель состояния опроса в нужное положение.

Создание вопросов

Чтобы добавить вопросы в опросник, выполните следующие действия:

1. В таблице шаблонов нажмите на ссылку с названием нужного шаблона.
2. Во вкладке **Опрос** (Survey) нажмите кнопку *Добавить* (Add).
3. Заполните форму **Добавить вопрос** (Add Question):
   - **Вопрос** (Question) – укажите текст вопроса, который выводится в форме при запуске задания или потока заданий.
   - **Описание** (Description) – укажите справочную информацию, которая поможет пользователю принять решение при ответе на вопрос.

- **Имя переменной ответа** (Answer variable name) – укажите имя переменной Ansible, значение которой пользователь задает при ответе на вопрос.

  Требования к имени переменной:

  - Имя переменной должно быть уникально на уровне шаблона.
  - Допускается использование символов латинского алфавита, цифр и знака _.
  - Использование пробелов запрещено.
- **Необходимо** (Required) – если эта настройка включена, ответ пользователя на вопрос обязателен.
- **Тип ответа** (Answer type) – выберите тип ответа, который может ввести пользователь.

  Подробности о поддерживаемых типах ответов см. в разделе *Опросы*.

4. В зависимости от выбранного типа ответа заполните соответствующие поля:

   - **Текст** (Text).

     Совет: Для ввода конфиденциальных данных рекомендуется использовать ответ типа «Пароль» (Password).

     Поддерживаются следующие опциональные поля:

     - **Минимальная длина** – минимальное количество символов в строке.

       Значение по умолчанию – `0`.
     - **Максимальная длина** – максимальное количество символов в строке.

       Значение по умолчанию – `1024`.
     - **Ответ по умолчанию** (Default answer)
   - **Текстовая область** (Textarea).

     Поддерживаются следующие опциональные поля:

     - **Минимальная длина** – минимальное количество символов во введенном тексте.

       Значение по умолчанию – `0`.
     - **Максимальная длина** – максимальное количество символов во введенном тексте.

       Значение по умолчанию – `1024`.
     - **Ответ по умолчанию** (Default answer)
   - **Пароль** (Password).

     Поддерживаются следующие опциональные поля:

     - **Минимальная длина** – минимальное количество символов в строке.

       Значение по умолчанию – `0`.
     - **Максимальная длина** – максимальное количество символов в строке.

       Значение по умолчанию – `1024`.

  - **Ответ по умолчанию** (Default answer)

- **Множественный выбор (одиночное выделение)** (Multiple Choice (single select)).

  Чтобы создать элемент списка, введите значение в поле **Варианты множественного выбора** (Muptiple Choice Options) и нажмите `Enter`.

  Чтобы сделать один из элементов списка значением по умолчанию, включите флаг в соответствующей строке.

- **Множественный выбор (множественное выделение)** (Multiple Choice (multiple select)).

  Чтобы создать элемент списка, введите значение в поле **Варианты множественного выбора** (Multiple Choice Options) и нажмите `Enter`.

  Чтобы сделать пункты выбранными по умолчанию, включите флаги в соответствующих строках.

- **Целое число** (Integer).

  Поддерживаются следующие опциональные поля:

  - **Минимум** (Minimum) – минимально допустимое значение.

    Значение по умолчанию – `0`.

  - **Максимум** (Maximum) – максимально допустимое значение.

    Значение по умолчанию – `1024`.

  - **Ответ по умолчанию** (Default answer).

- **Плавающее** (Float).

  Поддерживаются следующие опциональные поля:

  - **Минимум** (Minimum) – минимально допустимое значение.

    Значение по умолчанию – `0`.

  - **Максимум** (Maximum) – максимально допустимое значение.

    Значение по умолчанию – `1024`.

  - **Ответ по умолчанию** (Default answer).

5. Нажмите кнопку *Сохранить* (Save).

Удаление вопросов

Чтобы удалить вопросы, выполните следующие действия:

1. В таблице шаблонов включите флаги на удаляемых опросах.
2. Нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Полномочия

Окно **Полномочия** (Credentials) используется для управления *полномочиями*. Для создания полномочий и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Ресурсы* ▸ *Полномочия* (*Resources* ▸ *Credentials*).

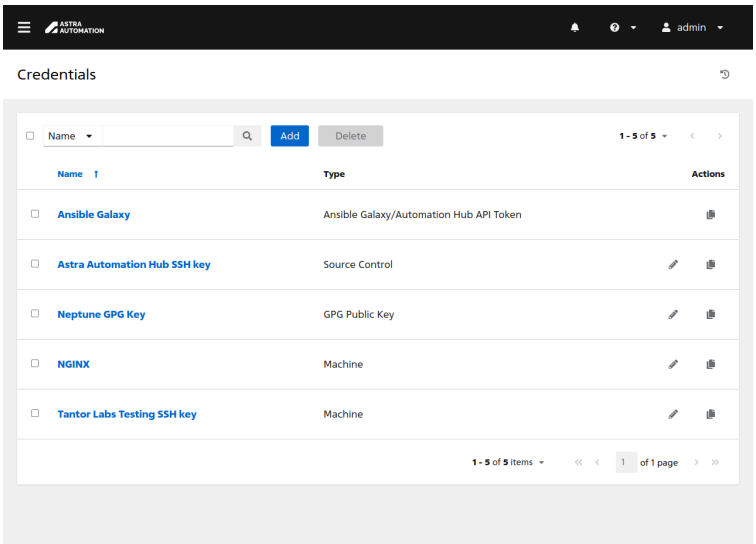


Таблица полномочий состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Тип** (Type) – тип полномочий;
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - переход в окно редактирования сведений о полномочии (не поддерживается для некоторых типов полномочий);
  - копирование полномочия.

Просмотр сведений о полномочиях

Для получения подробных сведений о полномочии нажмите на ссылку с его названием в таблице полномочий.

Внешний вид окна со сведениями о полномочии зависит от их типа и может состоять из следующих вкладок:

- **Подробности** (Details) – общие сведения о полномочии;
- **Доступ** (Access) – таблица пользователей и назначенных им пользовательских и командных *ролей*;
- **Шаблоны заданий** (Job Templates) – таблица *шаблонов заданий*, использующих полномочие.

Создание полномочия

Для создания полномочия выполните следующие действия:

1. В окне **Полномочия** (Credentials) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новые учетные данные** (Create New Credential):
   - **Название** (Name) – название полномочия.
   - **Описание** (Description) – дополнительная информация о полномочии, например, цель его создания или правила использования.
   - **Организация** (Organization) – организация, которой принадлежит полномочие.

     Если поле не заполнено, полномочие принадлежит всем организациям контроллера.
   - **Тип полномочия** (Credential Type) – *тип полномочия*.

     При выборе значения в этом поле в форме появляются соответствующие поля.
3. Заполните поля, соответствующие выбранному типу полномочия.

   Особенности заполнения полей для каждого типа полномочия приведены в следующих секциях.
4. Нажмите кнопку *Сохранить* (Save).

API-токен Ansible Galaxy/Automation Hub

При создании полномочия этого типа доступны следующие поля:

- **URL сервера Galaxy** (Galaxy Server URL) – URL *приватного Automation Hub*, используемого для распространения инфраструктурного кода.
- **URL сервера аутентификации** (Auth Server URL) – если для аутентификации на приватном Automation Hub используется сервер аутентификации Keycloak, укажите его URL в этом поле.
- **Токен API** (API Token) – токен, используемый для аутентификации в приватном Automation Hub.

Подробности о полномочиях этого типа см. в секции *API-токен Ansible Galaxy/Automation Hub*.

Centrify Vault Credential Provider Lookup

Важно: Эта часть документации находится в стадии разработки.

CyberArk Central Credential Provider Lookup

Важно: Эта часть документации находится в стадии разработки.

CyberArk Conjur Secrets Manager Lookup

Важно: Эта часть документации находится в стадии разработки.

Google Compute Engine

При создании полномочия этого типа доступны следующие поля:

- **Файл JSON сервисной учетной записи** (Service account JSON file) – файл с реквизитами сервисной учетной записи Google Compute Cloud. При выборе файла JSON, содержащего необходимые сведения, остальные поля формы заполняются автоматически.

  Пошаговые инструкции по созданию такого файла см. в документации Google Compute Cloud[134].
- **Адрес электронной почты сервисной учетной записи** (Service Account Email Address) – адрес электронной почты, связанный с сервисной учетной записью Google Compute Engine.
- **Project** – идентификатор проекта в Google Compute Engine.
- **Закрытый ключ RSA** (RSA Private Key) – содержимое приватного ключа SSH, используемого для доступа к Google Compute Engine с реквизитами сервисной учетной записи.

Подробности об этом типе полномочий см. в подразделе *Google Compute Engine*.

HashiCorp Vault Secret Lookup

Важно: Эта часть документации находится в стадии разработки.

[134] https://cloud.google.com/iam/docs/keys-create-delete

HashiCorp Vault Signed SSH

Важно: Эта часть документации находится в стадии разработки.

Insights

Важно: Эта часть документации находится в стадии разработки.

Microsoft Azure Key Vault

Важно: Эта часть документации находится в стадии разработки.

OpenStack

При создании полномочия этого типа доступны следующие поля:

- **Имя пользователя** (Username) – имя пользователя OpenStack.
- **Пароль (ключ API)** (Password (API Key)) – пароль или ключ API OpenStack.
- **Узел (URL для аутентификации)** (Host (Authentication URL)) – URL, используемый для аутентификации в OpenStack, например:

  ```
  https://openstack.example.com/auth/
  ```

- **Проект (имя арендатора)** (Project (Tenant Name)) – название или идентификатор проекта (тенанта) OpenStack. Как правило, совпадает с именем пользователя OpenStack.
- **Проект (доменное имя)** (Project (Domain Name)) – доменное имя проекта (тенанта) OpenStack.
- **Доменное имя** (Domain Name) – FQDN сервера OpenStack.

  Заполнение этого поля требуется только при использовании сервера аутентификации Keystone v3.
- **Название региона** (Region Name) – название региона OpenStack.
- **Проверка SSL** (Verify SSL) – если эта настройка включена, при подключении к OpenStack проверяется корректность сертификата SSL.

  Значение по умолчанию – включено.

Red Hat Satellite 6

Важно: Эта часть документации находится в стадии разработки.

Thycotic DevOps Secrets Vault

Важно: Эта часть документации находится в стадии разработки.

Thycotic Secret Server

Важно: Эта часть документации находится в стадии разработки.

Vault

Важно: Эта часть документации находится в стадии разработки.

VMware vCenter

Важно: Эта часть документации находится в стадии разработки.

Web-сервисы Amazon

Важно: Эта часть документации находится в стадии разработки.

Виртуализация Red Hat

Важно: Эта часть документации находится в стадии разработки.

Диспетчер ресурсов Microsoft Azure

Важно: Эта часть документации находится в стадии разработки.

Личный токен доступа к GitHub

При создании полномочий этого типа укажите в поле **Токен** (Token) значение токена, созданного в профиле GitHub.

Подробности о полномочиях этого типа см. в секции *Личный токен доступа к GitHub*.

Машина

При создании полномочия этого типа доступны следующие поля:

- **Имя пользователя** (Username) – имя пользователя, с реквизитами которого контроллер должен подключаться к управляемому узлу.
- **Пароль** (Password) – пароль пользователя, с реквизитами которого контроллер должен подключаться к управляемому узлу.
- **Закрытый ключ SSH** (SSH Private Key) – содержимое приватного ключа SSH.
- **Подписанный сертификат SSH** (Signed SSH Certificate) – содержимое сертификата, используемого для защиты подключения SSH.
- **Парольная фраза закрытого ключа** (Private Key Passphrase) – если приватный ключ SSH защищен паролем, укажите его в этом поле.
- **Способ повышения привилегий** (Privilege Escalation Method) – метод повышения привилегий.
- **Имя пользователя с повышенными привилегиями** (Privilege Escalation Username) – если для повышения привилегий команды следует выполнять от имени пользователя, отличающегося от указанного в поле **Имя пользователя** (Username), укажите его в этом поле.
- **Пароль с повышенными привилегиями** (Privilege Escalation Password) – пароль пользователя, имя которого указано в поле **Имя пользователя с повышенными привилегиями** (Privilege Escalation Username).

Подробности о полномочиях этого типа см. в секции *Машина*.

Открытый ключ GPG

При создании полномочия этого типа укажите в поле **Открытый ключ GPG** (GPG Public Key) содержимое ключа GPG, который контроллер должен использовать для проверки подписи содержимого.

Подробности о полномочиях этого типа см. в секции *GPG Public Key*.

Платформа автоматизации Red Hat Ansible

Важно: Эта часть документации находится в стадии разработки.

Реестр контейнеров

Важно: Эта часть документации находится в стадии разработки.

Сеть

Важно: Эта часть документации находится в стадии разработки.

Система управления версиями

При создании полномочия этого типа доступны следующие поля:

- **Имя пользователя** (Username) – имя пользователя, с реквизитами которого контроллер должен подключаться к источнику кода.
- **Пароль** (Password) – пароль пользователя, с реквизитами которого контроллер должен подключаться к источнику кода.
- **Закрытый ключ SCM** (SCM Private Key) – содержимое приватного ключа SSH, используемого для подключения к источнику кода.
- **Парольная фраза закрытого ключа** (Private Key Passphrase) – если приватный ключ SSH защищен паролем, укажите его в этом поле.

Подробности о полномочиях этого типа см. в секции *Управление источниками кода*.

Токен доступа OpenShift или Kubernetes API

Важно: Эта часть документации находится в стадии разработки.

Токен персонального доступа GitLab

При создании полномочий этого типа укажите в поле **Токен** (Token) значение токена, созданного в профиле GitLab.

Подробности о полномочиях этого типа см. в секции *Токен персонального доступа GitLab*.

Удаление полномочия

Чтобы удалить полномочие, выполните следующие действия:

1. В таблице полномочий нажмите на ссылку с названием удаляемого полномочия.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Проекты

Окно **Проекты** (Projects) используется для управления *проектами*. Для создания проектов и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Ресурсы* ▸ *Проекты* (*Resources* ▸ *Projects*).

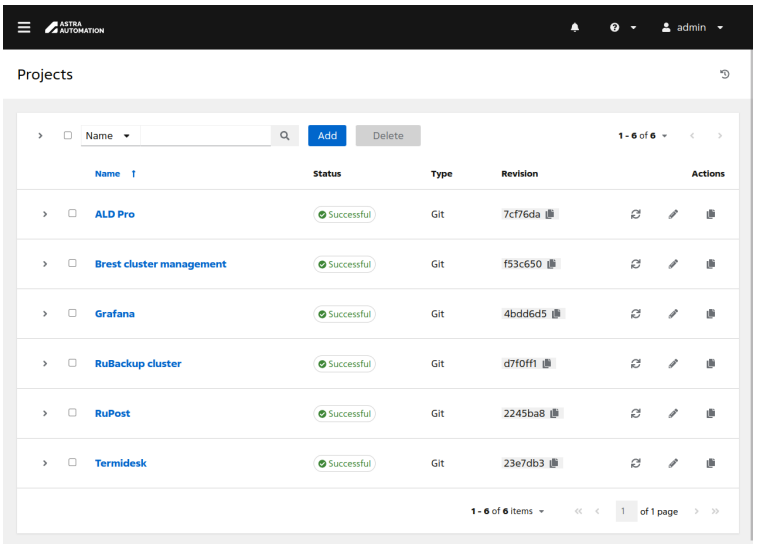


Таблица проектов состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Статус** (Status) – текущее состояние проекта;
- **Тип** (Type) – тип источника кода проекта;
- **Версия** (Revision) – версия кода проекта;
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - синхронизация кода проекта с источником;

- переход в окно редактирования свойств проекта;
- создание копии проекта.

Просмотр

Для получения подробных сведений о проекте нажмите на ссылку с его названием в таблице проектов.

Окно сведений о проекте состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о проекте;
- **Доступ** (Access) – таблица пользователей и назначенных им пользовательских и командных *ролей*;
- **Шаблоны заданий** (Job Templates) – таблица *шаблонов заданий*, использующих код проекта;
- **Уведомления** (Notifications) – таблица уведомлений;
- **Расписание** (Schedules) – таблица заданий по обновлению кода проекта из *источника*.

Примечание: Эта вкладка недоступна для локальных проектов.

Создание

Для создания проекта выполните следующие действия:

1. В окне **Проекты** (Projects) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новый проект** (Create New Project):
   - **Название** (Name) – укажите название проекта.

     Особенности заполнения поля:

     - В рамках одной организации не допускается создание проектов с одним и тем же названием.
     - Название проекта не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использование символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите описание проекта, например, его назначение или особенности использования.
   - **Организация** (Organization) – выберите организацию, которой принадлежит проект.
   - **Среда исполнения** (Execution Environment) – выберите *среду исполнения*, используемую для запуска заданий, созданных на основе проекта.

     Примечание: Это поле становится доступным для редактирования после выбора организации. Среда исполнения, выбранная на уровне проекта, имеет бо-

лее высокий приоритет, чем среды исполнения, выбранные на уровне контроллера и организации.

- **Тип системы управления исходными данными** (Source Control Type) – выберите тип источника кода проекта.

  Особенности работы контроллера с различными типами источников описаны в разделе *Источники кода*.

- **Учетные данные для проверки подписи содержимого** (Content Signature Validation Credential) – если для проверки аутентичности кода проекта следует использовать GPG, выберите соответствующие полномочия типа «Публичный ключ GPG».

3. Заполните поля, соответствующие выбранному источнику кода.

Ручное управление (Manual)

- **Базовый путь проекта** (Project Base Path)

  Значение в этом поле недоступно для изменения. Оно указывает на каталог, в котором выполняется поиск проекта.

- **Каталог playbook** (Playbook Directory) – выберите локальный каталог с кодом проекта.

Git

- **Тип системы управления исходными данными** (Source Control Type) – укажите ссылку на репозиторий Git, в котором хранится код проекта.
- **Ветка/тег/коммит системы управления исходными данными** (Source Control Branch/Tag/Commit) – если требуется использовать ревизию кода, отличную от последнего коммита в основной ветке, укажите в этом поле название нужной ветки, название тега или хэш коммита.
- **Спецификация (Refspec) системы управления исходными данными** (Source Control Refspec) – если требуется использовать специфичную ревизию кода, укажите в этом поле правила получения необходимых ссылок из репозитория Git.
- **Полномочия на систему управления исходными данными** (Source Control Credential)

  Если для доступа к репозиторию требуется аутентификация, выберите в этом поле соответствующие полномочия типа «Управление версиями» (Source Control).

- **Очистить** (Clean)

  Если эта настройка включена, перед обновлением будут сброшены все локальные изменения кода.

- **Удалить** (Delete)

  Если эта настройка включена, при обновлении локальная копия кода проекта будет удалена, после чего загружена заново.

- **Отслеживание подмодулей** (Track Submodules)

  Если эта настройка включена, при обновлении проекта будут также обновлены все его субмодули.

- **Обновить версию при запуске** (Update Revision on Launch)

  Если эта настройка включена, перед запуском заданий на основе проекта его код будет обновляться автоматически.

- **Разрешить переопределение ветки** (Allow Branch Override)

  Если эта настройка включена, в шаблоне задания можно выбрать ветку или версию кода, отличные от указанных в свойствах проекта.

Subversion

- **URL системы управления исходными данными** (Source Control URL) – укажите ссылку на репозиторий Subversion, в котором хранится код проекта.
- **Версия №** (Revision #) – если требуется использовать ревизию кода, отличную от последнего коммита в основной ветке, укажите в этом поле номер нужной ревизии.
- **Полномочия на систему управления исходными данными** (Source Control Credential)

  Если для доступа к репозиторию требуется аутентификация, выберите в этом поле соответствующие полномочия типа «Управление версиями» (Source Control).

- **Очистить** (Clean)

  Если эта настройка включена, перед обновлением будут сброшены все локальные изменения кода.

- **Удалить** (Delete)

  Если эта настройка включена, при обновлении локальная копия кода проекта будет удалена, после чего загружена заново.

- **Обновить версию при запуске** (Update Revision on Launch)

  Если эта настройка включена, перед запуском заданий на основе проекта его код будет обновляться автоматически.

- **Разрешить переопределение ветки** (Allow Branch Override)

  Если эта настройка включена, в шаблоне задания можно выбрать ветку или версию кода, отличные от указанных в свойствах проекта.

Внешний архив (Remote Archive)

- **URL системы управления исходными данными** (Source Control URL) – укажите ссылку на архив, в котором хранится код проекта.
- **Полномочия на систему управления исходными данными** (Source Control Credential)

  Если для доступа к репозиторию требуется аутентификация, выберите в этом поле соответствующие полномочия типа «Управление версиями» (Source Control).
- **Очистить** (Clean)

  Если эта настройка включена, перед обновлением будут сброшены все локальные изменения кода.
- **Удалить** (Delete)

  Если эта настройка включена, при обновлении локальная копия кода проекта будет удалена, после чего загружена заново.
- **Обновить версию при запуске** (Update Revision on Launch)

  Если эта настройка включена, перед запуском заданий на основе проекта его код будет обновляться автоматически.
- **Разрешить переопределение ветки** (Allow Branch Override)

  Если эта настройка включена, в шаблоне задания можно выбрать ветку или версию кода, отличные от указанных в свойствах проекта.

4. Нажмите кнопку *Сохранить* (Save).

После создания проекта его настройки можно изменять с помощью различных групп параметров, описанных в последующих секциях.

Настройка доступа

Во вкладке **Доступ** (Access) выводится список пользователей и назначенных им *ролей*. Нажатие на ссылку с именем пользователя приводит к переходу в окно просмотра сведений о пользователе.

Назначение ролей отдельным пользователям

Чтобы настроить доступ к проекту для отдельных пользователей, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Пользователи* (Users) и нажмите кнопку *Продолжить* (Next).
3. Выберите пользователей, которым хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли, которые хотите назначить выбранным на предыдущем шаге пользователям.
5. Нажмите кнопку *Сохранить* (Save).

Назначение командных ролей

Чтобы настроить доступ к проекту для всех участников одной или нескольких команд, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Команды* (Teams) и нажмите кнопку *Продолжить* (Next).
3. Выберите команды, участникам которых хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли и нажмите кнопку *Сохранить* (Save).

Отзыв ролей

Чтобы отозвать пользовательскую или командную роль, выполните следующие действия:

1. Во вкладке **Доступ** (Access) в строке с нужным пользователем нажмите кнопку × рядом с названием отзываемой роли.
2. Подтвердите отзыв роли.

Примечание: Роли «Системный администратор» (System Administrator) и «Системный аудитор» (System Auditor) нельзя отозвать через настройку доступа на уровне проекта, однако, можно *изменить тип пользователя*.

Удаление

Удаление проектов имеет следующие особенности:

- При удалении проекта связанные с ним шаблоны заданий и инвентарные списки не удаляются и могут быть связаны с новым проектом.
- При удалении проекта, использующего в качестве источника локальный каталог, файлы проекта не удаляется с диска.

Чтобы удалить проект, выполните следующие действия:

1. В таблице проектов нажмите на ссылку с названием удаляемого проекта.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Обновление

Если проект использует внешний источник кода, для поддержания локальной копии в актуальном состоянии необходимо обновлять проект. Astra Automation Controller позволяет обновлять код проектов вручную или автоматически, при этом автоматическое обновление может выполняться при запуске связанных с проектом заданий или по расписанию.

Чтобы обновить проект вручную, во вкладке **Подробности** (Details) нажмите кнопку *Синхронизировать* (Sync).

Совет: Для быстрого запуска обновления проекта можно использовать кнопку *Синхронизировать проект* (Sync Project), доступную в колонке **Действия** (Actions) таблицы проектов.

Чтобы проект автоматически обновлялся перед запуском связанных с ним заданий, в настройках источника включите соответствующие флаги в блоке **Опции** (Options).

Чтобы проект обновлялся по расписанию, настройте его следующим образом:

1. Во вкладке **Расписание** (Schedules) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новое расписание** (Create New Schedule):
   - **Название** (Name) – укажите название расписания.
   - **Описание** (Description) – укажите описание расписания.
   - **Дата и время начала** (Start date/time) – укажите дату и время первого запуска задания.
   - **Локальный часовой пояс** (Local time zone) – выберите часовой пояс, по которому рассчитывается время.

     Значение по умолчанию – UTC.
   - **Частота повторения** (Repeat frequency) – выберите частоту повторения запусков задания.

     Если выбрано значение **Нет (запускается один раз)** (None (run once)), задание будет выполнено один раз.

     Включение флагов **Мин** (Min), **Час** (Hour), **День** (Day), **Неделя** (Week), **Месяц** (Month) и **Год** (Year) приводит к добавлению в форму наборов полей, позволяющих выполнить настройку соответствующих временных интервалов. Для всех типов интервалов доступны следующие настройки:
     - **Выполнять каждые** (Run every) – количество выбранных временных интервалов между запусками задания.
     - **Завершено** (End) – условие прекращения запуска задания по расписанию. Доступны следующие настройки:
       * **Никогда** (Never) – не прекращать.
       * **После указанного количества вхождений** (After number of occurences) – после указанного количества запусков.
       * **На дату** (On date) – в указанные день и время.

Для недельных интервалов доступна дополнительная настройка **В дни** (Days). Она позволяет выбрать дни недели, по которым следует запускать задание.

Для месячных интервалов доступна дополнительная настройка **Выполнить в** (Run on). Она позволяет выбрать определенную дату или относительную позицию дня в месяце. Например, при выборе даты и указании числа «20» задание будет запускаться двадцатого числа каждого месяца. При выборе относительной позиции нет привязки к конкретной дате. Вместо этого настраивается запуск в дни, обозначаемые как, например, «второй понедельник» или «последняя пятница». Также поддерживаются значения, позволяющие выбрать рабочие или нерабочие дни.

Для годовых интервалов доступна дополнительная настройка **Выполнить в** (Run on). Она позволяет выбрать месяц и число либо относительную позицию дня в году. Например, при выборе месяца «Январь» и числа «10» задание будет запускаться каждый год десятого января. При выборе относительной позиции дня нет привязки к конкретной дате. Вместо этого настраивается запуск в дни, обозначаемые как, например, «первый четверг мая» или «последний день февраля». Также поддерживаются значения, позволяющие выбрать рабочие или нерабочие дни.

- **Исключения** (Exceptions) – настройки, предотвращающие запуск задания в указанные минуты, часы, дни, месяцы и годы. Эти настройки работают по тем же принципам, что и настройки частоты повторений.

3. Нажмите кнопку *Сохранить* (Save).

Инвентарь

Окно **Инветарь** (Inventories) используется для управления *инвентарными списками*. Для создания инвентарных списков и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Ресурсы* ▸ *Инвентарь* (*Resources* ▸ *Inventories*).

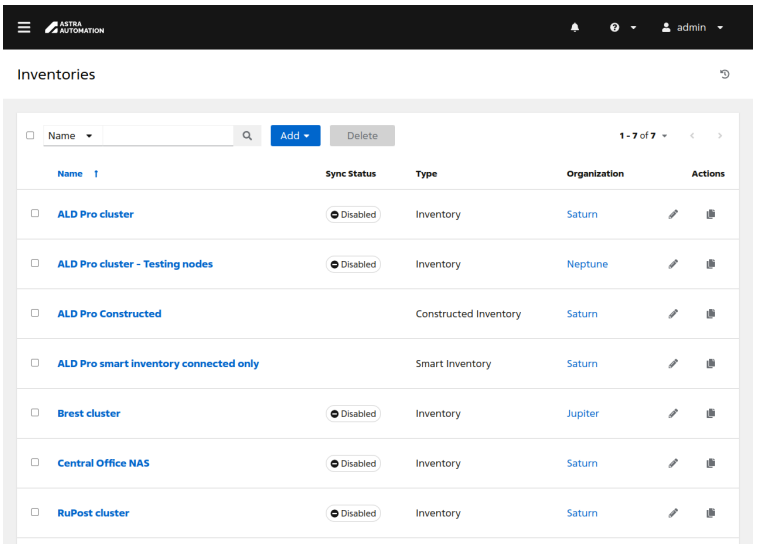


Таблица инвентарных списков состоит из следующих колонок:

- Флаги для выбора нескольких записей.
- **Название** (Name).

- **Состояние синхронизации** (Sync Status).

  Если для формирования инвентаря используется хотя бы один внешний источник, в этом поле выводится статус синхронизации. В противном случае выводится статус **Отключено** (Disabled).

  Для умных и сборных инвентарных списков понятие синхронизации не используется.

- **Тип** (Type) – *тип инвентарного списка*.
- **Организация** (Organization) – организация-владелец.
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - переход в окно редактирования свойств инвентарного списка;
  - создание копии инвентарного списка.

В этом документе приводятся инструкции, общие для инвентарных списков всех типов. Инструкции по управлению инвентарными списками различных типов описаны в соответствующих документах.

Просмотр

Для получения подробных сведений об инвентарном списке нажмите на ссылку с его названием в таблице инвентарных списков. Окно сведений об инвентарном списке состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения об инвентарном списке.
- **Доступ** (Access) – таблица пользователей и назначенных им ролей.
- **Группы** (Groups) – таблица групп узлов.

  Примечание: Эта вкладка недоступна при просмотре сведений об умных инвентарных списках.

- **Управляемые узлы** (Hosts) – список управляемых узлов, входящих в инвентарный список.
- **Источники** (Sources) – список источников сведений об управляемых узлах.

  Примечание: Эта вкладка доступна только при просмотре сведений об обычных инвентарных списках.

- **Задания** (Jobs) – список заданий, связанных с инвентарным списком.
- **Шаблоны заданий** (Job Templates) – список шаблонов заданий, связанных с инвентарным списком.

Настройка доступа

Во вкладке **Доступ** (Access) выводится таблица пользователей и назначенных им *ролей*. Нажатие на ссылку с именем пользователя приводит к переходу в окно просмотра сведений о пользователе.

Назначение ролей отдельным пользователям

Чтобы настроить доступ к инвентарному списку для отдельных пользователей, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Пользователи* (Users) и нажмите кнопку *Продолжить* (Next).
3. Выберите пользователей, которым хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли, которые хотите назначить выбранным на предыдущем шаге пользователям.
5. Нажмите кнопку *Сохранить* (Save).

Назначение командных ролей

Чтобы настроить доступ к инвентарному списку для всех участников одной или нескольких команд, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Команды* (Teams) и нажмите кнопку *Продолжить* (Next).
3. Выберите команды, участникам которых хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли и нажмите кнопку *Сохранить* (Save).

Отзыв ролей

Чтобы отозвать пользовательскую или командную роль, выполните следующие действия:

1. Во вкладке **Доступ** (Access) в строке с нужным пользователем нажмите кнопку × рядом с названием отзываемой роли.
2. Подтвердите отзыв роли.

Примечание: Роли «Системный администратор» (System Administrator) и «Системный аудитор» (System Auditor) нельзя отозвать через настройку доступа на уровне инвентарного списка, однако, можно *изменить тип пользователя*.

Выполнение специальных команд (Ad Hoc)

Для выполнения специальных команд выполните следующие действия:

1. Во вкладке **Управляемые узлы** (Hosts) выберите узлы или во вкладке **Группы** (Groups) выберите группы управляемых узлов, на которых хотите выполнить специальную команду.

   Отфильтровать узлы по названию можно позже.

2. Нажмите кнопку *Выполнить команду* (Run Command).

3. В открывшемся окне заполните форму во вкладке **Подробности** (Details):

   - **Модуль** (Module) – выберите модуль Ansible, используемый для выполнения команды.

     Поддерживаются следующие модули (в алфавитном порядке):

     - apt[135];
     - apt_key[136];
     - apt_repository[137];
     - apt_rpm[138];
     - command[139];
     - group[140];
     - mount[141];
     - ping[142];
     - selinux[143];
     - service[144];
     - setup[145];
     - shell[146];
     - user[147];
     - win_group[148];
     - win_ping[149];

[135] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html
[136] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_key_module.html
[137] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_repository_module.html
[138] https://docs.ansible.com/ansible/latest/collections/community/general/apt_rpm_module.html
[139] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html
[140] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/group_module.html
[141] https://docs.ansible.com/ansible/latest/collections/ansible/posix/mount_module.html
[142] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html
[143] https://docs.ansible.com/ansible/latest/collections/ansible/posix/selinux_module.html
[144] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/service_module.html
[145] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html
[146] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html
[147] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html
[148] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_group_module.html
[149] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_ping_module.html

  - win_service[150];
  - win_updates[151];
  - win_user[152];
  - yum[153].

- **Аргументы** (Arguments) – укажите аргументы, с которыми должна быть выполнена команда из выбранного модуля.

  Примечание: Для некоторых модулей заполнение этого поля обязательно.

- **Степень подробности** (Verbosity) – выберите степень детализации журнала выполнения задания.

  Поддерживаются следующие значения:

  - `0 (Нормальный) (0 Normal)`;
  - `1 (Подробный) (1 Verbose)`;
  - `2 (Более подробный) (2 More Verbose)`;
  - `3 (Отладка) (3 Debug)`;
  - `4 (Отладка подключения) (4 Connection Debug)`;
  - `5 (Отладка WinRM) (5 WinRM Debug)`.

  Значение по умолчанию – `0 (Нормальный) (0 Normal)`.

- **Лимит** (Limit) – укажите шаблон названий управляемых узлов, на которых следует выполнить команду. В качестве разделителя названий узлов используется запятая.

  Если значение не указано, либо равно `all` или `*`, команда выполняется на всех узлах, входящих в инвентарный список.

  Подробности о шаблонах названий узлов см. в документации Ansible[154].

- **Ответвления** (Forks) – укажите количество параллельных процессов, используемых для выполнения задания.

  Если для этого параметра задано значение меньше 1, используется значение по умолчанию, равное 5.

  Максимальное значение этого параметра задается в значении *системной настройки контроллера* **Максимальное количество ответвлений задания** (Maximum number of forks per job).

- **Показать изменения** (Show changes).

  Если эта настройка включена, в журнал выполнения задания выводятся изменения, сделанные на управляемых узлах при выполнении команды (если это поддерживается выбранным модулем или командой).

  Эквивалентно выполнению команды `ansible-playbook` с параметром `--diff`.

[150] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_service_module.html
[151] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_updates_module.html
[152] https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_user_module.html
[153] https://docs.ansible.com/ansible/latest/collections/ansible/builtin/yum_module.html
[154] https://docs.ansible.com/ansible/latest/inventory_guide/intro_patterns.html

- **Включить эскалацию привилегий** (Enable privilege escalation).

  Если эта настройка включена, задание запускается с повышенными привилегиями.

  Эквивалентно запуску команды `ansible-playbook` с параметром `--become`.

  Важно: Запрос пароля суперпользователя не производится. Убедитесь, что на управляемых узлах разрешено выполнение команды sudo без ввода пароля.

  Инструкции по настройке приведены в разделе *Использование sudo без ввода пароля*.

- **Дополнительные переменные** (Extra variables) – укажите дополнительные переменные, значения которых следует использовать при выполнении задания.

  Подробности о порядке разрешения значений переменных см. в разделе *Переменные*.

4. Нажмите кнопку *Продолжить* (Next).
5. Выберите *среду исполнения*, в которой следует запустить задание.
6. Нажмите кнопку *Продолжить* (Next).
7. Выберите полномочия, используемые для доступа к управляемым узлам, и нажмите кнопку *Продолжить* (Next).
8. Во вкладке **Предварительный просмотр** (Preview) проверьте корректность введенных данных и нажмите кнопку *Запустить* (Run).

Удаление

Предупреждение: Особенности удаления инвентарных списков различного типа описаны в подразделе *Особенности удаления*.

Чтобы удалить инвентарный список, выполните следующие действия:

1. В таблице инвентарных списков нажмите на ссылку с названием удаляемого инвентарного списка.
2. Во кладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Обычный инвентарный список

В этой секции приводятся инструкции по работе с *обычными инвентарными списками* через веб-интерфейс.

Создание обычного инвентарного списка

Для создания *обычного инвентарного списка* выполните следующие действия:

1. В окне **Инветарь** (Inventories) нажмите кнопку *Добавить* (Add) и в открывшемся меню выберите пункт **Добавить инвентарь** (Add Inventory).
2. Заполните форму **Создать новый инвентарь** (Create new inventory):
   - **Название** (Name) – укажите название инвентарного списка.

     Особенности заполнения поля:

     - Название инвентарного списка должно быть уникальным на уровне организации.
     - Название инвентарного списка не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использование символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите дополнительные данные об инвентарном списке, например, цель его создания или особенности использования.
   - **Организация** (Organization) – выберите организацию, которой принадлежит инвентарный список.
   - **Группы узлов контроллера** (Instance Groups) – выберите группы узлов, используемые для запуска заданий, связанных с инвентарным списком.

     Подробности о порядке выбора узлов для запуска заданий см. в секции *Выбор группы узлов для запуска заданий*.
   - **Метки** (Labels).

     С помощью меток можно группировать задания, связанные с инвентарным списком. Также метки удобно использовать для поиска нужных записей журнала выполнения заданий или их логической группировки.
   - **Запретить откат группы узлов управления** (Prevent Instance Group Fallback).

     Если эта настройка включена, при запуске заданий, связанных с инвентарным списком, выбор предпочтительных групп узлов исполнения работает следующим образом:

     - Нельзя добавить группы, указанные как предпочтительные в свойствах организации.
     - Если в инвентарном списке список предпочтительных групп узлов пуст, используются глобальные настройки предпочтительных групп узлов.
   - **Переменные** (Variables) – укажите переменные, действующие на уровне инвентарного списка.

     Подробности о порядке присвоения значений переменным см. в разделе *Переменные*.
3. Нажмите кнопку *Сохранить* (Save).

После создания инвентарного списка его настройки можно изменять с помощью различных групп параметров, описанных в последующих секциях.

Управление группами узлов

Обычный инвентарный список имеет полную поддержку групп управляемых узлов и позволяет выполнять с ними следующие операции:

- создание;
- изменение;
- удаление;
- заполнение управляемыми узлами;
- построение иерархии.

Примечание: Группы узлов из источников создаются автоматически при синхронизации.

Для перехода к списку групп узлов и управления ими выполните следующие действия:

1. В таблице инвентарных списков нажмите на ссылку с названием нужного обычного инвентарного списка.
2. Выберите вкладку **Группы** (Groups).

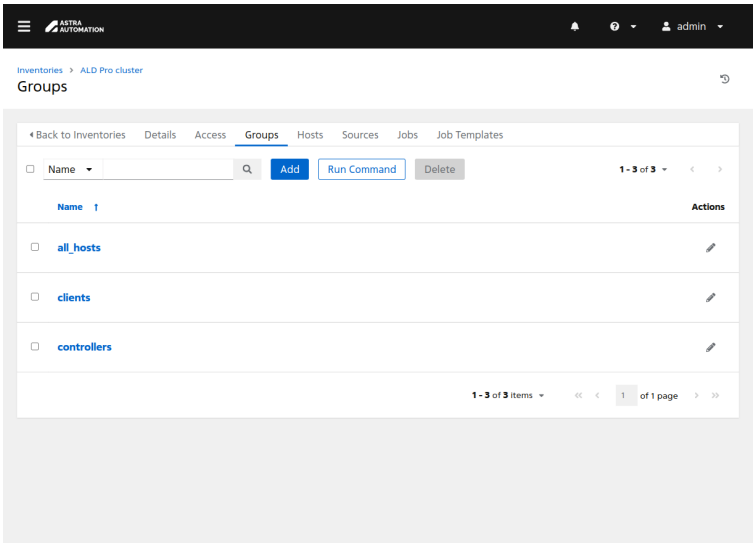


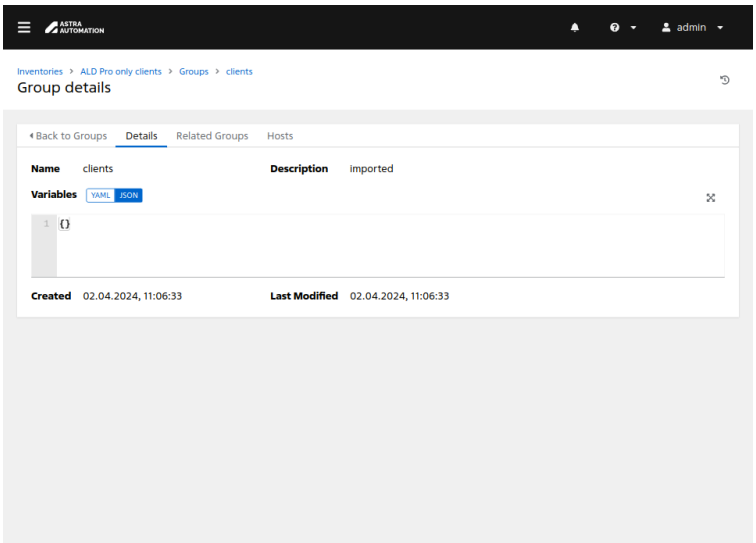
Таблица групп узлов обычного инвентарного списка состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Действия** (Actions) – кнопка перехода в окно редактирования свойств группы узлов.

Просмотр информации о группах узлов

Для просмотра информации о группе управляемых узлов выполните следующие действия:

1. В таблице инвентарных списков нажмите на ссылку с названием нужного обычного инвентарного списка.
2. Выберите вкладку **Группы** (Groups).
3. Нажмите на ссылку с названием нужной группы узлов.



Окно просмотра информации о группе управляемых узлов состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о группе управляемых узлов:
  - **Название** (Name) – название группы.
  - **Переменные** (Variables) – переменные группы.

    Подробности о порядке разрешения значений переменных см. в разделе *Переменные*.
  - **Создано** (Created) – дата и время создания группы узлов, а также ссылка на профиль выполнившего его пользователя.
  - **Последнее изменение** (Last Modified) – дата и время последнего изменения группы узлов, а также ссылка на профиль выполнившего его пользователя.
- **Связанные группы** (Related Groups) – таблица дочерних групп узлов.

  Таблица дочерних групп узлов состоит из следующих колонок:
  - флаги для выбора нескольких записей;
  - **Название** (Name) – ссылка на дочернюю группу;
  - **Действия** (Actions) – кнопка изменения свойств дочерней группы.
- **Управляемые узлы** (Hosts) – таблица управляемых узлов, входящих в группу.

  Таблица управляемых узлов группы состоит из следующих колонок:
  - Флаги для выбора нескольких записей.

- **Название** (Name) – ссылки для перехода в окно просмотра свойств управляемых узлов.
- **Описание** (Description).
- **Активность** (Activity).

  Для управляемых узлов, сведения о которых добавлены из внешних источников, в этом поле выводится значение `imported`.
- **Действия** (Actions) – кнопки вызова часто выполняемых действий:
  * изменение *статуса управляемого узла*;
  * переход в окно редактирования свойств управляемого узла.

Создание группы узлов

Чтобы создать группу управляемых узлов, выполните следующие действия:

1. Во вкладке **Группы** (Groups) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новую группу** (Create new group):
   - **Название** (Name) – укажите название группы управляемых узлов.

     Предупреждение: Название группы управляемых узлов должно быть уникально в пределах инвентарного списка.

   - **Описание** (Description) – укажите описание группы управляемых узлов.
   - **Переменные** (Variables) – укажите значения переменных, действующих на уровне группы управляемых узлов.

     Подробности о порядке разрешения значений переменных см. в разделе *Переменные*.
3. Нажмите кнопку *Сохранить* (Save).

Создание иерархии групп узлов

Чтобы создать иерархические отношения между группами управляемых узлов, выполните следующие действия:

1. Во вкладке **Группы** (Groups) нажмите на ссылку с названием группы узлов, которая должна быть родительской. Далее эта группа называется родительской группой.
2. В окне просмотра информации о родительской группе выберите вкладку **Связанные группы** (Related Groups).
3. Нажмите кнопку *Добавить* (Add) и в открывшемся меню выберите соответствующий пункт:
   - **Добавить существующую группу** (Add existing group) – чтобы сделать дочерними уже существующие группы узлов;
   - **Добавить новую группу** (Add new group) – чтобы создать новую дочернюю группу узлов.

4. Если вы выбрали добавление существующих групп узлов, в открывшемся окне **Выберите группы** (Select Groups) включите флаги в строках с названиями нужных групп узлов и нажмите кнопку *Сохранить* (Save).
5. Если вы выбрали создание новой дочерней группы узлов, заполните форму **Создать новую группу** (Create new group) и нажмите кнопку *Сохранить* (Save).

Удаление группы узлов

Чтобы удалить группу узлов, выполните следующие действия:

1. В таблице инвентарных списков нажмите на ссылку с названием нужного обычного инвентарного списка.
2. Выберите вкладку **Группы** (Groups).
3. В таблице групп управляемых узлов нажмите на ссылку с названием удаляемой группы.
4. В диалоговом окне **Удалить группу?** (Delete Group?) выполните следующие действия:
    1. Выберите действие, которое следует выполнить с дочерними группами и входящими в удаляемую группу узлами:
        - **Удалить все группы и узлы** (Delete All Groups and Hosts) – удалить;
        - **Добавить дочерние узлы и группы** (Promote Child Groups and Hosts) – сохранить.
    2. Нажмите кнопку *Удалить* (Delete).

Управление источниками

Обычный инвентарный список поддерживает получение сведений об управляемых узлах из одного или нескольких внешних источников.

Примечание: Чтобы в контроллере появились сведения об управляемых узлах из источника, после его добавления необходимо запустить синхронизацию.

Для перехода к списку источников и управления ими выполните следующие действия:

1. В таблице инвентарных списков нажмите на ссылку с названием нужного обычного инвентарного списка.
2. Выберите вкладку **Источники** (Sources).

Добавление источника

Чтобы добавить в обычный инвентарный список источник сведений об управляемых узлах, выполните следующие действия:

1. Во вкладке **Источники** (Sources) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новый источник** (Create new source):
   - **Название** (Name) – укажите название источника сведений об управляемых узлах.
   - **Описание** (Description) – укажите описание источника, например, название связанного проекта.
   - **Среда исполнения** (Execution Environment) – укажите среду исполнения, используемую для работы с источником.

     Указанное здесь значение имеет более высокий приоритет, чем среда исполнения, указанная на уровне организации, команды или инвентаря.
   - **Источник** (Source) – выберите тип источника сведений об управляемых узлах.
3. Заполните поля, соответствующие выбранному источнику.

Примечание: Поля **Проект** (Project) и **Инвентарный файл** (Inventory file) доступны только для источников типа «Взято из проекта» (Source from a Project). При этом значение в поле **Инвентарный файл** (Inventory file) можно выбрать только после выбора проекта.

   - **Полномочия** (Credential) – если для доступа к источнику требуется аутентификация, выберите полномочие соответствующего типа в этом поле.
   - **Проект** (Project) – выберите проект, служащий источником сведений об управляемых узлах.
   - **Инвентарный файл** (Inventory file) – выберите файл инвентаря из состава проекта.

     При выборе значения `/ (project root)` источниками сведений об управляемых узлах считаются все файлы инвентаря, которые контроллер найдет в проекте. Это значение рекомендуется выбирать только в том случае, когда инвентарь в проекте хранится в виде иерархической структуры из файлов и каталогов.
   - **Степень подробности** (Verbosity) – укажите степень подробности вывода Ansible в журнал выполнения задания синхронизации инвентаря.

     Поддерживаются следующие значения:

     - `0 (Предупреждение) (0 Warnning)` – выводится информация о событиях со степенью важности не ниже «Предупреждение».
     - `1 (Информация) (1 Info)` – выводится информация обо всех событиях, происходящих при синхронизации инвентаря.
     - `2 (Отладка) (2 Debug)` – выводится вся доступная информация обо всех событиях, происходящих при обработке инвентаря.

     Значение по умолчанию – `1 Информация (1 Info)`.

- **Фильтр узлов** (Hosts Filter) – укажите регулярное выражение, используемое для фильтрации узлов по названию. Этот фильтр применяется последним, после всех остальных фильтров, предоставляемых используемым расширением инвентаря.
- **Включенная переменная** (Enabled Variable) – укажите название переменной из данных об управляемом узле, значение которой следует использовать для определения статуса узла после импорта.
- **Разрешенное значение** (Enabled Value).

  Чтобы управляемый узел импортировался со статусом «Вкл» (On), должны выполниться два условия:

  – в свойствах управляемого узла есть переменная с названием, указанным в поле **Включенная переменная** (Enabled Variable);

  – значение этой переменной совпадает с указанным в поле **Разрешенное значение** (Enabled Value).

  Если такой переменной нет или ее значение отличается от указанного в этом поле, узел импортируется со статусом «Выкл» (Off).
- **Перезаписать** (Overwrite).

  Если эта настройка включена, синхронизация с источником работает следующим образом:

  – Импортированные ранее узлы и группы, удаленные в источнике, удаляются из инвентарного списка.

  – Узлы и группы, не привязанные к источнику, перемещаются в первую найденную группу, созданную вручную. Если такой группы нет, все не привязанные к источнику узлы и группы остаются в группе по умолчанию (`all`).

  Если эта настройка выключена, при синхронизации из инвентарного списка **не** удаляются узлы и группы, удаленные в источнике.
- **Перезаписывать переменные** (Overwrite variables).

  Если эта настройка включена, при синхронизации с источником значения переменных инвентаря, хранящихся в контроллере, удаляются и заменяются значениями из источника.

  Если эта настройка выключена, при синхронизации с источником хранящиеся в контроллере значения переменных объединяются со значениями в источнике.

Примечание: Для одноименных переменных значение из источника заменяет значение, хранящееся в контроллере.

- **Обновление при запуске** (Update on launch).

  Если эта настройка включена, при запуске связанного с инвентарем задания автоматически запускается задание синхронизации инвентаря с источником.
- **Таймаут кэширования (в секундах)** (Cache timeout, seconds).

Примечание: Эта настройка доступна только если включен флаг **Обновление при запуске** (Update on launch)

Время, в течение которого с момента последней синхронизации данные об управляемых узлах считаются свежими.

Эта настройка может быть полезна, если задания обновления инвентаря создаются быстрее, чем выполняются. Если данные в кэше свежие, задание синхронизации считается успешно выполненным без запуска.

При значении `0` задания синхронизации запускаются всегда.

Значение по умолчанию – `0`.

- **Исходные переменные** (Source variables).

  При использовании источника типа «Взято из проекта» (Sourced from a Project) переменные объединяются с переменными инвентаря.

  При использовании источников других типов в этом поле задаются настройки соответствующего расширения инвентаря.

  Ссылки на документацию расширений инвентаря см. в подразделе *Поддержка расширений*.

4. Нажмите кнопку *Сохранить* (Save).

Синхронизация списка узлов с источником

Синхронизация используется для первичного импорта сведений об управляемых узлах в контроллер и для актуализации сведений об импортированных ранее узлах.

Чтобы запустить синхронизацию инвентарного списка с источниками, во вкладке **Источники** (Sources) выполните следующие действия:

- Если нужно синхронизировать инвентарный список со всеми источниками, нажмите кнопку *Синхронизировать все* (Sync all).
- Если нужно синхронизировать инвентарный список только с определенным источником, выполните следующие действия:
  1. В таблице со списком источников нажмите на ссылку с названием нужного источника.
  2. В окне свойств источника нажмите кнопку *Синхронизировать* (Sync).
  3. Чтобы отслеживать ход выполнения задания синхронизации, нажмите на ссылку в поле **Статус** (Status).

Настройка расписания синхронизации списка узлов с источником

Чтобы синхронизация с источником выполнялась по расписанию, во вкладке **Источники** (Sources) выполните следующие действия:

1. В таблице со списком источников нажмите на ссылку с названием нужного источника.
2. Во вкладке **Расписание** (Schedules) нажмите кнопку *Добавить* (Add).
3. Заполните форму **Создать новое расписание** (Create New Schedule):
   - **Название** (Name) – укажите название расписания.
   - **Описание** (Description) – укажите описание расписания.

- **Дата и время начала** (Start date/time) – укажите дату и время первого запуска задания.
- **Локальный часовой пояс** (Local time zone) – выберите часовой пояс, по которому рассчитывается время.

  Значение по умолчанию – UTC.
- **Частота повторения** (Repeat frequency) – выберите частоту повторения запусков задания.

  Если выбрано значение **Нет (запускается один раз)** (None (run once)), задание будет выполнено один раз.

  Включение флагов **Мин** (Min), **Час** (Hour), **День** (Day), **Неделя** (Week), **Месяц** (Month) и **Год** (Year) приводит к добавлению в форму наборов полей, позволяющих выполнить настройку соответствующих временных интервалов. Для всех типов интервалов доступны следующие настройки:

  - **Выполнять каждые** (Run every) – количество выбранных временных интервалов между запусками задания.
  - **Завершено** (End) – условие прекращения запуска задания по расписанию. Доступны следующие настройки:
    - **Никогда** (Never) – не прекращать.
    - **После указанного количества вхождений** (After number of occurences) – после указанного количества запусков.
    - **На дату** (On date) – в указанные день и время.

  Для недельных интервалов доступна дополнительная настройка **В дни** (Days). Она позволяет выбрать дни недели, по которым следует запускать задание.

  Для месячных интервалов доступна дополнительная настройка **Выполнить в** (Run on). Она позволяет выбрать определенную дату или относительную позицию дня в месяце. Например, при выборе даты и указании числа «20» задание будет запускаться двадцатого числа каждого месяца. При выборе относительной позиции нет привязки к конкретной дате. Вместо этого настраивается запуск в дни, обозначаемые как, например, «второй понедельник» или «последняя пятница». Также поддерживаются значения, позволяющие выбрать рабочие или нерабочие дни.

  Для годовых интервалов доступна дополнительная настройка **Выполнить в** (Run on). Она позволяет выбрать месяц и число либо относительную позицию дня в году. Например, при выборе месяца «Январь» и числа «10» задание будет запускаться каждый год десятого января. При выборе относительной позиции дня нет привязки к конкретной дате. Вместо этого настраивается запуск в дни, обозначаемые как, например, «первый четверг мая» или «последний день февраля». Также поддерживаются значения, позволяющие выбрать рабочие или нерабочие дни.

  - **Исключения** (Exceptions) – настройки, предотвращающие запуск задания в указанные минуты, часы, дни, месяцы и годы. Эти настройки работают по тем же принципам, что и настройки частоты повторений.

4. Нажмите кнопку *Сохранить* (Save).

Удаление источника

Предупреждение: При удалении источника удаляются все импортированные из него управляемые узлы.

Чтобы удалить источник, выполните следующие действия:

1. В таблице инвентарных списков нажмите на ссылку с названием нужного обычного инвентарного списка.
2. Выберите вкладку **Источники** (Sources).
3. В таблице источников нажмите на ссылку с названием удаляемого источника.
4. В окне свойств источника нажмите кнопку *Удалить* (Delete).
5. Подтвердите удаление источника.

Умный инвентарный список

В этой секции приводятся инструкции по работе с *умными инвентарными списками* через веб-интерфейс.

Важно: Эта часть документации находится в стадии разработки.

Сборный инвентарный список

В этой секции приводятся инструкции по работе со *сборными инвентарными списками* через веб-интерфейс.

Создание сборного инвентарного списка

Для создания сборного инвентаря выполните следующие действия:

1. В окне **Инветарь** (Inventories) нажмите на кнопку *Добавить* (Add) и в открывшемся меню выберите пункт **Добавить сборный инвентарь** (Add constructed inventory).
2. Заполните форму **Создать новый сборный инвентарь** (Create new constructed inventory):
   - **Название** (Name) – укажите название инвентарного списка.

     Особенности заполнения поля:

     - Название инвентарного списка должно быть уникальным на уровне организации.
     - Название инвентарного списка не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использование символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите дополнительные данные об инвентарном списке, например, цель его создания или особенности использования.

- **Организация** (Organization) – выберите организацию, которой принадлежит инвентарный список.
- **Группы узлов контроллера** (Instance Groups) – выберите группы узлов, используемые для запуска заданий, связанных с инвентарным списком.

  Подробности о порядке выбора узлов для запуска заданий см. в секции *Выбор группы узлов для запуска заданий*.
- **Входные данные инвентаря** (Input Inventories) – выберите инвентарные списки, которые хотите использовать для получения сведений об управляемых узлах.
- **Таймаут кэширования (в секундах)** (Cache timeout (seconds)).

  При синхронизации сборного инвентаря автоматически создается источник сведений об управляемых узлах. Эта настройка задает период времени, в течение которого данные в источнике считаются свежими.

  Эта настройка может быть полезна, если задания обновления инвентаря создаются быстрее чем выполняются. Если данные в кэше свежие, задание синхронизации считается успешно выполненным без запуска.

  При значении `0` задания синхронизации запускаются всегда.

  Значение по умолчанию – `0`.
- **Степень подробности** (Verbosity) – выберите степень детализации журнала выполнения задания синхронизации инвентарного списка.

  Поддерживаются следующие значения:

  - `0 (Нормальный) (0 Normal)`;
  - `1 (Подробный) (1 Verbose)`;
  - `2 (Более подробный) (2 More Verbose)`;
  - `3 (Отладка) (3 Debug)`;
  - `4 (Отладка подключения) (4 Connection Debug)`;
  - `5 (Отладка WinRM) (5 WinRM Debug)`.

  Значение по умолчанию – `0 (Нормальный) (0 Normal)`.
- **Лимит** (Limit) – укажите выражение, используемое для фильтрации управляемых узлов.

  Важно: Для использования параметра `limit` требуется версия контроллера не ниже `1.0-upd2` и версия среды исполнения Control Plane Execution Environment не ниже `0.5.1`.

- **Исходные переменные** (Source vars) – укажите параметры используемого расширения Ansible и привила его использования.

3. Нажмите кнопку *Сохранить* (Save).
4. Во вкладке **Подробности** (Details) нажмите кнопку *Синхронизировать* (Sync).

Управляемые узлы

Окно **Управляемые узлы** (Hosts) используется для управления сведениями об управляемых узлах и запуска на них специальных (Ad-Hoc) команд. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Ресурсы ▸ Управляемые узлы* (*Resources ▸ Hosts*).

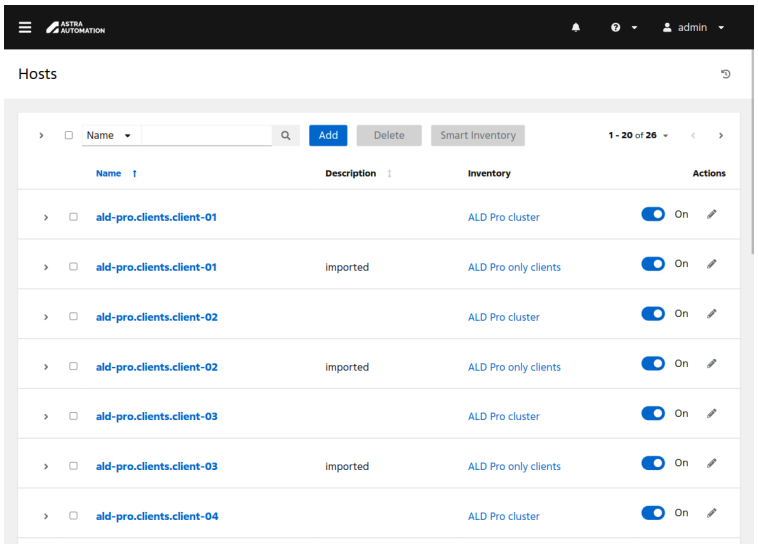


Таблица управляемых узлов состоит из следующих колонок:

- Переключатели вывода информации о статусах заданий, связанных с управляемым узлом.

  При нажатии на переключатель выводятся иконки, показывающие статус выполнения заданий, связанных с управляемым узлом. Нажатие на иконку приводит к переходу в окно просмотра вывода Ansible при выполнении задания.

- Флаги для выбора нескольких записей.
- **Название** (Name) – ссылка на окно просмотра свойств управляемого узла.
- **Описание** (Description).

  Для управляемых узлов, сведения о которых добавлены из внешних источников, в этом поле выводится значение `imported`.

- **Инвентарь** (Inventory) – ссылка на окно просмотра свойств *инвентарного списка*, в который входит управляемый узел.
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - переключатель *статуса управляемого узла*;
  - кнопка перехода в окно редактирования свойств управляемого узла.

Просмотр

Для получения подробных сведений об управляемом узле нажмите на ссылку с его названием в таблице управляемых узлов. Окно сведений об управляемом узле состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения об управляемом узел.
- **Факты** (Facts) – факты Ansible.

  Примечание: Факты доступны для просмотра после сбора.

- **Группы** (Groups) – таблица групп узлов, в которые входит управляемый узел. Состоит из следующих колонок:
  - флаги для выбора нескольких записей;
  - **Название** (Name) – ссылка на окно просмотра свойств группы узлов;
  - **Действия** (Actions) – кнопка перехода в окно редактирования свойств группы узлов.
- **Задания** – таблица заданий, связанных с управляемым узлом. Состоит из следующих колонок:
  - переключатель подробности вывода основных сведений о задании;
  - флаги для выбора нескольких записей;
  - **Название** (Name) – ссылка на окно просмотра журнала выполнения задания;
  - **Статус** (Status) – текущий статус выполнения задания;
  - **Время начала** (Start Time) – время запуска задания;
  - **Время завершения** (Finish Time) – время завершения, отмены или прерывания задания;
  - **Действия** (Actions) – кнопка перезапуска задания.

Создание

Для создания записи об управляемом узле выполните следующие действия:

1. В окне **Управляемые узлы** (Hosts) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новый узел** (Create New Host):
   - **Название** (Name) – укажите название управляемого узла.
   - **Описание** (Description) – укажите дополнительные сведения об управляемом узле.
   - **Инвентарь** (Inventory) – выберите *обычный инвентарный список*, в состав которого добавляется управляемый узел.
   - **Переменные** (Variables) – укажите переменные, действующие на уровне управляемого узла.

     Подробности о порядке присвоения значений переменным см. в разделе *Переменные*.

3. Нажмите кнопку *Сохранить* (Save).

Включение в группу

Управляемые узлы можно включать только в существующие группы в том же инвентарном списке, которому принадлежат сами узлы. Пошаговые инструкции по управлению группами управляемых узлов представлены в документе *Управление группами узлов*.

Чтобы включить управляемый узел в группы узлов, выполните следующие действия:

1. Во вкладке **Группы** (Groups) нажмите кнопку *Добавить* (Add).
2. В открывшемся окне **Выберите группы** (Select Groups) включите флаги в строках с теми группами узлов, в которые хотите включить управляемый узел.
3. Нажмите кнопку *Сохранить* (Save).

Исключение из группы

Чтобы исключить управляемый узел из групп узлов, выполните следующие действия:

1. Во вкладке **Группы** (Groups) включите флаги для групп, из которых хотите исключить управляемый узел.
2. Нажмите кнопку *Отвязать* (Disassociate).
3. Подтвердите исключение узла из групп узлов.

Удаление

Чтобы удалить запись об управляемом узле, выполните следующие действия:

1. В таблице управляемых узлов нажмите на ссылку с названием удаляемого управляемого узла.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление записи об управляемом узле.

Доступ

Раздел панели навигации **Доступ** (Access) содержит подразделы, используемые для управления организациями, пользователями и командами.

Организации

Окно **Организации** (Organizations) используется для управления *организациями*. Организация объединяет в себе множество пользователей и команд, а также является владельцем различных ресурсов. Для создания организаций и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Доступ* ‣ *Организации* (*Access* ‣ *Organizations*).

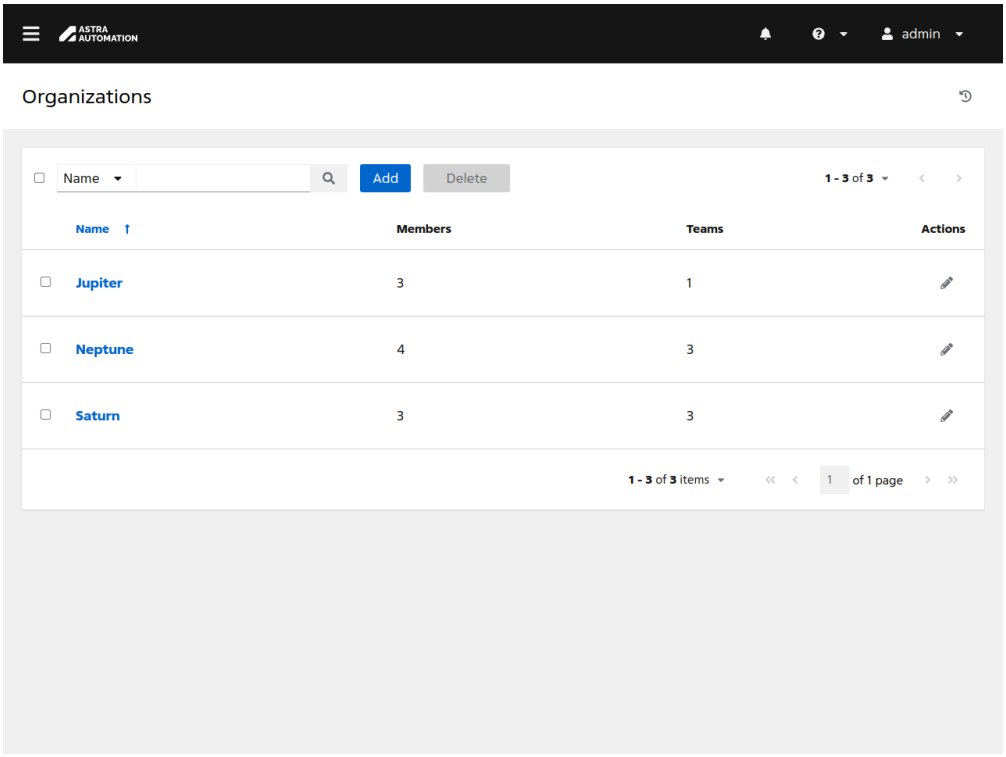


Таблица организаций состоит из следующих колонок:

- Флаги для выбора нескольких записей.
- **Название** (Name).
- **Участники** (Members).

  При расчете числа участников не учитываются пользователи, которым не назначена роль «Участник» (Member). Однако, эти пользователи имеют на управление ресурсами организации привилегии, предоставляемые остальными назначенными ролями.
- **Команды** (Teams) – количество команд, связанных с организацией.
- **Действия** (Actions) – кнопка перехода в окно редактирования свойств организации.

Просмотр

Для получения подробных сведений об организации нажмите на ссылку с ее названием в таблице организаций.

Окно сведений об организации состоит из следующих вкладок:

- **Подробности** (Details)– общие сведения об организации;
- **Доступ** (Access) – список пользователей и назначенных им ролей;
- **Команды** (Teams) – список команд организации;
- **Среды исполнения** (Execution Environments) – список сред исполнения, принадлежащих организации;
- **Уведомления** (Notifications) – список уведомлений, связанных с организацией.

Создание

Для создания организации выполните следующие действия:

1. В окне **Организации** (Organizations) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новую организацию** (Create New Organization):
   - **Название** (Name) – укажите название организации, например, `Jupiter`.

     Особенности заполнения поля:

     - Не допускается создание двух организаций с одним и тем же названием.
     - Название организации не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использование символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите описание организации.

     В описании можно указать дополнительные данные об организации, используемые при администрировании контроллера. Например, контакты ответственных лиц или назначение организации.
   - **Среда исполнения** – выберите *среду исполнения*, которую следует использовать на уровне организации по умолчанию. Эта среда исполнения имеет больший приоритет, чем среда исполнения, заданная в *общесистемных настройках контроллера*.
   - **Учетные данные Galaxy** (Galaxy Credentials) – выберите полномочия, используемые для подключения к Ansible Galaxy.

     Значение в этом поле формируется автоматически. Однако, если вы хотите использовать для доступа к Ansible Galaxy собственные учетные данные, укажите их в значении этого поля.
3. Нажмите кнопку *Сохранить* (Save).

Удаление

Предупреждение: Вместе с организацией удаляются связанные с ней ресурсы. Принадлежащие организации пользователи и команды при этом не удаляются, а открепляются от удаляемой организации и могут быть в дальнейшем привязаны к другой.

Чтобы удалить организацию, выполните следующие действия:

1. В окне **Организации** (Organizations) нажмите на ссылку с названием удаляемой организации.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Настройка доступа

Во вкладке **Доступ** (Access) выводится список связанных с организацией *пользователей* и назначенных им *ролей*. Нажатие на ссылку с именем пользователя приводит к переходу в окно просмотра сведений о пользователе.

Назначение ролей

Чтобы назначить пользователям или командам роли на уровне организации, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов, которым назначаются роли:
   - *Пользователи* (Users) – если вы хотите назначить роли одному или нескольким пользователям;
   - *Команды* (Teams) – если вы хотите назначить роли всем пользователям команд.
3. Нажмите кнопку *Продолжить* (Next).
4. Выберите пользователей или команды, которым хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
5. Выберите роли, которые хотите назначить выбранным на предыдущем шаге пользователям или командам.

   Примечание: При *связывании пользователя и команды* выбранные командные роли будут назначены пользователю автоматически.

6. Нажмите кнопку *Сохранить* (Save).

Отзыв ролей

Чтобы отозвать роль у пользователя, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку × рядом с названием отзываемой роли.
2. Подтвердите отзыв.

Примечание: Роли «Системный администратор» (System Administrator) и «Системный аудитор» (System Auditor) нельзя отозвать через настройку доступа на уровне организации. Используйте для этого изменение свойств пользователя.

Уведомления

Во вкладке **Уведомления** (Notifications) выводится список уведомлений, связанных с организацией. Для каждого уведомления доступны переключатели, управляющие типом событий, при наступлении которых должно быть отправлено уведомление:

- **Согласование** (Approval) – согласован переход к очередному заданию в потоке заданий;
- **Пуск** (Start) – запущено задание или поток заданий;
- **Успех** (Success) – успешно выполнено задание или поток заданий;
- **Сбой** (Failure) – при выполнении задания произошла ошибка, для которой не предусмотрен обработчик.

Примечание: Если уведомления для организации не настроены, отображается пустой список. Для создания уведомлений и управления ими выберите на панели навигации *Администрирование* ▸ *Уведомления* (*Administration* ▸ *Notifications*).

Примеры

Изучите управление организациями на примерах.

Назначение роли отдельному пользователю

Пусть пользователю `alex` необходимо назначить роль, позволяющую управлять всеми проектами организации. Для этого в окне организации выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. В открывшемся окне выберите *Пользователи* (Users) и нажмите кнопку *Продолжить* (Next).
3. Из списка пользователей выберите `alex` и нажмите кнопку *Продолжить* (Next).
4. Из списка ролей выберите **Администратор проекта** (Project Admin).
5. Нажмите кнопку *Сохранить* (Save).

Назначение ролей команде

Пусть для управления инфраструктурой используется команда с названием `DevOps Team`. Всем ее участникам необходимо предоставить следующие привилегии:

- управление средами исполнения;
- управление проектами.

Для этого в окне организации выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. В открывшемся окне выберите *Команды* (Teams) и нажмите кнопку *Продолжить* (Next).

3. Из списка команд выберите `DevOps Team` и нажмите кнопку *Продолжить* (Next).
4. Из списка ролей выберите **Администратор проекта** (Project Admin) и **Администратор среды исполнения** (Execution Environment Admin).
5. Нажмите кнопку *Сохранить* (Save).

Пользователи

Окно **Пользователи** (Users) используется для управления *пользователями*. Для создания пользователей и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Доступ* ▸ *Пользователи* (*Access* ▸ *Users*).

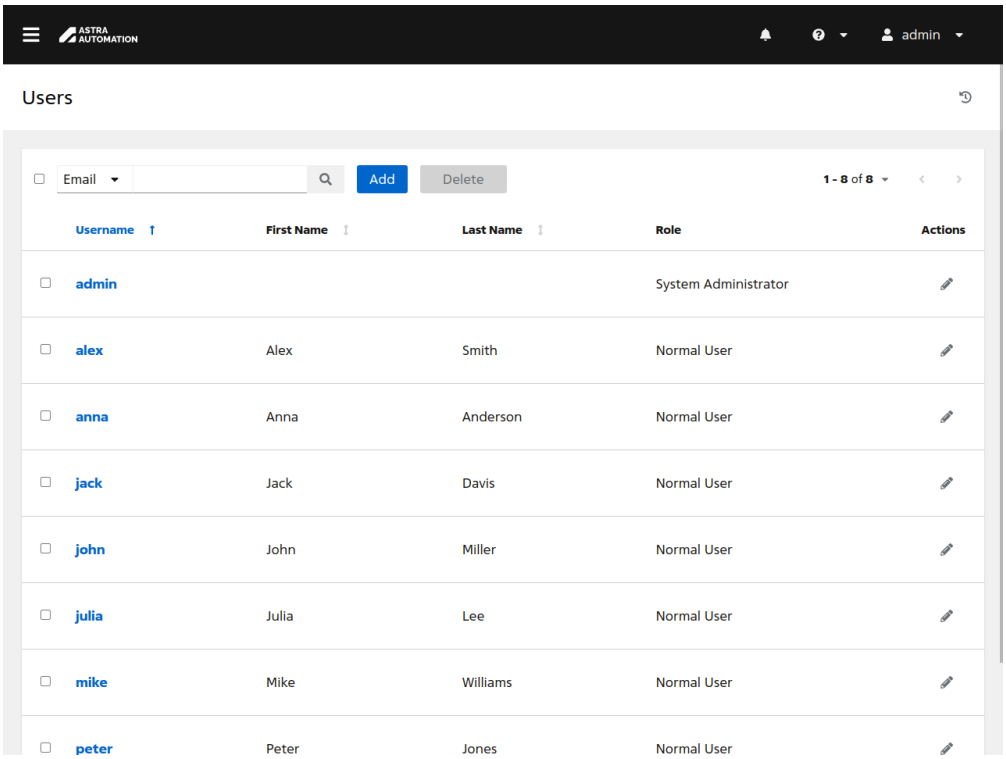


Таблица пользователей состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Имя пользователя** (Username) – уникальное имя пользователя, используемое для работы с контроллером;
- **Имя** (First Name) – имя пользователя;
- **Фамилия** (Last Name) – фамилия пользователя;
- **Роль** (Role) – тип пользователя;
- **Действия** (Actions) – кнопка перехода в окно редактирования сведений о пользователе.

Просмотр

Для получения подробных сведений о пользователе нажмите на ссылку с его именем в таблице пользователей.

Окно сведений о пользователе состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о пользователе.
- **Организации** (Organizations) – список организаций, участником которых является пользователь.

  Чтобы сделать пользователя участником организации, *назначьте ему роль* «Участник» (Member) на ее уровне.
- **Команды** (Teams) – список команд, участником которых является пользователь.

  Чтобы сделать пользователя участником команды, *свяжите его с нужной командой*.
- **Токены** (Tokens) – список токенов доступа, созданных пользователем.

  Примечание: Эта вкладка доступна только при просмотре собственной учетной записи.

Создание

Для создания пользователя выполните следующие действия:

1. В окне **Пользователи** (Users) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать нового пользователя**:
   - **Имя** (First Name) – укажите имя пользователя.
   - **Фамилия** (Last Name) – укажите фамилию пользователя.
   - **Электронная почта** (Email) – укажите адрес электронной почты пользователя.
   - **Имя пользователя** (Username) – укажите имя пользователя, используемое для работы с контроллером.

     Требования к имени пользователя:

     - Максимально допустимая длина – 150 символов.
     - Разрешено использование букв, цифр и символов `_`, `@`, `+`, `.`, `-`.
     - Уникальность на уровне контроллера.
   - **Пароль** (Password) и **Подтвердить пароль** (Confirm Password) – укажите пароль и подтверждение пароля пользователя.

     Единственное требование к паролю и его подтверждению – длина не менее 8 символов.
   - **Тип пользователя** (User Type) – выберите тип пользователя.

     Подробности о типах пользователей см. в разделе *Типы пользователей*.
   - **Организация** (Organization) – выберите организацию, которой принадлежит пользователь.
3. Нажмите кнопку *Сохранить* (Save).

Удаление

Чтобы удалить пользователя, выполните следующие действия:

1. В таблице пользователей нажмите на ссылку с именем удаляемого пользователя.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Настройка доступа

Окно **Пользователи** (Users) предоставляет функциональность, позволяющую выполнять с пользователями следующие действия:

- связывать с командами;
- назначение индивидуальные роли на доступ к экземплярам ресурсов.

Связывание с командой

Чтобы связать пользователя с командой, выполните следующие действия:

1. Во вкладке **Команды** (Teams) нажмите кнопку *Связать* (Associate).
2. Выберите команды, с которыми хотите связать пользователя, и нажмите кнопку *Сохранить* (Save).

Назначение индивидуальных ролей

Чтобы назначить пользователю роль, выполните следующие действия:

1. Во вкладке **Роли** (Roles) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов.

   Поддерживаются следующие значения:

   - *шаблоны заданий (job templates)*;
   - *шаблоны потока заданий (workflow job templates)*;
   - полномочия (credentials);
   - *инвентарь (inventories)*;
   - *проекты (projects)*;
   - *организации (organizations)*;
   - группы узлов контроллера (instance groups).
3. Нажмите кнопку *Продолжить* (Next).
4. Выберите экземпляры объектов и нажмите кнопку *Продолжить* (Next).
5. Выберите роли. Список доступных ролей зависит от выбранного ранее типа объектов.
6. Нажмите кнопку *Сохранить* (Save).

Отзыв ролей

Чтобы отозвать у пользователя роль, выполните следующие действия:

1. Во вкладке **Роли** (Roles) нажмите кнопку × рядом с названием отзываемой роли.
2. Подтвердите отзыв.

Смена пароля

Чтобы изменить пароль пользователя, выполните следующие действия:

1. В таблице пользователей нажмите на ссылку с именем нужного пользователя.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Редактировать* (Edit).
3. Укажите новый пароль в полях **Пароль** (Password) и **Подтвердить пароль** (Confirm Password).
4. Нажмите кнопку *Сохранить* (Save).

Смена типа пользователя

Чтобы изменить *тип пользователя*, выполните следующие действия:

1. В таблице пользователей нажмите на ссылку с именем нужного пользователя.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Редактировать* (Edit).
3. В поле **Тип пользователя** (User Type) выберите нужное значение.
4. Нажмите кнопку *Сохранить* (Save).

Управление токенами

Токены позволяют сторонним приложениям получить доступ к контроллеру от имени выбранного пользователя. Каждый пользователь создает свои токены самостоятельно.

Для управления токенами выполните следующие действия:

1. Войдите в веб-интерфейс контроллера от имени нужного пользователя.
2. На панели навигации выберите *Доступ* ‣ *Пользователи* (*Access* ‣ *Users*).
3. Нажмите на ссылку с именем активного пользователя.

   Совет: Для быстрого перехода в нужное окно также можно выбрать в пользовательском меню пункт **Сведения о пользователе** (User Details).

4. Выберите вкладку **Токены** (Tokens).

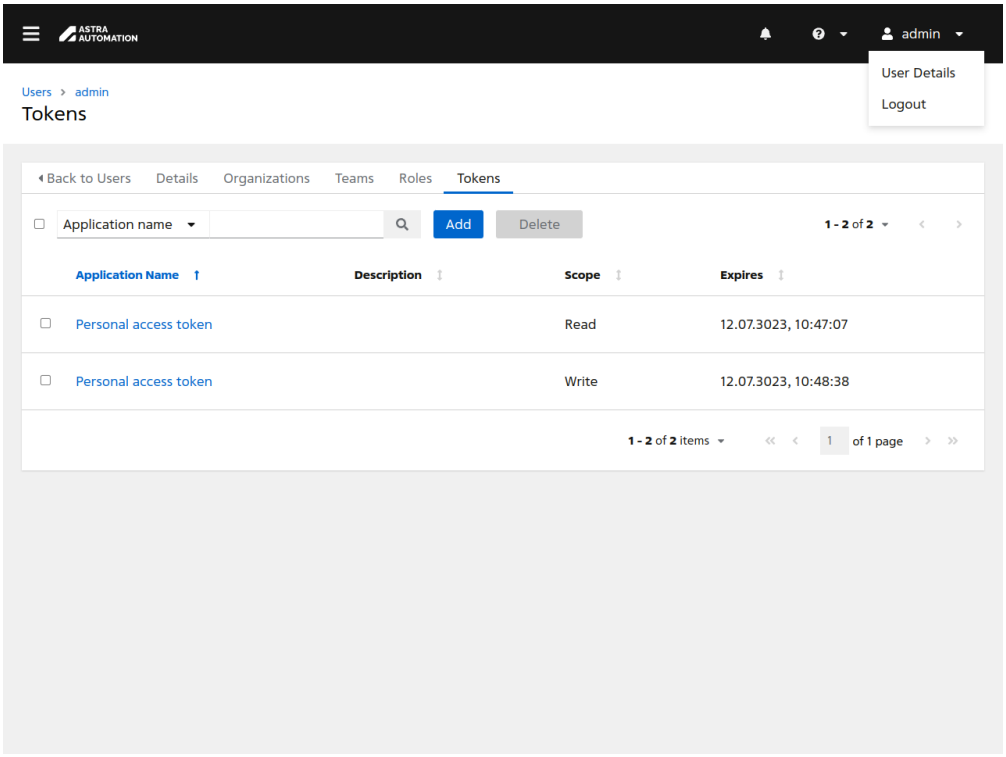


Таблица токенов состоит из следующих колонок:

- **Название приложения** (Application Name) – название приложения, связанного с токеном. Если с токеном не связано ни одно приложение, выводится название **Личный токен доступа** (Personal access token).
- **Описание** (Description) – описание токена.
- **Область** (Scope) – режим доступа к данным контроллера, предоставляемый токеном.
- **Истекает** (Expires) – дата и время истечения срока действия токена.

  Для изменения срока действия токенов измените значение настройки **Истечение срока действия токена доступа** (Access Token Expiration) в *общих параметрах аутентификации*.

Создание токена

Для создания токена выполните следующие действия:

1. Во вкладке **Токены** (Tokens) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать пользовательский токен** (Create user token):
   - **Приложение** (Application) – выберите приложение, для которого создается токен.

Примечание: Приложение должно быть предварительно добавлено в список приложений.

- **Описание** (Description) – добавьте краткое описание токена, например, название приложения, для которого он создается.
- **Область** (Scope) – выберите режим доступа к данным контроллера, предоставляемый токеном:
  - **Чтение** (Read);
  - **Запись** (Write).

3. Нажмите кнопку *Сохранить* (Save).
4. В открывшемся диалоговом окне скопируйте значение из поля **Токен** (Token) в буфер обмена и сохраните в надежное место

Важно: После закрытия диалогового окна получить доступ к значению токена невозможно.

---

5. Закройте окно **Информация о токене** (Token information).

Удаление токена

Предупреждение: При удалении токена все приложения, которые его используют, теряют доступ к контроллеру.

Для удаления пользовательского токена выполните следующие действия:

1. Во вкладке **Токены** (Tokens) нажмите на ссылку с названием удаляемого токена.
2. Нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Команды

Окно **Команды** (Teams) используется для управления *командами*. Команда объединяет в себе множество пользователей и используется для разграничения доступа к существующим ресурсам. Для создания команд и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Доступ* ▸ *Команды* (*Access* ▸ *Teams*).

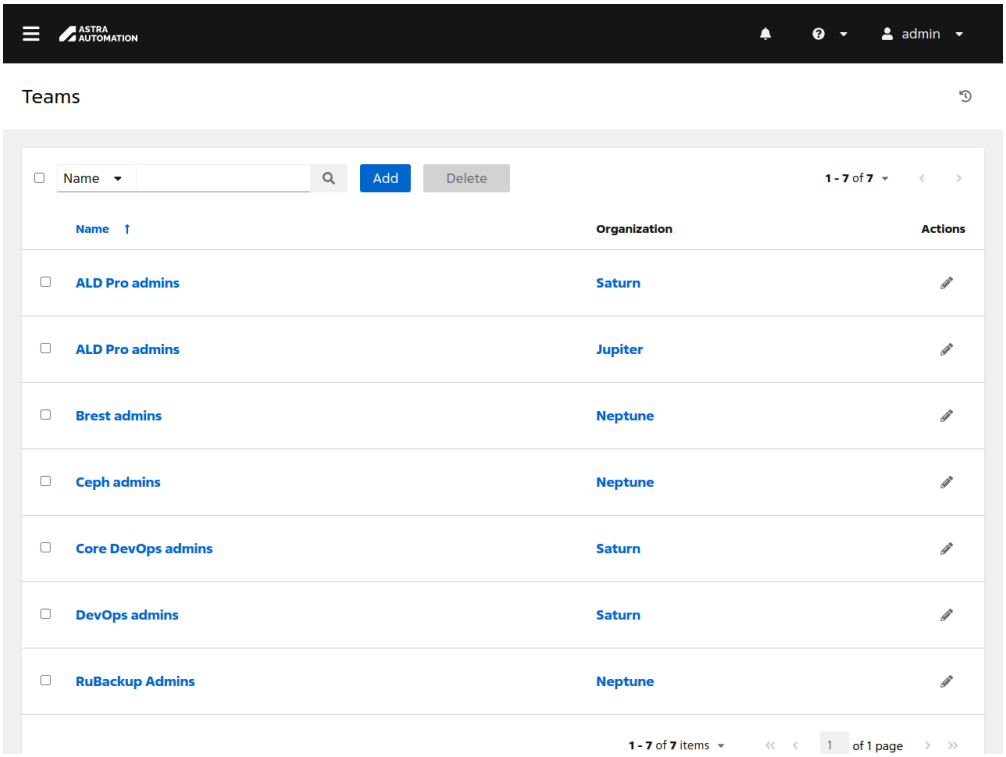


Таблица команд состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Организация** (Organization) – название организации, которой принадлежит команда;
- **Действия** (Actions) – кнопка перехода в окно редактирования свойств команды.

Просмотр

Для получения подробных сведений о команде нажмите на ссылку с ее названием в таблице команд.

Окно сведений о команде состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о команде;
- **Доступ** (Access) – список пользователей и назначенных им ролей.
- **Роли** (Roles) – список командных ролей. Указанные здесь роли автоматически назначаются всем участникам команды.

Создание

Для создания команды выполните следующие действия:

1. В окне **Команды** (Teams) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новую команду** (Create New Team):
   - **Название** (Name) – укажите название команды.

     Особенности заполнения поля:

     - Название команды должно быть уникальным на уровне организации.
     - Название команды не может состоять из одних пробельных символов (пробелы, табуляции и так далее).
     - Допускается использование символов кириллицы и специальных символов.
   - **Описание** (Description) – укажите дополнительные сведения о команде, например, контакты ответственных лиц или назначение команды.
   - **Организация** (Organization) – выберите организацию, которой принадлежит команда.
3. Нажмите кнопку *Сохранить* (Save).

Удаление

Предупреждение: При удалении команды ее участники теряют предоставленные командой привилегии.

Чтобы удалить команду, выполните следующие действия:

1. В таблице команд нажмите на ссылку с названием удаляемой команды.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление

Настройка доступа

Во вкладке **Доступ** (Access) выводится список *пользователей* и назначенных им *ролей*. Нажатие на ссылку с именем пользователя приводит к переходу в окно просмотра сведений о пользователе.

Назначение ролей отдельным пользователям

Чтобы назначить отдельным пользователям роль на уровне команды, выполните следующие действия:

1. Во вкладке **Доступ** (Access) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов *Пользователи* (Users) и нажмите кнопку *Продолжить* (Next).
3. Выберите пользователей, которым хотите назначить роли, и нажмите кнопку *Продолжить* (Next).
4. Выберите роли, которые хотите назначить выбранным на предыдущем шаге пользователям.
5. Нажмите кнопку *Сохранить* (Save).

Назначение командных ролей

Чтобы предоставить участникам команды доступ к существующим ресурсам, выполните следующие действия:

1. Во вкладке **Роли** (Roles) нажмите кнопку *Добавить* (Add).
2. Выберите тип объектов и нажмите кнопку *Продолжить* (Next).
3. Выберите экземпляры ресурсов и нажмите кнопку *Продолжить* (Next).
4. Выберите роли и нажмите кнопку *Сохранить* (Save).

Отзыв ролей

Чтобы отозвать пользовательскую или командую роль, выполните следующие действия:

1. Во вкладке **Доступ** (Access) или **Роли** (Roles) соответственно нажмите кнопку × рядом с названием отзываемой роли.
2. Подтвердите отзыв.

Примечание: Роли «Системный администратор» (System Administrator) и «Системный аудитор» (System Auditor) нельзя отозвать через настройку доступа на уровне команды, однако, можно *изменить тип пользователя*.

Администрирование

Раздел *Администрирование* (Administration) содержит подразделы, используемые для управления нестандартными типами полномочий, уведомлениями, служебными заданиями, отдельными узлами контроллера и их группами, приложениями, средами исполнения и топологией.

Типы полномочий

Окно **Типы полномочий** (Credential Types) используется для управления пользовательскими (нестандартными) типами *полномочий*. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Типы полномочий* (*Administration* ▸ *Credential Types*).

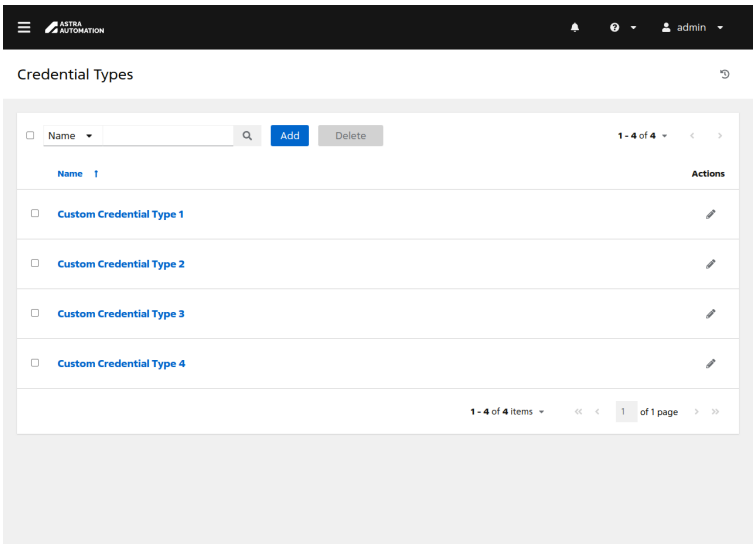


Таблица типов полномочий состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Действия** (Actions) – кнопка перехода в окно изменения свойств типа полномочий.

Примечание: Для управления пользовательскими типами полномочий необходимы привилегии администратора контроллера.

Создание

Для создания нового типа полномочий выполните следующие действия:

1. В окне **Типы полномочий** (Credential Types) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новый тип учетных данных** (Create new credential Type):
   - **Название** (Name) – укажите название типа полномочий.

     Название типа полномочий должно быть уникальным на уровне контроллера.
   - **Описание** (Description) – укажите описание типа полномочий, например, их назначение или особенности использования.
   - **Настройка входных данных** (Input configuration) – укажите параметры входных данных, которые необходимо указать при использовании этого типа полномочий.

- **Конфигурация инжектора** (Injector configuration) – укажите данные, которые передаются во внешнюю систему аутентификации при использовании полномочий этого типа.

Примечание: Особенности и примеры заполнения полей **Настройка входных данных** (Input configuration) и **Конфигурация инжектора** (Injector configuration) см. в секции *Создание типа полномочий*.

---

3. Нажмите кнопку *Сохранить* (Save).

Удаление

Чтобы удалить тип полномочий, выполните следующие действия:

1. *Удалите* все полномочия соответствующего типа.
2. В таблице типов полномочий нажмите на ссылку с названием удаляемого типа полномочий.
3. Нажмите кнопку *Удалить* (Delete).
4. Подтвердите удаление.

Уведомления

Окно **Уведомления** (Notifications) используется для управления уведомлениями. Для создания уведомлений и управления ими необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Уведомления* (*Administration* ▸ *Notifications*).

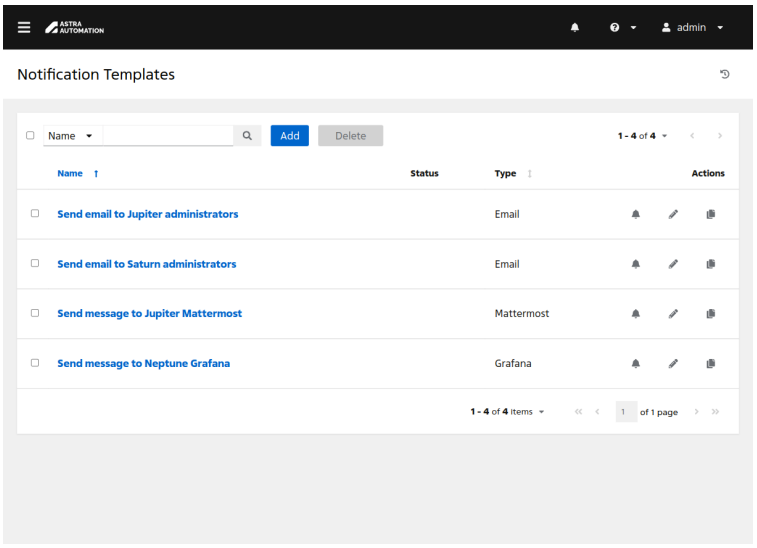


Таблица уведомлений состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Статус** (Status);

- **Тип** (Type) – способ доставки уведомлений;
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - проверка корректности настроек уведомления;
  - переход в окно редактирования свойств уведомления;
  - копирование уведомления.

Просмотр

Для получения подробных сведений об уведомлении нажмите на ссылку с его названием в таблице уведомлений.

Внешний вид окна со сведениями об уведомлении зависит от типа уведомления.

Создание

Для создания уведомления выполните следующие действия:

1. В окне **Уведомления** (Notifications) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новый шаблон уведомления** (Create New Notification Template):
   - **Название** (Name) – укажите название уведомления.
   - **Описание** (Description) – укажите описание уведомления.
   - **Организация** (Organization) – выберите организацию, для которой настраивается уведомление.
   - **Тип** (Type) – выберите тип создаваемого уведомления.

     После выбора типа в форме отображаются соответствующие поля ввода.

     #### Электронная почта (Email)

     - **Имя пользователя** (Username) – укажите имя пользователя для доступа к почтовому серверу.
     - **Пароль** (Password) – укажите пароль пользователя для доступа к почтовому серверу.
     - **Узел** (Host) – укажите IP-адрес или *FQDN* почтового сервера.
     - **Список получателей** (Recipient list) – укажите адреса электронной почты получателей, по одному на строку.
     - **Адрес электронной почты отправителя** (Sender e-mail) – укажите адрес электронной почты, с которого следует отправлять письма.
     - **Порт** (Port) – укажите номер порта для подключения к почтовому серверу.
     - **Таймаут** (Timeout) – укажите период ожидания почтового сервера (в секундах). Если в течение указанного времени контроллер не сможет связаться с почтовым сервером, отправка уведомления будет отменена.

       Минимальное значение – 1.

Максимальное значение – 120.

- **Использовать SSL** (Use SSL) – защита подключения к почтовому серверу с помощью SSL.
- **Использовать TLS** (Use TLS) – защита подключения к почтовому серверу с помощью TLS.

Grafana

- **URL Grafana** (Grafana URL) – укажите URL сервера Grafana без пути `/api/annotations/`. При создании уведомлений он добавляется к указанному URL автоматически.
- **Ключ API Grafana** (Grafana API key) – укажите ключ доступа к Grafana API.
- **Идентификатор информационной панели (необязательно)** (ID of the dashboard (optional)) – укажите идентификатор информационной панели Grafana для вывода уведомлений.
- **Идентификатор панели (необязательно)** (ID of the panel (optional)) – укажите идентификатор панели Grafana для вывода уведомлений.
- **Теги для аннотации (необязательно)** (Tags for the annotation (optional)) – укажите тэги без команд, по одному на строку.
- **Отключить проверку SSL** (Disable SSL verification) – укажите, должен ли контроллер проверять сертификат SSL сервера Grafana.

IRC

- **Пароль на сервере IRC** (IRC server password) – укажите пароль для подключения к серверу IRC.
- **Порт сервера IRC** (IRC server port) – укажите номер порта для подключения к серверу IRC.

  Значение по умолчанию – 80.
- **Адрес сервера IRC** (IRC server address) – укажите IP-адрес или *FQDN* сервера IRC.
- **IRC Nick** (IRC nick) – укажите имя пользователя для доступа к IRC.
- **Целевые каналы или пользователи** (Destination channels or users) – укажите названия каналов и имена пользователей, которым должны быть отправлены сообщения с уведомлениями, по одному на строку.

  Важно: При вводе названий каналов и имен пользователей не указывайте символы @ и # в начале строки.

- **Отключить проверку SSL** (Disable SSL verification) – укажите, должен ли контроллер проверять сертификат SSL сервера IRC.

Mattermost

- **Целевой URL** (Target URL) – укажите URL сервера Mattermost.
- **Имя пользователя** (Username) – укажите имя пользователя для доступа к серверу Mattermost.
- **Канал** (Channel) – укажите название канала Mattermost для отправки уведомлений.
- **URL значка** (Icon URL) – укажите ссылку на иконку, используемую как аватар пользователя Mattermost, от имени которого отправляются сообщения.
- **Отключить проверку SSL** (Disable SSL verification) – укажите, должен ли контроллер проверять сертификат SSL сервера Mattermost.

Pagerduty

- **API Token** (Токен API) – укажите токен для подключения к PagerDuty.
- **Поддомен Pagerduty** (Pagerduty subdomain) – укажите поддомен PagerDuty.
- **Ключ к сервису API/интеграции** (API service/intergration key) – укажите ключ к сервису API или ключ интеграции.
- **Идентификатор клиента** (Client identifier) – укажите идентификатор клиента, от имени которого контроллер будет отправлять уведомления в PagerDuty.

Rocket.Chat

- **Целевой URL** (Target URL) – укажите *FQDN* для подключения к Rocket.Chat.
- **Имя пользователя** (Username) – укажите имя пользователя для доступа к Rocket.Chat.
- **URL значка** (Icon URL) – укажите ссылку на иконку, используемую как аватар пользователя, от имени которого отправляются сообщения.
- **Отключить проверку SSL** (Disable SSL verification) – укажите, должен ли контроллер проверять сертификат SSL сервера Rocket.Chat.

Slack

- **Целевые каналы** (Destination channels) – укажите названия каналов Slack для отправки в них уведомлений, по одному каналу на строку.

  Важно: Название канала должно начинаться с символа #.

  Чтобы ответить на определенное сообщение или запустить тред, укажите идентификатор родительского сообщения – строку из 16 цифр. После 10-й цифры добавьте символ `.`, например, `#dev-ops, 0000000555.111111`.

  Подробности см. в документации Slack[155].

[155] https://api.slack.com/messaging/retrieving#individual_messages

- **Токен** (Token) – укажите токен, используемый для подключения к Slack.
- **Цвет уведомления** (Notification color) – укажите цвет уведомления в шестнадцатеричном формате, например, #12F или FF0000.

Twilio

- **Токен учетной записи** (Account token) – укажите токен для доступа к серверу Twilio.
- **Исходный номер телефона** (Source phone number) – укажите номер телефона, с которого отправляются сообщения.

  Важно: Допускается использование цифр и символа + в начале строки.

- **Номер(а) для получения SMS** (Destination SMS number(s)) – укажите номера телефонов для отправки на них SMS с уведомлениями, по одному на строку.

  Важно: Допускается использование цифр и символа + в начале строки.

  Подробности см. в документации Twilio[156].
- **Идентификатор сеанса связи** (Account SID) – укажите идентификатор сеанса связи.

Webhook

- **Имя пользователя** (Username) – укажите имя пользователя для доступа к сервису WebHook.
- **Пароль basic auth** (Basic auth password) – укажите пароль пользователя для доступа к сервису WebHook.
- **Целевой URL** (Target URL) – укажите URL WebHook.
- **Отключить проверку SSL** (Disalbe SSL verification) – укажите, должен ли контроллер проверять сертификат SSL сервера WebHook.
- **Заголовки HTTP** (HTTP Headers) – укажите в формате JSON дополнительные заголовки HTTP, которые должны быть добавлены в запрос.
- **Метод HTTP** (HTTP Method) – выберите метод HTTP, используемый для отправки запросов к WebHook. Поддерживаются методы POST и PUT.

- **Настройка сообщений...** (Customize messages...) – переведите переключатель во включенное состояние, чтобы настроить шаблоны, используемые для формирования сообщений о событиях.

  Если переключатель включен, в форме отображаются дополнительные поля:

[156] https://www.twilio.com/docs/messaging

Поле	Событие
Начальное сообщение (Start message)	Запуск задания
Сообщение об успехе (Success message)	Успешное выполнение задания
Сообщение об ошибке (Error message)	Ошибка при выполнении задания
Сообщение о согласовании потока заданий (Workflow approved message)	Согласование этапа потока заданий
Сообщение об отказе в выполнении потока заданий (Workflow denied message)	Отклонение этапа потока заданий
Сообщение о состоянии ожидания потока заданий (Workflow pending message)	Ожидание согласования этапа потока заданий
Сообщение об истечении времени ожидания потока заданий (Workflow timed out message)	Истечение времени ожидания согласования этапа потока заданий

Примечание: Подробности о синтаксисе шаблонов и доступных переменных см. в документе *Уведомления*.

3. Нажмите кнопку *Сохранить* (Save).

Удаление

Чтобы удалить уведомление, выполните следующие действия:

1. В таблице уведомлений нажмите на ссылку с названием удаляемого уведомления.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Служебные задания

Окно **Служебные задания** (Management Jobs) используется для управления запуском *служебных заданий*. Чтобы запустить служебное задание немедленно, настроить его запуск по расписанию или получать уведомления, необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Служебные задания* (*Administration* ▸ *Management Jobs*).

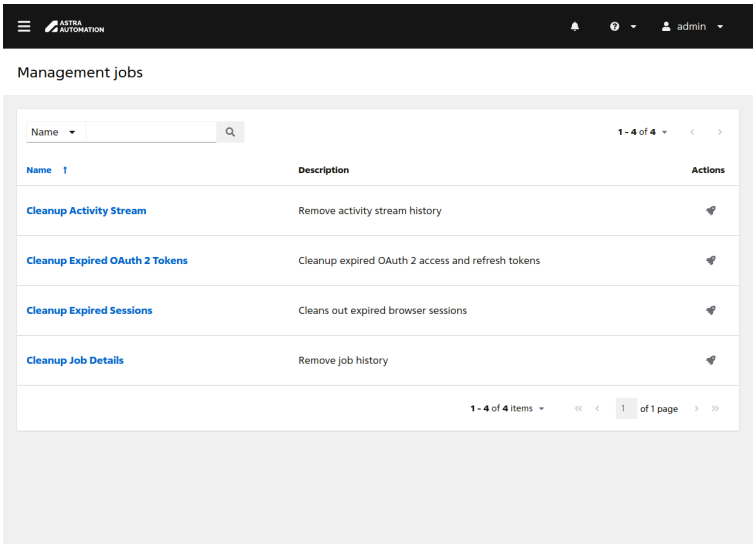


Таблица служебных заданий состоит из следующих колонок:

- **Название** (Name);
- **Описание** (Description);
- **Действия** (Actions) – кнопка запуска задания.

Запуск

Для запуска служебного задания нажмите кнопку запуска в колонке **Действия** (Actions).

Для заданий типа «Cleanup Activity Stream» и «Cleanup Job Details» при этом открывается диалоговое окно **Запуск служебного задания** (Launch management job).

1. Укажите период хранения записей (количество дней). Все записи старше указанного периода будут удалены.

   Значение по умолчанию – 30.

2. Нажмите кнопку *Запустить* (Run).

Настройка расписания

При развертывании контроллера расписание запуска служебных заданий настраивается автоматически.

Предупреждение: Некорректные настройки расписания запуска служебных заданий могут привести к отказу контроллера.

Чтобы изменить расписание запуска служебного задания, выполните следующие действия:

1. В таблице служебных заданий нажмите на ссылку с названием нужного задания.
2. Во вкладке **Расписание** (Schedules) нажмите на ссылку с названием существующего расписания.

3. Во вкладке **Подробности** (Details) нажмите кнопку *Редактировать* (Edit).
4. Измените значения в форме **Редактировать детали** (Edit Details):
   - **Название** (Name) – укажите название расписания.
   - **Описание** (Description) – укажите описание расписания.
   - **Дата и время начала** (Start date/time) – укажите дату и время первого запуска задания.
   - **Локальный часовой пояс** (Local time zone) – выберите часовой пояс, по которому рассчитывается время.

     Значение по умолчанию – UTC.
   - **Частота повторения** (Repeat frequency) – выберите частоту повторения запусков задания.

     Если выбрано значение **Нет (запускается один раз)** (None (run once)), задание будет выполнено один раз.

     Включение флагов **Мин** (Min), **Час** (Hour), **День** (Day), **Неделя** (Week), **Месяц** (Month) и **Год** (Year) приводит к добавлению в форму наборов полей, позволяющих выполнить настройку соответствующих временных интервалов. Для всех типов интервалов доступны следующие настройки:

     – **Выполнять каждые** (Run every) – количество выбранных временных интервалов между запусками задания.

     – **Завершено** (End) – условие прекращения запуска задания по расписанию. Доступны следующие настройки:

       * **Никогда** (Never) – не прекращать.
       * **После указанного количества вхождений** (After number of occurences) – после указанного количества запусков.
       * **На дату** (On date) – в указанные день и время.

     Для недельных интервалов доступна дополнительная настройка **В дни** (Days). Она позволяет выбрать дни недели, по которым следует запускать задание.

     Для месячных интервалов доступна дополнительная настройка **Выполнить в** (Run on). Она позволяет выбрать определенную дату или относительную позицию дня в месяце. Например, при выборе даты и указании числа «20» задание будет запускаться двадцатого числа каждого месяца. При выборе относительной позиции нет привязки к конкретной дате. Вместо этого настраивается запуск в дни, обозначаемые как, например, «второй понедельник» или «последняя пятница». Также поддерживаются значения, позволяющие выбрать рабочие или нерабочие дни.

     Для годовых интервалов доступна дополнительная настройка **Выполнить в** (Run on). Она позволяет выбрать месяц и число либо относительную позицию дня в году. Например, при выборе месяца «Январь» и числа «10» задание будет запускаться каждый год десятого января. При выборе относительной позиции дня нет привязки к конкретной дате. Вместо этого настраивается запуск в дни, обозначаемые как, например, «первый четверг мая» или «последний день февраля». Также поддерживаются значения, позволяющие выбрать рабочие или нерабочие дни.

  - **Исключения** (Exceptions) – настройки, предотвращающие запуск задания в указанные минуты, часы, дни, месяцы и годы. Эти настройки работают по тем же принципам, что и настройки частоты повторений.

- **Дни хранения данных** (Days of Data to Keep)

  Примечание: Эта настройка доступна только для заданий «Cleanup Activity Stream» и «Cleanup Job Details».

  Период (количество дней), в течение которого должны храниться данные о действиях пользователей и выполнении заданий соответственно. Все записи, созданные ранее этого периода, будут удалены.

  Значения по умолчанию:

  - Cleanup Activity Schedule – 355;
  - Cleanup Job Schedule – 120.

5. Нажмите кнопку *Сохранить* (Save).

Группы узлов контроллера

Окно **Группы узлов контроллера** (Instance Groups) используется для управления группами узлов контроллера и группами контейнеров. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Группы узлов контроллера* (*Administration* ▸ *Instance Groups*).

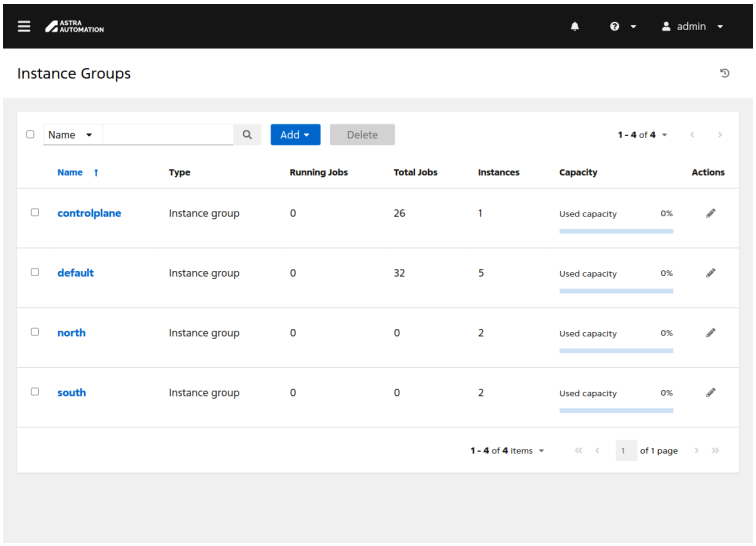


Таблица групп состоит из следующих колонок:

- Флаги для выбора нескольких записей.
- **Название** (Name) – ссылка для перехода в окно просмотра подробных сведений о группе узлов или группе контейнеров.
- **Тип** (Type) – тип записи (группа узлов контроллера или группа контейнеров).
- **Запущенные задания** (Running Jobs) – количество заданий, запущенных на узлах группы или группе контейнеров в настоящее время.

- **Всего задач** (Total Jobs) – общее количество заданий, запущенных на узлах группы или группе контейнеров когда-либо.
- **Узлы контроллера** (Instances) – количество узлов или контейнеров в группе.

Примечание: Один и тот же узел может входить в несколько групп узлов.

- **Ресурсы** (Capacity) – степень утилизации ресурсов группы узлов. Чем выше значение в этой колонке, тем выше нагрузка на узлы группы.
- **Действия** (Actions) – кнопка перехода в окно редактирования группы узлов или контейнеров.

Просмотр группы узлов

Для получения подробных сведений о группе узлов контроллера нажмите на ссылку с ее названием в таблице групп. Окно сведений о группе узлов контроллера состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о группе.
- **Узлы контроллера** (Instances) – таблица узлов группы и кнопки управления ими.

  Таблица состоит из следующих колонок:

  - Переключатели вывода подробной информации об узлах.
  - Флаги для выбора нескольких записей.
  - **Название** (Name) – ссылка для перехода в окно просмотра подробных сведений об узле.
  - **Статус** (Status) – текущий статус узла.

    Для проверки статуса узла выполните *проверку его работоспособности*.
  - **Тип узла** (Node Type).

    Подробности о типах узлов см. в разделе *Особенности архитектуры*.
  - **Настройка ресурсов** (Capacity Adjustment) – ползунок изменения максимального количества ветвлений для всех процессов, запущенных на узле.

    Максимальное значение этого параметра зависит от количества ядер CPU и объема оперативной памяти на узле.
  - **Потребляемые ресурсы** (User capacity) – индикатор утилизации ресурсов узла. Чем выше значение в этом поле, тем больше нагрузка на узел.
  - **Действия** (Actions) – переключатели статусов узлов.
- **Задания** (Jobs) – таблица заданий, запущенных на узлах группы.

  Таблица состоит из следующих колонок:

  - переключатели вывода подробной информации о задании;
  - флаги для выбора нескольких записей;
  - **Название** (Name) – ссылка для перехода в окно просмотра вывода задания;
  - **Статус** (Status) – текущий статус выполнения задания;

- **Тип** (Type) – тип задания;
- **Время начала** (Start Time) – время запуска задания;
- **Время завершения** (Finish Time) – время завершения, отмены или прерывания задания;
- **Действия** (Actions) – кнопка перезапуска задания.

Создание группы узлов

Для создания группы узлов выполните следующие действия:

1. В окне **Группы узлов контроллера** (Instance Groups) нажмите кнопку *Добавить* (Add).
2. В открывшемся меню выберите **Добавить группу узлов контроллера** (Add instance group).
3. Заполните форму **Создать новую группу узлов управления** (Create new instance group):
   - **Название** (Name) – укажите название группы узлов.

     Требования к названию:

     - название должно быть уникальным;
     - запрещено создание группы с названием `default`.
   - **Политика минимального количества** (Policy instance minimum) – укажите минимальное количество узлов, которые должны быть включены в состав группы при подключении к кластеру новых узлов.

     Значение по умолчанию – `0`.
   - **Политика процентного количества** (Policy instance percentage) – укажите минимальный процент узлов от общего количества, которые должны быть включены в состав группы при подключении к кластеру новых узлов.

     Значение по умолчанию – `0`.
   - **Максимальное количество одновременных заданий** (Max concurrent jobs) – укажите максимальное количество заданий, которые могут быть одновременно запущены на узлах группы.

     При значении `0` количество заданий не ограничено.

     Значение по умолчанию – `0`.
   - **Максимальное количество ветвлений процесса** (Max forks) – максимальное количество ответвленных процессов всех заданий, запущенных на узлах группы.

     При значении `0` количество ветвлений не ограничено.

     Значение по умолчанию – `0`.
4. Нажмите кнопку *Сохранить* (Save).

Включение узлов в группу

Чтобы включить узлы в группу, выполните следующие действия:

1. В таблице групп узлов и контейнеров нажмите на ссылку с названием нужной группы узлов.
2. Выберите вкладку **Узлы контроллера** (Instances).
3. Нажмите кнопку *Связать* (Associate).
4. В диалоговом окне **Выберите узлы контроллера** (Select Instances) включите флаги в строках с нужными узлами и нажмите кнопку *Сохранить* (Save).

Отвязка узлов от группы

Чтобы исключить узлы из группы, выполните следующие действия:

1. В таблице групп узлов и контейнеров нажмите на ссылку с названием нужной группы узлов.
2. Выберите вкладку **Узлы контроллера** (Instances).
3. Включите флаги в строках с исключаемыми узлами.
4. Нажмите кнопку *Отвязать* (Disassociate).
5. Подтвердите исключение узлов из группы.

Проверка работоспособности узлов

Чтобы проверить работоспособность узлов группы, выполните следующие действия:

1. В таблице групп узлов и контейнеров нажмите на ссылку с названием нужной группы узлов.
2. Выберите вкладку **Узлы контроллера** (Instances).
3. Включите флаги в строках с узлами, работоспособность которых хотите проверить.
4. Нажмите кнопку *Выполнить проверку работоспособности* (Run health check).
5. Перезагрузите страницу.

Создание группы контейнеров

Для создания группы контейнеров выполните следующие действия:

1. В окне **Группы узлов контроллера** (Instance Groups) нажмите кнопку *Добавить* (Add).
2. В открывшемся меню выберите **Добавить группу контейнеров** (Add container group).
3. Заполните форму **Создать новую группу узлов контейнеров** (Create new instance container):
    - **Название** (Name) – укажите название группы контейнеров.

- **Полномочия** (Credential) – если для доступа к группе контейнеров требуется аутентификация, выберите соответствующее *полномочие* типа «Токен доступа OpenShift или Kubernetes API» (OpenShift or Kubernetes API Bearer Token).
- **Максимальное количество одновременных заданий** (Max concurrent jobs) – максимальное количество заданий, которые могут быть одновременно запущены на группе контейнеров.

  При значении `0` количество заданий не ограничено.

  Значение по умолчанию – `0`.
- **Максимальное количество ветвлений процесса** (Max forks) – максимальное количество ответвленных процессов всех заданий, запущенных на группе контейнеров.

  При значении `0` количество ветвлений не ограничено.

  Значение по умолчанию – `0`.
- **Настройка спецификации pod** (Customize pod specification). Если эта настройка включена, становятся доступными для изменения настройки спецификации подов. Укажите их в поле **Измененная спецификация pod** (Custom pod spec).

4. Нажмите кнопку *Сохранить* (Save).

Удаление группы узлов или группы контейнеров

Чтобы удалить группу узлов или группу контейнеров, выполните следующие действия:

1. В таблице групп узлов и контейнеров нажмите на ссылку с названием удаляемой группы.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление группы.

Узлы контроллера

Окно **Узлы контроллера** (Instances) используется для просмотра статуса узлов контроллера и запуска диагностических команд. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Узлы контроллера* (*Administration* ▸ *Instances*).

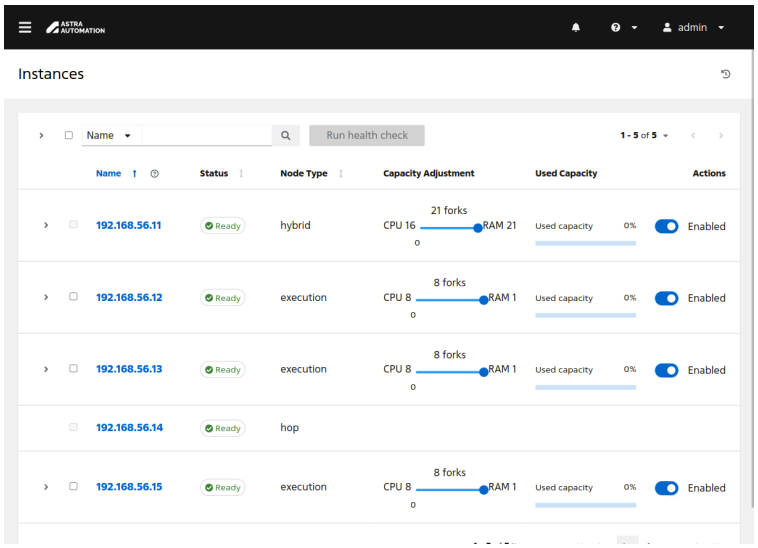


Таблица узлов состоит из следующих колонок:

- Переключатели вывода подробной информации об узлах.
- Флаги для выбора нескольких записей.
- **Название** (Name) – ссылка для перехода в окно просмотра подробных сведений об узле.
- **Статус** (Status) – текущий статус узла.

  Для проверки и обновления статуса узла выполните *проверку его работоспособности*.
- **Тип узла** (Node Type) – тип узла.

  Подробности о типах узлов см. в разделе *Особенности архитектуры*.
- **Настройка ресурсов** (Capacity Adjustment) – ползунок изменения максимального количества ветвлений для всех процессов, запущенных на узле.

  Максимальное значение этого параметра зависит от количества ядер CPU и объема оперативной памяти на узле.
- **Потребляемые ресурсы** (Used Capacity) – индикатор утилизации ресурсов узла. Чем выше значение в этом поле, тем больше нагрузка на узел.
- **Действия** (Actions) – переключатели статусов узлов.

Проверка работоспособности узлов

Чтобы проверить работоспособность узлов, выполните следующие действия:

1. В таблице узлов включите флаги в строках с теми узлами, работоспособность которых хотите проверить.
2. Нажмите кнопку *Выполнить проверку работоспособности* (Run health check).
3. Обновите страницу браузера.

Приложения

Окно **Приложения** (Applications) используется для управления приложениями и их токенами. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Приложения* (*Administration* ▸ *Applications*).

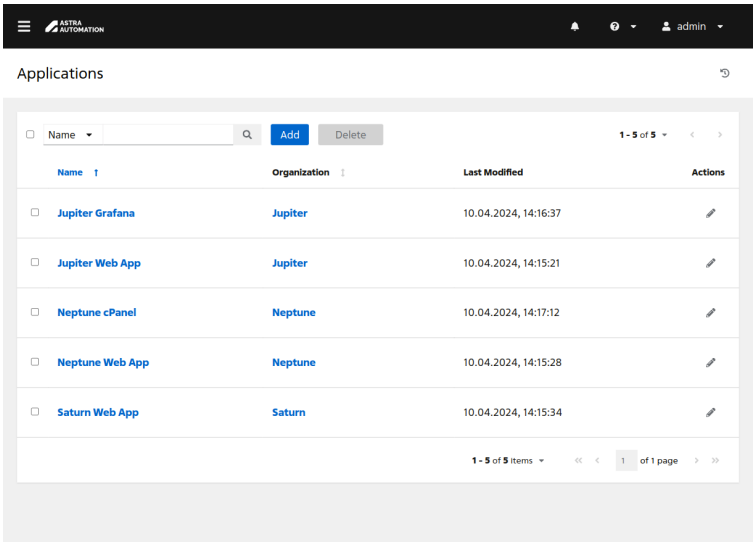


Таблица приложений состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name) – ссылка для перехода в окно просмотра сведений о приложении;
- **Организация** (Organization) – ссылка для перехода в окно просмотра сведений об организации-владельце;
- **Последнее изменение** (Last Modified) – дата и время последнего изменения записи;
- **Действия** (Actions) – кнопка перехода в окно редактирования свойств приложения.

Просмотр сведений о приложении

Для получения подробных сведений о приложении нажмите на ссылку с его названием в таблице приложений.

Окно сведений о приложении состоит из следующих вкладок:

- **Подробности** (Details) – общие сведения о приложении.
- **Токены** (Tokens) – таблица связанных с приложением токенов.

  Таблица токенов состоит из следующих колонок:

  - флаги для выбора нескольких записей;
  - **Название** (Name) – ссылка для перехода в окно просмотра сведений о пользователе-владельце токена;
  - **Область** (Scope) – режим доступа к данным контроллера, предоставляемый токеном;
  - **Истекает** (Expires) – дата и время истечения срока действия токена.

    Для изменения срока действия токенов измените значение настройки **Истечение срока действия токена доступа** (Access Token Expiration) в *общих параметрах аутентификации*.

Создание приложения

Для создания приложения выполните следующие действия:

1. В окне **Приложения** (Applications) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создать новое приложение** (Create New Application):
   - **Название** (Name) – должно быть уникальным в рамках организации.
   - **Описание** (Description) – укажите описание приложения.
   - **Организация** (Organization) – выберите организацию-владельца.
   - **Тип предоставляемых полномочий** (Authorization grant type) – выберите тип доступа к контроллеру.

     Поддерживаются следующие значения:

     - **Код авторизации** (Authorization code);
     - **Владелец ресурса с доступом по паролю** (Resource owner password-based).
   - **URI перенаправления** (Redirect URIs) – укажите список разрешенных URI. В качестве разделителя используйте символ пробела.

     Примечание: Это поле обязательно для заполнения, если в поле **Тип предоставляемых полномочий** (Authorization grant type) выбрано значение **Код авторизации** (Authorization code).

   - **Тип клиента** (Client type) – укажите уровень защищенности приложения, которому предоставляете доступ.

     Поддерживаются следующие значения:

- **Конфиденциальный** (Confidential) – приложение имеет механизмы защиты доступа к своим данным;
- **Публичный** (Public) – данные приложения публично доступны.

3. Нажмите кнопку *Сохранить* (Save).
4. В открывшемся окне **Информация о приложении** (Application information) выводятся следующие данные, необходимые для использования приложения:
   - **Идентификатор клиента** (Client ID);
   - **Секретная фраза клиента** (Client secret) (только если при создании приложения выбран тип клиента «Конфиденциальный»).

   Важно: Сохраните показанные значения в надежном месте – после закрытия диалогового окна получить к ним доступ повторно невозможно.

5. Создайте токены согласно инструкций, приведенных в подразделе *Создание токена*.

Удаление приложения

Опасно: Вместе с приложением удаляются все связанные с ним токены. Для восстановления доступа приложений к контроллеру все токены нужно будет создать заново.

Чтобы удалить приложение, выполните следующие действия:

1. В таблице приложений нажмите на ссылку с названием удаляемого приложения.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление.

Удаление токенов приложения

Предупреждение: При удалении токена использующие его приложения теряют доступ к контроллеру.

Чтобы удалить токен приложения, выполните следующие действия:

1. В таблице приложений нажмите на ссылку с названием нужного приложения.
2. Во вкладке **Токены** (Tokens) включите флаги в строках с удаляемыми токенами.
3. Нажмите кнопку *Удалить* (Delete).
4. Подтвердите удаление.

Среды исполнения

Окно **Среды исполнения** (Execution Environments) используется для управления *средами исполнения*. Для управления средами исполнения необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Среды исполнения* (*Administration* ▸ *Execution Environments*).

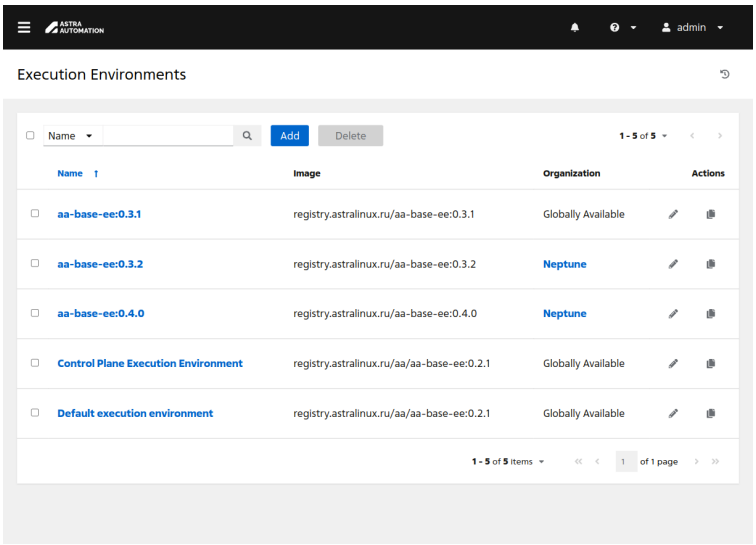


Таблица сред исполнения состоит из следующих колонок:

- флаги для выбора нескольких записей;
- **Название** (Name);
- **Образ** (Image) – ссылка на образ в реестре образов;
- **Организация** (Organization) – ссылка на организацию, которой принадлежит среда исполнения, или текст «Доступно глобально» (Globally Available), если среда исполнения доступна глобально.
- **Действия** (Actions) – кнопки для быстрого вызова часто выполняемых действий:
  - переход в окно редактирования свойств среды исполнения;
  - создание копии среды исполнения.

Просмотр

Для получения подробных сведений о среде исполнения нажмите на ссылку с ее названием в таблице сред исполнения.

Окно сведений о среде исполнения состоит из двух вкладок:

- **Подробности** (Details) – общие сведения о среде исполнения.
- **Шаблоны** (Templates) – *шаблоны заданий*, связанные со средой исполнения.

Добавление

Для добавления среды исполнения выполните следующие действия:

1. В окне **Среды исполнения** (Execution Environments) нажмите кнопку *Добавить* (Add).
2. Заполните форму **Создайте новую среду исполнения** (Create new execution environment):
   - **Название** (Name) – укажите название среды исполнения.

     Примечание: Название должно быть уникальным на уровне контроллера.

   - **Образ** (Image) – укажите ссылку на образ в реестре образов.
   - **Выбрать** (Pull) – укажите *параметры загрузки образа*.
   - **Описание** (Description) – укажите описание среды исполнения, например, версии компонентов.
   - **Организация** (Organization) – выберите *организацию*, которой принадлежит среда исполнения.

     Если это поле не заполнено, среду исполнения могут использовать все организации.
   - **Полномочия на реестр** (Registry credential) – если для доступа к реестру образов требуется аутентификация, выберите в этом поле *полномочия* типа «Реестр контейнеров» (Container Registry).

     Инструкции по созданию полномочий этого типа приведены в подразделе *Реестр контейнеров*.
3. Нажмите кнопку *Сохранить* (Save).

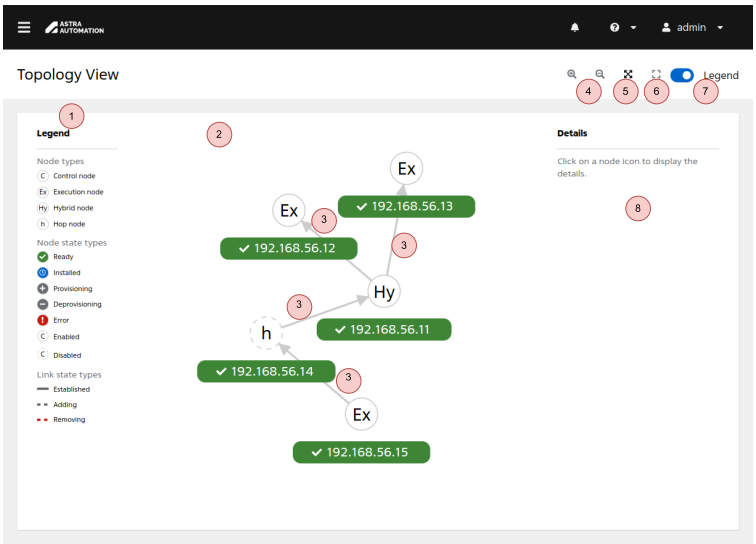
Удаление

Чтобы удалить среду исполнения, выполните следующие действия:

1. В таблице сред исполнения нажмите на ссылку с названием удаляемой среды исполнения.
2. Во вкладке **Подробности** (Details) нажмите кнопку *Удалить* (Delete).
3. Подтвердите удаление среды исполнения.

Топология

Окно **Топология** (Topology View) используется для просмотра топологии узлов кластера. Для этого необходимо перейти в соответствующий раздел с помощью панели навигации: *Администрирование* ▸ *Топология* (*Administration* ▸ *Topology View*).

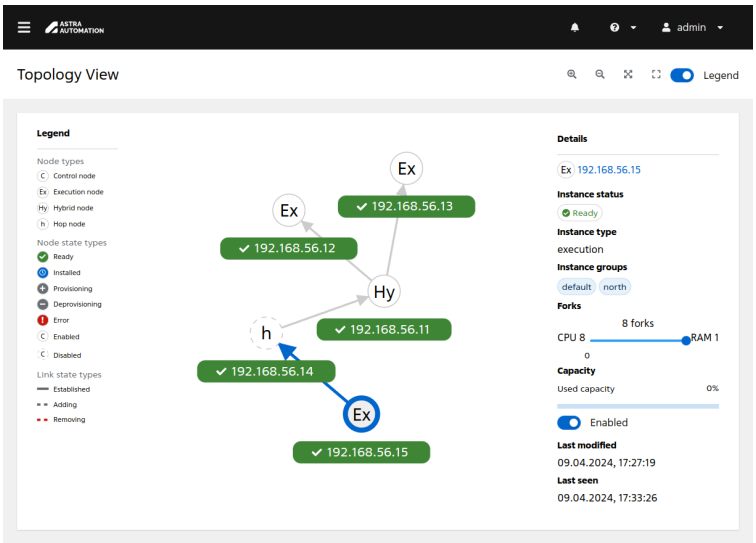


Окно **Топология** (Topology View) состоит из следующих элементов:

1. легенда;
2. схема узлов;
3. связи между узлами;
4. кнопки управления масштабом;
5. кнопка подгонки масштаба под размеры экрана;
6. кнопка сброса масштаба к значению по умолчанию;
7. переключатель видимости панели легенды;
8. панель сведений о выбранном узле.

   Содержимое панели **Подробности** (Details) зависит от типа выбранного узла.

Для просмотра сведений об узле выделите его.



Здесь представлена топология контроллера, состоящего из следующих узлов:

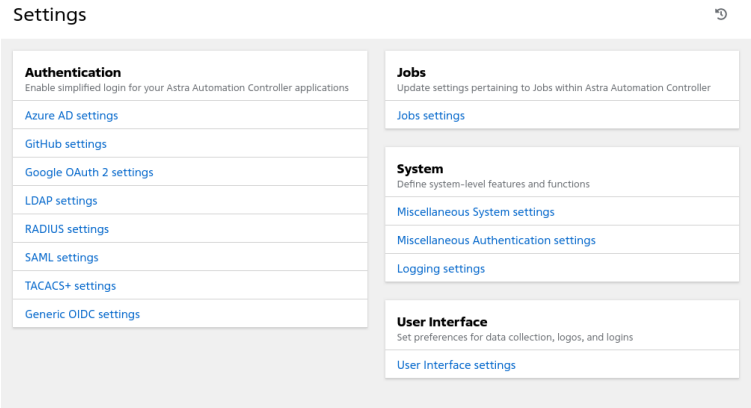
- гибридный узел 192.168.56.11, выполняющий роль управляющего узла;

- исполняющие узлы 192.168.56.12 и 192.168.56.13, соединенные с плоскостью управления (в данном случае состоящей из одного узла 192.168.56.11) напрямую;
- промежуточный узел 192.168.56.14;
- исполняющий узел 192.168.56.15, соединенный с плоскостью управления через промежуточный узел 192.168.56.14.

Подробности о типах узлов см. в разделе *Особенности архитектуры*.

Настройки

Окно **Настройки** (Settings) предназначено для изменения общих настроек контроллера.



Окно **Настройки** (Settings) состоит из четырех панелей:

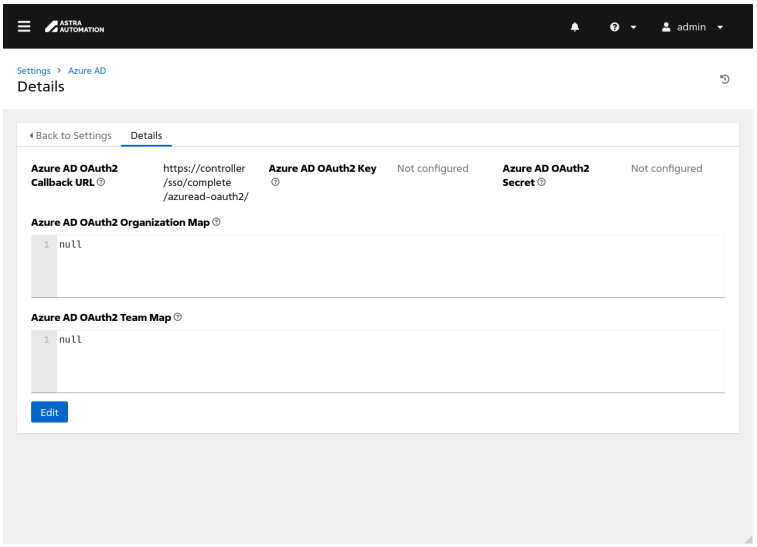
- **Аутентификация** (Authentication) – настройки аутентификации с использованием внешних провайдеров.
- **Задания** (Jobs) – настройки выполнения заданий.
- **Система** (System) – настройки встроенного провайдера аутентификации, общесистемные настройки и настройки журналирования.
- **Пользовательский интерфейс** (User Interface) – настройки внешнего вида страницы аутентификации.

Аутентификация

Панель **Аутентификация** (Authentication) содержит ссылки на страницы настроек внешних провайдеров аутентификации.

Azure AD

Окно **Azure AD** используется для настройки аутентификации с помощью провайдера Microsoft Azure Active Directory.



Форма **Редактировать** (Edit) содержит следующие поля:

- **Ключ Azure AD OAuth2** (Azure AD OAuth2 Key) – код авторизации приложения в Azure AD.
- **Секрет Azure AD OAuth2** (Azure AD OAuth2 Secret) – секретный ключ авторизации приложения в Azure AD.
- **Ассоциация организаций в контроллере с Azure AD с помощью OAuth2** (Azure AD OAuth2 Organization Map) – настройки ассоциации пользователей Azure AD с организациями контроллера.

Пример

Пусть настройки ассоциации пользователей Azure AD с организациями контроллера заданы следующим образом:

```
{
  "Jupiter": {
    "admins": [
      "/^admin_.*?@jupiter\\.example\\.ru$/"
    ],
    "users": [
      ".*?@jupiter\\.example\\.ru$/"
    ],
    "remove_admins": true,
    "remove_users": true
  },
  "Saturn": {
    "admins": [
      "/^admin_.*?@saturn\\.example\\.ru$/"
    ],
    "users": true,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "remove_admins": true,
    "remove_users": true
  }
}
```

В список пользователей организации Jupiter будут добавлены все пользователи Azure AD, названия учетных записей которых заканчиваются на `@jupiter.example.ru`. В администраторы организации Juputer будут добавлены только те пользователи Azure AD, названия учетных записей которых начинаются с `admin_` и заканчиваются на `@jupiter.example.ru`. Все пользователи Azure AD, не удовлетворяющие указанным условиям, будут удалены из списка администраторов организации Jupiter и списка ее пользователей.

Аналогичные настройки используются для организации Saturn, с той лишь разницей, что названия учетных записей пользователей этой организации должны заканчиваться на `@saturn.example.ru`.

- **Ассоциация команд в контроллере с Azure AD с помощью OAuth2** (Azure AD OAuth2 Team Map) – настройки ассоциации пользователей Azure AD с командами контроллера.

Пример

Пусть настройки ассоциации пользователей Azure AD с командами контроллера заданы следующим образом:

```
{
  "DevOps Team": {
    "organization": "Jupiter",
    "users": [
        ".*?@jupiter\\.example\\.ru$/"
    ],
    "remove": true
  }
}
```

Если организация Jupiter не существует, она будет создана.

Если в организации Jupiter не существует команда DevOps Team, она будет создана.

В список участников команды DevOps Team будут добавлены все пользователи Azure AD, названия учетных записей которых заканчиваются на `@jupiter.example.ru`. Все пользователи Azure AD, не удовлетворяющие указанным условиям, будут удалены из команды DevOps Team.

Подробности об ассоциации внешних пользователей с командами контроллера см. в секции *Ассоциация с командами*.

GitHub

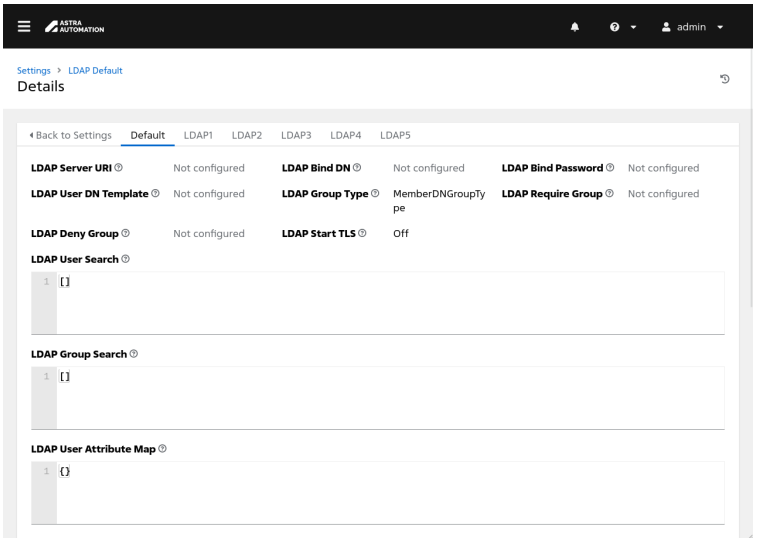
Важно: Эта часть документации находится в стадии разработки.

Google OAuth 2

Важно: Эта часть документации находится в стадии разработки.

LDAP

Окно **LDAP** используется для настройки аутентификации с помощью *провайдера LDAP*.



Оно состоит из вкладки **По умолчанию** (Default), содержащей настройки для сервера аутентификации LDAP по умолчанию, и пяти вкладок для настройки дополнительных серверов аутентификации LDAP.

Все вкладки содержат одинаковые наборы настроек. Форма **Редактировать** (Edit) содержит следующие поля:

- **URI сервера LDAP** (LDAP Server URI) – URI сервера LDAP в следующем формате:

  ```
  <protocol>://<domain>:<port>
  ```

  где:

  - <protocol> – тип протокола для подключения.

    Поддерживаются следующие значения:

    * `ldap` – подключение без защиты;
    * `ldaps` – защита подключения с помощью TLS.

  - <domain> – *FQDN* сервера LDAP.

  - <port> – порт для подключения. Как правило, порт 389 используется для подключения по протоколу `ldap`, а порт 636 – `ldaps`.

  Если необходимо ввести URI нескольких серверов, используйте символы пробела или запятой в качестве разделителя.

  Пустое значение в этом поле отключает аутентификацию через LDAP.

- **Пароль привязки к LDAP** (LDAP Bind Password) – пароль для подключения к серверу LDAP.
- **Тип группы LDAP** (LDAP Group Type) – тип обработчика, используемого для групп LDAP.

  Поддерживаются следующие значения:

  - `ActiveDirectoryGroupType`;
  - `GroupOfNamesType`;
  - `GroupOfUniqueNamesType`;
  - `MemberDNGroupType`;
  - `NestedActiveDirectoryGroupType`;
  - `NestedGroupOfNamesType`;
  - `NestedGroupOfUniqueNamesType`;
  - `NestedMemberDNGroupType`;
  - `NestedOrganizationalRoleGroupType`;
  - `OrganizationalRoleGroupType`;
  - `PosixGroupType`;
  - `PosixUIDGroupType`.

  Подробное описание каждого типа см. в документации библиотеки django-auth-ldap[157].

  Значение по умолчанию – `MemberDNGroupType`.

- **Запуск LDAP с TLS** (LDAP Start TLS) – использование TLS для подключения к серверу LDAP по протоколу `ldap` (без защиты соединения).

  Значение по умолчанию – Выкл (Off).

- **LDAP Bind DN** – отличительное наименование пользователя для привязки ко всем поисковым запросам, например:

```
CN=users,DC=example,DC=com
```

- **Шаблон уникального наименования (DN) пользователя в LDAP** (LDAP User DN Template) – если все пользователи в LDAP имеют одно и то же отличительное наименование, укажите в этом поле шаблон для более эффективного поиска.

  Важно: Указанное в этом поле значение используется вместо значения, заданного в поле **LDAP Bind DN**.

  ---

[157] https://django-auth-ldap.readthedocs.io/en/stable/groups.html#types-of-groups

- **Обязательная группа LDAP** (LDAP Require Group) – отличительное наименование группы LDAP, в которую должны входить пользователи. Если пользователь не является членом указанной группы, его аутентификация в контроллере невозможна.

  Если поле не заполнено, все пользователи LDAP, соответствующие поисковому запросу, смогут аутентифицироваться в контроллере.

  Поддерживается только одна обязательная группа.

- **Запрещенная группа LDAP** (LDAP Deny Group) – отличительное наименование группы LDAP, членам который запрещено аутентифицироваться в контроллере.

  Поддерживается только одна запрещенная группа.

- **LDAP-запрос по поиску пользователей** (LDAP User Search) – параметры запроса, используемого для поиска пользователей в LDAP, например:

```
[
  "OU=Users,DC=example,DC=com",
  "SCOPE_BASE",
  "(cn=$(user)s)"
]
```

  Поддерживается использование LDAPSearchUnion для создания сложных запросов.

- **Групповой поиск по LDAP** (LDAP Group Search) – настройки поиска групп в LDAP, например:

```
[
  "DC=example,DC=com",
  "SCOPE_SUBTREE",
  "(ObjectClass=group)"
]
```

  Использование LDAPSearchUnion в этом поле не поддерживается.

- **Ассоциация пользовательских атрибутов с LDAP** (LDAP User Attribute Map) – правила ассоциирования данных пользователя в LDAP с данными пользователя в контроллере, например:

```
{
  "first_name", "givenName",
  "last_name", "sn",
  "email": "mail"
}
```

  По умолчанию используются настройки для работы с Microsoft Active Directory.

- **Параметры типа группы LDAP** (LDAP Group Type Parameters) – дополнительные параметры типа группы LDAP.

  Список поддерживаемых параметров зависит от значения, выбранного в поле **Тип группы LDAP** (LDAP Group Type). Подробности см. в документации библиотеки django-auth-ldap[158].

  Значение по умолчанию:

[158] https://django-auth-ldap.readthedocs.io/en/stable/groups.html#types-of-groups

```
{
  "member_attr": "member",
  "name_attr": "cn"
}
```

- **Пользовательские флаги LDAP по группам** (LDAP User Flags By Group) – критерии, при выполнении которых пользователям LDAP в контроллере будут автоматически присваиваться *типы* «Системный администратор» (System Administrator) и «Системный аудитор» (System Auditor), например:

```
{
  "is_superuser": "cn=superusers,ou=groups,db=example,dc=com",
  "is_system_auditor": "cn=auditors,ou=groups,dc=example,dc=com"
}
```

- **Ассоциация организации с LDAP** (LDAP Organization Map) – параметры ассоциации организаций LDAP с *организациями* контроллера.

Пример

Пусть настройки ассоциации пользователей LDAP с организациями контроллера заданы следующим образом:

```
{
  "Jupiter": {
    "admins": "CN=admins,OU=groups,DC=jupiter,DC=example,DC=com",
    "users": [
      "CN=devops,OU=groups,DC=jupiter,DC=example,DC=com",
      "CN=testers,OU=groups,DC=jupiter,DC=example,DC=com"
    ],
    "remove_admins": true,
    "remove_users": true
  },
  "Saturn": {
    "admins": "CN=admins,OU=groups,DC=saturn,DC=example,DC=com",
    "users": true,
    "remove_admins": true,
    "remove_users": false
  }
}
```

Если организации Jupiter и Saturn не существуют, они будут созданы.

В список администраторов организации Jupiter будут добавлены все пользователи LDAP, которые удовлетворяют следующим условиям:

- состоят в группе LDAP `admins`;
- имеют учетную запись в домене `jupiter.example.com`.

Все пользователи LDAP, не удовлетворяющие указанным условиям, будут удалены из списка администраторов организации Jupiter.

В список пользователей организации Jupiter будут добавлены все пользователи LDAP, которые удовлетворяют следующим условиям:

- состоят в группах LDAP `devops` или `testers`;
- имеют учетную запись в домене `jupiter.example.com`.

Все пользователи LDAP, не удовлетворяющие указанным условиям, будут удалены из списка пользователей организации Jupiter.

В список администраторов организации Saturn будут добавлены все пользователи LDAP, которые удовлетворяют следующим условиям:

- состоят в группе LDAP `admins`;
- имеют учетную запись в домене `saturn.example.ru`.

Все пользователи LDAP, не удовлетворяющие указанным условиям, будут удалены из списка администраторов организации Saturn.

Пользователями организации Saturn являются любые пользователи, найденные в LDAP.

Пользователи организации Saturn, не найденные в LDAP, не будут удалены из контроллера.

Подробности об ассоциации внешних пользователей с организациями контроллера см. в секции *Ассоциация с организациями*.

- **Ассоциация группы с LDAP** (LDAP Team Map) – параметры ассоциации групп LDAP с *командами* контроллера.

Пример

Пусть настройки ассоциации пользователей LDAP с командами контроллера заданы следующим образом:

```
{
  "DevOps Team": {
    "organization": "Jupiter",
    "users": "CN=devops,OU=groups,DC=jupiter,DC=example,DC=com",
    "remove": true
  }
}
```

Если организация Jupiter не существует, она будет создана.

Если в организации не существует команда DevOps Team, она будет создана.

В список участников команды DevOps Team будут добавлены все пользователи LDAP, которые удовлетворяют следующим условиям:

- входят в группу LDAP `devops`;
- имеют учетную запись в домене `jupiter.example.com`.

Все пользователи LDAP, не удовлетворяющие указанным условиям, будут удалены из команды DevOps Team.

Подробности об ассоциации внешних пользователей с командами контроллера см. в секции *Ассоциация с командами*.

RADIUS

Важно: Эта часть документации находится в стадии разработки.

SAML

Важно: Эта часть документации находится в стадии разработки.

TACACS+

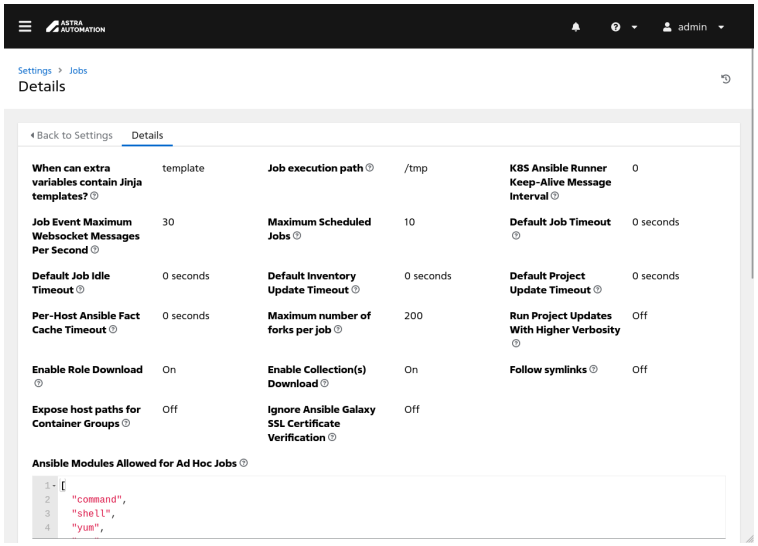
Важно: Эта часть документации находится в стадии разработки.

Общий OIDC

Важно: Эта часть документации находится в стадии разработки.

Задания

Окно **Задания** (Jobs) содержит настройки, которые управляют запуском и исполнением всех заданий в Astra Automation Controller.



Форма **Редактировать** (Edit) содержит следующие поля:

- **Путь выполнения заданий** (Job execution path)

Путь к каталогу, в котором контроллер должен создать временный подкаталог для хранения данных, связанных с заданием, например, файлы полномочий.

Значение по умолчанию – `/tmp`.

- **Максимальное количество запланированных заданий** (Maximum Scheduled Jobs)

  Количество заданий на основе одного и того же шаблона, которые можно поставить в очередь выполнения. Если указанное значение будет достигнуто, создавать новые задания будет нельзя.

  Значение по умолчанию – `10`.

- **K8S Ansible Runner Keep-Alive Message Interval**

  Интервал в секундах, через который узлам из группы контейнеров отправляется сообщение для поддержания соединения. При значении `0` сообщения не отправляются.

  Значение по умолчанию – `0`.

- **Тайм-аут задания по умолчанию** (Default Job Timeout)

  Время в секундах, отведенное на исполнение каждого задания. Значение этой настройки может быть переопределено в настройках шаблона задания. При значении `0` задания выполняются без ограничений по времени.

  Значение по умолчанию – `0`.

- **Default Job Idle Timeout** (Default Job Idle Timeout)

  Если за указанный период времени в секундах не происходит вывода Ansible, исполнение задания прерывается. При значении `0` задания могут выполняться без вывода неограниченно долго.

  Значение по умолчанию – `0`.

- **Тайм-аут обновления инвентаря по умолчанию** (Default Inventory Update Timeout)

  Время в секундах, отведенное на выполнение задачи обновления инвентаря из внешнего источника. Значение этой настройки может быть переопределено в настройках инвентаря. При значении `0` задания обновления инвентаря выполняются без ограничений по времени.

  Значение по умолчанию – `0`.

- **Тайм-аут обновления проекта по умолчанию** (Default Project Update Timeout)

  Время в секундах, отведенное на выполнение задачи обновления проекта из внешнего источника. Значение этой настройки может быть переопределено в настройках проекта. При значении `0` задания обновления проекта выполняются без ограничений по времени.

  Значение по умолчанию – `0`.

- **Тайм-аут кэширования ansible фактов для каждого узла** (Per-Host Ansible Fact Cache Timeout)

  Период времени в секундах, в течение которого факты Ansible считаются действительными с момента их последнего изменения. В playbook будут доступны только действительные, не устаревшие факты. При значении `0` факты не устаревают.

Важно: Эта настройка не влияет на удаление фактов в базе данных Astra Automation Controller.

Значение по умолчанию – `0`.

- **Максимальное количество ответвлений задания** (Maximum number of forks per job)

  Максимальное количество ответвлений, которое можно задать в настройках шаблона задания. Попытка сохранить шаблон задания со значением больше указанного приведет к ошибке.

  Значение по умолчанию – `200`.

- **В каких случаях дополнительные переменные могут содержать шаблоны Jinja?** (When can extra variables contain Jinja templates?)

  Разрешение на использование шаблонов Jinja2 в дополнительных переменных. Поддерживаются следующие значения:

  - `Всегда` (Always);
  - `Никогда` (Never);
  - `В шаблонах` (Template).

Предупреждение: Для этой настройки рекомендуется использовать значение `Никогда` или `В шаблонах`. Это связано с тем, что шаблоны Jinja2 потенциально могут быть использованы для запуска произвольного кода на языке Python, что создает угрозу безопасности.

Значение по умолчанию – `В шаблонах` (Template).

- **Выполнить обновление проекта с большей детализацией** (Run Project Updates With Higher Verbosity)

  Если эта настройка включена, используется более детализированный вывод при выполнении заданий обновления проектов. Использование этой настройки эквивалентно выполнению команды `ansible-playbook project_update.yml` с параметром `-vvv`.

  Значение по умолчанию – выключено.

- **Игнорировать проверку SSL-сертификата Ansible Galaxy** (Ingore Ansible Galaxy SSL Certificate Verification)

  Если эта настройка включена, при установке содержимого из Ansible Galaxy не выполняется проверка сертификата SSL.

  Значение по умолчанию – выключено.

- **Включить загрузку ролей** (Enable Role Download)

  Если эта настройка включена и источником кода проекта является система контроля версий, разрешена загрузка ролей, перечисленных в файле проекта `roles/requirements.yml`.

  Значение по умолчанию – включено.

- **Разрешить загрузку коллекции(й)** (Enable Collection(s) Download)

  Если эта настройка включена и источником кода проекта является система контроля версий, разрешена загрузка коллекций, перечисленных в файле проекта `collections/requirements.yml`.

  Значение по умолчанию – включено.

- **Переходы по символическим ссылкам** (Follow symlinks)

  Если эта настройка включена, при обработке playbook разрешен переход по символическим ссылкам.

  > **Предупреждение:** Переход по ссылкам может привести к бесконечной рекурсии, если ссылка указывает на свой родительский каталог.

  Значение по умолчанию – выключено.

- **Expose host paths for Container Groups**

  Если эта настройка включена, разрешено использование `hostPath` для подов, созданных в группе контейнеров.

  > **Предупреждение:** Использование томов `hostPath` содержит потенциальные риски безопасности. Рекомендуется избегать использования `hostPath`, где это возможно.

  Значение по умолчанию – выключено.

- **Модули Ansible, разрешенные для выполнения специальных заданий** (Ansible Modules Allowed for Ad Hoc Jobs)

  Список модулей Ansible, которые разрешено использовать при выполнении специальных (ad-hoc) команд.

  Значение по умолчанию:

```
[
  "command",
  "shell",
  "yum",
  "apt",
  "apt_key",
  "apt_repository",
  "apt_rpm",
  "service",
  "group",
  "user",
  "mount",
  "ping",
  "selinux",
  "setup",
  "win_ping",
  "win_service",
  "win_updates",
  "win_group",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "win_user"
]
```

- **Расширения Ansible Callback** (Ansible Callback Plugins)

  Список путей для поиска расширений обратного вызова Ansible, по одному на строку. При значении `[]` расширения обратного вызова не используются.

  Значение по умолчанию – `[]`.

- **Пути для доступа к изолированным заданиям** (Path to expose to isolated jobs)

  Список путей, монтируемых на исполняющих узлах в контейнеры как отдельные тома, по одному на строку. Пути, не указанные в этом списке, недоступны в контейнерах и не могут быть использованы при выполнении заданий.

  Формат записей:

```
<HOST_DIR>[:<CONTAINER_DIR>[:<OPTIONS>]]
```

  Здесь:

  - <HOST_DIR> – путь к каталогу в файловой системе исполняющего узла;
  - <CONTAINER_DIR> – точка монтирования каталога в файловой системе контейнера;
  - <OPTIONS> – опциональные параметры монтирования.

  Значение по умолчанию:

```
[
   "/etc/pki/ca-trust:/etc/pki/ca-trust:O",
   "/usr/share/pki:/usr/share/pki:O"
]
```

- **Environment Variables for Galaxy Commands**

  Дополнительные переменные окружения, используемые при вызове команды `ansible-galaxy` для обновления проекта. Использование этих переменных может быть полезно, если команда `ansible-galaxy` работает через прокси-сервер, но не Git.

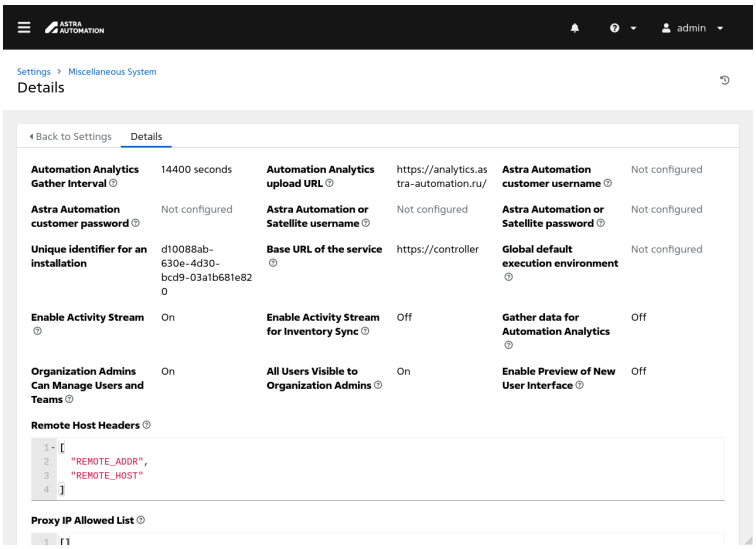
  Значение по умолчанию:

```
{
  "ANSIBLE_FORCE_COLOR": "false",
  "GIT_SSH_COMMAND": "ssh -o StrictHostKeyChecking=no"
}
```

Система

Панель **Система** (System) содержит ссылки на разделы, используемые для управления общими параметрами системы, параметрами встроенного провайдера аутентификации и журналированием.

Общие системные настройки



Окно **Общие системные настройки** (Miscellaneous System settings) содержит следующие настройки:

- **Включить ленту активности** (Enable Activity Stream)

  Если эта настройка включена, ведется журнал активности (*Режимы просмотра ▸ Лента активности*).

  Значение по умолчанию – включено.

- **Включить ленту активности для синхронизации инвентаря** (Enable Acitivity Stream for Inventory Sync)

  Если эта настройка включена, в ленте активности выводятся сведения о событиях синхронизации инвентаря.

  Значение по умолчанию – выключено.

- **Глобальная среда исполнения по умолчанию** (Global default execution environment)

  Среда исполнения, используемая по умолчанию на уровне контроллера.

- **Базовый URL сервиса** (Base URL of the service)

  URL, используемый при создании ссылок на страницу контроллера. Укажите в этом поле доменное имя, используемое для доступа к контроллеру.

  Значение по умолчанию – `https://controller/`.

- **Пользователи, доступные администраторам организации** (All Users Visible to Organization Admins)

Если эта настройка включена, администраторы организаций имеют возможность управлять пользователями других организаций.

Значение по умолчанию – включено.

- **Администраторы организации могут управлять пользователями и командами** (Organization Admins Can Manage Users and Teams)

  Если эта настройка включена, администраторы организаций могут управлять своими пользователями и командами.

  Значение по умолчанию – выключено.

- **Gather data for Automation Analytics**

  Сбор данных для сервиса Automation Analytics.

  Значение по умолчанию – выключено.

- **Astra Automation customer username**

  Имя пользователя, используемое для подключения к серверу Automation Analytics.

- **Astra Automation customer password**

  Пароль пользователя, используемый для подключения к серверу Automation Analytics.

- **Astra Automation or Satellite username**
- **Astra Automation or Satellite password**
- **Automation Analytics upload URL**

  URL для передачи информации, собранной службой Automation Analytics.

- **Automation Analytics Gather Inverval**

  Период (в секундах) отправки информации, собранной службой Automation Analytics.

  Значение по умолчанию – `14400` (4 часа).

- **Enable Preview of New User Interface**

  Если эта настройка включена, используется экспериментальный пользовательский интерфейс.

  Значение по умолчанию – выключено.

- **Last gathered entries from the data collection service of Automation Analytics**
- **Заголовки (headers) удаленного узла** (Remote Host Headers)

  Заголовки HTTP и мета-ключи, используемые для определения имени удаленного хоста или его IP-адреса. Если доступ к контроллеру осуществляется через обратный прокси-сервер, добавьте в список дополнительные заголовки, например, `HTTP_X_FORWARDED_FOR`.

  Значение по умолчанию:

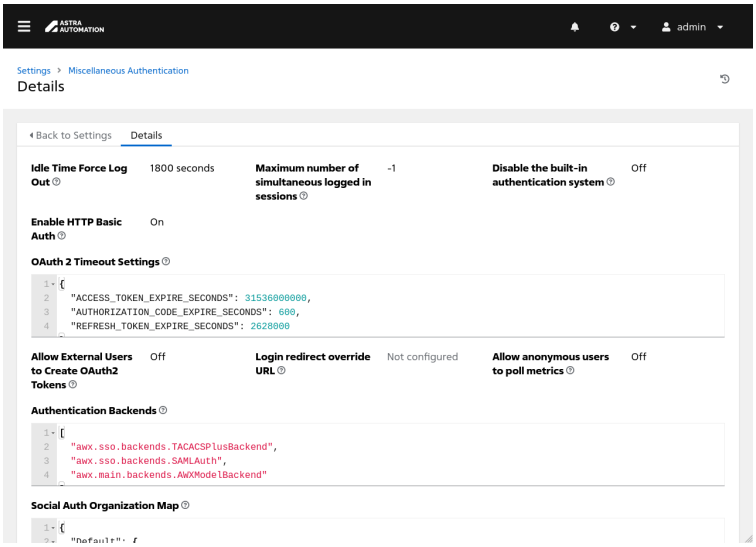
```
[
    "REMOTE_ADDR",
    "REMOTE_HOST"
]
```

- **IP-адреса разрешенных прокси-серверов** (Proxy IP Allowed List)

  Список IP-адресов разрешенных прокси-серверов или балансировщика нагрузки. При значении [] запрещено использование прокси-серверов или балансировщика нагрузки.

  Значение по умолчанию – [].

Общие параметры аутентификации



Окно **Общие параметры аутентификации** (Miscellaneous Authentication settings) содержит настройки, которые управляют встроенным провайдером аутентификации:

- **Отключить встроенную систему аутентификации** (Disable the built-in authentication system)

  Если эта настройка выключена, в контроллере используется встроенная система аутентификации. Если для аутентификации в контроллере используется внешний провайдер аутентификации, например, LDAP или SAML, эту настройку необходимо включить.

  Значение по умолчанию – выключено.

- **Время бездействия до принудительного выхода** (Idle Time Force Log Out)

  Время в секундах, по истечение которого сессии неактивных пользователей автоматически завершаются.

  Значение по умолчанию – 1800 (30 минут).

- **Максимальное количество одновременных сеансов входа в систему** (Maxumum number of simultaneous logged in sessions)

  Максимальное количество одновременных сеансов пользователя. При значении -1 количество сеансов не ограничено.

  Значение по умолчанию – -1.

- **Включить HTTP Basic Auth** (Enable HTTP Basic Auth)

  Если эта настройка включена, разрешено использование браузерного API «HTTP Basic Auth».

Примечание: Подробности об API «HTTP Basic Auth» см. в RFC 7617[159].

Значение по умолчанию – включено.

- **Разрешить внешним пользователям создавать токены OAuth2** (Allow External Users to Create OAuth2 Tokens)

  Если эта настройка включена, пользователи от внешних провайдеров аутентификации (LDAP, SAML и другие) получают возможность создавать токены OAuth2.

  Значение по умолчанию – выключено.

- **Перенаправление URL при входе** (Login redirect override URL)

  URL, по которому для входа в систему будут перенаправлены неавторизованные пользователи. Если значение не указано, используется перенаправление на страницу входа.

  Значение по умолчанию – пустая строка.

- **Истечение срока действия токена доступа** (Access Token Expiration)

  Время в секундах, в течение которого действует токен доступа (access token).

  Значение по умолчанию – `31536000000` (~1000 лет).

- **Срок действия токена обновления истек** (Refresh Token Expiration)

  Время в секундах, в течение которого действует токен обновления (refresh token) после истечения токена доступа.

  Значение по умолчанию – `2628000` (~30 дней).

- **Истечение срока действия кода авторизации** (Authorization Code Expiration)

  Время в секундах, в течение которого действует код авторизации.

  Значение по умолчанию – `600` (10 минут).

- **Ассоциация организаций с публичными системами аутентификации** (Social Auth Organization Map)
- **Ассоциация команд с публичными системами** (Social Auth Team Map)
- **Поля для идентификации пользователя через публичные сервисы аутентификации** (Social Auth User Fields)
- **Use Email address for usernames**

  Если эта настройка включена, для аутентификации в качестве имени пользователя используется его адрес электронной почты.

  Значение по умолчанию – выключено.

- **Minimum number of characters in local password**

  Минимальное количество буквенных символов в паролях, используемых для аутентификации через встроенную систему.

  Значение по умолчанию – `0`.

[159] https://datatracker.ietf.org/doc/html/rfc7617

- **Minimum number of digit characters in local password**

  Минимальное количество цифр в паролях, используемых для аутентификации через встроенную систему.

  Значение по умолчанию – `0`.

- **Minimum number of uppercase characters in local password**

  Минимальное количество буквенных символов верхнего регистра в паролях, используемых для аутентификации через встроенную систему.

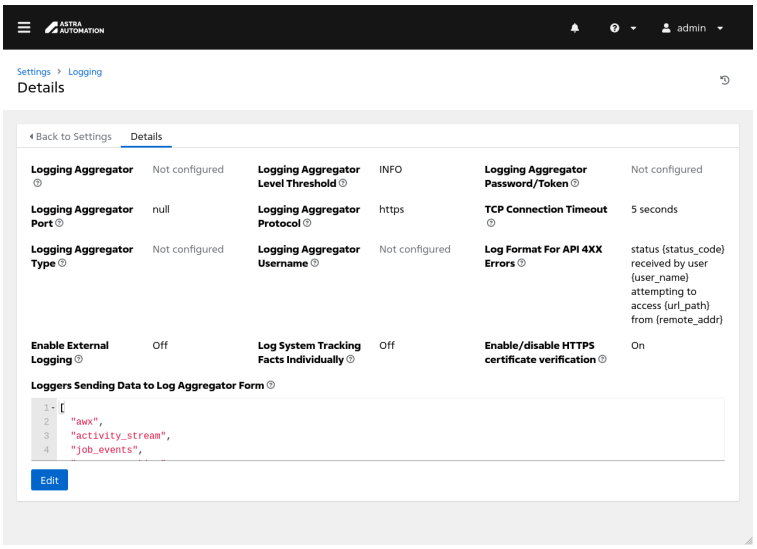
  Значение по умолчанию – `0`.

- **Minimum number of special characters in local password**

  Минимальное количество специальных символов в паролях, используемых для аутентификации через встроенную систему.

  Значение по умолчанию – `0`.

Настройки журналирования



Окно **Настройки журналирования** (Logging settings) содержит настройки, управляющие ведением журнала контроллера:

- **Включить внешнее ведение журнала** (Enable External Logging)

  Если эта настройка включена, для хранения журналов используется внешняя система журналирования.

  Примечание: Для включения этой настройки необходимо задать значения настроек **Агрегатор протоколирования** и **Тип агрегатора ведения журнала**.

- **Агрегатор протоколирования** (Logging Aggregator)

  FQDN или IP-адрес сервера, на котором размещена внешняя система журналирования.

- **Порт агрегатора ведения журнала** (Logging Aggregator Port)

  Номер порта, через который осуществляется подключение к внешней системе журналирования.

- **Тип агрегатора ведения журнала** (Logging Aggregator Type)

  Тип внешней системы журналирования. Поддерживаются следующие значения:

  - `---------` – внутренняя система журналирования;
  - `logstash` – Elastic Logstash;
  - `splunk` – Splunk;
  - `loggly` – Solarwinds Loggly;
  - `sumologic` – Sumo Logic;
  - `other` – внешняя система журналирования с настройками, заданными вручную.

- **Имя пользователя агрегатора ведения журнала** (Logging Aggregator Username)

  Имя пользователя, используемое для доступа к внешней системе журналирования.

- **Пароль/токен агрегатора ведения журнала** (Logging Aggregator Password/Token)

  Пароль или токен для доступа к внешней системе журналирования.

- **Вести индивидуальный журнал по фактам, собранным системой** (Log System Tracking Facts Individually)

  Если этот параметр включен, данные системного отслеживания фактов будут отправляться для каждого пакета, сервиса или другой сущности, найденной при сканировании, что обеспечивает большую детализацию поискового запроса. Если этот параметр выключен, факты будут отправляться в виде единого словаря, что обеспечивает большую эффективность обработки фактов.

  Значение по умолчанию – выключено.

- **Протокол ведения журналов** (Logging Aggregator Protocol)

  Протокол соединения с внешней системов журналирования. Поддерживаются следующие значения:

  - `HTTPS/HTTP`.

    Примечание: При выборе этого значения используется протокол HTTPS, если только протокол HTTP не указан явно в значении настройки **Агрегатор протоколирования** (Logging Aggregator).

  - `TCP`.
  - `UDP`.

  Значение по умолчанию – `HTTPS/HTTP`.

- **Порог уровня агрегатора ведения журнала** (Logging Aggregator Level Threshold)

  Минимальный уровень сообщения, необходимый для его записи в журнал. Уровни сообщений, от наименьшего к наибольшему:

  1. `DEBUG`;
  2. `INFO`;

3. WARNING;
4. ERROR;
5. CRITICAL.

Сообщения с уровнем ниже указанного будут пропущены обработчиком журнала.

Примечание: Действие этого параметра не распространяется на сообщения из категории awx.analytics.

Значение по умолчанию – INFO.

- **Тайм-аут TCP-соединения** (TCP Connection Timeout)

  Время в секундах, по истечении которого соединение с внешней системой журналирования разрывается. Значение этой настройки используется только если для настройки **Протокол ведения журналов** задано значение HTTPS/HTTP или TCP.

  Значение по умолчанию – 5.

- **Включить/отключить проверку сертификата HTTPS** (Enable/disable HTTPS certificate verification)

  Если эта настройка включена, при соединении с внешней системой журналирования по протоколу HTTPS выполняется проверка сертификата.

  Значение по умолчанию – включено.

- **Регистраторы, отправляющие данные в форму агрегатора журналов** (Loggers Sending Data to Log Aggregator Form)

  Список типов регистраторов, отправляющих данные в агрегатор журналов.

  Поддерживаются следующие значения:

  - awx – внутренние события контроллера;
  - activity_stream – изменение ресурсов;
  - job_events – события, связанные с заданиями;
  - system_tracking – факты, собранные при выполнении заданий.

    Примечание: Для сбора фактов необходимо включить настройку **Включить хранилище фактов** (Enable Fact Storage) в свойствах шаблона задания.

  - broadcast_websocket – ошибки широковещательных рассылок веб-сокетов.

  Значение по умолчанию:

```
[
    "awx",
    "activity_strem",
    "job_events",
    "system_tracking",
    "broadcast_websocket"
]
```

- **Log Format For API 4XX Errors**

Шаблон записей о событиях, связанных с кодами HTTP 4XX при обращении к конечным точкам API.

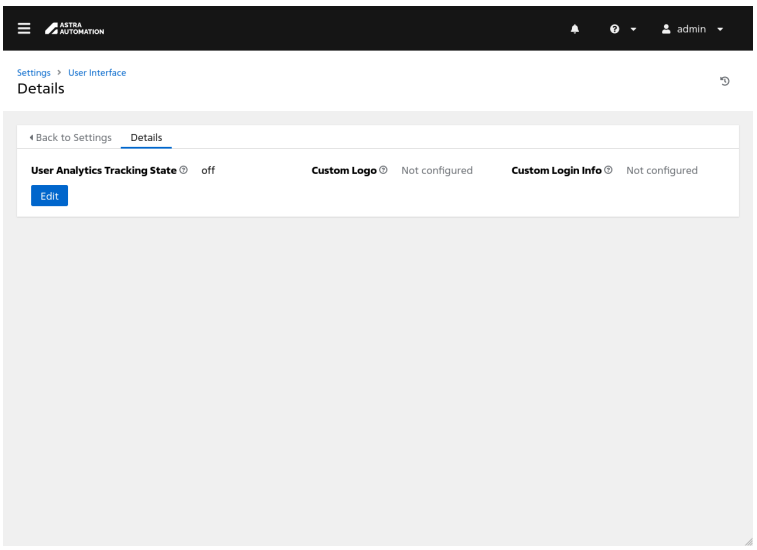
В шаблоне поддерживается использование следующих переменных:

- `status_code` – код статуса HTTP;
- `user_name` – имя пользователя, выполнившего запрос к конечной точке API;
- `url_path` – URL конечной точки API, обращение к которой привело к ошибке;
- `remote_addr` – удаленный адрес, отображаемый для ошибки пользователя;
- `error` – сообщение об ошибке, возвращенное конечной точкой API.

Если конечная точка API не вернула текст сообщения об ошибке, используется строковое название статуса HTTP.

Пользовательский интерфейс

Окно **Пользовательский интерфейс** (User Interface) содержит настройки, управляющие внешним видом страницы аутентификации.



Форма **Редактировать** (Edit) содержит следующие поля:

- **Пользовательская информация для входа в систему** (Custom Login Info) – текст, выводимый на странице аутентификации. Поддерживаются форматы Plain Text и HTML.
- **Пользовательский логотип** (Custom Logo) – изображение, отображаемое на странице аутентификации.

  Поддерживаются следующие форматы:

  - GIF;
  - JPEG;
  - PNG (рекомендуется).

6.4 Программный интерфейс (API)

В этом разделе приводятся инструкции по управлению Astra Automation Controller с помощью программного интерфейса – API (Application Programming Interface).

Для взаимодействия с API достаточно отправлять HTTP-запросы к определенным точкам доступа API (API endpoint), которые предоставляют доступ к нужным данным или выполняют определенные действия.

API используется для интеграции между различными приложениями, создания расширений, автоматизации задач, доступа к сторонним сервисам и многих других целей. API позволяет различным системам и приложениям обмениваться данными и ресурсами, чтобы улучшить их функциональность и эффективность.

Преимущества использования API:

- Масштабируемость – поддерживает горизонтальное масштабирование, что позволяет расширять его для обработки больших объемов запросов и данных.
- Независимость от языка и платформы – может быть использован с любым языком программирования и на любой платформе, что делает его универсальным и легко интегрируемым.
- Безопасность – использует стандартные методы аутентификации, такие как OAuth, что обеспечивает безопасность передачи данных.
- Открытость – имеет открытую структуру, что позволяет разработчикам легко получать доступ к различным данным и ресурсам.

6.4.1 Общий синтаксис

В этом документе рассматривается синтаксис запросов к API и получаемых ответов.

Структура запроса

Запрос API состоит из следующих элементов:

```
<method> <URI>[<query>]
   [<headers>]
   [{<body>}]
```

- <method> – указывает, какого рода действие нужно выполнить с ресурсом:
  - GET – получение данных об объекте (ресурсе) или данных в виде списка объектов по указанному URI.
  - POST – создание нового объекта или выполнение команды контроллера.
  - PUT и PATCH – обновление указанного объекта.
  - DELETE – удаление объекта.
- <URI> – указывает на конкретный ресурс, с которым необходимо выполнить действие. При необходимости к URI добавляют параметры запроса (query string).
- <headers> – заголовки HTTP содержат дополнительные метаданные запроса, такие как:
  - тип контента;

  - параметры авторизации;
  - язык;
  - формат.
- <body> – тело запроса содержит данные, которые нужно передать на сервер.

Например, в результате следующего запроса будет создан пользователь с именем <your_username> и паролем <your_password>:

```
POST https://<address>/api/v2/users/ HTTP/1.1
Content-Type: application/json
Authorization: Bearer <your_token>

   {
      "username": "<your_username>",
      "password": "<your_password>"
   }
```

Здесь:

- POST – указывает на то, что новый объект должен быть создан на сервере.
- `https://<address>/api/v2/users/` – указывает на конкретный ресурс, к которому обращается запрос. В данном случае запрос отправляется к ресурсу `users` в рамках API.
- `HTTP/1.1` – название и версия протокола HTTP, используемые для связи между клиентом и сервером.
- `Content-Type: application/json` – заголовок, указывающий на то, что данные отправляются в формате JSON.
- `Authorization: Bearer <your_token>` – заголовок, указывающий на использование bearer token для авторизации и содержащий сам токен.

Структура ответа

Ответ на HTTP-запрос состоит из следующих частей:

```
<status_line>
   [<headers>]
   [{<body>}]
```

- <status_line> – статусная строка. Содержит название и номер версии используемого протокола, код статуса и его краткое описание. Самыми распространенными кодами статуса являются:
  - 200 – запрос успешно выполнен.
  - 201 – в результате запроса был создан новый объект.
  - 202 – запрос получен, но еще не обработан.
  - 301 – URL запрошенного ресурса был изменен навсегда. Новый URL будет указан в ответе.
  - 302 – URL запрошенного ресурса был временно изменен.
  - 400 – запрос был сформирован с ошибками.

  - 403 – нет прав доступа к запрошенному ресурсу.
  - 404 – запрошенный ресурс не существует.
  - 500 – на сервере произошла ошибка, в результате которой он не может успешно обработать запрос.

- <headers> – заголовки, содержащие сведения об ответе. Могут включать информацию о типе содержимого, кэшировании, дате и времени, сервере, длине тела ответа и другие сведения.
- <body> – тело ответа, содержащее данные, отправленные от сервера к клиенту в ответ на запрос. Может содержать данные различных типов, например, текст, JSON и так далее, в зависимости от запроса и целей обмена информацией.

Пример ответа на запрос о создании пользователя:

```
HTTP/1.1 201 Created
Content-Type: application/json

   {
   "id": 7,
   "type": "user",
   "url": "/api/v2/users/7/",
   "related": {
      "named_url": "/api/v2/users/<your_username>/",
      "teams": "/api/v2/users/7/teams/",
      "organizations": "/api/v2/users/7/organizations/",
      "admin_of_organizations": "/api/v2/users/7/admin_of_organizations/",
      "projects": "/api/v2/users/7/projects/",
      "credentials": "/api/v2/users/7/credentials/",
      "roles": "/api/v2/users/7/roles/",
      "activity_stream": "/api/v2/users/7/activity_stream/",
      "access_list": "/api/v2/users/7/access_list/",
      "tokens": "/api/v2/users/7/tokens/",
      "authorized_tokens": "/api/v2/users/7/authorized_tokens/",
      "personal_tokens": "/api/v2/users/7/personal_tokens/"
   },
   "summary_fields": {
      "user_capabilities": {
            "edit": true,
            "delete": true
      }
   },
   "created": "2024-03-28T22:53:18.939698Z",
   "modified": null,
   "username": "<your_username>",
   "first_name": "",
   "last_name": "",
   "email": "",
   "is_superuser": false,
   "is_system_auditor": false,
   "ldap_dn": "",
   "last_login": null,
   "external_account": null,
   "auth": []
   }
```

Здесь:

- `id` – идентификатор созданного пользователя;
- `type` – тип пользователя;
- `url` – идентификатор ресурса;
- `related` – связанные ссылки на другие ресурсы, такие как команды, организации, проекты, роли и другие;
- `summaryfields` – сводные поля с информацией о возможностях пользователя, в данном случае: редактирование, удаление;
- `created` – дата и время создания пользователя;
- `modified` – дата и время последней модификации пользователя;
- `username` – название учетной записи пользователя;
- `firstname` – имя пользователя;
- `lastname` – фамилия пользователя;
- `email` – электронная почта пользователя;
- `issuperuser` – сведения о том, является ли пользователь системным администратором;
- `issystemauditor` – сведения о том, является ли пользователь системным аудитором;
- `ldapdn` – уникальное имя пользователя в системе LDAP;
- `lastlogin` – дата последнего входа пользователя;
- `externalaccount` – внешний аккаунт;
- `auth` – данные аутентификации пользователя.

6.4.2 Методы аутентификации

API Astra Automation Controller поддерживает аутентификацию следующими способами:

- аутентификация сеанса (session authentication);
- базовая аутентификация (basic authentication);
- аутентификация через токен OAuth 2 (OAuth 2 token authentication).

Примечание: Для доступа к API рекомендуется использовать аутентификацию через токен OAuth 2.

Аутентификация сеанса

Аутентификация сеанса используется при входе в API или пользовательский интерфейс Astra Automation Controller. С помощью утилиты `curl` можно увидеть активность, которая происходит при входе в Astra Automation Controller:

1. Получите X-CSRFToken с помощью команды:

   ```
   curl -k -c - https://<address>/api/login/
   ```

   Пример вывода:

   ```
   192.168.56.11   FALSE   /       TRUE    0       csrftoken       ␣
   ↪n97Hk5MjncJ45qHYmPFoqx8dLy8MPQMtfOafEZZlO9oPTJIztoD8qXRVuJY1OVsQ
   ```

   Здесь:

   - FALSE – указывает, что cookie не требует безопасного соединения (HTTPS);
   - `/` – URI, к которому относится установленный cookie;
   - TRUE – указывает, что cookie доступен для всех поддоменов этого домена;
   - `0` – срок действия cookie (в секундах) отсутствует, поэтому cookie действителен только в рамках текущей сессии;
   - `csrftoken` – название cookie;
   - `n97Hk5MjncJ45qHYmPFoqx8dLy8MPQMtfOafEZZlO9oPTJIztoD8qXRVuJY1OVsQ` – CSRF-токен.

2. Выполните запрос аутентификации:

   ```
   curl -X POST -H 'Content-Type: application/x-www-form-urlencoded' \
        --referer https://<address>/api/login/ \
        -H 'X-CSRFToken: <your_CSRFToken>' \
        --data 'username=<username>&password=<password>' \
        --cookie 'csrftoken=<your_CSRFToken>' \
        https://<address>/api/login/ -k -D - -o /dev/null
   ```

   Здесь:

   - <your_CSRFToken> – CSRF-токен, полученный ранее;
   - <username> – имя пользователя;
   - <password> – пароль.

   Если пользователь успешно аутентифицирован, сервер возвращает ответ, в заголовке которого в поле `X-API-Session-Cookie-Name` указывает название ключа, определяющего идентификатор сеанса связи, в составе cookie. По умолчанию значение этого параметра равно `awx_sessionid`.

   Пример ответа:

   ```
   Server: nginx
   Date: Mon, 21 Mar 2024 18:19:04 GMT
   Content-Type: text/html; charset=utf-8
   Content-Length: 0
   Connection: keep-alive
   Location: /accounts/profile/
   X-API-Session-Cookie-Name: awx_sessionid
   ```

(continues on next page)

(продолжение с предыдущей страницы)

```
Expires: Mon, 21 Mar 2024 18:19:04 GMT
Cache-Control: max-age=0, no-cache, no-store, must-revalidate, private
Vary: Cookie, Accept-Language, Origin
Session-Timeout: 1800
Content-Language: en
X-API-Total-Time: 0.888s
X-API-Request-Id: 0d9f18d32e6d40efb6e0427353c073d6
Access-Control-Expose-Headers: X-API-Request-Id
Set-Cookie: userLoggedIn=true; Path=/
Set-Cookie:␣
↪csrftoken=ZG2vbmRfTDksShZVBoHltV7pRtCpZcK9G2WBir1hsgOq6vpsM7IRvH9QmjSSBDBm;␣
↪Path=/; SameSite=Lax; Secure
Set-Cookie: awx_sessionid=89174mqdxcd2etvhp5umdysjxpr5fxdu; expires=Mon, 21 Mar␣
↪2024 18:49:04 GMT; HttpOnly; Max-Age=1800; Path=/; SameSite=Lax; Secure
Strict-Transport-Security: max-age=63072000
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, must-revalidate
Expires: 0
Pragma: no-cache
```

Базовая аутентификация

При использовании базовой аутентификации состояние аутентификации не сохраняется. Поэтому имя пользователя (username) и пароль (password) в формате Base64 должны отправляться вместе с каждым запросом через заголовок авторизации. Это применимо для локальных записей и для учетных записей LDAP.

Пример запроса:

```
curl -X GET --user '<username>:<password>' https://<address>/api/v2/credentials -k -L
```

В целях безопасности вы можете отключить базовую аутентификацию в графическом интерфейсе. Для этого выполните следующие действия:

1. Перейдите в раздел **Настройки** (Settings).
2. На панели **Система** (System) нажмите на ссылку *Общие параметры аутентификации* (Miscellaneous Authentication settings).
3. Нажмите кнопку *Редактировать* (Edit).
4. Выключите флаг **Включить HTTP Basic Auth** (Enable HTTP Basic Auth).
5. Нажмите кнопку *Сохранить* (Save).

Аутентификация через токен OAuth 2

Аутентификация через токен OAuth 2 обычно используется при программном взаимодействии с API Astra Automation Controller. Как и при базовой аутентификации, токен OAuth 2 предоставляется с каждым запросом API через заголовок авторизации. Токены имеют настраиваемый срок действия и при необходимости могут быть отозваны администратором для одного пользователя или для всей системы. По умолчанию внешним пользователям, например, созданным с помощью единого входа, не разрешено создавать токены OAuth 2 в целях безопасности.

Чтобы разрешить внешним пользователям создавать токены OAuth 2, выполните следующие действия в графическом интерфейсе контроллера:

1. Перейдите в раздел **Настройки** (Settings).
2. На панели **Система** (System) нажмите на ссылку *Общие параметры аутентификации* (Miscellaneous Authentication settings).
3. Нажмите кнопку *Редактировать* (Edit).
4. Включите флаг **Разрешить внешним пользователям создавать токены OAuth2** (Allow External Users to Create OAuth2 Tokens).
5. Нажмите кнопку *Сохранить* (Save).

Создать токен доступа OAuth 2 можно с помощью следующих средств:

- графический интерфейс;
- запрос API.

Создание токена в графическом интерфейсе

Пошаговые инструкции по управлению токенами через графический интерфейс доступны в подразделе *Управление токенами*.

Создание токена с помощью запроса

Создайте токен с помощью команды:

```
curl -u <username>:<password> -k -X POST https://<address>/api/v2/tokens/
```

Пример вывода команды:

```
{
   "id": 6,
   "type": "o_auth2_access_token",
   "url": "/api/v2/tokens/6/",
   "related": {
      "user": "/api/v2/users/1/",
      "activity_stream": "/api/v2/tokens/6/activity_stream/"
   },
   "summary_fields": {
      "user": {
         "id": 1,
         "username": "admin",
         "first_name": "",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
        "last_name": ""
      }
   },
   "created": "2024-03-21T20:53:09.035133Z",
   "modified": "2024-03-21T20:53:09.041729Z",
   "description": "",
   "user": 1,
   "token": "<your_token>",
   "refresh_token": null,
   "application": null,
   "expires": "3023-07-23T20:53:09.030794Z",
   "scope": "write"
}
```


Здесь <your_token> – токен аутентификации, который теперь можно использовать для выполнения запроса, например:

```
curl -k -X POST \
   https://<address>/api/v2/hosts/ \
   -H “Content-Type: application/json” \
   -H “Authorization: Bearer <your_token>`
```

6.4.3 Клиенты API

Для взаимодействия с контроллером по API можно использовать следующие типы клиентов:

- браузер;
- утилиту curl[160];
- специальные приложения, часто называемые REST API clients[161];
- расширения для браузера (например, Postman[162]);
- программу на любом языке программирования (например, Python[163]).

Браузер

Примечание: Доступ к API с помощью браузера возможен только после прохождения авторизации в графическом интерфейсе контроллера.

В адресной строке браузера введите `https://<address>/api/v2/`, где <address> – *FQDN* или IP-адрес контроллера. При выборе любой точки доступа API (API endpoint) отображается страница с результатом GET-запроса к ней. Например, по адресу `https://<address>/api/v2/` вы увидите результат запроса на получение списка всех доступных точек доступа. А по адресу `https://<address>/api/v2/ping/` вы увидите результат выполнения команды GET `/api/v2/ping/`, которая используется для проверки доступности сервера через API.

[160] https://curl.se/
[161] https://hevodata.com/learn/rest-clients/
[162] https://www.postman.com/
[163] https://www.python.org/

Утилита curl

Чтобы улучшить формат вывода JSON, в дополнение к утилите `curl` установите утилиту `jq`, с помощью команды:

```
sudo apt install jq
```

Для составления curl-запроса используйте шаблон:

```
curl -X GET --user <username>:<password> https://<address>/api/v2/<api_endpoint> -k -
↪s | jq .
```

Здесь:

- `-X GET` – метод HTTP, который вы хотите использовать в своем запросе. GET используется для запроса данных с сервера.
- `--user <username>:<password>` – ключ, в котором необходимо указать имя пользователя и пароль для аутентификации на сервере.
- `https://<address>/api/v2/<api_endpoint>` – URI, на который отправляется запрос.
- <address> – FQDN или IP-адрес контроллера.
- <api_endpoint> – точка доступа API (URI), к которой делается запрос.
- `-k` или `--insecure` – флаг, позволяющий утилите `curl` игнорировать проверки сертификата SSL. Это может быть полезно для тестов или в случаях, когда используются самоподписанные сертификаты.
- `-s` или `--silent` – флаг, отключающий индикаторы прогресса и сообщения об ошибках.
- `jq` – утилита для фильтрации и преобразования данных JSON.

Примеры использования curl

Обращение к корневой точке доступа API:

```
curl -X GET -k https://<address>/api/ | jq .
```

Пример вывода команды:

```
{
   "description": "AWX REST API",
   "current_version": "/api/v2/",
   "available_versions": {
      "v2": "/api/v2/"
   },
   "oauth2": "/api/o/",
   "custom_logo": "",
   "custom_login_info": "",
   "login_redirect_override": ""
}
```

Получение списка всех точек доступа:

```
curl -X GET -k https://<address>/api/v2/ | jq .
```

Пример вывода команды:


```
{
  "ping": "/api/v2/ping/",
  "instances": "/api/v2/instances/",
  "instance_groups": "/api/v2/instance_groups/",
  "config": "/api/v2/config/",
  "settings": "/api/v2/settings/",
  "me": "/api/v2/me/",
  "dashboard": "/api/v2/dashboard/",
  "organizations": "/api/v2/organizations/",
  "users": "/api/v2/users/",
  "execution_environments": "/api/v2/execution_environments/",
  "projects": "/api/v2/projects/",
  "project_updates": "/api/v2/project_updates/",
  "teams": "/api/v2/teams/",
  "credentials": "/api/v2/credentials/",
  "credential_types": "/api/v2/credential_types/",
  "credential_input_sources": "/api/v2/credential_input_sources/",
  "applications": "/api/v2/applications/",
  "tokens": "/api/v2/tokens/",
  "metrics": "/api/v2/metrics/",
  "inventory": "/api/v2/inventories/",
  "constructed_inventory": "/api/v2/constructed_inventories/",
  "inventory_sources": "/api/v2/inventory_sources/",
  "inventory_updates": "/api/v2/inventory_updates/",
  "groups": "/api/v2/groups/",
  "hosts": "/api/v2/hosts/",
  "host_metrics": "/api/v2/host_metrics/",
  "job_templates": "/api/v2/job_templates/",
  "jobs": "/api/v2/jobs/",
  "ad_hoc_commands": "/api/v2/ad_hoc_commands/",
  "system_job_templates": "/api/v2/system_job_templates/",
  "system_jobs": "/api/v2/system_jobs/",
  "schedules": "/api/v2/schedules/",
  "roles": "/api/v2/roles/",
  "notification_templates": "/api/v2/notification_templates/",
  "notifications": "/api/v2/notifications/",
  "labels": "/api/v2/labels/",
  "unified_job_templates": "/api/v2/unified_job_templates/",
  "unified_jobs": "/api/v2/unified_jobs/",
  "activity_stream": "/api/v2/activity_stream/",
  "workflow_job_templates": "/api/v2/workflow_job_templates/",
  "workflow_jobs": "/api/v2/workflow_jobs/",
  "workflow_approvals": "/api/v2/workflow_approvals/",
  "workflow_job_template_nodes": "/api/v2/workflow_job_template_nodes/",
  "workflow_job_nodes": "/api/v2/workflow_job_nodes/",
  "mesh_visualizer": "/api/v2/mesh_visualizer/",
  "bulk": "/api/v2/bulk/",
  "analytics": "/api/v2/analytics/"
}
```


Получение списка шаблонов заданий:

```
curl -X GET --user <admin>:<password> -k https://<address>/api/v2/job_templates/ | jq␣
↪.
```

Пример вывода команды:


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 7,
      "type": "job_template",
      "url": "/api/v2/job_templates/7/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "labels": "/api/v2/job_templates/7/labels/",
        "inventory": "/api/v2/inventories/1/",
        "project": "/api/v2/projects/6/",
        "organization": "/api/v2/organizations/1/",
        "credentials": "/api/v2/job_templates/7/credentials/",
        "last_job": "/api/v2/jobs/18/",
        "jobs": "/api/v2/job_templates/7/jobs/",
        "schedules": "/api/v2/job_templates/7/schedules/",
        "activity_stream": "/api/v2/job_templates/7/activity_stream/",
        "launch": "/api/v2/job_templates/7/launch/",
        "webhook_key": "/api/v2/job_templates/7/webhook_key/",
        "webhook_receiver": "",
        "notification_templates_started": "/api/v2/job_templates/7/notification_
↪templates_started/",
        "notification_templates_success": "/api/v2/job_templates/7/notification_
↪templates_success/",
        "notification_templates_error": "/api/v2/job_templates/7/notification_
↪templates_error/",
        "access_list": "/api/v2/job_templates/7/access_list/",
        "survey_spec": "/api/v2/job_templates/7/survey_spec/",
        "object_roles": "/api/v2/job_templates/7/object_roles/",
        "instance_groups": "/api/v2/job_templates/7/instance_groups/",
        "slice_workflow_jobs": "/api/v2/job_templates/7/slice_workflow_jobs/",
        "copy": "/api/v2/job_templates/7/copy/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "inventory": {
          "id": 1,
          "name": "Demo Inventory",
          "description": "",
          "has_active_failures": false,
          "total_hosts": 1,
          "hosts_with_active_failures": 0,
          "total_groups": 0,
          "has_inventory_sources": false,
          "total_inventory_sources": 0,
          "inventory_sources_with_failures": 0,
          "organization_id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "kind": ""
    },
    "project": {
      "id": 6,
      "name": "Demo Project",
      "description": "",
      "status": "successful",
      "scm_type": "git",
      "allow_override": false
    },
    "last_job": {
      "id": 18,
      "name": "Demo Job Template",
      "description": "",
      "finished": "2024-03-15T12:21:31.648339Z",
      "status": "successful",
      "failed": false
    },
    "last_update": {
      "id": 18,
      "name": "Demo Job Template",
      "description": "",
      "status": "successful",
      "failed": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the job template",
        "name": "Admin",
        "id": 37
      },
      "execute_role": {
        "description": "May run the job template",
        "name": "Execute",
        "id": 38
      },
      "read_role": {
        "description": "May view settings for the job template",
        "name": "Read",
        "id": 39
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "start": true,
        "schedule": true,
        "copy": true
      },
      "labels": {
        "count": 0,
        "results": []
      },
      "recent_jobs": [
        {
          "id": 18,
          "status": "successful",
          "finished": "2024-03-15T12:21:31.648339Z",
          "canceled_on": null,
          "type": "job"
        },
        {
          "id": 1,
          "status": "successful",
          "finished": "2024-02-16T00:15:25.600100Z",
          "canceled_on": null,
          "type": "job"
        }
      ],
      "credentials": [
        {
          "id": 1,
          "name": "Demo Credential",
          "description": "",
          "kind": "ssh",
          "cloud": false
        }
      ]
    },
    "created": "2024-02-16T00:06:03.362775Z",
    "modified": "2024-02-16T00:06:03.362794Z",
    "name": "Demo Job Template",
    "description": "",
    "job_type": "run",
    "inventory": 1,
    "project": 6,
    "playbook": "hello_world.yml",
    "scm_branch": "",
    "forks": 0,
    "limit": "",
    "verbosity": 0,
    "extra_vars": "",
    "job_tags": "",
    "force_handlers": false,
    "skip_tags": "",
    "start_at_task": "",
    "timeout": 0,
    "use_fact_cache": false,
    "organization": 1,
    "last_job_run": "2024-03-15T12:21:31.648339Z",
    "last_job_failed": false,
    "next_job_run": null,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "status": "successful",
      "execution_environment": null,
      "host_config_key": "",
      "ask_scm_branch_on_launch": false,
      "ask_diff_mode_on_launch": false,
      "ask_variables_on_launch": false,
      "ask_limit_on_launch": false,
      "ask_tags_on_launch": false,
      "ask_skip_tags_on_launch": false,
      "ask_job_type_on_launch": false,
      "ask_verbosity_on_launch": false,
      "ask_inventory_on_launch": false,
      "ask_credential_on_launch": false,
      "ask_execution_environment_on_launch": false,
      "ask_labels_on_launch": false,
      "ask_forks_on_launch": false,
      "ask_job_slice_count_on_launch": false,
      "ask_timeout_on_launch": false,
      "ask_instance_groups_on_launch": false,
      "survey_enabled": false,
      "become_enabled": false,
      "diff_mode": false,
      "allow_simultaneous": false,
      "custom_virtualenv": null,
      "job_slice_count": 1,
      "webhook_service": "",
      "webhook_credential": null,
      "prevent_instance_group_fallback": false
    },
    {
      "id": 9,
      "type": "job_template",
      "url": "/api/v2/job_templates/9/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "labels": "/api/v2/job_templates/9/labels/",
        "inventory": "/api/v2/inventories/2/",
        "project": "/api/v2/projects/8/",
        "organization": "/api/v2/organizations/1/",
        "credentials": "/api/v2/job_templates/9/credentials/",
        "last_job": "/api/v2/jobs/17/",
        "jobs": "/api/v2/job_templates/9/jobs/",
        "schedules": "/api/v2/job_templates/9/schedules/",
        "activity_stream": "/api/v2/job_templates/9/activity_stream/",
        "launch": "/api/v2/job_templates/9/launch/",
        "webhook_key": "/api/v2/job_templates/9/webhook_key/",
        "webhook_receiver": "",
        "notification_templates_started": "/api/v2/job_templates/9/notification_
↪templates_started/",
        "notification_templates_success": "/api/v2/job_templates/9/notification_
↪templates_success/",
        "notification_templates_error": "/api/v2/job_templates/9/notification_
↪templates_error/",
        "access_list": "/api/v2/job_templates/9/access_list/",
        "survey_spec": "/api/v2/job_templates/9/survey_spec/",
        "object_roles": "/api/v2/job_templates/9/object_roles/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "instance_groups": "/api/v2/job_templates/9/instance_groups/",
        "slice_workflow_jobs": "/api/v2/job_templates/9/slice_workflow_jobs/",
        "copy": "/api/v2/job_templates/9/copy/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "inventory": {
          "id": 2,
          "name": "NGINX inventory",
          "description": "",
          "has_active_failures": true,
          "total_hosts": 1,
          "hosts_with_active_failures": 1,
          "total_groups": 0,
          "has_inventory_sources": false,
          "total_inventory_sources": 0,
          "inventory_sources_with_failures": 0,
          "organization_id": 1,
          "kind": ""
        },
        "project": {
          "id": 8,
          "name": "Развертывание NGINX",
          "description": "",
          "status": "successful",
          "scm_type": "git",
          "allow_override": false
        },
        "last_job": {
          "id": 17,
          "name": "Развертывание NGINX",
          "description": "",
          "finished": "2024-03-15T12:11:55.017427Z",
          "status": "failed",
          "failed": true
        },
        "last_update": {
          "id": 17,
          "name": "Развертывание NGINX",
          "description": "",
          "status": "failed",
          "failed": true
        },
        "created_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "modified_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "last_name": ""
      },
      "object_roles": {
        "admin_role": {
          "description": "Can manage all aspects of the job template",
          "name": "Admin",
          "id": 64
        },
        "execute_role": {
          "description": "May run the job template",
          "name": "Execute",
          "id": 65
        },
        "read_role": {
          "description": "May view settings for the job template",
          "name": "Read",
          "id": 66
        }
      },
      "user_capabilities": {
        "edit": true,
        "delete": true,
        "start": true,
        "schedule": true,
        "copy": true
      },
      "labels": {
        "count": 0,
        "results": []
      },
      "recent_jobs": [
        {
          "id": 17,
          "status": "failed",
          "finished": "2024-03-15T12:11:55.017427Z",
          "canceled_on": null,
          "type": "job"
        }
      ],
      "credentials": []
    },
    "created": "2024-03-15T12:11:27.168941Z",
    "modified": "2024-03-15T12:11:27.168956Z",
    "name": "Развертывание NGINX",
    "description": "",
    "job_type": "run",
    "inventory": 2,
    "project": 8,
    "playbook": "nginx/site.yml",
    "scm_branch": "",
    "forks": 0,
    "limit": "",
    "verbosity": 0,
    "extra_vars": "---",
    "job_tags": "",
    "force_handlers": false,
    "skip_tags": "",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "start_at_task": "",
      "timeout": 0,
      "use_fact_cache": false,
      "organization": 1,
      "last_job_run": "2024-03-15T12:11:55.017427Z",
      "last_job_failed": true,
      "next_job_run": null,
      "status": "failed",
      "execution_environment": null,
      "host_config_key": "",
      "ask_scm_branch_on_launch": false,
      "ask_diff_mode_on_launch": false,
      "ask_variables_on_launch": false,
      "ask_limit_on_launch": false,
      "ask_tags_on_launch": false,
      "ask_skip_tags_on_launch": false,
      "ask_job_type_on_launch": false,
      "ask_verbosity_on_launch": false,
      "ask_inventory_on_launch": false,
      "ask_credential_on_launch": false,
      "ask_execution_environment_on_launch": false,
      "ask_labels_on_launch": false,
      "ask_forks_on_launch": false,
      "ask_job_slice_count_on_launch": false,
      "ask_timeout_on_launch": false,
      "ask_instance_groups_on_launch": false,
      "survey_enabled": false,
      "become_enabled": false,
      "diff_mode": false,
      "allow_simultaneous": false,
      "custom_virtualenv": null,
      "job_slice_count": 1,
      "webhook_service": "",
      "webhook_credential": null,
      "prevent_instance_group_fallback": false
    }
  ]
}
```


Запуск задания из шаблона:

```
curl -X POST --user <admin>:<password> -k https://<address>/api/v2/job_templates/7/
↪launch/ | jq .
```

Пример вывода команды:


```
{
  "job": 26,
  "ignored_fields": {},
  "id": 26,
  "type": "job",
  "url": "/api/v2/jobs/26/",
  "related": {
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "labels": "/api/v2/jobs/26/labels/",
    "inventory": "/api/v2/inventories/1/",
    "project": "/api/v2/projects/6/",
    "organization": "/api/v2/organizations/1/",
    "credentials": "/api/v2/jobs/26/credentials/",
    "unified_job_template": "/api/v2/job_templates/7/",
    "stdout": "/api/v2/jobs/26/stdout/",
    "job_events": "/api/v2/jobs/26/job_events/",
    "job_host_summaries": "/api/v2/jobs/26/job_host_summaries/",
    "activity_stream": "/api/v2/jobs/26/activity_stream/",
    "notifications": "/api/v2/jobs/26/notifications/",
    "create_schedule": "/api/v2/jobs/26/create_schedule/",
    "job_template": "/api/v2/job_templates/7/",
    "cancel": "/api/v2/jobs/26/cancel/",
    "relaunch": "/api/v2/jobs/26/relaunch/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
      "name": "Default",
      "description": ""
    },
    "inventory": {
      "id": 1,
      "name": "Demo Inventory",
      "description": "",
      "has_active_failures": false,
      "total_hosts": 1,
      "hosts_with_active_failures": 0,
      "total_groups": 0,
      "has_inventory_sources": false,
      "total_inventory_sources": 0,
      "inventory_sources_with_failures": 0,
      "organization_id": 1,
      "kind": ""
    },
    "project": {
      "id": 6,
      "name": "Demo Project",
      "description": "",
      "status": "successful",
      "scm_type": "git",
      "allow_override": false
    },
    "job_template": {
      "id": 7,
      "name": "Demo Job Template",
      "description": ""
    },
    "unified_job_template": {
      "id": 7,
      "name": "Demo Job Template",
      "description": "",
      "unified_job_type": "job"
    },
    "created_by": {
      "id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "user_capabilities": {
      "delete": true,
      "start": true
    },
    "labels": {
      "count": 0,
      "results": []
    },
    "credentials": [
      {
        "id": 1,
        "name": "Demo Credential",
        "description": "",
        "kind": "ssh",
        "cloud": false
      }
    ]
  },
  "created": "2024-03-19T06:41:24.347473Z",
  "modified": "2024-03-19T06:41:24.389125Z",
  "name": "Demo Job Template",
  "description": "",
  "job_type": "run",
  "inventory": 1,
  "project": 6,
  "playbook": "hello_world.yml",
  "scm_branch": "",
  "forks": 0,
  "limit": "",
  "verbosity": 0,
  "extra_vars": "{}",
  "job_tags": "",
  "force_handlers": false,
  "skip_tags": "",
  "start_at_task": "",
  "timeout": 0,
  "use_fact_cache": false,
  "organization": 1,
  "unified_job_template": 7,
  "launch_type": "manual",
  "status": "pending",
  "execution_environment": null,
  "failed": false,
  "started": null,
  "finished": null,
  "canceled_on": null,
  "elapsed": 0,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  "job_args": "",
  "job_cwd": "",
  "job_env": {},
  "job_explanation": "",
  "execution_node": "",
  "controller_node": "",
  "result_traceback": "",
  "event_processing_finished": false,
  "launched_by": {
    "id": 1,
    "name": "admin",
    "type": "user",
    "url": "/api/v2/users/1/"
  },
  "work_unit_id": null,
  "job_template": 7,
  "passwords_needed_to_start": [],
  "allow_simultaneous": false,
  "artifacts": {},
  "scm_revision": "",
  "instance_group": null,
  "diff_mode": false,
  "job_slice_number": 0,
  "job_slice_count": 1,
  "webhook_service": "",
  "webhook_credential": null,
  "webhook_guid": ""
}
```


Получение токена авторизации:

```
curl -X POST \
https://<address>/api/v2/tokens/ \
-H 'Content-Type: application/json' \
-d '{
      "description": "Token description",
      "application": null,
      "user": "admin",
      "scope": "write"
   }' \
-u '<admin>:<password>' | jq .
```

Пример вывода команды:


```
{
  "id": 2,
  "type": "o_auth2_access_token",
  "url": "/api/v2/tokens/2/",
  "related": {
    "user": "/api/v2/users/1/",
    "activity_stream": "/api/v2/tokens/2/activity_stream/"
  },
  "summary_fields": {
    "user": {
      "id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "username": "admin",
      "first_name": "",
      "last_name": ""
    }
  },
  "created": "2024-03-19T06:55:01.493262Z",
  "modified": "2024-03-19T06:55:01.506786Z",
  "description": "Token description",
  "user": 1,
  "token": "<your_token>",
  "refresh_token": null,
  "application": null,
  "expires": "3023-07-27T06:55:01.487431Z",
  "scope": "write"
}
```


Здесь <your_token> – токен авторизации.

Проверка статуса задания, запущенного ранее:

```
curl -X GET \
https://<address>/api/v2/jobs/<jobs_id>/ \
-H 'Authorization: Bearer <your_token>' \
-k -s | jq .
```

Здесь <jobs_id> – идентификатор задания, запущенного ранее.

Пример вывода команды:


```
{
  "id": 26,
  "type": "job",
  "url": "/api/v2/jobs/26/",
  "related": {
    "created_by": "/api/v2/users/1/",
    "labels": "/api/v2/jobs/26/labels/",
    "inventory": "/api/v2/inventories/1/",
    "project": "/api/v2/projects/6/",
    "organization": "/api/v2/organizations/1/",
    "credentials": "/api/v2/jobs/26/credentials/",
    "unified_job_template": "/api/v2/job_templates/7/",
    "stdout": "/api/v2/jobs/26/stdout/",
    "execution_environment": "/api/v2/execution_environments/1/",
    "job_events": "/api/v2/jobs/26/job_events/",
    "job_host_summaries": "/api/v2/jobs/26/job_host_summaries/",
    "activity_stream": "/api/v2/jobs/26/activity_stream/",
    "notifications": "/api/v2/jobs/26/notifications/",
    "create_schedule": "/api/v2/jobs/26/create_schedule/",
    "job_template": "/api/v2/job_templates/7/",
    "cancel": "/api/v2/jobs/26/cancel/",
    "relaunch": "/api/v2/jobs/26/relaunch/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "name": "Default",
      "description": ""
    },
    "inventory": {
      "id": 1,
      "name": "Demo Inventory",
      "description": "",
      "has_active_failures": false,
      "total_hosts": 1,
      "hosts_with_active_failures": 0,
      "total_groups": 0,
      "has_inventory_sources": false,
      "total_inventory_sources": 0,
      "inventory_sources_with_failures": 0,
      "organization_id": 1,
      "kind": ""
    },
    "execution_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    },
    "project": {
      "id": 6,
      "name": "Demo Project",
      "description": "",
      "status": "successful",
      "scm_type": "git",
      "allow_override": false
    },
    "job_template": {
      "id": 7,
      "name": "Demo Job Template",
      "description": ""
    },
    "unified_job_template": {
      "id": 7,
      "name": "Demo Job Template",
      "description": "",
      "unified_job_type": "job"
    },
    "instance_group": {
      "id": 1,
      "name": "controlplane",
      "is_container_group": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "user_capabilities": {
      "delete": true,
      "start": true
    },
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "labels": {
      "count": 0,
      "results": []
    },
    "credentials": [
      {
        "id": 1,
        "name": "Demo Credential",
        "description": "",
        "kind": "ssh",
        "cloud": false
      }
    ]
  },
  "created": "2024-03-19T06:41:24.347473Z",
  "modified": "2024-03-19T06:41:24.596205Z",
  "name": "Demo Job Template",
  "description": "",
  "job_type": "run",
  "inventory": 1,
  "project": 6,
  "playbook": "hello_world.yml",
  "scm_branch": "",
  "forks": 0,
  "limit": "",
  "verbosity": 0,
  "extra_vars": "{}",
  "job_tags": "",
  "force_handlers": false,
  "skip_tags": "",
  "start_at_task": "",
  "timeout": 0,
  "use_fact_cache": false,
  "organization": 1,
  "unified_job_template": 7,
  "launch_type": "manual",
  "status": "successful",
  "execution_environment": 1,
  "failed": false,
  "started": "2024-03-19T06:41:24.694396Z",
  "finished": "2024-03-19T06:41:28.962513Z",
  "canceled_on": null,
  "elapsed": 4.268,
  "job_args": "[\"podman\", \"run\", \"--rm\", \"--tty\", \"--interactive\", \"--
↪workdir\", \"/runner/project\", \"-v\", \"/tmp/awx_26_b59w6psk/:/runner/:Z\", \"--
↪env-file\", \"/tmp/awx_26_b59w6psk/artifacts/26/env.list\", \"--quiet\", \"--name\",
↪ \"ansible_runner_26\", \"--user=root\", \"--network\", \"slirp4netns:enable_
↪ipv6=true\", \"registry.astralinux.ru/aa/aa-base-ee:0.2.1\", \"ansible-playbook\", \
↪"-u\", \"admin\", \"-i\", \"/runner/inventory/hosts\", \"-e\", \"@/runner/env/
↪extravars\", \"hello_world.yml\"]",
  "job_cwd": "/runner/project",
  "job_env": {
    "ANSIBLE_UNSAFE_WRITES": "1",
    "AWX_ISOLATED_DATA_DIR": "/runner/artifacts/26",
    "ANSIBLE_FORCE_COLOR": "True",
    "ANSIBLE_HOST_KEY_CHECKING": "False",
    "ANSIBLE_INVENTORY_UNPARSED_FAILED": "True",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "ANSIBLE_PARAMIKO_RECORD_HOST_KEYS": "False",
    "HOME": "/var/lib/awx",
    "AWX_PRIVATE_DATA_DIR": "/tmp/awx_26_b59w6psk",
    "JOB_ID": "26",
    "INVENTORY_ID": "1",
    "PROJECT_REVISION": "347e44fea036c94d5f60e544de006453ee5c71ad",
    "ANSIBLE_RETRY_FILES_ENABLED": "False",
    "MAX_EVENT_RES": "700000",
    "AWX_HOST": "https://controller",
    "ANSIBLE_SSH_CONTROL_PATH_DIR": "/runner/cp",
    "ANSIBLE_COLLECTIONS_PATHS": "/runner/requirements_collections:~/.ansible/
↪collections:/usr/share/ansible/collections",
    "ANSIBLE_ROLES_PATH": "/runner/requirements_roles:~/.ansible/roles:/usr/share/
↪ansible/roles:/etc/ansible/roles",
    "ANSIBLE_CALLBACK_PLUGINS": "/runner/artifacts/26/callback",
    "ANSIBLE_STDOUT_CALLBACK": "awx_display",
    "RUNNER_OMIT_EVENTS": "False",
    "RUNNER_ONLY_FAILED_EVENTS": "False"
  },
  "job_explanation": "",
  "execution_node": "192.168.56.11",
  "controller_node": "192.168.56.11",
  "result_traceback": "",
  "event_processing_finished": true,
  "launched_by": {
    "id": 1,
    "name": "admin",
    "type": "user",
    "url": "/api/v2/users/1/"
  },
  "work_unit_id": "Vt4HYQqA",
  "job_template": 7,
  "passwords_needed_to_start": [],
  "allow_simultaneous": false,
  "artifacts": {},
  "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
  "instance_group": 1,
  "diff_mode": false,
  "job_slice_number": 0,
  "job_slice_count": 1,
  "webhook_service": "",
  "webhook_credential": null,
  "webhook_guid": "",
  "host_status_counts": {
    "ok": 1
  },
  "playbook_counts": {
    "play_count": 1,
    "task_count": 2
  },
  "custom_virtualenv": null
}
```


Создание проекта:

```
curl -X POST \
https://<address>/api/v2/projects/ \
```

(continues on next page)

(продолжение с предыдущей страницы)

```
-H 'Authorization: Bearer <your_token>' \
-H 'Content-Type: application/json' \
-d '{
   "name": "New Project",
   "organization": 1,
   "scm_type": "git",
   "scm_url": "https://github.com/ansible/ansible-tower-samples"
}' \
-k -s | jq .
```

Пример вывода команды:

```
{
  "id": 13,
  "type": "project",
  "url": "/api/v2/projects/13/",
  "related": {
    "named_url": "/api/v2/projects/New Project++Default/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "current_job": "/api/v2/project_updates/29/",
    "teams": "/api/v2/projects/13/teams/",
    "playbooks": "/api/v2/projects/13/playbooks/",
    "inventory_files": "/api/v2/projects/13/inventories/",
    "update": "/api/v2/projects/13/update/",
    "project_updates": "/api/v2/projects/13/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/13/scm_inventory_sources/",
    "schedules": "/api/v2/projects/13/schedules/",
    "activity_stream": "/api/v2/projects/13/activity_stream/",
    "notification_templates_started": "/api/v2/projects/13/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/13/notification_templates_
↪success/",
    "notification_templates_error": "/api/v2/projects/13/notification_templates_error/
↪",
    "access_list": "/api/v2/projects/13/access_list/",
    "object_roles": "/api/v2/projects/13/object_roles/",
    "copy": "/api/v2/projects/13/copy/",
    "organization": "/api/v2/organizations/1/",
    "current_update": "/api/v2/project_updates/29/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
      "name": "Default",
      "description": ""
    },
    "current_update": {
      "id": 29,
      "name": "New Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "current_job": {
```

(continues on next page)

(продолжение с предыдущей страницы)


```
      "id": 29,
      "name": "New Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the project",
        "name": "Admin",
        "id": 79
      },
      "use_role": {
        "description": "Can use the project in a job template",
        "name": "Use",
        "id": 80
      },
      "update_role": {
        "description": "May update the project",
        "name": "Update",
        "id": 81
      },
      "read_role": {
        "description": "May view settings for the project",
        "name": "Read",
        "id": 82
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "start": true,
      "schedule": true,
      "copy": true
    }
  },
  "created": "2024-03-19T14:26:16.980872Z",
  "modified": "2024-03-19T14:26:16.980886Z",
  "name": "New Project",
  "description": "",
  "local_path": "_13__new_project",
  "scm_type": "git",
  "scm_url": "https://github.com/ansible/ansible-tower-samples",
  "scm_branch": "",
  "scm_refspec": "",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "",
  "last_job_run": null,
  "last_job_failed": false,
  "next_job_run": null,
  "status": "pending",
  "organization": 1,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": null,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": null
}
```


Изменение проекта, созданного ранее:

```
curl -X PATCH \
https://<address>/api/v2/projects/<project_id>/ \
-H 'Authorization: Bearer <your_token>' \
-H 'Content-Type: application/json' \
-d '{
   "name": "Updated Project",
   "scm_url": "https://github.com/ansible/ansible-examples"
}' \
-k -s | jq .
```

Здесь <project_id> – идентификатор проекта, созданного ранее.

Пример вывода команды:


```
{
  "id": 13,
  "type": "project",
  "url": "/api/v2/projects/13/",
  "related": {
    "named_url": "/api/v2/projects/Updated Project++Default/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "current_job": "/api/v2/project_updates/30/",
    "last_job": "/api/v2/project_updates/29/",
    "teams": "/api/v2/projects/13/teams/",
    "playbooks": "/api/v2/projects/13/playbooks/",
    "inventory_files": "/api/v2/projects/13/inventories/",
    "update": "/api/v2/projects/13/update/",
    "project_updates": "/api/v2/projects/13/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/13/scm_inventory_sources/",
    "schedules": "/api/v2/projects/13/schedules/",
    "activity_stream": "/api/v2/projects/13/activity_stream/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "notification_templates_started": "/api/v2/projects/13/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/13/notification_templates_
↪success/",
    "notification_templates_error": "/api/v2/projects/13/notification_templates_error/
↪",
    "access_list": "/api/v2/projects/13/access_list/",
    "object_roles": "/api/v2/projects/13/object_roles/",
    "copy": "/api/v2/projects/13/copy/",
    "organization": "/api/v2/organizations/1/",
    "current_update": "/api/v2/project_updates/30/",
    "last_update": "/api/v2/project_updates/29/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
      "name": "Default",
      "description": ""
    },
    "last_job": {
      "id": 29,
      "name": "New Project",
      "description": "",
      "finished": "2024-03-19T14:26:21.930637Z",
      "status": "successful",
      "failed": false
    },
    "last_update": {
      "id": 29,
      "name": "New Project",
      "description": "",
      "status": "successful",
      "failed": false
    },
    "current_update": {
      "id": 30,
      "name": "Updated Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "current_job": {
      "id": 30,
      "name": "Updated Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the project",
        "name": "Admin",
        "id": 79
      },
      "use_role": {
        "description": "Can use the project in a job template",
        "name": "Use",
        "id": 80
      },
      "update_role": {
        "description": "May update the project",
        "name": "Update",
        "id": 81
      },
      "read_role": {
        "description": "May view settings for the project",
        "name": "Read",
        "id": 82
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "start": true,
      "schedule": true,
      "copy": true
    },
    "resolved_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    }
  },
  "created": "2024-03-19T14:26:16.980872Z",
  "modified": "2024-03-19T14:28:32.951463Z",
  "name": "Updated Project",
  "description": "",
  "local_path": "_13__new_project",
  "scm_type": "git",
  "scm_url": "https://github.com/ansible/ansible-examples",
  "scm_branch": "",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
  "last_job_run": "2024-03-19T14:26:21.930637Z",
  "last_job_failed": false,
  "next_job_run": null,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  "status": "pending",
  "organization": 1,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": null,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": "2024-03-19T14:26:21.930637Z"
}
```


Удаление проекта:

```
curl -X DELETE \
https://<address>/api/v2/projects/<project_id>/ \
-H 'Authorization: Bearer <your-token>'
```

Расширения для браузера

Применение расширения для браузера Google Chrome выглядит следующим образом:

1. Откройте Chrome Web Store и установите расширение Postman на ваш браузер.
2. Запустите Postman из списка расширений в вашем браузере.
3. Создайте новый запрос.
4. Введите URI объекта, который вы хотите запросить, и выберите нужный метод HTTP (GET, POST, PUT, PATCH, DELETE).
5. Отправьте запрос.
6. После отправки запроса вы увидите ответ от сервера в формате JSON, XML или другом формате. Вы также можете просмотреть статус-код ответа и другие детали.

Программа на языке программирования

Рассмотрим в качестве примера скрипт Python, который выполняет следующие действия в контроллере:

1. Создает новый проект.
2. Получает информацию о всех проектах.
3. Получает информацию о созданном проекте.
4. Обновляет проект из источника.
5. Проверяет статус обновления проекта.
6. Изменяет проект.
7. Получает информацию о созданом проекте после изменения.
8. Удаляет проект.
9. Получает информацию о всех проектах.
10. Получает конфигурацию Astra Automation Controller.

После команд, запускающих какое-либо действие на сервере, добавлена тридцатисекундная пауза для наглядности выполнения команд.

Для запуска скрипта необходимо:

- Использовать Python не ниже версии 3.8.
- Установить пакет `requests`.

```
"""Demo scenario"""

import json
import time
import requests

# Define base URL and headers
base_url = "https://<address>/api/v2/"
headers = {"Authorization": "Bearer <your_token>", "Content-type": "application/json"}


def button_press():
    """Prompt the user to press Enter to continue."""
    input("\nНажмите Enter чтобы продолжить...\n")


def print_json(response):
    """
    Print the response JSON returned by a REST API request.

    Args:
        response: The response object returned by the REST API request
    """
    if response.text:
        print(json.dumps(response.json(), indent=4))
    else:
        print("Status code:", response.status_code)


# Define the project data
project_data = {
    "name": "New Project",
    "organization": 1,
    "scm_type": "git",
    "scm_url": "https://github.com/ansible/ansible-tower-samples",
}

# Post new project
response_post_new_project = requests.post(
    f"{base_url}projects/",
    headers=headers,
    data=json.dumps(project_data),
    verify=False,
    timeout=10,
)
print("\n--- Создание проекта ---")
print_json(response_post_new_project)
project_id = response_post_new_project.json()["id"]
time.sleep(30)
print("\n --- Проект создан ---")
```

(continues on next page)

(продолжение с предыдущей страницы)

```
button_press()

# Get all projects
response_get_all_projects = requests.get(
    f"{base_url}projects/", headers=headers, verify=False, timeout=10
)
print("\n--- Информация о проектах ---")
print_json(response_get_all_projects)
print("\n --- Выведена информация по всем проектам ---")
button_press()

# Get project details
response_get_project_detail = requests.get(
    f"{base_url}projects/{project_id}/", headers=headers, verify=False, timeout=10
)
print("\n--- Получение деталей проекта ---")
print_json(response_get_project_detail)
print("\n --- Выведена информация о созданном проекте ---")
button_press()

# Update project from the version control source
response_update_project = requests.post(
    f"{base_url}projects/{project_id}/update/",
    headers=headers,
    verify=False,
    timeout=10,
)
print("\n--- Обновление проекта из источника контроля версий ---")
print_json(response_update_project)
time.sleep(30)
print("\n --- Проект обновлен ---")
button_press()

# Check update status information
response_project_update_status = requests.get(
    f"{base_url}projects/{project_id}/project_updates/",
    headers=headers,
    verify=False,
    timeout=10,
)
print("\n--- Проверка информации о статусе обновления ---")
print_json(response_project_update_status)
print("\n --- Выведена информация о статусе обновления ---")
button_press()

# Change the project
patch_data = {
    "name": "Updated Project",
    "scm_url": "https://github.com/ansible/ansible-examples",
}
response_patch_project = requests.patch(
    f"{base_url}projects/{project_id}/",
    headers=headers,
    data=json.dumps(patch_data),
    verify=False,
    timeout=10,
)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print("\n--- Изменение проекта ---")
print_json(response_patch_project)
time.sleep(30)
print("\n --- Проект изменён ---")
button_press()

# Get project details after the change
response_get_project_detail_after_change = requests.get(
    f"{base_url}projects/{project_id}/", headers=headers, verify=False, timeout=10
)
print("\n--- Получение деталей проекта (после обновления) ---")
print_json(response_get_project_detail_after_change)
print("\n --- Выведена информация о созданном проекте после изменения ---")
button_press()

# Delete the project
response_delete_project = requests.delete(
    f"{base_url}projects/{project_id}/", headers=headers, verify=False, timeout=10
)
print("\n--- Удаление проекта ---")
print_json(response_delete_project)
print("\n --- Проект удалён ---")
button_press()

# List all projects after deletion
response_get_all_projects_after_deletion = requests.get(
    f"{base_url}projects/", headers=headers, verify=False, timeout=10
)
print("\n--- Список проектов (после удаления) ---")
print_json(response_get_all_projects_after_deletion)
print("\n --- Выведена информация по всем проектам после удаления ---")
button_press()

# Get Astra Automation Controller Configuration
response_get_configuration = requests.get(
    f"{base_url}config/", headers=headers, verify=False, timeout=10
)
print("\n--- Получение конфигурации Astra Automation Controller ---")
print_json(response_get_configuration)
print("\n --- Выведена информация по серверу Astra Automation Controller ---")
```

В демонстрационном скрипте замените значения <address> и <your_token> на свои.

Пример вывода:

Создание проекта:

```
{
  "id": 15,
  "type": "project",
  "url": "/api/v2/projects/15/",
  "related": {
    "named_url": "/api/v2/projects/New Project++Default/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
    "current_job": "/api/v2/project_updates/35/",
    "teams": "/api/v2/projects/15/teams/",
    "playbooks": "/api/v2/projects/15/playbooks/",
    "inventory_files": "/api/v2/projects/15/inventories/",
    "update": "/api/v2/projects/15/update/",
    "project_updates": "/api/v2/projects/15/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/15/scm_inventory_sources/",
    "schedules": "/api/v2/projects/15/schedules/",
    "activity_stream": "/api/v2/projects/15/activity_stream/",
    "notification_templates_started": "/api/v2/projects/15/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/15/notification_templates_
↪success/",
    "notification_templates_error": "/api/v2/projects/15/notification_templates_error/
↪",
    "access_list": "/api/v2/projects/15/access_list/",
    "object_roles": "/api/v2/projects/15/object_roles/",
    "copy": "/api/v2/projects/15/copy/",
    "organization": "/api/v2/organizations/1/",
    "current_update": "/api/v2/project_updates/35/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
      "name": "Default",
      "description": ""
    },
    "current_update": {
      "id": 35,
      "name": "New Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "current_job": {
      "id": 35,
      "name": "New Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the project",
        "name": "Admin",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "id": 100
      },
      "use_role": {
        "description": "Can use the project in a job template",
        "name": "Use",
        "id": 101
      },
      "update_role": {
        "description": "May update the project",
        "name": "Update",
        "id": 102
      },
      "read_role": {
        "description": "May view settings for the project",
        "name": "Read",
        "id": 103
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "start": true,
      "schedule": true,
      "copy": true
    }
  },
  "created": "2024-03-29T07:08:16.321624Z",
  "modified": "2024-03-29T07:08:16.321638Z",
  "name": "New Project",
  "description": "",
  "local_path": "_15__new_project",
  "scm_type": "git",
  "scm_url": "https://github.com/ansible/ansible-tower-samples",
  "scm_branch": "",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "",
  "last_job_run": null,
  "last_job_failed": false,
  "next_job_run": null,
  "status": "pending",
  "organization": 1,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": null,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": null
}
```


Получение информации о проектах:

```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 6,
      "type": "project",
      "url": "/api/v2/projects/6/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "teams": "/api/v2/projects/6/teams/",
        "playbooks": "/api/v2/projects/6/playbooks/",
        "inventory_files": "/api/v2/projects/6/inventories/",
        "update": "/api/v2/projects/6/update/",
        "project_updates": "/api/v2/projects/6/project_updates/",
        "scm_inventory_sources": "/api/v2/projects/6/scm_inventory_sources/",
        "schedules": "/api/v2/projects/6/schedules/",
        "activity_stream": "/api/v2/projects/6/activity_stream/",
        "notification_templates_started": "/api/v2/projects/6/notification_templates_
↪started/",
        "notification_templates_success": "/api/v2/projects/6/notification_templates_
↪success/",
        "notification_templates_error": "/api/v2/projects/6/notification_templates_
↪error/",
        "access_list": "/api/v2/projects/6/access_list/",
        "object_roles": "/api/v2/projects/6/object_roles/",
        "copy": "/api/v2/projects/6/copy/",
        "organization": "/api/v2/organizations/1/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "created_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "modified_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "object_roles": {
          "admin_role": {
            "description": "Can manage all aspects of the project",
            "name": "Admin",
            "id": 22
          },
          "use_role": {
            "description": "Can use the project in a job template",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "name": "Use",
            "id": 23
          },
          "update_role": {
            "description": "May update the project",
            "name": "Update",
            "id": 24
          },
          "read_role": {
            "description": "May view settings for the project",
            "name": "Read",
            "id": 25
          }
        },
        "user_capabilities": {
          "edit": true,
          "delete": true,
          "start": true,
          "schedule": true,
          "copy": true
        }
      },
      "created": "2024-02-16T00:06:02.670809Z",
      "modified": "2024-02-16T00:06:02.670834Z",
      "name": "Demo Project",
      "description": "",
      "local_path": "_6__demo_project",
      "scm_type": "git",
      "scm_url": "https://github.com/ansible/ansible-tower-samples",
      "scm_branch": "",
      "scm_refspec": "",
      "scm_clean": false,
      "scm_track_submodules": false,
      "scm_delete_on_update": false,
      "credential": null,
      "timeout": 0,
      "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
      "last_job_run": null,
      "last_job_failed": false,
      "next_job_run": null,
      "status": "successful",
      "organization": 1,
      "scm_update_on_launch": false,
      "scm_update_cache_timeout": 0,
      "allow_override": false,
      "custom_virtualenv": null,
      "default_environment": null,
      "signature_validation_credential": null,
      "last_update_failed": false,
      "last_updated": null
    },
    {
      "id": 8,
      "type": "project",
      "url": "/api/v2/projects/8/",
      "related": {
        "created_by": "/api/v2/users/1/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "modified_by": "/api/v2/users/1/",
        "credential": "/api/v2/credentials/4/",
        "last_job": "/api/v2/project_updates/16/",
        "teams": "/api/v2/projects/8/teams/",
        "playbooks": "/api/v2/projects/8/playbooks/",
        "inventory_files": "/api/v2/projects/8/inventories/",
        "update": "/api/v2/projects/8/update/",
        "project_updates": "/api/v2/projects/8/project_updates/",
        "scm_inventory_sources": "/api/v2/projects/8/scm_inventory_sources/",
        "schedules": "/api/v2/projects/8/schedules/",
        "activity_stream": "/api/v2/projects/8/activity_stream/",
        "notification_templates_started": "/api/v2/projects/8/notification_templates_
↪started/",
        "notification_templates_success": "/api/v2/projects/8/notification_templates_
↪success/",
        "notification_templates_error": "/api/v2/projects/8/notification_templates_
↪error/",
        "access_list": "/api/v2/projects/8/access_list/",
        "object_roles": "/api/v2/projects/8/object_roles/",
        "copy": "/api/v2/projects/8/copy/",
        "organization": "/api/v2/organizations/1/",
        "last_update": "/api/v2/project_updates/16/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "credential": {
          "id": 4,
          "name": "Astra Automation Hub",
          "description": "",
          "kind": "scm",
          "cloud": false,
          "kubernetes": false,
          "credential_type_id": 2
        },
        "last_job": {
          "id": 16,
          "name": "\u0420\u0430\u0437\u0432\u0435\u0440\u0442\u044b\u0432\u0430\u043d\
↪u0438\u0435 NGINX",
          "description": "",
          "finished": "2024-03-15T12:09:44.270752Z",
          "status": "successful",
          "failed": false
        },
        "last_update": {
          "id": 16,
          "name": "\u0420\u0430\u0437\u0432\u0435\u0440\u0442\u044b\u0432\u0430\u043d\
↪u0438\u0435 NGINX",
          "description": "",
          "status": "successful",
          "failed": false
        },
        "created_by": {
          "id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "username": "admin",
        "first_name": "",
        "last_name": ""
      },
      "modified_by": {
        "id": 1,
        "username": "admin",
        "first_name": "",
        "last_name": ""
      },
      "object_roles": {
        "admin_role": {
          "description": "Can manage all aspects of the project",
          "name": "Admin",
          "id": 60
        },
        "use_role": {
          "description": "Can use the project in a job template",
          "name": "Use",
          "id": 61
        },
        "update_role": {
          "description": "May update the project",
          "name": "Update",
          "id": 62
        },
        "read_role": {
          "description": "May view settings for the project",
          "name": "Read",
          "id": 63
        }
      },
      "user_capabilities": {
        "edit": true,
        "delete": true,
        "start": true,
        "schedule": true,
        "copy": true
      }
    },
    "created": "2024-03-15T11:57:40.851309Z",
    "modified": "2024-03-15T11:57:40.851323Z",
    "name": "\u0420\u0430\u0437\u0432\u0435\u0440\u0442\u044b\u0432\u0430\u043d\
↪u0438\u0435 NGINX",
    "description": "",
    "local_path": "_8__nginx",
    "scm_type": "git",
    "scm_url": "ssh://git@hub.astra-automation.ru:2222/aa-gca/AA/aac-samples.git",
    "scm_branch": "",
    "scm_refspec": "",
    "scm_clean": false,
    "scm_track_submodules": false,
    "scm_delete_on_update": false,
    "credential": 4,
    "timeout": 0,
    "scm_revision": "d7f0ff18cf8ffb44d64e8acd995402d545e2de90",
    "last_job_run": "2024-03-15T12:09:44.270752Z",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "last_job_failed": false,
      "next_job_run": null,
      "status": "successful",
      "organization": 1,
      "scm_update_on_launch": false,
      "scm_update_cache_timeout": 0,
      "allow_override": false,
      "custom_virtualenv": null,
      "default_environment": null,
      "signature_validation_credential": null,
      "last_update_failed": false,
      "last_updated": "2024-03-15T12:09:44.270752Z"
    },
    {
      "id": 15,
      "type": "project",
      "url": "/api/v2/projects/15/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "last_job": "/api/v2/project_updates/35/",
        "teams": "/api/v2/projects/15/teams/",
        "playbooks": "/api/v2/projects/15/playbooks/",
        "inventory_files": "/api/v2/projects/15/inventories/",
        "update": "/api/v2/projects/15/update/",
        "project_updates": "/api/v2/projects/15/project_updates/",
        "scm_inventory_sources": "/api/v2/projects/15/scm_inventory_sources/",
        "schedules": "/api/v2/projects/15/schedules/",
        "activity_stream": "/api/v2/projects/15/activity_stream/",
        "notification_templates_started": "/api/v2/projects/15/notification_templates_
↪started/",
        "notification_templates_success": "/api/v2/projects/15/notification_templates_
↪success/",
        "notification_templates_error": "/api/v2/projects/15/notification_templates_
↪error/",
        "access_list": "/api/v2/projects/15/access_list/",
        "object_roles": "/api/v2/projects/15/object_roles/",
        "copy": "/api/v2/projects/15/copy/",
        "organization": "/api/v2/organizations/1/",
        "last_update": "/api/v2/project_updates/35/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "last_job": {
          "id": 35,
          "name": "New Project",
          "description": "",
          "finished": "2024-03-29T07:08:21.153077Z",
          "status": "successful",
          "failed": false
        },
        "last_update": {
          "id": 35,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "name": "New Project",
        "description": "",
        "status": "successful",
        "failed": false
      },
      "created_by": {
        "id": 1,
        "username": "admin",
        "first_name": "",
        "last_name": ""
      },
      "modified_by": {
        "id": 1,
        "username": "admin",
        "first_name": "",
        "last_name": ""
      },
      "object_roles": {
        "admin_role": {
          "description": "Can manage all aspects of the project",
          "name": "Admin",
          "id": 100
        },
        "use_role": {
          "description": "Can use the project in a job template",
          "name": "Use",
          "id": 101
        },
        "update_role": {
          "description": "May update the project",
          "name": "Update",
          "id": 102
        },
        "read_role": {
          "description": "May view settings for the project",
          "name": "Read",
          "id": 103
        }
      },
      "user_capabilities": {
        "edit": true,
        "delete": true,
        "start": true,
        "schedule": true,
        "copy": true
      }
    },
    "created": "2024-03-29T07:08:16.321624Z",
    "modified": "2024-03-29T07:08:16.321638Z",
    "name": "New Project",
    "description": "",
    "local_path": "_15__new_project",
    "scm_type": "git",
    "scm_url": "https://github.com/ansible/ansible-tower-samples",
    "scm_branch": "",
    "scm_refspec": "",
    "scm_clean": false,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "scm_track_submodules": false,
      "scm_delete_on_update": false,
      "credential": null,
      "timeout": 0,
      "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
      "last_job_run": "2024-03-29T07:08:21.153077Z",
      "last_job_failed": false,
      "next_job_run": null,
      "status": "successful",
      "organization": 1,
      "scm_update_on_launch": false,
      "scm_update_cache_timeout": 0,
      "allow_override": false,
      "custom_virtualenv": null,
      "default_environment": null,
      "signature_validation_credential": null,
      "last_update_failed": false,
      "last_updated": "2024-03-29T07:08:21.153077Z"
    }
  ]
}
```


Получение деталей проекта:


```
{
  "id": 15,
  "type": "project",
  "url": "/api/v2/projects/15/",
  "related": {
    "named_url": "/api/v2/projects/New Project++Default/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "last_job": "/api/v2/project_updates/35/",
    "teams": "/api/v2/projects/15/teams/",
    "playbooks": "/api/v2/projects/15/playbooks/",
    "inventory_files": "/api/v2/projects/15/inventories/",
    "update": "/api/v2/projects/15/update/",
    "project_updates": "/api/v2/projects/15/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/15/scm_inventory_sources/",
    "schedules": "/api/v2/projects/15/schedules/",
    "activity_stream": "/api/v2/projects/15/activity_stream/",
    "notification_templates_started": "/api/v2/projects/15/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/15/notification_templates_
↪success/",
    "notification_templates_error": "/api/v2/projects/15/notification_templates_error/
↪",
    "access_list": "/api/v2/projects/15/access_list/",
    "object_roles": "/api/v2/projects/15/object_roles/",
    "copy": "/api/v2/projects/15/copy/",
    "organization": "/api/v2/organizations/1/",
    "last_update": "/api/v2/project_updates/35/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
      "name": "Default",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "description": ""
  },
  "last_job": {
    "id": 35,
    "name": "New Project",
    "description": "",
    "finished": "2024-03-29T07:08:21.153077Z",
    "status": "successful",
    "failed": false
  },
  "last_update": {
    "id": 35,
    "name": "New Project",
    "description": "",
    "status": "successful",
    "failed": false
  },
  "created_by": {
    "id": 1,
    "username": "admin",
    "first_name": "",
    "last_name": ""
  },
  "modified_by": {
    "id": 1,
    "username": "admin",
    "first_name": "",
    "last_name": ""
  },
  "object_roles": {
    "admin_role": {
      "description": "Can manage all aspects of the project",
      "name": "Admin",
      "id": 100
    },
    "use_role": {
      "description": "Can use the project in a job template",
      "name": "Use",
      "id": 101
    },
    "update_role": {
      "description": "May update the project",
      "name": "Update",
      "id": 102
    },
    "read_role": {
      "description": "May view settings for the project",
      "name": "Read",
      "id": 103
    }
  },
  "user_capabilities": {
    "edit": true,
    "delete": true,
    "start": true,
    "schedule": true,
    "copy": true
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    },
    "resolved_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    }
  },
  "created": "2024-03-29T07:08:16.321624Z",
  "modified": "2024-03-29T07:08:16.321638Z",
  "name": "New Project",
  "description": "",
  "local_path": "_15__new_project",
  "scm_type": "git",
  "scm_url": "https://github.com/ansible/ansible-tower-samples",
  "scm_branch": "",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
  "last_job_run": "2024-03-29T07:08:21.153077Z",
  "last_job_failed": false,
  "next_job_run": null,
  "status": "successful",
  "organization": 1,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": null,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": "2024-03-29T07:08:21.153077Z"
}
```


Обновление проекта из источника контроля версий:


```
{
  "project_update": 36,
  "id": 36,
  "type": "project_update",
  "url": "/api/v2/project_updates/36/",
  "related": {
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "unified_job_template": "/api/v2/projects/15/",
    "stdout": "/api/v2/project_updates/36/stdout/",
    "project": "/api/v2/projects/15/",
    "cancel": "/api/v2/project_updates/36/cancel/",
    "scm_inventory_updates": "/api/v2/project_updates/36/scm_inventory_updates/",
    "notifications": "/api/v2/project_updates/36/notifications/",
    "events": "/api/v2/project_updates/36/events/"
  },
  "summary_fields": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "organization": {
      "id": 1,
      "name": "Default",
      "description": ""
    },
    "project": {
      "id": 15,
      "name": "New Project",
      "description": "",
      "status": "pending",
      "scm_type": "git",
      "allow_override": false
    },
    "unified_job_template": {
      "id": 15,
      "name": "New Project",
      "description": "",
      "unified_job_type": "project_update"
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "user_capabilities": {
      "delete": true,
      "start": true
    }
  },
  "created": "2024-03-29T07:10:02.005597Z",
  "modified": "2024-03-29T07:10:02.013457Z",
  "name": "New Project",
  "description": "",
  "local_path": "_15__new_project",
  "scm_type": "git",
  "scm_url": "https://github.com/ansible/ansible-tower-samples",
  "scm_branch": "",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "",
  "unified_job_template": 15,
  "launch_type": "manual",
  "status": "pending",
  "execution_environment": null,
  "failed": false,
  "started": null,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  "finished": null,
  "canceled_on": null,
  "elapsed": 0.0,
  "job_args": "",
  "job_cwd": "",
  "job_env": {},
  "job_explanation": "",
  "execution_node": "",
  "result_traceback": "",
  "event_processing_finished": false,
  "launched_by": {
    "id": 1,
    "name": "admin",
    "type": "user",
    "url": "/api/v2/users/1/"
  },
  "work_unit_id": null,
  "project": 15,
  "job_type": "check",
  "job_tags": "update_git,install_roles,install_collections"
}
```


Проверка информации о статусе обновления:


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 35,
      "type": "project_update",
      "url": "/api/v2/project_updates/35/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "unified_job_template": "/api/v2/projects/15/",
        "stdout": "/api/v2/project_updates/35/stdout/",
        "execution_environment": "/api/v2/execution_environments/2/",
        "project": "/api/v2/projects/15/",
        "cancel": "/api/v2/project_updates/35/cancel/",
        "scm_inventory_updates": "/api/v2/project_updates/35/scm_inventory_updates/",
        "notifications": "/api/v2/project_updates/35/notifications/",
        "events": "/api/v2/project_updates/35/events/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "execution_environment": {
          "id": 2,
          "name": "Control Plane Execution Environment",
          "description": "",
          "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
        },
        "project": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "id": 15,
        "name": "New Project",
        "description": "",
        "status": "successful",
        "scm_type": "git",
        "allow_override": false
      },
      "unified_job_template": {
        "id": 15,
        "name": "New Project",
        "description": "",
        "unified_job_type": "project_update"
      },
      "instance_group": {
        "id": 1,
        "name": "controlplane",
        "is_container_group": false
      },
      "created_by": {
        "id": 1,
        "username": "admin",
        "first_name": "",
        "last_name": ""
      },
      "user_capabilities": {
        "delete": true,
        "start": true
      }
    },
    "created": "2024-03-29T07:08:16.402906Z",
    "modified": "2024-03-29T07:08:16.813818Z",
    "name": "New Project",
    "description": "",
    "unified_job_template": 15,
    "launch_type": "manual",
    "status": "successful",
    "execution_environment": 2,
    "failed": false,
    "started": "2024-03-29T07:08:16.878716Z",
    "finished": "2024-03-29T07:08:21.153077Z",
    "canceled_on": null,
    "elapsed": 4.274,
    "job_explanation": "",
    "execution_node": "192.168.56.11",
    "launched_by": {
      "id": 1,
      "name": "admin",
      "type": "user",
      "url": "/api/v2/users/1/"
    },
    "work_unit_id": "kROVdZEb",
    "local_path": "_15__new_project",
    "scm_type": "git",
    "scm_url": "https://github.com/ansible/ansible-tower-samples",
    "scm_branch": "",
    "scm_refspec": "",
    "scm_clean": false,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "scm_track_submodules": false,
    "scm_delete_on_update": false,
    "credential": null,
    "timeout": 0,
    "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
    "project": 15,
    "job_type": "check",
    "job_tags": "update_git,install_roles,install_collections"
  },
  {
    "id": 36,
    "type": "project_update",
    "url": "/api/v2/project_updates/36/",
    "related": {
      "created_by": "/api/v2/users/1/",
      "unified_job_template": "/api/v2/projects/15/",
      "stdout": "/api/v2/project_updates/36/stdout/",
      "execution_environment": "/api/v2/execution_environments/2/",
      "project": "/api/v2/projects/15/",
      "cancel": "/api/v2/project_updates/36/cancel/",
      "scm_inventory_updates": "/api/v2/project_updates/36/scm_inventory_updates/",
      "notifications": "/api/v2/project_updates/36/notifications/",
      "events": "/api/v2/project_updates/36/events/"
    },
    "summary_fields": {
      "organization": {
        "id": 1,
        "name": "Default",
        "description": ""
      },
      "execution_environment": {
        "id": 2,
        "name": "Control Plane Execution Environment",
        "description": "",
        "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
      },
      "project": {
        "id": 15,
        "name": "New Project",
        "description": "",
        "status": "successful",
        "scm_type": "git",
        "allow_override": false
      },
      "unified_job_template": {
        "id": 15,
        "name": "New Project",
        "description": "",
        "unified_job_type": "project_update"
      },
      "instance_group": {
        "id": 1,
        "name": "controlplane",
        "is_container_group": false
      },
      "created_by": {
        "id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "user_capabilities": {
          "delete": true,
          "start": true
        }
      },
      "created": "2024-03-29T07:10:02.005597Z",
      "modified": "2024-03-29T07:10:02.188768Z",
      "name": "New Project",
      "description": "",
      "unified_job_template": 15,
      "launch_type": "manual",
      "status": "successful",
      "execution_environment": 2,
      "failed": false,
      "started": "2024-03-29T07:10:02.256356Z",
      "finished": "2024-03-29T07:10:06.368318Z",
      "canceled_on": null,
      "elapsed": 4.112,
      "job_explanation": "",
      "execution_node": "192.168.56.11",
      "launched_by": {
        "id": 1,
        "name": "admin",
        "type": "user",
        "url": "/api/v2/users/1/"
      },
      "work_unit_id": "5bmmFdvg",
      "local_path": "_15__new_project",
      "scm_type": "git",
      "scm_url": "https://github.com/ansible/ansible-tower-samples",
      "scm_branch": "",
      "scm_refspec": "",
      "scm_clean": false,
      "scm_track_submodules": false,
      "scm_delete_on_update": false,
      "credential": null,
      "timeout": 0,
      "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
      "project": 15,
      "job_type": "check",
      "job_tags": "update_git,install_roles,install_collections"
    }
  ]
}
```


Изменение проекта:


```
{
  "id": 15,
  "type": "project",
  "url": "/api/v2/projects/15/",
  "related": {
    "named_url": "/api/v2/projects/Updated Project++Default/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "current_job": "/api/v2/project_updates/37/",
    "last_job": "/api/v2/project_updates/36/",
    "teams": "/api/v2/projects/15/teams/",
    "playbooks": "/api/v2/projects/15/playbooks/",
    "inventory_files": "/api/v2/projects/15/inventories/",
    "update": "/api/v2/projects/15/update/",
    "project_updates": "/api/v2/projects/15/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/15/scm_inventory_sources/",
    "schedules": "/api/v2/projects/15/schedules/",
    "activity_stream": "/api/v2/projects/15/activity_stream/",
    "notification_templates_started": "/api/v2/projects/15/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/15/notification_templates_
↪success/",
    "notification_templates_error": "/api/v2/projects/15/notification_templates_error/
↪",
    "access_list": "/api/v2/projects/15/access_list/",
    "object_roles": "/api/v2/projects/15/object_roles/",
    "copy": "/api/v2/projects/15/copy/",
    "organization": "/api/v2/organizations/1/",
    "current_update": "/api/v2/project_updates/37/",
    "last_update": "/api/v2/project_updates/36/"
  },
  "summary_fields": {
    "organization": {
      "id": 1,
      "name": "Default",
      "description": ""
    },
    "last_job": {
      "id": 36,
      "name": "New Project",
      "description": "",
      "finished": "2024-03-29T07:10:06.368318Z",
      "status": "successful",
      "failed": false
    },
    "last_update": {
      "id": 36,
      "name": "New Project",
      "description": "",
      "status": "successful",
      "failed": false
    },
    "current_update": {
      "id": 37,
      "name": "Updated Project",
      "description": "",
      "status": "pending",
      "failed": false
    },
    "current_job": {
      "id": 37,
      "name": "Updated Project",
      "description": "",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "status": "pending",
      "failed": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the project",
        "name": "Admin",
        "id": 100
      },
      "use_role": {
        "description": "Can use the project in a job template",
        "name": "Use",
        "id": 101
      },
      "update_role": {
        "description": "May update the project",
        "name": "Update",
        "id": 102
      },
      "read_role": {
        "description": "May view settings for the project",
        "name": "Read",
        "id": 103
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "start": true,
      "schedule": true,
      "copy": true
    },
    "resolved_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    }
  },
  "created": "2024-03-29T07:08:16.321624Z",
  "modified": "2024-03-29T07:11:25.337583Z",
  "name": "Updated Project",
  "description": "",
  "local_path": "_15__new_project",
  "scm_type": "git",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  "scm_url": "https://github.com/ansible/ansible-examples",
  "scm_branch": "",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
  "last_job_run": "2024-03-29T07:10:06.368318Z",
  "last_job_failed": false,
  "next_job_run": null,
  "status": "pending",
  "organization": 1,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": null,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": "2024-03-29T07:10:06.368318Z"
}
```


Получение деталей проекта (после обновления):


```
{
  "id": 15,
  "type": "project",
  "url": "/api/v2/projects/15/",
  "related": {
    "named_url": "/api/v2/projects/Updated Project++Default/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "last_job": "/api/v2/project_updates/37/",
    "teams": "/api/v2/projects/15/teams/",
    "playbooks": "/api/v2/projects/15/playbooks/",
    "inventory_files": "/api/v2/projects/15/inventories/",
    "update": "/api/v2/projects/15/update/",
    "project_updates": "/api/v2/projects/15/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/15/scm_inventory_sources/",
    "schedules": "/api/v2/projects/15/schedules/",
    "activity_stream": "/api/v2/projects/15/activity_stream/",
    "notification_templates_started": "/api/v2/projects/15/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/15/notification_templates_
↪success/",
    "notification_templates_error": "/api/v2/projects/15/notification_templates_error/
↪",
    "access_list": "/api/v2/projects/15/access_list/",
    "object_roles": "/api/v2/projects/15/object_roles/",
    "copy": "/api/v2/projects/15/copy/",
    "organization": "/api/v2/organizations/1/",
    "last_update": "/api/v2/project_updates/37/"
  },
  "summary_fields": {
    "organization": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "id": 1,
    "name": "Default",
    "description": ""
  },
  "last_job": {
    "id": 37,
    "name": "Updated Project",
    "description": "",
    "finished": "2024-03-29T07:11:51.297476Z",
    "status": "successful",
    "failed": false
  },
  "last_update": {
    "id": 37,
    "name": "Updated Project",
    "description": "",
    "status": "successful",
    "failed": false
  },
  "created_by": {
    "id": 1,
    "username": "admin",
    "first_name": "",
    "last_name": ""
  },
  "modified_by": {
    "id": 1,
    "username": "admin",
    "first_name": "",
    "last_name": ""
  },
  "object_roles": {
    "admin_role": {
      "description": "Can manage all aspects of the project",
      "name": "Admin",
      "id": 100
    },
    "use_role": {
      "description": "Can use the project in a job template",
      "name": "Use",
      "id": 101
    },
    "update_role": {
      "description": "May update the project",
      "name": "Update",
      "id": 102
    },
    "read_role": {
      "description": "May view settings for the project",
      "name": "Read",
      "id": 103
    }
  },
  "user_capabilities": {
    "edit": true,
    "delete": true,
    "start": true,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "schedule": true,
      "copy": true
    },
    "resolved_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    }
  },
  "created": "2024-03-29T07:08:16.321624Z",
  "modified": "2024-03-29T07:11:25.337583Z",
  "name": "Updated Project",
  "description": "",
  "local_path": "_15__new_project",
  "scm_type": "git",
  "scm_url": "https://github.com/ansible/ansible-examples",
  "scm_branch": "",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": null,
  "timeout": 0,
  "scm_revision": "b50586543c6c4be907fdc88f9f78a2b35d2a895f",
  "last_job_run": "2024-03-29T07:11:51.297476Z",
  "last_job_failed": false,
  "next_job_run": null,
  "status": "successful",
  "organization": 1,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": null,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": "2024-03-29T07:11:51.297476Z"
}
```


Удаление проекта:

```
Status code: 204
```

Получение списка проектов (после удаления):


```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 6,
      "type": "project",
      "url": "/api/v2/projects/6/",
      "related": {
        "created_by": "/api/v2/users/1/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "modified_by": "/api/v2/users/1/",
        "teams": "/api/v2/projects/6/teams/",
        "playbooks": "/api/v2/projects/6/playbooks/",
        "inventory_files": "/api/v2/projects/6/inventories/",
        "update": "/api/v2/projects/6/update/",
        "project_updates": "/api/v2/projects/6/project_updates/",
        "scm_inventory_sources": "/api/v2/projects/6/scm_inventory_sources/",
        "schedules": "/api/v2/projects/6/schedules/",
        "activity_stream": "/api/v2/projects/6/activity_stream/",
        "notification_templates_started": "/api/v2/projects/6/notification_templates_
↪started/",
        "notification_templates_success": "/api/v2/projects/6/notification_templates_
↪success/",
        "notification_templates_error": "/api/v2/projects/6/notification_templates_
↪error/",
        "access_list": "/api/v2/projects/6/access_list/",
        "object_roles": "/api/v2/projects/6/object_roles/",
        "copy": "/api/v2/projects/6/copy/",
        "organization": "/api/v2/organizations/1/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "created_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "modified_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "object_roles": {
          "admin_role": {
            "description": "Can manage all aspects of the project",
            "name": "Admin",
            "id": 22
          },
          "use_role": {
            "description": "Can use the project in a job template",
            "name": "Use",
            "id": 23
          },
          "update_role": {
            "description": "May update the project",
            "name": "Update",
            "id": 24
          },
          "read_role": {
            "description": "May view settings for the project",
            "name": "Read",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "id": 25
          }
        },
        "user_capabilities": {
          "edit": true,
          "delete": true,
          "start": true,
          "schedule": true,
          "copy": true
        }
      },
      "created": "2024-02-16T00:06:02.670809Z",
      "modified": "2024-02-16T00:06:02.670834Z",
      "name": "Demo Project",
      "description": "",
      "local_path": "_6__demo_project",
      "scm_type": "git",
      "scm_url": "https://github.com/ansible/ansible-tower-samples",
      "scm_branch": "",
      "scm_refspec": "",
      "scm_clean": false,
      "scm_track_submodules": false,
      "scm_delete_on_update": false,
      "credential": null,
      "timeout": 0,
      "scm_revision": "347e44fea036c94d5f60e544de006453ee5c71ad",
      "last_job_run": null,
      "last_job_failed": false,
      "next_job_run": null,
      "status": "successful",
      "organization": 1,
      "scm_update_on_launch": false,
      "scm_update_cache_timeout": 0,
      "allow_override": false,
      "custom_virtualenv": null,
      "default_environment": null,
      "signature_validation_credential": null,
      "last_update_failed": false,
      "last_updated": null
    },
    {
      "id": 8,
      "type": "project",
      "url": "/api/v2/projects/8/",
      "related": {
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "credential": "/api/v2/credentials/4/",
        "last_job": "/api/v2/project_updates/16/",
        "teams": "/api/v2/projects/8/teams/",
        "playbooks": "/api/v2/projects/8/playbooks/",
        "inventory_files": "/api/v2/projects/8/inventories/",
        "update": "/api/v2/projects/8/update/",
        "project_updates": "/api/v2/projects/8/project_updates/",
        "scm_inventory_sources": "/api/v2/projects/8/scm_inventory_sources/",
        "schedules": "/api/v2/projects/8/schedules/",
        "activity_stream": "/api/v2/projects/8/activity_stream/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "notification_templates_started": "/api/v2/projects/8/notification_templates_
↪started/",
        "notification_templates_success": "/api/v2/projects/8/notification_templates_
↪success/",
        "notification_templates_error": "/api/v2/projects/8/notification_templates_
↪error/",
        "access_list": "/api/v2/projects/8/access_list/",
        "object_roles": "/api/v2/projects/8/object_roles/",
        "copy": "/api/v2/projects/8/copy/",
        "organization": "/api/v2/organizations/1/",
        "last_update": "/api/v2/project_updates/16/"
      },
      "summary_fields": {
        "organization": {
          "id": 1,
          "name": "Default",
          "description": ""
        },
        "credential": {
          "id": 4,
          "name": "Astra Automation Hub",
          "description": "",
          "kind": "scm",
          "cloud": false,
          "kubernetes": false,
          "credential_type_id": 2
        },
        "last_job": {
          "id": 16,
          "name": "\u0420\u0430\u0437\u0432\u0435\u0440\u0442\u044b\u0432\u0430\u043d\
↪u0438\u0435 NGINX",
          "description": "",
          "finished": "2024-03-15T12:09:44.270752Z",
          "status": "successful",
          "failed": false
        },
        "last_update": {
          "id": 16,
          "name": "\u0420\u0430\u0437\u0432\u0435\u0440\u0442\u044b\u0432\u0430\u043d\
↪u0438\u0435 NGINX",
          "description": "",
          "status": "successful",
          "failed": false
        },
        "created_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "modified_by": {
          "id": 1,
          "username": "admin",
          "first_name": "",
          "last_name": ""
        },
        "object_roles": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "admin_role": {
          "description": "Can manage all aspects of the project",
          "name": "Admin",
          "id": 60
        },
        "use_role": {
          "description": "Can use the project in a job template",
          "name": "Use",
          "id": 61
        },
        "update_role": {
          "description": "May update the project",
          "name": "Update",
          "id": 62
        },
        "read_role": {
          "description": "May view settings for the project",
          "name": "Read",
          "id": 63
        }
      },
      "user_capabilities": {
        "edit": true,
        "delete": true,
        "start": true,
        "schedule": true,
        "copy": true
      }
    },
    "created": "2024-03-15T11:57:40.851309Z",
    "modified": "2024-03-15T11:57:40.851323Z",
    "name": "\u0420\u0430\u0437\u0432\u0435\u0440\u0442\u044b\u0432\u0430\u043d\
↪u0438\u0435 NGINX",
    "description": "",
    "local_path": "_8__nginx",
    "scm_type": "git",
    "scm_url": "ssh://git@hub.astra-automation.ru:2222/aa-gca/AA/aac-samples.git",
    "scm_branch": "",
    "scm_refspec": "",
    "scm_clean": false,
    "scm_track_submodules": false,
    "scm_delete_on_update": false,
    "credential": 4,
    "timeout": 0,
    "scm_revision": "d7f0ff18cf8ffb44d64e8acd995402d545e2de90",
    "last_job_run": "2024-03-15T12:09:44.270752Z",
    "last_job_failed": false,
    "next_job_run": null,
    "status": "successful",
    "organization": 1,
    "scm_update_on_launch": false,
    "scm_update_cache_timeout": 0,
    "allow_override": false,
    "custom_virtualenv": null,
    "default_environment": null,
    "signature_validation_credential": null,
    "last_update_failed": false,
```


(continues on next page)

(продолжение с предыдущей страницы)

```
      "last_updated": "2024-03-15T12:09:44.270752Z"
    }
  ]
}
```

Получение конфигурации Astra Automation Controller:

```
{
  "time_zone": "UTC",
  "license_info": {
    "license_type": "open",
    "valid_key": true,
    "subscription_name": "OPEN",
    "product_name": "AWX"
  },
  "version": "1.0.0+22.1.0",
  "eula": "",
  "analytics_status": "off",
  "analytics_collectors": {
    "config": {
      "name": "config",
      "version": "1.6",
      "description": "General platform configuration."
    },
    "counts": {
      "name": "counts",
      "version": "1.2",
      "description": "Counts of objects such as organizations, inventories, and␣
↪projects"
    },
    "cred_type_counts": {
      "name": "cred_type_counts",
      "version": "1.0",
      "description": "Counts of credentials by credential type"
    },
    "events_table": {
      "name": "events_table",
      "version": "1.5",
      "description": "Automation task records"
    },
    "host_metric_table": {
      "name": "host_metric_table",
      "version": "1.0",
      "description": "Host Metric data, incremental/full sync"
    },
    "instance_info": {
      "name": "instance_info",
      "version": "1.3",
      "description": "Cluster topology and capacity"
    },
    "inventory_counts": {
      "name": "inventory_counts",
      "version": "1.2",
      "description": "Inventories, their inventory sources, and host counts"
    },
    "org_counts": {
      "name": "org_counts",
```

(continues on next page)

(продолжение с предыдущей страницы)


```
      "version": "1.0",
      "description": "Counts of users and teams by organization"
    },
    "projects_by_scm_type": {
      "name": "projects_by_scm_type",
      "version": "1.0",
      "description": "Counts of projects by source control type"
    },
    "query_info": {
      "name": "query_info",
      "version": "1.0",
      "description": "Metadata about the analytics collected"
    },
    "unified_job_template_table": {
      "name": "unified_job_template_table",
      "version": "1.1",
      "description": "Data on job templates"
    },
    "unified_jobs_table": {
      "name": "unified_jobs_table",
      "version": "1.4",
      "description": "Data on jobs run"
    },
    "workflow_job_node_table": {
      "name": "workflow_job_node_table",
      "version": "1.0",
      "description": "Data on workflow runs"
    },
    "workflow_job_template_node_table": {
      "name": "workflow_job_template_node_table",
      "version": "1.0",
      "description": "Data on workflows"
    }
  },
  "become_methods": [
    ["sudo", "Sudo"],
    ["su", "Su"],
    ["pbrun", "Pbrun"],
    ["pfexec", "Pfexec"],
    ["dzdo", "DZDO"],
    ["pmrun", "Pmrun"],
    ["runas", "Runas"],
    ["enable", "Enable"],
    ["doas", "Doas"],
    ["ksu", "Ksu"],
    ["machinectl", "Machinectl"],
    ["sesu", "Sesu"]
  ],
  "ui_next": false,
  "project_base_dir": "/var/lib/awx/projects",
  "project_local_paths": [],
  "custom_virtualenvs": []
}
```

6.4.4 Базовый сценарий

В этом документе приведен базовый сценарий использования API Astra Automation Controller. В ходе выполнения сценария производятся следующие действия:

1. Получение токена авторизации.
2. Создание организации.
3. Добавление полномочий Ansible Galaxy для созданной организации.
4. Создание инвентаря.
5. Добавление управляемого узла в инвентарь.
6. Создание проекта.
7. Создание шаблона задания для установки NGINX.
8. Запуск шаблона задания.
9. Проверка статуса выполнения задания.

Предварительные требования

Перед выполнением сценария произведите следующие действия в графическом интерфейсе контроллера:

1. Создайте полномочия для доступа к следующим ресурсам:
   - управляемый узел;
   - Ansible Galaxy;
   - Astra Automation Hub.

   Примечание: С подробной информацией о работе с полномочиями можно ознакомиться в документе *Полномочия*.

   ---

2. Получите идентификаторы созданных полномочий:
   a. На панели навигации выберите *Ресурсы* ▸ *Полномочия* (*Resources* ▸ *Credentials*).
   b. Нажмите на ссылку с названием нужного полномочия, например, Ansible Galaxy.
   c. Скопируйте <credentials_id> в адресной строке браузера, которая имеет следующий вид `https://<address>/#/credentials/<credentials_id>/details`.

Подготовка организационной структуры

Для подготовки организационной структуры выполните следующие действия:

1. Получите токен авторизации:

```
curl -X POST \
   https://<address>/api/v2/tokens/ \
   -H 'Content-Type: application/json' \
   -d '{
            "description": "Token description",
            "application": null,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            "user": "<username>",
            "scope": "write"
        }' \
    -u '<username>:<password>' | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере:

- <address> – FQDN или IP-адрес контроллера, например, `192.168.56.11`;
- <username> – имя пользователя, используемое для аутентификации в контроллере, например, `admin`;
- <password> – пароль, например, `awx`.

Эта команда возвращает JSON следующего вида:

```
{
   "id": 13,
   "type": "o_auth2_access_token",
   "url": "/api/v2/tokens/13/",
   "related": {
      "user": "/api/v2/users/1/",
      "activity_stream": "/api/v2/tokens/13/activity_stream/"
   },
   "summary_fields": {
      "user": {
         "id": 1,
         "username": "admin",
         "first_name": "",
         "last_name": ""
      }
   },
   "created": "2024-03-21T08:38:43.412155Z",
   "modified": "2024-03-21T08:38:43.418814Z",
   "description": "Token description",
   "user": 1,
   "token": "<your_token>",
   "refresh_token": null,
   "application": null,
   "expires": "3023-07-23T08:38:43.406429Z",
   "scope": "write"
}
```

Здесь `token` – значение созданного токена. Сохраните его в надежном месте, оно понадобится позже.

2. Получите идентификатор среды исполнения в графическом интерфейсе контроллера:

   a. На панели навигации выберите *Администрирование* ▸ *Среды исполнения* (*Administration* ▸ *Execution Environments*).

   b. Нажмите на ссылку с названием нужной среды исполнения, например, Default execution environment.

   c. Скопируйте <execution_environment_id> в адресной строке браузера, которая имеет следующий вид `https://<address>/#/execution_environments/<execution_environments_id>/details`.

3. Создайте организацию:

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <your_token>' \
   -d '{
            "name": "<your_organization_name>",
            "default_environment": "<execution_environment_id>"
       }' \
   https://<address>/api/v2/organizations/ | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере:

- <your_token> – токен, полученный на предыдущем шаге, например, 3wijiG7********VtL2gseJESzjJsD;
- <your_organization_name> – название организации, например, Jupiter;
- <execution_environment_id> – идентификатор среды исполнения, например, 1.

Organization


```
{
  "id": 4,
  "type": "organization",
  "url": "/api/v2/organizations/4/",
  "related": {
    "named_url": "/api/v2/organizations/Jupiter/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "execution_environments": "/api/v2/organizations/4/execution_environments/",
    "projects": "/api/v2/organizations/4/projects/",
    "inventories": "/api/v2/organizations/4/inventories/",
    "job_templates": "/api/v2/organizations/4/job_templates/",
    "workflow_job_templates": "/api/v2/organizations/4/workflow_job_templates/",
    "users": "/api/v2/organizations/4/users/",
    "admins": "/api/v2/organizations/4/admins/",
    "teams": "/api/v2/organizations/4/teams/",
    "credentials": "/api/v2/organizations/4/credentials/",
    "applications": "/api/v2/organizations/4/applications/",
    "activity_stream": "/api/v2/organizations/4/activity_stream/",
    "notification_templates": "/api/v2/organizations/4/notification_templates/",
    "notification_templates_started": "/api/v2/organizations/4/notification_
↪templates_started/",
    "notification_templates_success": "/api/v2/organizations/4/notification_
↪templates_success/",
    "notification_templates_error": "/api/v2/organizations/4/notification_
↪templates_error/",
    "notification_templates_approvals": "/api/v2/organizations/4/notification_
↪templates_approvals/",
    "object_roles": "/api/v2/organizations/4/object_roles/",
    "access_list": "/api/v2/organizations/4/access_list/",
    "instance_groups": "/api/v2/organizations/4/instance_groups/",
    "galaxy_credentials": "/api/v2/organizations/4/galaxy_credentials/",
    "default_environment": "/api/v2/execution_environments/1/"
  },
  "summary_fields": {
    "default_environment": {
      "id": 1,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the organization",
        "name": "Admin",
        "id": 103,
        "user_only": true
      },
      "execute_role": {
        "description": "May run any executable resources in the organization",
        "name": "Execute",
        "id": 104
      },
      "project_admin_role": {
        "description": "Can manage all projects of the organization",
        "name": "Project Admin",
        "id": 105
      },
      "inventory_admin_role": {
        "description": "Can manage all inventories of the organization",
        "name": "Inventory Admin",
        "id": 106
      },
      "credential_admin_role": {
        "description": "Can manage all credentials of the organization",
        "name": "Credential Admin",
        "id": 107
      },
      "workflow_admin_role": {
        "description": "Can manage all workflows of the organization",
        "name": "Workflow Admin",
        "id": 108
      },
      "notification_admin_role": {
        "description": "Can manage all notifications of the organization",
        "name": "Notification Admin",
        "id": 109
      },
      "job_template_admin_role": {
        "description": "Can manage all job templates of the organization",
        "name": "Job Template Admin",
        "id": 110
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      },
      "execution_environment_admin_role": {
        "description": "Can manage all execution environments of the organization
↪",
        "name": "Execution Environment Admin",
        "id": 111
      },
      "auditor_role": {
        "description": "Can view all aspects of the organization",
        "name": "Auditor",
        "id": 112
      },
      "member_role": {
        "description": "User is a member of the organization",
        "name": "Member",
        "id": 113,
        "user_only": true
      },
      "read_role": {
        "description": "May view settings for the organization",
        "name": "Read",
        "id": 114
      },
      "approval_role": {
        "description": "Can approve or deny a workflow approval node",
        "name": "Approve",
        "id": 115
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true
    },
    "related_field_counts": {
      "inventories": 0,
      "teams": 0,
      "users": 0,
      "job_templates": 0,
      "admins": 0,
      "projects": 0
    }
  },
  "created": "2024-03-21T08:48:05.919683Z",
  "modified": "2024-03-21T08:48:05.919698Z",
  "name": "Jupiter",
  "max_hosts": 0,
  "custom_virtualenv": null,
  "default_environment": 1
}
```


4. Добавьте полномочия Ansible Galaxy для созданной организации:

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <your_token>' \
   -d '{
          "id":<credentials_id>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      }' \
  https://<address>/api/v2/organizations/<organization_id>/galaxy_credentials/ |␣
↪jq .
```

Подставьте собственные значения параметров вместо употребленных в примере:

- <credentials_id> – идентификатор полномочий для доступа к Ansible Galaxy, например, 2;
- <organization_id> – идентификатор организации, созданной ранее, например, 4.

Подготовка инвентаря

Для подготовки инвентаря выполните следующие действия:

1. Создайте инвентарь:

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <address>' \
   -d '{
            "name": "<your_inventory_name>",
            "organization": <organization_id>
       }' \
   https://<address>/api/v2/inventories/ | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере. Здесь <your_inventory_name> – название инвентаря, например, `Demo Inventory`.

Inventory


```
{
  "id": 4,
  "type": "inventory",
  "url": "/api/v2/inventories/4/",
  "related": {
    "named_url": "/api/v2/inventories/Demo Inventory++Jupiter/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "hosts": "/api/v2/inventories/4/hosts/",
    "variable_data": "/api/v2/inventories/4/variable_data/",
    "script": "/api/v2/inventories/4/script/",
    "activity_stream": "/api/v2/inventories/4/activity_stream/",
    "job_templates": "/api/v2/inventories/4/job_templates/",
    "ad_hoc_commands": "/api/v2/inventories/4/ad_hoc_commands/",
    "access_list": "/api/v2/inventories/4/access_list/",
    "object_roles": "/api/v2/inventories/4/object_roles/",
    "instance_groups": "/api/v2/inventories/4/instance_groups/",
    "copy": "/api/v2/inventories/4/copy/",
    "labels": "/api/v2/inventories/4/labels/",
    "groups": "/api/v2/inventories/4/groups/",
    "root_groups": "/api/v2/inventories/4/root_groups/",
    "update_inventory_sources": "/api/v2/inventories/4/update_inventory_sources/",
    "inventory_sources": "/api/v2/inventories/4/inventory_sources/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "tree": "/api/v2/inventories/4/tree/",
    "organization": "/api/v2/organizations/4/"
  },
  "summary_fields": {
    "organization": {
      "id": 4,
      "name": "Jupiter"
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the inventory",
        "name": "Admin",
        "id": 116
      },
      "update_role": {
        "description": "May update the inventory",
        "name": "Update",
        "id": 117
      },
      "adhoc_role": {
        "description": "May run ad hoc commands on the inventory",
        "name": "Ad Hoc",
        "id": 118
      },
      "use_role": {
        "description": "Can use the inventory in a job template",
        "name": "Use",
        "id": 119
      },
      "read_role": {
        "description": "May view settings for the inventory",
        "name": "Read",
        "id": 120
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "copy": true,
      "adhoc": true
    },
    "labels": {
      "count": 0,
      "results": []
    }
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  },
  "created": "2024-03-21T08:53:27.613062Z",
  "modified": "2024-03-21T08:53:27.613074Z",
  "name": "Jupiter",
  "organization": 4,
  "kind": "",
  "host_filter": null,
  "variables": "",
  "has_active_failures": false,
  "total_hosts": 0,
  "hosts_with_active_failures": 0,
  "total_groups": 0,
  "has_inventory_sources": false,
  "total_inventory_sources": 0,
  "inventory_sources_with_failures": 0,
  "pending_deletion": false,
  "prevent_instance_group_fallback": false
}
```


2. Добавьте управляемый узел в инвентарь:

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <your_token>' \
   -d '{
            "name": "<node_ip>",
            "description": "<node_name>",
            "inventory": <inventory_id>,
            "enabled": true
       }' \
   https://<address>/api/v2/hosts/ | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере:

- <node_ip> – IP-адрес управляемого узла например, `10.2.0.6`;
- <node_name> – название управляемого узла, например, `node01`;
- <inventory_id> – идентификатор инвентаря, например, 4.

Host


```
{
  "id": 7,
  "type": "host",
  "url": "/api/v2/hosts/7/",
  "related": {
    "named_url": "/api/v2/hosts/10.2.0.6++Demo Inventory++Jupiter/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "variable_data": "/api/v2/hosts/7/variable_data/",
    "groups": "/api/v2/hosts/7/groups/",
    "all_groups": "/api/v2/hosts/7/all_groups/",
    "job_events": "/api/v2/hosts/7/job_events/",
    "job_host_summaries": "/api/v2/hosts/7/job_host_summaries/",
    "activity_stream": "/api/v2/hosts/7/activity_stream/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "inventory_sources": "/api/v2/hosts/7/inventory_sources/",
    "smart_inventories": "/api/v2/hosts/7/smart_inventories/",
    "ad_hoc_commands": "/api/v2/hosts/7/ad_hoc_commands/",
    "ad_hoc_command_events": "/api/v2/hosts/7/ad_hoc_command_events/",
    "ansible_facts": "/api/v2/hosts/7/ansible_facts/",
    "inventory": "/api/v2/inventories/4/"
  },
  "summary_fields": {
    "inventory": {
      "id": 4,
      "name": "Demo Inventory",
      "has_active_failures": false,
      "total_hosts": 0,
      "hosts_with_active_failures": 0,
      "total_groups": 0,
      "has_inventory_sources": false,
      "total_inventory_sources": 0,
      "inventory_sources_with_failures": 0,
      "organization_id": 4,
      "kind": ""
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "user_capabilities": {
      "edit": true,
      "delete": true
    },
    "groups": {
      "count": 0,
      "results": []
    },
    "recent_jobs": []
  },
  "created": "2024-03-21T10:03:33.967091Z",
  "modified": "2024-03-21T10:03:33.967104Z",
  "name": "10.2.0.6",
  "description": "node01",
  "inventory": 4,
  "enabled": true,
  "instance_id": "",
  "variables": "",
  "has_active_failures": false,
  "has_inventory_sources": false,
  "last_job": null,
  "last_job_host_summary": null,
  "ansible_facts_modified": null
}
```

Подготовка задания

Для подготовки задания выполните следующие действия:

1. Создайте проект:

Примечание: В качестве источника при создании проекта используется репозиторий с примерами с Astra Automation Hub.

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <your_token>' \
   -d '{
            "name": "<your_project_name>",
            "scm_type": "git",
            "scm_url": "ssh://git@hub.astra-automation.ru:2222/aa-gca/AA/aac-
↪samples.git",
            "scm_branch": "master",
            "default_environment": <environment_id>,
            "credential": <credential_id>,
            "organization": <organization_id>
       }' \
   https://<address>/api/v2/projects/ | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере:

- <your_project_name> – название проекта, например, `Demo NGINX Project`;
- <credentials_id> – идентификатор полномочий для доступа к Astra Automation Hub, например, 3.

Project


```
{
  "id": 14,
  "type": "project",
  "url": "/api/v2/projects/14/",
  "related": {
    "named_url": "/api/v2/projects/Demo NGINX Project++Jupiter/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "credential": "/api/v2/credentials/3/",
    "current_job": "/api/v2/project_updates/9/",
    "teams": "/api/v2/projects/14/teams/",
    "playbooks": "/api/v2/projects/14/playbooks/",
    "inventory_files": "/api/v2/projects/14/inventories/",
    "update": "/api/v2/projects/14/update/",
    "project_updates": "/api/v2/projects/14/project_updates/",
    "scm_inventory_sources": "/api/v2/projects/14/scm_inventory_sources/",
    "schedules": "/api/v2/projects/14/schedules/",
    "activity_stream": "/api/v2/projects/14/activity_stream/",
    "notification_templates_started": "/api/v2/projects/14/notification_templates_
↪started/",
    "notification_templates_success": "/api/v2/projects/14/notification_templates_
↪success/",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "notification_templates_error": "/api/v2/projects/14/notification_templates_
↪error/",
    "access_list": "/api/v2/projects/14/access_list/",
    "object_roles": "/api/v2/projects/14/object_roles/",
    "copy": "/api/v2/projects/14/copy/",
    "organization": "/api/v2/organizations/4/",
    "default_environment": "/api/v2/execution_environments/1/",
    "current_update": "/api/v2/project_updates/9/"
  },
  "summary_fields": {
    "organization": {
      "id": 4,
      "name": "Jupiter"
    },
    "default_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    },
    "credential": {
      "id": 3,
      "name": "Credentials for AA Hub",
      "description": "",
      "kind": "scm",
      "cloud": false,
      "kubernetes": false,
      "credential_type_id": 2
    },
    "current_update": {
      "id": 9,
      "name": "Demo NGINX Project",
      "status": "pending",
      "failed": false
    },
    "current_job": {
      "id": 9,
      "name": "Demo NGINX Project",
      "status": "pending",
      "failed": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the project",
        "name": "Admin",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "id": 134
      },
      "use_role": {
        "description": "Can use the project in a job template",
        "name": "Use",
        "id": 135
      },
      "update_role": {
        "description": "May update the project",
        "name": "Update",
        "id": 136
      },
      "read_role": {
        "description": "May view settings for the project",
        "name": "Read",
        "id": 137
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "start": true,
      "schedule": true,
      "copy": true
    }
  },
  "created": "2024-03-21T09:44:26.824017Z",
  "modified": "2024-03-21T09:44:26.824031Z",
  "name": "Demo NGINX Project",
  "local_path": "_14__demo_nginx_project",
  "scm_type": "git",
  "scm_url": "ssh://git@hub.astra-automation.ru:2222/aa-gca/AA/aac-samples.git",
  "scm_branch": "master",
  "scm_refspec": "",
  "scm_clean": false,
  "scm_track_submodules": false,
  "scm_delete_on_update": false,
  "credential": 3,
  "timeout": 0,
  "scm_revision": "",
  "last_job_run": null,
  "last_job_failed": false,
  "next_job_run": null,
  "status": "pending",
  "organization": 4,
  "scm_update_on_launch": false,
  "scm_update_cache_timeout": 0,
  "allow_override": false,
  "custom_virtualenv": null,
  "default_environment": 1,
  "signature_validation_credential": null,
  "last_update_failed": false,
  "last_updated": null
}
```


2. Создайте шаблон задания для установки NGINX:

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <your_token>' \
   -d '{
            "name": "NGINX Deployment",
            "job_type": "run",
            "inventory": <inventory_id>,
            "project": <project_id>,
            "playbook": "nginx/site.yml"
       }' \
   https://<address>/api/v2/job_templates/ | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере. Здесь <project_id> – идентификатор проекта, созданного ранее, например, 14.

Job template


```
{
  "id": 15,
  "type": "job_template",
  "url": "/api/v2/job_templates/15/",
  "related": {
    "named_url": "/api/v2/job_templates/NGINX Deployment++Jupiter/",
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "labels": "/api/v2/job_templates/15/labels/",
    "inventory": "/api/v2/inventories/4/",
    "project": "/api/v2/projects/14/",
    "organization": "/api/v2/organizations/4/",
    "credentials": "/api/v2/job_templates/15/credentials/",
    "jobs": "/api/v2/job_templates/15/jobs/",
    "schedules": "/api/v2/job_templates/15/schedules/",
    "activity_stream": "/api/v2/job_templates/15/activity_stream/",
    "launch": "/api/v2/job_templates/15/launch/",
    "webhook_key": "/api/v2/job_templates/15/webhook_key/",
    "webhook_receiver": "",
    "notification_templates_started": "/api/v2/job_templates/15/notification_
↪templates_started/",
    "notification_templates_success": "/api/v2/job_templates/15/notification_
↪templates_success/",
    "notification_templates_error": "/api/v2/job_templates/15/notification_
↪templates_error/",
    "access_list": "/api/v2/job_templates/15/access_list/",
    "survey_spec": "/api/v2/job_templates/15/survey_spec/",
    "object_roles": "/api/v2/job_templates/15/object_roles/",
    "instance_groups": "/api/v2/job_templates/15/instance_groups/",
    "slice_workflow_jobs": "/api/v2/job_templates/15/slice_workflow_jobs/",
    "copy": "/api/v2/job_templates/15/copy/"
  },
  "summary_fields": {
    "organization": {
      "id": 4,
      "name": "Jupiter"
    },
    "inventory": {
      "id": 4,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "name": "Demo Inventory",
      "has_active_failures": false,
      "total_hosts": 0,
      "hosts_with_active_failures": 0,
      "total_groups": 0,
      "has_inventory_sources": false,
      "total_inventory_sources": 0,
      "inventory_sources_with_failures": 0,
      "organization_id": 4,
      "kind": ""
    },
    "project": {
      "id": 14,
      "name": "Demo NGINX Project",
      "status": "successful",
      "scm_type": "git",
      "allow_override": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "object_roles": {
      "admin_role": {
        "description": "Can manage all aspects of the job template",
        "name": "Admin",
        "id": 138
      },
      "execute_role": {
        "description": "May run the job template",
        "name": "Execute",
        "id": 139
      },
      "read_role": {
        "description": "May view settings for the job template",
        "name": "Read",
        "id": 140
      }
    },
    "user_capabilities": {
      "edit": true,
      "delete": true,
      "start": true,
      "schedule": true,
      "copy": true
    },
    "labels": {
      "count": 0,
      "results": []
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    },
    "recent_jobs": [],
    "credentials": []
  },
  "created": "2024-03-21T09:52:41.673531Z",
  "modified": "2024-03-21T09:52:41.673545Z",
  "name": "NGINX Deployment",
  "description": "",
  "job_type": "run",
  "inventory": 4,
  "project": 14,
  "playbook": "nginx/site.yml",
  "scm_branch": "",
  "forks": 0,
  "limit": "",
  "verbosity": 0,
  "extra_vars": "",
  "job_tags": "",
  "force_handlers": false,
  "skip_tags": "",
  "start_at_task": "",
  "timeout": 0,
  "use_fact_cache": false,
  "organization": 4,
  "last_job_run": null,
  "last_job_failed": false,
  "next_job_run": null,
  "status": "never updated",
  "execution_environment": null,
  "host_config_key": "",
  "ask_scm_branch_on_launch": false,
  "ask_diff_mode_on_launch": false,
  "ask_variables_on_launch": false,
  "ask_limit_on_launch": false,
  "ask_tags_on_launch": false,
  "ask_skip_tags_on_launch": false,
  "ask_job_type_on_launch": false,
  "ask_verbosity_on_launch": false,
  "ask_inventory_on_launch": false,
  "ask_credential_on_launch": false,
  "ask_execution_environment_on_launch": false,
  "ask_labels_on_launch": false,
  "ask_forks_on_launch": false,
  "ask_job_slice_count_on_launch": false,
  "ask_timeout_on_launch": false,
  "ask_instance_groups_on_launch": false,
  "survey_enabled": false,
  "become_enabled": false,
  "diff_mode": false,
  "allow_simultaneous": false,
  "custom_virtualenv": null,
  "job_slice_count": 1,
  "webhook_service": "",
  "webhook_credential": null,
  "prevent_instance_group_fallback": false
}
```

3. Добавьте полномочия для шаблона задания:

```
curl -X POST \
   -H 'Content-Type: application/json' \
   -H 'Authorization: Bearer <your_token>' \
   -d '{
            "id":<credential_id>
       }' \
   https://<address>/api/v2/job_templates/<job_template_id>/credentials/ | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере:

- <credentials_id> – идентификатор полномочий для доступа к управляемому узлу, например, 1;
- <job_template_id> – идентификатор созданного ранее шаблона задания, например, 15.

Выполнение и проверка задания

Для запуска и проверки статуса задания выполните следующие действия:

1. Запустите задание, используя созданный шаблон:

```
curl -X POST \
   -H 'Authorization: Bearer <your_token>' \
   https://<address>/api/v2/job_templates/<job_template_id>/launch/ | jq .
```

Job template


```
{
  "job": 10,
  "ignored_fields": {},
  "id": 10,
  "type": "job",
  "url": "/api/v2/jobs/10/",
  "related": {
    "created_by": "/api/v2/users/1/",
    "modified_by": "/api/v2/users/1/",
    "labels": "/api/v2/jobs/10/labels/",
    "inventory": "/api/v2/inventories/4/",
    "project": "/api/v2/projects/14/",
    "organization": "/api/v2/organizations/4/",
    "credentials": "/api/v2/jobs/10/credentials/",
    "unified_job_template": "/api/v2/job_templates/15/",
    "stdout": "/api/v2/jobs/10/stdout/",
    "job_events": "/api/v2/jobs/10/job_events/",
    "job_host_summaries": "/api/v2/jobs/10/job_host_summaries/",
    "activity_stream": "/api/v2/jobs/10/activity_stream/",
    "notifications": "/api/v2/jobs/10/notifications/",
    "create_schedule": "/api/v2/jobs/10/create_schedule/",
    "job_template": "/api/v2/job_templates/15/",
    "cancel": "/api/v2/jobs/10/cancel/",
    "relaunch": "/api/v2/jobs/10/relaunch/"
  },
  "summary_fields": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "organization": {
      "id": 4,
      "name": "Jupiter"
    },
    "inventory": {
      "id": 4,
      "name": "Demo Inventory",
      "has_active_failures": false,
      "total_hosts": 0,
      "hosts_with_active_failures": 0,
      "total_groups": 0,
      "has_inventory_sources": false,
      "total_inventory_sources": 0,
      "inventory_sources_with_failures": 0,
      "organization_id": 4,
      "kind": ""
    },
    "project": {
      "id": 14,
      "name": "Demo NGINX Project",
      "status": "successful",
      "scm_type": "git",
      "allow_override": false
    },
    "job_template": {
      "id": 15,
      "name": "NGINX Deployment",
      "description": ""
    },
    "unified_job_template": {
      "id": 15,
      "name": "NGINX Deployment",
      "description": "",
      "unified_job_type": "job"
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "modified_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "user_capabilities": {
      "delete": true,
      "start": true
    },
    "labels": {
      "count": 0,
      "results": []
    },
    "credentials": [
      {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "id": 4,
      "name": "NGINX Machine Credential",
      "description": "",
      "kind": "ssh",
      "cloud": false
    }
  ]
},
"created": "2024-03-21T10:02:14.206401Z",
"modified": "2024-03-21T10:02:14.230669Z",
"name": "NGINX Deployment",
"description": "",
"job_type": "run",
"inventory": 4,
"project": 14,
"playbook": "nginx/site.yml",
"scm_branch": "",
"forks": 0,
"limit": "",
"verbosity": 0,
"extra_vars": "{}",
"job_tags": "",
"force_handlers": false,
"skip_tags": "",
"start_at_task": "",
"timeout": 0,
"use_fact_cache": false,
"organization": 4,
"unified_job_template": 15,
"launch_type": "manual",
"status": "pending",
"execution_environment": null,
"failed": false,
"started": null,
"finished": null,
"canceled_on": null,
"elapsed": 0,
"job_args": "",
"job_cwd": "",
"job_env": {},
"job_explanation": "",
"execution_node": "",
"controller_node": "",
"result_traceback": "",
"event_processing_finished": false,
"launched_by": {
  "id": 1,
  "name": "admin",
  "type": "user",
  "url": "/api/v2/users/1/"
},
"work_unit_id": null,
"job_template": 15,
"passwords_needed_to_start": [],
"allow_simultaneous": false,
"artifacts": {},
"scm_revision": "",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
  "instance_group": null,
  "diff_mode": false,
  "job_slice_number": 0,
  "job_slice_count": 1,
  "webhook_service": "",
  "webhook_credential": null,
  "webhook_guid": ""
}
```


2. Проверьте статус выполнения задания:

```
curl -X GET \
   https://<address>/api/v2/jobs/<id_job>/ \
   -H 'Authorization: Bearer <your_token>' \
   -k -s | jq .
```

Подставьте собственные значения параметров вместо употребленных в примере. Здесь <id_job> – идентификатор задания, запущенного ранее, например, 11.

Job


```
{
  "id": 11,
  "type": "job",
  "url": "/api/v2/jobs/11/",
  "related": {
    "created_by": "/api/v2/users/1/",
    "labels": "/api/v2/jobs/11/labels/",
    "inventory": "/api/v2/inventories/4/",
    "project": "/api/v2/projects/14/",
    "organization": "/api/v2/organizations/4/",
    "credentials": "/api/v2/jobs/11/credentials/",
    "unified_job_template": "/api/v2/job_templates/15/",
    "stdout": "/api/v2/jobs/11/stdout/",
    "execution_environment": "/api/v2/execution_environments/1/",
    "job_events": "/api/v2/jobs/11/job_events/",
    "job_host_summaries": "/api/v2/jobs/11/job_host_summaries/",
    "activity_stream": "/api/v2/jobs/11/activity_stream/",
    "notifications": "/api/v2/jobs/11/notifications/",
    "create_schedule": "/api/v2/jobs/11/create_schedule/",
    "job_template": "/api/v2/job_templates/15/",
    "cancel": "/api/v2/jobs/11/cancel/",
    "relaunch": "/api/v2/jobs/11/relaunch/"
  },
  "summary_fields": {
    "organization": {
      "id": 4,
      "name": "Jupiter"
    },
    "inventory": {
      "id": 4,
      "name": "Demo Inventory",
      "has_active_failures": false,
      "total_hosts": 1,
      "hosts_with_active_failures": 0,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      "total_groups": 0,
      "has_inventory_sources": false,
      "total_inventory_sources": 0,
      "inventory_sources_with_failures": 0,
      "organization_id": 4,
      "kind": ""
    },
    "execution_environment": {
      "id": 1,
      "name": "Default execution environment",
      "description": "",
      "image": "registry.astralinux.ru/aa/aa-base-ee:0.2.1"
    },
    "project": {
      "id": 14,
      "name": "Demo NGINX Project",
      "status": "successful",
      "scm_type": "git",
      "allow_override": false
    },
    "job_template": {
      "id": 15,
      "name": "NGINX Deployment",
      "description": ""
    },
    "unified_job_template": {
      "id": 15,
      "name": "NGINX Deployment",
      "description": "",
      "unified_job_type": "job"
    },
    "instance_group": {
      "id": 2,
      "name": "default",
      "is_container_group": false
    },
    "created_by": {
      "id": 1,
      "username": "admin",
      "first_name": "",
      "last_name": ""
    },
    "user_capabilities": {
      "delete": true,
      "start": true
    },
    "labels": {
      "count": 0,
      "results": []
    },
    "credentials": [
      {
        "id": 4,
        "name": "NGINX Machine Credentials",
        "description": "",
        "kind": "ssh",
        "cloud": false
```


(continues on next page)

(продолжение с предыдущей страницы)


```
      }
    ]
  },
  "created": "2024-03-21T10:03:39.307771Z",
  "modified": "2024-03-21T10:03:39.534635Z",
  "name": "NGINX Deployment",
  "description": "",
  "job_type": "run",
  "inventory": 4,
  "project": 14,
  "playbook": "nginx/site.yml",
  "scm_branch": "",
  "forks": 0,
  "limit": "",
  "verbosity": 0,
  "extra_vars": "{}",
  "job_tags": "",
  "force_handlers": false,
  "skip_tags": "",
  "start_at_task": "",
  "timeout": 0,
  "use_fact_cache": false,
  "organization": 4,
  "unified_job_template": 15,
  "launch_type": "manual",
  "status": "successful",
  "execution_environment": 1,
  "failed": false,
  "started": "2024-03-21T10:03:39.605438Z",
  "finished": "2024-03-21T10:03:49.709550Z",
  "canceled_on": null,
  "elapsed": 10.104,
  "job_args": "[\"podman\", \"run\", \"--rm\", \"--tty\", \"--interactive\", \"--
↪workdir\", \"/runner/project\", \"-v\", \"/tmp/awx_11_9_5zjzyv/:/runner/:Z\", \
↪"--env-file\", \"/tmp/awx_11_9_5zjzyv/artifacts/11/env.list\", \"--quiet\", \"--
↪name\", \"ansible_runner_11\", \"--user=root\", \"--network\", \
↪"slirp4netns:enable_ipv6=true\", \"registry.astralinux.ru/aa/aa-base-ee:0.2.1\",
↪ \"ssh-agent\", \"sh\", \"-c\", \"trap 'rm -f /runner/artifacts/11/ssh_key_data
↪' EXIT && ssh-add /runner/artifacts/11/ssh_key_data && rm -f /runner/artifacts/
↪11/ssh_key_data && ansible-playbook -u astra -i /runner/inventory/hosts -e @/
↪runner/env/extravars nginx/site.yml\"]",
  "job_cwd": "/runner/project",
  "job_env": {
    "ANSIBLE_UNSAFE_WRITES": "1",
    "AWX_ISOLATED_DATA_DIR": "/runner/artifacts/11",
    "ANSIBLE_FORCE_COLOR": "True",
    "ANSIBLE_HOST_KEY_CHECKING": "False",
    "ANSIBLE_INVENTORY_UNPARSED_FAILED": "True",
    "ANSIBLE_PARAMIKO_RECORD_HOST_KEYS": "False",
    "HOME": "/var/lib/awx",
    "AWX_PRIVATE_DATA_DIR": "/tmp/awx_11_9_5zjzyv",
    "JOB_ID": "11",
    "INVENTORY_ID": "4",
    "PROJECT_REVISION": "d7f0ff18cf8ffb44d64e8acd995402d545e2de90",
    "ANSIBLE_RETRY_FILES_ENABLED": "False",
    "MAX_EVENT_RES": "700000",
    "AWX_HOST": "https://controller",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
    "ANSIBLE_SSH_CONTROL_PATH_DIR": "/runner/cp",
    "ANSIBLE_COLLECTIONS_PATHS": "/runner/requirements_collections:~/.ansible/
↪collections:/usr/share/ansible/collections",
    "ANSIBLE_ROLES_PATH": "/runner/requirements_roles:~/.ansible/roles:/usr/share/
↪ansible/roles:/etc/ansible/roles",
    "ANSIBLE_CALLBACK_PLUGINS": "/runner/artifacts/11/callback",
    "ANSIBLE_STDOUT_CALLBACK": "awx_display",
    "RUNNER_OMIT_EVENTS": "False",
    "RUNNER_ONLY_FAILED_EVENTS": "False"
  },
  "job_explanation": "",
  "execution_node": "10.2.0.5",
  "controller_node": "10.2.0.5",
  "result_traceback": "",
  "event_processing_finished": true,
  "launched_by": {
    "id": 1,
    "name": "admin",
    "type": "user",
    "url": "/api/v2/users/1/"
  },
  "work_unit_id": "EQMxvG0G",
  "job_template": 15,
  "passwords_needed_to_start": [],
  "allow_simultaneous": false,
  "artifacts": {},
  "scm_revision": "d7f0ff18cf8ffb44d64e8acd995402d545e2de90",
  "instance_group": 2,
  "diff_mode": false,
  "job_slice_number": 0,
  "job_slice_count": 1,
  "webhook_service": "",
  "webhook_credential": null,
  "webhook_guid": "",
  "host_status_counts": {
    "ok": 1
  },
  "playbook_counts": {
    "play_count": 1,
    "task_count": 16
  },
  "custom_virtualenv": null
}
```


Если в поле `failed` указано значение `false`, значит задание завершилось успешно.

Примечание: Подробное формальное описание HTTP API представлено в спецификации.

6.5 Интерфейс командной строки (CLI)

Важно: Эта часть документации находится в стадии разработки.

6.6 Настройка контроллера через Ansible

Важно: Эта часть документации находится в стадии разработки.

6.7 Среда исполнения

Astra Automation Controller использует *среды исполнения* для запуска заданий на исполняющих и гибридных узлах.

При развертывании контроллера в нем автоматически создаются две среды исполнения:

- Control Plane Execution Environment:
  - Используется при развертывании и обновлении платформы.
  - Используется для выполнения служебных заданий Control Plane.
  - Образ среды невозможно изменить через пользовательский интерфейс.
- Default execution environment:
  - Используется по умолчанию для выполнения заданий из шаблонов.
  - Его образ можно заменить на другой.

Для создания дополнительных сред исполнения можно использовать как образы из реестра Astra Automation[164] (рекомендуется), так и собранные самостоятельно.

Поддерживается использование *полномочий* для доступа к реестрам образов, требующим аутентификацию пользователей.

Вместо среды по умолчанию можно назначить другую среду на следующих уровнях (в порядке возрастания приоритета использования):

1. контроллер;
2. организация;
3. проект;
4. шаблон задания.

При создании среды исполнения необходимо указать следующие параметры:

- Название. Должно быть уникальным в рамках контроллера.
- Образ. Ссылка на образ в реестре образов. Указывается в следующем формате:

[164] https://registry.astralinux.ru/browse/aa/#!/taglist/aa-base-ee

```
<registry>/<name>:<tag>
```

где:

- <registry> – URL реестра образов;
- <name> – название образа;
- <tag> – тег, указывающий версию образа.

6.7.1 Параметры загрузки

Astra Automation Controller позволяет задать настройки загрузки образа для каждой среды исполнения. Поддерживаются три варианта управления загрузкой:

- Перед запуском контейнера образ всегда загружается заново.

  Эта настройка может быть полезна при использовании в ссылке на образ тега `latest`.

- Перед запуском контейнера образ загружается только в том случае, если он отсутствует.
- Никогда – образ загружается только один раз, в момент создания среды исполнения.

6.7.2 Получение версий компонентов

Для получения сведений о версиях компонентов выполните следующие действия:

1. Запустите контейнер с образом среды исполнения в интерактивном режиме:

   ```
   docker run \
       --tty \
       --interactive \
       --rm \
       registry.astralinux.ru/aa/aa-base-ee:<tag> \
       bash
   ```

   где <tag> – версия образа, например, `latest` или `0.5.1`.

2. Для получения сведений о версиях компонентов выполните следующие команды:

   - Python:

     ```
     python3 --version
     ```

   - Пакеты Python:

     ```
     pip3 freeze
     ```

   - Компоненты Ansible:

     ```
     pip3 freeze | grep ansible
     ```

   - OpenTofu:

```
tofu --version
```

3. Для завершения работы с контейнером выполните команду:

```
exit
```

Архитектура *Структура управления* Элементы управления и связь между ними.

Графический интерфейс Управление через веб-интерфейс.

Программный интерфейс Формальное описание API в стандарте Open API.

ГЛАВА 7

Разработка контента и среды исполнения

Значимой частью Astra Automation является набор средств создания собственного контента Ansible и контейнерных образов для среды исполнения. Данный раздел представляет концепции и инструкции по созданию образов среды исполнения и коллекций Ansible, удовлетворяющих потребностям организации.

7.1 Среда исполнения

Среда исполнения Astra Automation (Execution Environment, EE) – это контейнер с образом на базе Astra Linux Special Edition, который рекомендуется использовать для развертывания и тестирования кода, запускаемого с помощью Ansible и Terraform.

Предупреждение: Для запуска контейнера с указанным образом необходима версия Docker не ниже 19.03.

Использование EE дает следующие преимущества:

- Нет необходимости устанавливать дополнительное ПО: все необходимое уже есть в составе образа.
- Не требуется ручная настройка окружения.
- Контейнер с образом EE может быть запущен в любой ОС, поддерживаемой Docker.
- EE используется для тестирования кода коллекций Ansible, размещенных в реестре Astra Automation Hub.

Совет: Актуальный список версий образа EE доступен в реестре Astra Automation[165].

[165] https://registry.astralinux.ru/browse/aa/#!/taglist/aa-base-ee

7.1.1 Загрузка образа

Для загрузки образа среды исполнения используйте команду:

```
docker pull registry.astralinux.ru/aa/aa-base-ee:<version>
```

где <version> – версия образа. Если версия не указана, загружается образ с меткой latest, которой соответствует новейшая стабильная версия образа.

Совет: С помощью команды tag для образа можно создать псевдоним (alias), например:

```
docker tag registry.astralinux.ru/aa/aa-base-ee aa-base-ee
```

Теперь к образу registry.astralinux.ru/aa/aa-base-ee:latest можно обращаться через псевдоним – aa-base-ee.

7.1.2 Использование

Для выполнения определенной команды используйте следующий синтаксис:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume <source>:<target> \
   registry.astralinux.ru/aa/aa-base-ee:<version> \
   <command>
```

где

- --rm – автоматическое удаление контейнера после выполнения команды.
- --interactive, -i – интерактивный режим работы контейнера.

  Необходим, если требуется ручное взаимодействие с созданным процессом, например, для ввода пароля.
- --tty, -t – включение поддержки стандартных функций терминала, например, форматирование вывода и скрытие символов вводимых паролей.
- --volume, -v – монтирование объекта файловой системы управляющего узла (<source>) в файловую систему контейнера (<target>).
- registry.astralinux.ru/aa/aa-base-ee – ссылка на образ.
- <version> – версия образа. Если не указана, используется самая свежая (latest).
- <command> – команда, которая должна быть выполнена в контейнере.

Подробное описание параметров команды **docker run** см. в документации Docker[166].

[166] https://docs.docker.com/reference/cli/docker/container/run/

7.1.3 Примеры

Следующие примеры демонстрируют использование EE для выполнения различных команд.

Запуск Terraform

Для развертывания инфраструктуры с помощью Terraform выполните в каталоге проекта команду:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume ~/.ssh/hub.astra-automation.ru:/root/.ssh/id_rsa \
   --volume "$(pwd):/app" \
   registry.astralinux.ru/aa/aa-base-ee \
   bash -c 'terraform init && terraform apply'
```

где

- `~/.ssh/hub.astra-automation.ru` – путь к файлу приватного ключа SSH, используемого для доступа к реестру образов Astra Automation Hub;
- `$(pwd)` – команда, возвращающая полный путь к текущему каталогу;
- `/app` – каталог в файловой системе контейнера, в который будет смонтирован каталог проекта из файловой системы управляющего узла.

Запуск Ansible playbook

Для настройки управляемых узлов с помощью Ansible выполните в каталоге проекта команду:

```
docker run \
   --rm \
   --interactive \
   --tty \
   --volume ~/.ssh/hub.astra-automation.ru:/root/.ssh/id_rsa \
   --volume "$(pwd):/app" \
   registry.astralinux.ru/aa/aa-base-ee:latest \
   bash -c 'ansible-galaxy install -r requirements.yml && `
           `ansible-playbook -i inventory.yml playbook.yml'
```

где

- `~/.ssh/hub.astra-automation.ru` – путь к файлу приватного ключа SSH, используемого для доступа к реестру коллекций Astra Automation Hub;
- `$(pwd)` – команда, возвращающая полный путь к текущему каталогу;
- `/app` – каталог в файловой системе контейнера, в который будет смонтирован каталог проекта из файловой системы управляющего узла;
- `ansible-galaxy` – утилита Ansible, используемая для управления зависимостями;
- `requirements.yml` – имя файла с перечнем зависимостей Ansible;
- `ansible-playbook` – утилита Ansible, используемая для запуска playbook;

- `inventory.yml` – файл инвентаря;
- `playbook.yml` – файл playbook.

7.2 Создание образа среды исполнения

Added in version 1.0-upd3.

Для создания образов среды исполнения EE (execution environment) следует использовать утилиту Ansible Builder.

Инструментом для непосредственного создания образа может быть Podman или Docker.

После установки утилиты и подготовки инструмента контейнеризации необходимо создать файл *определения среды исполнения (EE definition)*. По умолчанию файл называется `execution-environment.yml` (или `execution-environment.yaml`).

7.2.1 Установка утилиты ansible-builder

Установка утилиты `ansible-builder` в Astra Linux Special Edition 1.7.4 и 1.7.5 состоит из следующих шагов:

1. В каталоге `/etc/apt/sources.list.d/` создайте файл `astra-automation.list` со ссылкой на репозиторий `aa-debs-for-alse`:

   ```
   echo "deb https://dl.astralinux.ru/aa/aa-debs-for-alse-1.7 <version> main | sudo␣
   ↪tee -a /etc/apt/sources.list.d/astra-automation.list"
   ```

   где `<version>` – номер необходимой версии, которая должна быть не ранее `1.0-upd3`.

2. Обновите список доступных пакетов:

   ```
   sudo apt update
   ```

3. Установите пакет `ansible-builder`:

   ```
   sudo apt install ansible-builder --yes
   ```

   Added in version 1.0-upd3.

4. При использовании Docker для корректной работы `ansible-builder` добавьте активного пользователя в группу `docker`:

   ```
   sudo usermod -aG docker $USER && newgrp docker
   ```

7.2.2 Этапы сборки

Ansible Builder выполняет несколько этапов при запуске инструмента контейнеризации для создания образа контейнера:

1. **Base**: использует Docker или Podman для загрузки базового образа, который был определен в файле `execution-environment.yml`, затем устанавливает версию Python (если она определена и отличается от всех версий Python в базовом образе), `pip`, `ansible-runner` и `ansible-core` или `ansible`. Все три последующих этапа сборки строятся на выводе этапа Base.
2. **Galaxy**: загружает определенные в файле `execution-environment.yml` коллекции с Galaxy и сохраняет их локально.
3. **Builder**: загружает Python-пакеты и системные пакеты, определенные в файле `execution-environment.yml`, и сохраняет их локально.
4. **Final**: производит окончательную сборку образа.

Если необходима настройка размера разделяемой памяти (`--shm-size`) во время сборки EE, рекомендуется использовать Docker.

7.2.3 Файл определения среды исполнения

Параметры файла `execution-environment.yml` описаны в документации Ansible[167].

7.2.4 Настройка размера разделяемой памяти в Docker

1. Для настройки размера разделяемой памяти `/dev/shm` отредактируйте файл конфигурации Docker `/etc/docker/daemon.json`. Если такой файл отсутствует, создайте его:

   ```
   { "default-shm-size": "2g" }
   ```

   Значение 2g устанавливает размер разделяемой памяти в 2 гигабайта. Это значение следует подобрать исходя из потребностей и доступных системных ресурсов.

2. После внесения изменений перезапустите Docker:

   ```
   sudo systemctl restart docker
   ```

7.2.5 Сборка образа

Для сборки образа можно использовать Docker или Podman.

Пример команды для запуска сборки:

```
sudo ansible-builder build \
   -t my-ee:latest \
   -f execution-environment.yml \
   -c ./context \
   --container-runtime [docker|podman] \
   --build-arg RUN_OPTS="--privileged" \
   -vvv
```

[167] https://ansible.readthedocs.io/projects/builder/en/latest/definition/

где:

- `ansible-builder build` запускает процесс сборки образа.
- `-t my-ee:latest` – локальная метка образа, включающая название и версию.
- `-f execution-environment.yml` – путь к файлу с параметрами среды исполнения.
- `-c ./context` – путь к каталогу с дополнительными файлами, которые должны быть включены в образ.
- `--container-runtime [docker|podman]` – инструмент контейнеризации для сборки образа. Поддерживаются значения `podman` (по умолчанию) и `docker`.
- `--build-arg RUN_OPTS="--privileged"` добавляет аргументы к процессу сборки, например, запуск контейнера в привилегированном режиме. Это может быть необходимо для выполнения определенных операций, требующих расширенных прав в контейнере.
- `-vvv` – максимально подробный вывод сообщений о ходе сборки.

7.3 Разработка коллекций

Использование Ansible Navigator и Ansible Creator для разработки контента.

Важно: Эта часть документации находится в стадии разработки.

7.4 Проверка стиля и синтаксиса

Использование Ansible Linter для проверки стиля и синтаксиса коллекций и playbook.

Важно: Эта часть документации находится в стадии разработки.

7.5 Тестирование

Использование Ansible Molecule для тестирования контента.

Важно: Эта часть документации находится в стадии разработки.

7.6 Поиск неисправностей

Использование Ansible Navigator для проверки и поиска неисправностей в контенте, EE и определениях инвентаря.

Важно: Эта часть документации находится в стадии разработки.

7.7 Контроль целостности

Использование Ansible Sign для контроля подлинности и целостности контента.

Важно: Эта часть документации находится в стадии разработки.

ГЛАВА 8

Управление событиями

Обработчик событий Event-Driven Ansible оснащает платформу интеллектуальной составляющей по обработке различных событий, возникающих в различных компонентах инфраструктуры пользователя. Преимуществами применения обработчика событий являются:

- существенное снижение объема ручной работы;
- быстрое и своевременное реагирование на изменившуюся ситуацию в инфраструктуре.

Важно: Эта часть документации находится в стадии разработки.

ГЛАВА 9

Средства аналитики

Важно: Эта часть документации находится в стадии разработки.

ГЛАВА 10

Примеры решений

10.1 Защита информации

Операционная система Astra Linux Special Edition является сертифицированным средством защиты информации от несанкционированного доступа и реализует функции защиты информации, предусмотренные для 1 класса защиты следующими требованиями:

- Требования по безопасности информации к операционным системам (утв. приказом ФСТЭК России[168] России от 19.08.2016 г. № 119),
- Требования по безопасности информации к средствам виртуализации (утв. приказом ФСТЭК России от 27.10.2022 г. № 187),
- Требования по безопасности информации к средствам контейнеризации (утв. приказом ФСТЭК России от 4.07.2022 г. № 118),

Кроме того, Astra Linux Special Edition реализует функции системы управления базами данных (СУБД), обеспечивающие выполнение ряда функций безопасности СУБД.

Функции защиты реализованы встроенным комплексом средств защиты информации Astra Linux Special Edition и могут применяться для реализации мер защиты информации ограниченного доступа в соответствии с требованиями приказов ФСТЭК России от 11.02.2013 г. № 17, 18.02.2013 г. №21, от 14.03.2014 № 31, от 25.12.2017 г. № 239.

[168] https://fstec.ru/

10.1.1 Назначение

Настоящий раздел содержит общую информацию о возможных вариантах использования Astra Automation для автоматизации настройки системы защиты Astra Linux в информационных системах, обрабатывающих информацию ограниченного доступа.

10.1.2 Требования к мерам защиты

В соответствии с требованиями ФСТЭК России в информационных системах, обрабатывающих информацию ограниченного доступа: в государственных информационных системах, информационных системах персональных данных, автоматизированных системах управления, в составе значимых объектов критической информационной инфраструктуры, — должны быть реализованы меры защиты информации, состав которых определяется исходя из класса (уровня, категории) защищенности (значимости) системы и актуальной модели угроз. Соответствие системы защиты информации информационной системы требованиям по безопасности подтверждается на этапе аттестационных испытаний.

Возможности Astra Linux для реализации мер защиты информации, предусмотренных указанными выше приказами и описание используемых встроенных средств защиты приведены на официальном справочном ресурсе[169].

Примечание: При этом важно понимать, что требования приказов ФСТЭК России от 11.02.2013 г. № 17, 18.02.2013 г. № 21, от 14.03.2014 № 31, от 25.12.2017 г. № 239 не регламентируют настройки средств защиты, а определяют базовый набор мер защиты, который для реализации на объектах информатизации адаптируется и уточняется в зависимости от структурно-функциональных характеристик и модели угроз.

С учетом того, что меры защиты представлены в требованиях приказов без конкретных параметров настроек, наши рекомендации по настройке операционной системы могут оказаться неподходящими для вашей информационной системы или недостаточными для предотвращения несанкционированного доступа. Решение о применение настроек безопасности остается за администратором безопасности вашей информационной системы.

Важно: Решения по автоматизации применения мер защиты в виде коллекций Ansible находятся в активной стадии разработки. Этот раздел документации будет расширен описанием возможностей и подробностей применения этих решений.

---

[169] https://wiki.astralinux.ru/pages/viewpage.action?pageId=181666117

10.2 Пример организации процесса CI/CD

Рассмотрим организацию простейшего процесса непрерывной разработки и развертывания программных продуктов. При необходимости вы можете воспроизвести данный сценарий в собственном окружении.

10.2.1 Описание сценария

Приведен пример планирования и настройки непрерывного процесса разработки программного проекта (CI, continuous integration) и его развертывания (CD, continuous deployment).

Особенности сценария:

- Проект хранится в публично доступном репозитории под управлением Git, используя GitHub.
- Обработку процессов CI/CD выполняет контроллер Astra Automation.
- При создании запроса (PR, pull request) на внесение изменений в какую-либо ветку репозитория GitHub вызывает процесс CI, целью которого является проверка корректности разработанного программного продукта. При исполнении контролируемая программа оповещает о корректности путем возврата соответствующего кода:
  - `0` – программа работает корректно;
  - `1` – в программе обнаружена ошибка.

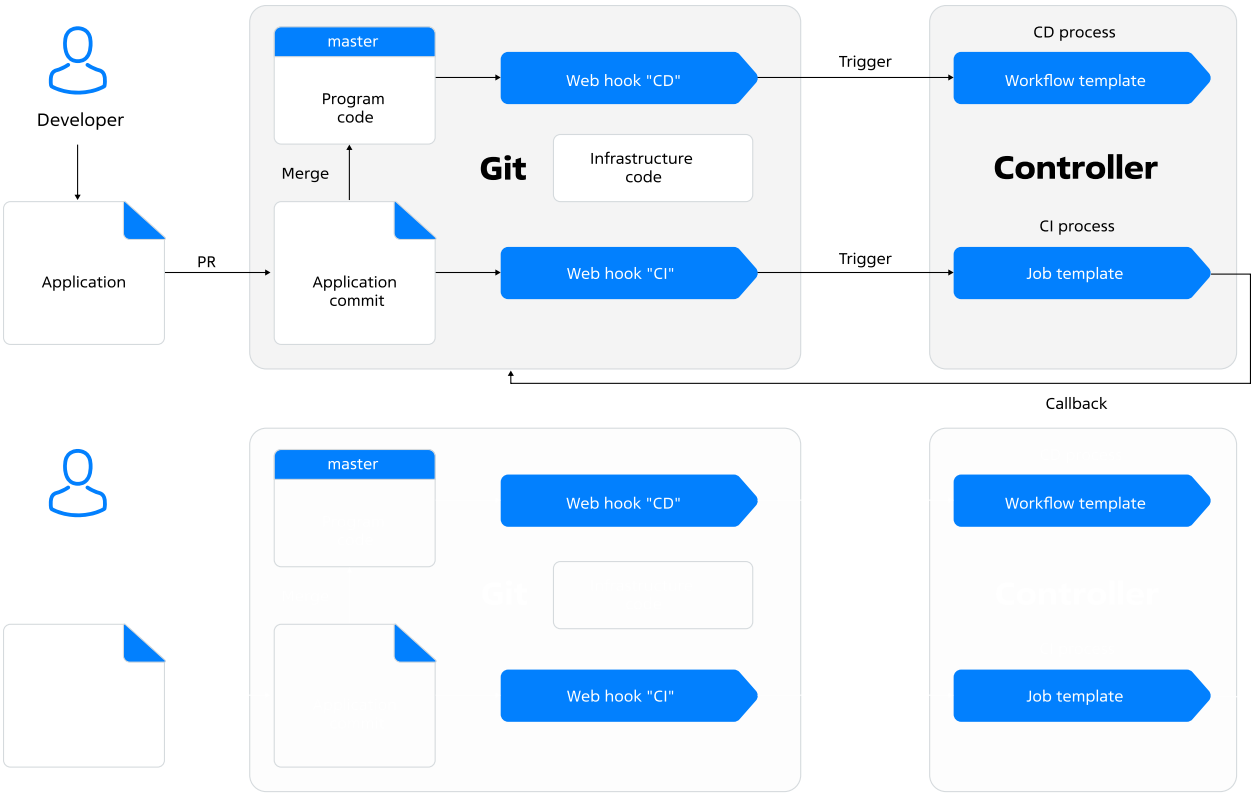
  При обнаружении ошибки в программе разработчик имеет возможность исправить код и повторить процесс CI.
- После слияния рабочей ветки с веткой `master` GitHub вызывает процесс CD, в котором контроллер проверяет, что объектом изменений является ветка `master`. В этом случае он запрашивает разрешение на обновление продукта и после одобрения выполняет обновление продукта.

10.2.2 Структура проекта

Проект реализации сценария включает разработку структуры и инфраструктурного кода.

Взаимодействие компонентов

Структурная схема сценария представляется в следующем виде:



Взаимодействие компонентов в процессе CI происходит следующим образом:

1. В репозитории GitHub разработчик вносит изменения в демонстрационную программу и создает запрос на внесение изменений в ветку `master` (PR).
2. GitHub перехватывает PR с помощью соответствующего объекта webhook и направляет запрос на выполнение процесса CI в контроллере. Запрос PR переходит в состояние ожидания ответа от контроллера.
3. Контроллер получает запрос на запуск задания из шаблона, реализующего процесс CI. В данном сценарии задание выводит содержимое программы в выходной поток и запускает программу для проверки корректности ее выполнения.
4. Контроллер возвращает результат обработки запроса в GitHub.
5. GitHub получает результат запроса и сообщает о готовности внести изменения в `master`.

CD является продолжением предыдущего процесса. Его можно выполнить в случае успешного завершения последнего:

1. Разработчик принимает решение о внесении изменений, представленных PR, в ветку `master` и начинает процесс слияния веток (Git merge).
2. После завершения обновления ветки `master` GitHub перехватывает запрос с помощью соответствующего объекта webhook и направляет запрос на выполнение процесса CD в контроллере.
3. Контроллер получает запрос на запуск заданий из шаблона потока, реализующего процесс CD. Он запускает выполнение цепочки заданий, включая получение одобрения от ответственного лица.

Примечание: В отличие от процесса CI, контроллер не возвращает результат обработки запроса в GitHub. Поэтому результаты процесса CD необходимо анализировать средствами контроллера.

Репозиторий проекта

Для простоты исполняемый код обслуживаемой демонстрационной программы разработчика (`project.sh`) и инфраструктурный код для реализации сценария находятся в одном репозитории в виде следующей структуры каталогов и файлов:

```
./
├── aac
│   ├── cd_check.yml
│   ├── cd_deploy.yml
│   └── ci.yml
└── project.sh
```

Назначение и состав файлов представлены в следующих секциях. Для воспроизведения сценария загрузите эти файлы в свой репозиторий GitHub в соответствии с представленной структурой.

Демонстрационная программа

Демонстрационная программа, к которой применены процессы CI/CD, состоит из одного скрипта shell `project.sh` (`download`):

Demo program

```
#!/bin/bash

# Use "exit 0" to simulate correct code
# Use "exit 1" to "exit 255" to simulate incorrect code
exit 1
```

CI/CD playbooks

Каждая часть процесса использует отдельные файлы playbook.

Процесс CI состоит из одного задания, которое выполняется с помощью playbook `aac/ci.yml` (`download`) на локальном узле, то есть на самом контроллере:

CI playbook

```
---
- hosts: localhost
  become: false
  gather_facts: false
  tasks:
    - name: Get code revision "{{ awx_webhook_event_ref }}"
      ansible.builtin.git:
        repo: "{{ awx_webhook_payload.repository.clone_url }}"
        dest: /tmp/project
        version: "{{ awx_webhook_event_ref }}"

    - name: Show main project
      ansible.builtin.shell:
        chdir: /tmp/project
        cmd: cat ./project.sh

    - name: Check main project
      ansible.builtin.shell:
        chdir: /tmp/project
        cmd: ./project.sh
```

Playbook выполняет следующие действия:

1. Создает клон репозитория, адрес которого содержит переменная `awx_webhook_payload.repository.clone_url`, в каталоге `/tmp/project/`.
2. Выводит содержимое файла проекта в выходной поток командой `cat`.
3. Запускает на выполнение скрипт программы для тестирования ее работоспособности. По результатам выполнения контроллер определяет корректность внесенных изменений.

Процесс CD состоит из потока заданий, в который вовлечены следующие файлы playbook:

1. `aac/cd_check.yml` (`download`) – проверка того, что вызов задания связан с обновлением ветки `master` в репозитории GitHub:

   ### CD check playbook

   ```
   ---
   - hosts: localhost
     become: false
     gather_facts: false
     tasks:
       - name: Check if push is into master branch
         ansible.builtin.fail:
           msg: This push is not into master branch
         when: awx_webhook_payload.ref != "refs/heads/master"
   ```

   Playbook завершает аварийно процесс CD, если обновление было не в ветке `master`. В противном случае процесс продолжается.

2. `aac/cd_deploy.yml` (`download`) – демонстрационное (фиктивное) развертывание продукта:

CD deployment playbook

```
---
- hosts: localhost
  become: false
  gather_facts: false
  tasks:
    - name: Deploy project
      ansible.builtin.debug:
        msg: The project is being deployed.
```

Playbook только выводит сообщение о выполнении развертывания.

10.2.3 Настройка процесса

После создания публичного репозитория необходимо выполнить настройки основных компонентов.

Настройка прав доступа

Для обеспечения взаимодействия компонентов проекта необходимо предоставить контроллеру полномочия на доступ к репозиторию GitHub. Это позволит контроллеру возвращать состояние процесса (callback) в GitHub. Полномочия необходимо предоставить через персональный токен доступа к репозиторию GitHub. Для его создания в GitHub существует следующая последовательность шагов в настройках:

profile > settings > developer settings > personal access tokens > tokens(classic) > generate new token

---

Примечание: Сгенерированный токен показывается только один раз. Для повторного использования необходимо сохранить его в надежном месте.

Контроллер

В контроллере необходимо произвести следующие настройки:

1. Задайте базовый URL сервиса, который для контроллера, установленном на одном узле, совпадает с URL этого контроллера. Шаги настройки:

   Настройки (Settings) > Система (System) > Общие системные настройки (Miscellaneous System settings) > Базовый URL сервиса (Base URL of the service)

   Примечание: Базовый URL будет передан в GitHub для использования его в качестве ссылки на соответствующее задание контроллера.

   ---

2. Создайте организацию. Шаги настройки:

   Access > Organizations > Add

   Параметры настройки:

- **Название**: *cicd*.

3. Создайте полномочие на доступ к репозиторию GitHub. Шаги настройки:

   Resources > Credentials > Add

   Параметры настройки:

   - **Название** (Name): *cicd*;
   - **Тип полномочия** (Credential Type): **Личный токен доступа к GitHub** (GitHub Personal Access Token);
   - **Сведения о типе** (Type Details) > **Токен** (Token): <содержимое токена, созданного для организации или учетной записи в GitHub>;
   - **Организация** (Organization): *cicd*.

4. Создайте проект на основе репозитория, созданного в GitHub. Шаги настройки:

   Resources > Projects > Add

   Параметры настройки:

   - **Название**: *cicd*;
   - **Организация** (Organization): *cicd*;
   - **Среда исполнения** (Execution Environment): *Default execution environment*;
   - **Тип системы управления исходными данными** (Source Control Type): *Git*;
   - **URL системы управления исходными данными** (Source Control URL): *https://github.com/<GitHub account or organization>/<repo-name>.git*.

5. Создайте шаблоны заданий. Шаги настройки:

   Resources > Templates > Add > Add job template

   Параметры настройки для шаблона CI:

   - **Название** (Name): *ci*;
   - **Тип задания** (Job Type): **Выполнение** (Run);
   - **Инвентарь** (Inventory): *Demo Inventory*;
   - **Проект** (Project): *cicd*;
   - **Playbook**: `aac/ci.yml`.
   - В секции **Опции** (Options) включите следующие флаги:
     - **Включить webhook** (Enable Webhook);
     - **Параллельные задания** (Enable Concurrent Jobs).
   - В секции **Подробности о webhook** (Webhook details) настройте следующие параметры:
     - **Сервис webhook** (Webhook Service): **GitHub**;
     - **Полномочия webhook** (Webhook Credential): *cicd*.

   Сохраните настройки.

   Параметры настройки для шаблона задания на шаге проверки процесса CD:

   - **Название** (Name): *cd_check*;

- **Тип задания** (Job Type): **Выполнение** (Run);
- **Инвентарь** (Inventory): *Demo Inventory*;
- **Проект** (Project): *cicd*;
- **Playbook**: `aac/cd_check.yml`.

Сохраните настройки.

Параметры настройки для шаблона задания на шаге развертывания процесса CD:

- **Название** (Name): *cd_deploy*;
- **Тип задания** (Job Type): **Выполнение** (Run);
- **Инвентарь** (Inventory): *Demo Inventory*;
- **Проект** (Project): *cicd*;
- **Playbook**: `aac/cd_deploy.yml`.

Сохраните настройки.

6. Создайте шаблон потока заданий для процесса CD. Шаги настройки:

   Resources > Templates > Add > Add workflow template

   Параметры настройки:

   - **Название** (Name): *cd*;
   - **Организация** (Organization): *cicd*;
   - **Инвентарь** (Inventory): *Demo Inventory*.
   - В секции **Опции** (Options) выберите следующие опции:
     - **Включить webhook** (Enable Webhook);
     - **Параллельные задания** (Enable Concurrent Jobs).
   - В секции **Подробности о webhook** (Webhook details) настройте следующие параметры:
     - **Сервис webhook** (Webhook Service): **GitHub**;
     - **Полномочия webhook** (Webhook Credential): *cicd*.

   Сохраните настройки.

   Во всплывающем окне **Пожалуйста, нажмите кнопку «Пуск», чтобы начать** (Please click the Start button to begin) нажмите кнопку **Пуск** (Start) для настройки потока с помощью визуализатора.

   - Создайте узел на основе шаблона `cd_check` со следующими параметрами:
     - **Тип узла** (Node Type): **Шаблон задания** (Job Template);
     - **Название** (Name): *cd_check*.
   - Создайте узел согласования со следующими параметрами:
     - **Тип узла** (Node Type): **Согласование** (Approval);
     - **Название** (Name): *Согласование развертывания продукта*.
   - Создайте узел на основе шаблона `cd_deploy` со следующими параметрами:
     - **Тип узла** (Node Type): **Job Template**;

  - **Название** (Name): *cd_deploy*.

После сохранения каждого из двух шаблонов контроллер сгенерирует параметры доступа для шаблона в следующем виде:

- **Webhook URL**: <webhook URL>;
- **Ключ webhook** (Webhook key): <hash code>.

Параметры доступа необходимы для настройки объектов webhook в GitHub.

GitHub

В репозитории проекта необходимо настроить перехват событий с помощью объектов webhook.

Шаги настройки:

Settings > Webhooks > Add webhook

Создайте объекты перехвата событий со следующими параметрами:

- Webhook для CI:
  - **Payload URL**: <URL, использованный в настройках шаблона задания CI в контроллере>;
  - **Content type**: **application/json**;
  - **Secret**: <значение Webhook key из настроек шаблона задания CI в контроллере>;
  - **SSL verification**: **Disable**.

    Примечание: Здесь и далее необходимо отключить проверку сертификата, если не установлен корректный сертификат TLS/SSL на сервере, которому вы доверяете.

  - В секции **Which events would you like to trigger this webhook** выберите опцию **Let me select individual events**.
  - В открывшемся списке выберите **Pull requests**.
  - Выберите опцию **Active**.
- Webhook для CD:
  - **Payload URL**: <URL, использованный в настройках шаблона потока заданий CD в контроллере>;
  - **Content type**: **application/json**;
  - **Secret**: <значение Webhook key из настроек шаблона задания CI в контроллере>;
  - **SSL verification**: **Disable**.
  - В секции **Which events would you like to trigger this webhook** выберите опцию **Let me select individual events**.
  - В открывшемся списке выберите **Pushes**.
  - Выберите опцию **Active**.

10.2.4 Проверка работоспособности

Проверке подлежат отдельно два процесса.

Проверка CI

Инициируйте процесс CI следующими действиями:

1. В репозитории GitHub внесите изменение в файл `project.sh`, так чтобы в нем оставалась команда `exit 1`, имитирующая некорректный код.
2. Нажмите кнопку **Commit changes…**.
3. Выберите опцию **Create a new branch…** и введите комментарий в поле сообщения.
4. Нажмите кнопку **Propose changes**.
5. Нажмите кнопку **Create pull request**.

Сервис GitHub выведет сообщения о начале обработке события. Эта обработка завершается сообщением об ошибке «All checks have failed, 1 failing check».

Проверьте запуск процесса CI в контроллере:

1. В панели навигации откройте *Views* > *Jobs*. В верху списка вы видите появление новой записи о запуске задания из шаблона CI.
2. Откройте задание и изучите результат его работы. Поскольку скрипт `project.sh` возвращает код 1, то последняя задача в playbook `ci.yml` возвращает ошибку.

Повторите весь процесс, обеспечив в скрипте `project.sh` выполнение команды `exit 0`, сообщающей об успешном выполнении программы. Отличия от предыдущей попытки:

- Процесс следует начать с редактирования последней ветки, созданной в предыдущей попытке.
- Не надо создавать еще одну ветку, а выберите ту, в которой вы внесли изменения.

В этот раз процесс CI завершается успешно. Для того, чтобы убедиться в этом, откройте в вкладку **Pull requests** и затем откройте новейший PR.

Проверка CD

Процесс CD является продолжением процесса CI. После успешного завершения CI сервис GitHub предоставит возможность начать процесс обновления ветки `master` с последующим обновлением продукта.

После появления зеленой кнопки **Merge pull request** выполните обновление ветки `master`. Для этого нажмите кнопку **Merge pull request** и затем **Confirm merge**.

На этом работа в GitHub завершена.

Проверьте запуск процесса CD в контроллере:

1. В панели навигации откройте *Views* > *Jobs*. В верху списка вы видите появление новой записи о запуске потока заданий из шаблона CD.
2. Откройте задание и изучите результат ее работы.
3. Используйте один из следующих способов для определения узла графа, требующего разрешения на выполнение дальнейших действий:

- визуализатор открытого потока заданий;
- иконка уведомлений (в виде колокольчика);
- список согласований, который можно открыть с помощью панели навигации: *Режимы просмотра* ‣ *Согласование потоков заданий* (*Views* ‣ *Workflow Approvals*).

4. Выберите соответствующий узел и нажмите кнопку **Согласовать** (Approve). Убедитесь, что выполнение потока заданий (процесс CD) завершено успешно.

10.3 Применение конструктора кода

Важно: Эта часть документации находится в стадии разработки.

Одним из преимуществ контроллера являются его возможности по управлению сложной цепочкой заданий, которые можно создавать как через графический интерфейс, так и через API.

Данный раздел иллюстрирует подобные возможности с помощью различных сценариев.

ГЛАВА 11

Обеспечение безопасности

В этом разделе рассматривается обеспечение безопасности Astra Automation.

11.1 Сетевые настройки

Существуют различные способы атак и защиты в сети, которые учитывают в корпоративных инфраструктурах. Некоторые из них следует учитывать при настройке узлов контроллера.

11.1.1 Использование специальных каналов связи

В контурах сети, где накладываются особые ограничения на связь с внешним миром, используют такие каналы связи, которые обеспечивают минимальный размер периметра возможных атак.

Особенности ограничений

Специальный канал связи типично включает следующие ограничения (примеры приведены применительно к Astra Automation):

- Разрешается подключение только к определенным серверам, например, контроллеру необходим доступ к Astra Automation Hub.
- Разрешается подключение только по определенному протоколу, например, по SSH.
- Подключение разрешено определенному пользователю.

Реализация такого канала для контроллера требует учета следующих особенностей:

- Сервисы контроллера работают с привилегиями пользователя awx.

- Синхронизация проектов контроллера с Astra Automation Hub или другими репозиториями происходит с использованием той системной среды исполнения (EE), которая добавляется к контроллеру в процессе его установки. Системная среда исполнения использует по умолчанию образ `aa-base-ee`, версия которого может изменяться вместе с версией контроллера.

  Примечание: Добавление какой-либо другой EE в настройках проекта влияет только на исполнение playbook, но не на синхронизацию проектов с репозиториями.

Пример настройки

Предположим, что выход в интернет по SSH должен происходить через прокси-сервер с адресом 10.111.222.1 по порту 2022. Для установления связи через прокси используется универсальная утилита ncat[170].

Примечание: Это внешняя утилита, которая должна выполняться внутри EE, однако она не установлена там по умолчанию.

Для использования утилиты `ncat` необходимо создать собственный образ EE и использовать его в качестве системного. Соответственно процесс состоит из следующих шагов:

1. *Создайте собственный образ EE*, включающий утилиту `ncat`, с помощью Ansible Builder.
2. Обеспечьте использование собственного образа EE в качестве системного при развертывании платформы. Для этого воспользуйтесь дополнительной переменной Ansible `control_plane_execution_environment`, значением которой должен быть URL требуемого образа EE, например:

   ```
   ./aa-setup -- -e control_plane_execution_environment=registry.astralinux.ru/aa/aa-
   ↪base-ee:<version>
   ```

Для подключения контроллера к Astra Automation Hub по SSH выполните следующие настройки в контроллере:

1. Создайте файл `config` в каталоге настроек SSH пользователя awx, то есть в каталоге `/var/lib/awx/.ssh/`, со следующим примерным содержимым:

   ```
   Host hub.astra-automation.ru
      ProxyCommand ncat --proxy 10.111.222.1:2022 %h %p
      PreferredAuthentications publickey
      IdentityFile /var/lib/awx/.ssh/id_key
   ```

   Особенности настройки:

   - Подключение к прокси происходит с помощью утилиты `ncat`.
   - `IdentityFile` ссылается на приватный ключ, публичная часть которого должна быть расположена на сервере, указанном в параметре `HOST`, в данном примере – `hub.astra-automation.ru`.

[170] https://nmap.org/ncat/

2. Обеспечьте подключение к EE тех компонентов, которые в нем отсутствуют. Это необходимо выполнить через графический интерфейс контроллера.

   Путь навигации: **Настройки** (Settings) > **Настройки заданий** (Jobs) > **Пути доступа к изолированным заданиям** (Paths to expose to isolated jobs).

   Дополните список монтирования следующими парами:

   ```
   "/var/lib/awx/.ssh/config:/root/.ssh/config:ro",
   "/var/lib/awx/.ssh/id_key:/root/.ssh/id_key:ro".
   ```

   В этом примере в EE попадут следующие недостающие компоненты:

   - файл настроек SSH `config`,
   - приватный ключ `id_key`.

С использованием приведенных настроек контроллер будет иметь доступ к Astra Automation Hub через указанный прокси-сервер для получения необходимого инфраструктурного кода.

11.2 Реквизиты в сети Mesh

Рецепторы выполняют функцию отправки и получения сообщений от узлов Astra Automation Controller.

По умолчанию для рецепторов генерируется самоподписанный сертификат. Если необходимо установить доверительные сеансы связи в кластере (сеть Mesh) с помощью собственных сертификатов SSL/TLS и секретных ключей, выполните следующие действия:

1. Сгенерируйте ключ для рецептора, например:

   ```
   openssl genrsa -out /tmp/mesh-CA.pem 2048
   ```

2. Используя созданный ключ, сгенерируйте сертификат для рецептора, например:

   ```
   openssl req -x509 -sha256 -new -nodes -key /tmp/mesh-CA.pem -days 3650 -out /tmp/
   ↪mesh-CA.crt -subj "/C=RU/ST=Moscow/L=Moscow/O=Company/OU=IT/
   ↪CN=CertificateAuthority"
   ```

   где:

   - `-new` – создание нового сертификата;
   - `-key` – указание файла с закрытым ключом;
   - `-out` – указание местоположения и названия файла вывода;
   - `-subj` – указание заданных значений сертификата:
     - `C` – страна, в виде двухсимвольного ISO-кода;
     - `ST` – регион, в котором официально зарегистрирована организация;
     - `L` – город, где официально зарегистрирована организация;
     - `O` – наименование организации в соответствии с уставом организации;
     - `OU` – наименование подразделения;
     - `CN` – FQDN веб-сервера.

3. На узле, откуда выполнялась установка продукта, перейдите в каталог `/opt/rbta/aa/astra-automation-setup/`:

```
cd /opt/rbta/aa/astra-automation-setup/
```

4. В файле `inventory` добавьте в секцию `[all:vars]` переменные `mesh_ca_keyfile` и `mesh_ca_certfile`, в значении которых укажите пути к сертификату и ключу соответственно, например:

```
[all:vars]
mesh_ca_keyfile=/tmp/mesh-CA.pem
mesh_ca_certfile=/tmp/mesh-CA.crt
```

5. Запустите команду `./aa-setup`, как указано в разделе *Установка*. Во время ее выполнения на основе указанного сертификата формируются сертификаты для всех узлов рецептора.

6. На любом узле, где запущена служба рецептора и прослушивается порт 27199, выполните команду проверки сертификата TLS:

```
openssl s_client localhost:27199
```

Команда выводит на экран терминала сообщение вида:

Ответ

```
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = 10.123.0.12
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 CN = 10.123.0.12
verify error:num=21:unable to verify the first certificate
verify return:1
depth=0 CN = 10.123.0.12
verify return:1
Certificate chain
0 s:CN = 10.123.0.12
i:CN = Astra Automation Nodes Mesh ROOT CA
Server certificate
-----BEGIN CERTIFICATE-----
MIIFQTCCAymgAwIBAgIEZflYiDANBgkqhkiG9w0BAQsFADAuMSwwKgYDVQQDEyNB
c3RyYSBBdXRvbWF0aW9uIE5vZGVzIE1lc2ggUk9PVCBDQTAeFw0yNDAzMTkwOTE5
MDRaFw0zNDAzMDkwNjE4NTFaMBYxFDASBgNVBAMTCzEwLjEyMy4wLjEyMIICIjAN
BgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA6u6t7rs9gkJKWxoFlwSzOMdSNkJG
ppUQxAt2eyaPYUWWGhA1GkCIpw3NyhptRspqH5H1ZUGs0tMGYHebGHaBU0N0wthr
QMW9pV084VFnoam8fxGtFCGxAXNE1Ti6dx5YBLRRscrtRCdjQCAWaDxCSeoGDkko
Ux0vnLow4QR6H7KSOyRijLjdwDMZFUCu/jUfeLywfPa+0dpGqy5/RKKmv92yZ9Du
oTP/12gKDwQSTVHo1bSfC0i7lsSVpQ9eCs0aNdgB97yawwEFnCidBTDNV8YpQpCr
azYJwTHTtT09u/S2c4rNVpVM6kwAGiz58fffG0CRR1MPspmE7IYeCdUJ4LWn/xHT
Hqi4F7T4un30rN5mZwGkjRzrV4js9TxAKAWKvY4Rp6aKwL4Ie5KI4o5K8we5JFtK
5MVyGEeg4JQD+74QnUU6i2Y/ESAFD4BjNC6krdGU/Aao5Ia0hYv8Qat1QQzRIDoN
DEE2+5kmv/KqxMKlPFwQgCYVevGum+fYxat+fv/d/kVLa/6dpdjUjhb6/dQkyIBZ
hek/Uifk87y5aPeNZfBPh+DEW4rCSL13qwuRPdnJkb9k9ONakhrhtGtzgegaQJfi
wCuFet1DfEuoS9n4RMXk8nXFnZt5rJGq2amPyf0lPm29jc424ogUJNUVgtaVZScN
F+C/wYPlhWDpaU8CAwEAAaN/MH0wDgYDVR0PAQH/BAQDAgeAMB0GA1UdJQQWMBQG
```

(continues on next page)

(продолжение с предыдущей страницы)

```
CCsGAQUFBwMCBggrBgEFBQcDATAfBgNVHSMEGDAWgBQHRhlDJBpd0gwyDemRsSzG
9qdc0TArBgNVHREEJDAihwQKewAMoBoGCSsGAQQBkggTAaANDAsxMC4xMjMuMC4x
MjANBgkqhkiG9w0BAQsFAAOCAgEAiWI5QFfd6gyYNGrMhVUr9GR17C9nnnvAwoSx
ZoKmxkO3tSaFJCq3FZSXERbRuDQ1+d21JjHcvChamLQt29iWfjuH4GvLHUwxs9SQ
+oTv7IWz+jD4iRA1NYSSZ8QoJ0ZVu+WrVgHz1lzrr1LSj0Vo2tdx5ptiw7d09sp+
c2MAgv/MHQjBOJVTCRU7d0yhV/tAXqpdWZVAeWB5YbeP4cfTaqGVOJOJUW5gvkCI
4WI9x/6Ks2vB0kA7jtOYTTL935vMUchRCdW7UhOmVoYGjHqSeq7JPF1nu8nWfcMk
SzHT+ksqVWbO0LlRK3KxgotfQDgDIg2QFk0qAAN2wJN045GlYohu4shDZVEq6Ywp
NV8pwYZaLSoWxjK+2GjdF625H/bw+ZMeXw1eq2BX02Ub6YD9kJcrZNQZMwWfa7Bz
Chkp5CR49zTmAIfMlo/qELP6RrUSqhxoNDJhw1UeSjilXsO2xKu1OuY0TAdN7oz/
JELaVyA/wvarOLM57oxwGYc/1cYHlKEY4GYSGmaanHl9Elq0J4VGQ7PpL67l1Ztj
ofiPVVS2pkVe8PPCrjR88ZAsYpex3GokgJzOl5sXSeG/JaOjniLOhab5OfnMFN6j
fZCnpUWIXCFwJ7QFfvDxo+sSUclse1FPZtPiLygdeqayoRUVpuxiwP/WPsRwIyXE
iViG1l4=
-----END CERTIFICATE-----
subject=CN = 10.123.0.12
issuer=CN = Astra Automation Nodes Mesh ROOT CA
Acceptable client certificate CA names
CN = Astra Automation Nodes Mesh ROOT CA
Requested Signature Algorithms: RSA-PSS+SHA256:ECDSA+SHA256:Ed25519:RSA-
 ↪PSS+SHA384:RSA-
 ↪PSS+SHA512:RSA+SHA256:RSA+SHA384:RSA+SHA512:ECDSA+SHA384:ECDSA+SHA512:RSA+SHA1:ECDSA+SHA1
Shared Requested Signature Algorithms: RSA-PSS+SHA256:ECDSA+SHA256:Ed25519:RSA-
 ↪PSS+SHA384:RSA-
 ↪PSS+SHA512:RSA+SHA256:RSA+SHA384:RSA+SHA512:ECDSA+SHA384:ECDSA+SHA512
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: ECDH, P-521, 521 bits
SSL handshake has read 2462 bytes and written 761 bytes
Verification error: unable to verify the first certificate
New, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
Server public key is 4096 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 21 (unable to verify the first certificate)
127922245240000:error:1409445C:SSL routines:ssl3_read_bytes:tlsv13 alert␣
 ↪certificate required:../ssl/record/rec_layer_s3.c:1544:SSL alert number 11
```

11.3 Управление сертификатами TLS

Если при развертывании контроллера в конфигурационном файле `inventory` не были указаны пути к файлу сертификата TLS (далее – сертификат) и файлу соответствующего ему ключа (далее – ключ), для защиты подключения используется самоподписанный сертификат.

Для защиты подключения к контроллеру, работающему в продуктовой среде, необходим сертификат, выданный удостоверяющим центром.

В этом документе приводятся инструкции по перевыпуску самоподписанного сертификата и управлению сертификатами, выданными удостоверяющим центром.

Важно: Все описанные действия необходимо выполнять на узле, с которого выполнялось развертывание контроллера.

11.3.1 Перевыпуск самоподписанного сертификата

Чтобы перевыпустить и переустановить самоподписанный сертификат, выполните следующие действия:

1. Перейдите в каталог `/opt/rbta/aa/astra-automation-setup/`:

```
cd /opt/rbta/aa/astra-automation-setup/
```

2. В файле `inventory` добавьте в секцию `[all:vars]` переменную `aap_service_regen_cert` со значением `true`:

```
[all:vars]
aap_service_regen_cert=true
```

3. Запустите утилиту `aa-setup`:

```
sudo ./aa-setup
```

11.3.2 Замена сертификата TLS

Чтобы использовать сертификат, выданный удостоверяющим центром, выполните следующие действия:

1. Перейдите в каталог `/opt/rbta/aa/astra-automation-setup/`:

```
cd /opt/rbta/aa/astra-automation-setup/
```

2. Скопируйте файлы сертификата и соответствующего ему ключа в каталог `/opt/rbta/aa/astra-automation-setup/` или один из его подкаталогов.

3. В файле `inventory` в секцию `[all:vars]` добавьте переменные `web_servercertificates_cert` и `web_servercertificates_key`, в значении которых укажите полные пути к файлам сертификата и ключа соответственно, например:

```
[all:vars]
web_servercertificates_cert = /opt/rbta/aa/astra-automation-setup/cert.pem
web_servercertificates_key = /opt/rbta/aa/astra-automation-setup/cert.key
```

4. Если в секции `[all:vars]` присутствует параметр `aap_service_regen_cert`, закомментируйте или удалите строку с ним.

5. Запустите утилиту `aa-setup`:

```
sudo ./aa-setup
```

6. Для проверки корректности замены сертификата выполните команду:

```
openssl s_client -connect <controller>:443
```

где <controller> – *FQDN* контроллера.

Если сертификат заменен корректно, в терминал выводится набор строк следующего вида (часть вывода опущена с целью сокращения):

```
CONNECTED(00000003)
depth=1 C = RU, O = Astra Automation by RusBITech-Astra, CN = Astra Automation
verify return:1
depth=0 CN = 192.168.56.11
verify return:1
---
Certificate chain
 0 s:CN = 192.168.56.11
   i:C = RU, O = Astra Automation by RusBITech-Astra, CN = Astra Automation
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFcTCCA1mgAwIBAgIUDpdgb2x2Rgm0GjT+c1XYNaVlWvYwDQYJKoZIhvcNAQEL
BQAwVjELMAkGA1UEBhMCUlUxLDAqBgNVBAoMI0FzdHJhIEF1dG9tYXRpb24gYnkg
...
Fm8gAXMhafc2dKQh7VDtR2c4aq7YR/QyfxjDxJ4CXdXSnpVjuVXyOM0Y0WnZnL1A
uF6oS+U=
-----END CERTIFICATE-----
subject=CN = 192.168.56.11

issuer=C = RU, O = Astra Automation by RusBITech-Astra, CN = Astra Automation

---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 2135 bytes and written 399 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 4096 bit
...
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 798F4BB7D362B626B60828B0FAC738EF58C370A4C39F7A2094AA7965B3B1D77E
    Session-ID-ctx:
    Master-Key:␣
↪841E644233A530EF049B7C28EF67D552B18732C712B8547D38E3C75E9673320EE2E17E1554FD30AB466B87068DD073
    ...
---
```

ГЛАВА 12

Производительность

Важно: Эта часть документации находится в стадии разработки.

12.1 Концепции

Важно: Эта часть документации находится в стадии разработки.

12.2 Мониторинг

Важно: Эта часть документации находится в стадии разработки.

12.3 Управление

Важно: Эта часть документации находится в стадии разработки.

ГЛАВА 13

Техническое обслуживание

Astra Automation представляет собой сложную платформу со значительным количеством различных компонентов. В связи с большой значимостью процессов автоматизации вопросы технической поддержки платформы являются одними из наиболее важных.

13.1 Журналы

Ведение журнала – это функция, позволяющая отправлять подробные отчеты в различные службы агрегации журналов.

Сервис агрегатора журналов работает со следующими системами мониторинга и анализа данных:

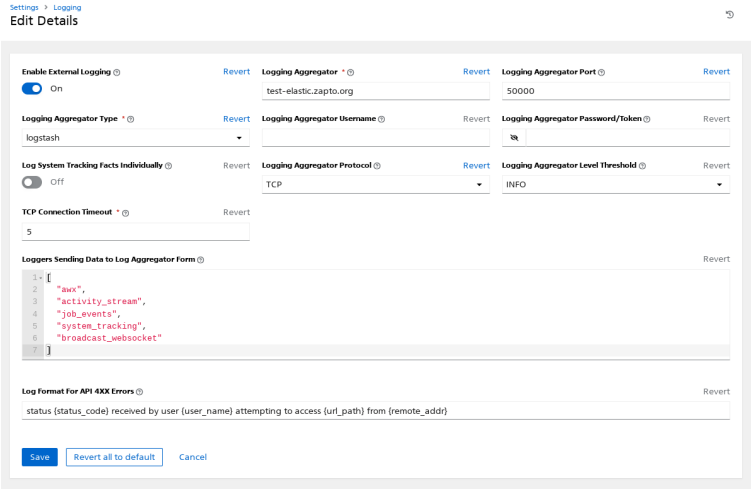
- Logstash[171];
- Splunk[172];
- Loggly[173];
- Sumo Logic[174].

В Astra Automation Controller в качестве системы централизованного сбора и хранения журналов используется syslog-ng[175].

Все настройки ведения журналов рекомендуется производить в веб-интерфейсе Astra Automation Controller. Установленные значения записываются в файл конфигурации `/var/lib/awx/syslogng/syslog-ng.conf`. После каждого изменения файла конфигурации сервис `awx-syslogng` перезапускается, применяя новые настройки.

[171] https://www.elastic.co/guide/en/logstash/current/index.html
[172] https://docs.splunk.com/Documentation
[173] https://documentation.solarwinds.com/en/success_center/loggly/content/loggly_documentation.htm
[174] https://help.sumologic.com/docs/get-started/
[175] https://github.com/syslog-ng/syslog-ng

Чтобы настроить ведение журнала, укажите необходимые значения в окне **Журналирование** в настройках контроллера:



Контроллер передает в агрегатор сообщения в соответствии с настроенным порогом:

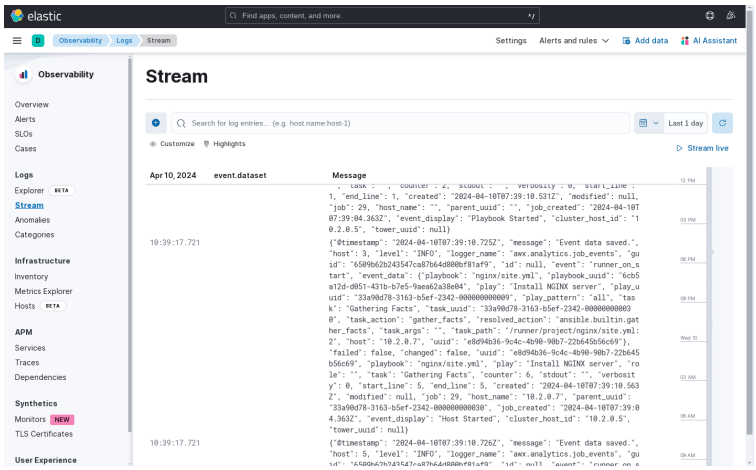
- **DEBUG** – все сообщения журнала;
- **INFO** – информационные сообщения;
- **WARNING** – предупреждения;
- **ERROR** – информация об ошибках;
- **CRITICAL** – информация о критических событиях.

Сообщения ниже порогового значения будут пропущены обработчиком журнала.

По умолчанию контроллер отправляет в агрегатор записи, полученные от всех внутренних регистраторов:

- awx – общие журналы сервера;
- `activity_stream` – запись изменений объектов в Astra Automation Controller;
- `job_events` – данные, возвращаемые модулем обратного вызова Ansible;
- `system_tracking` – данные о фактах, собранные модулем `setup`;
- `broadcast_websocket` – данные об отправленных сообщениях по WebSocket.

Пример записей, полученных в режиме stream, в агрегаторе типа Logstash:



13.2 Поиск причин неисправностей

Важно: Эта часть документации находится в стадии разработки.

13.3 Примеры сценариев

Важно: Эта часть документации находится в стадии разработки.

13.4 Использование базы знаний

Важно: Эта часть документации находится в стадии разработки.

13.5 Техническая поддержка

Положение о технической поддержке Astra Automation доступно на сайте ПАО Группа Астра по ссылке: https://astragroup.ru/support/polozheniya-o-tekhnicheskoy-podderzhke-astraautomation.php

Перед созданием обращения в службу технической поддержки убедитесь, что подходящего решения нет в документации Astra Automation и Базе знаний[176]. При создании обращения через Личный кабинет[177] решения из Базы знаний предлагаются автоматически.

[176] https://wiki.astralinux.ru/kb/
[177] https://lk-new.astralinux.ru/

Инструкции по использованию Личного кабинета доступны в Справочном центре в разделе Личный кабинет пользователя Astra Linux[178].

[178] https://wiki.astralinux.ru/pages/viewpage.action?pageId=128615802

ГЛАВА 14

Термины и определения

ALSE
: Astra Linux Special Edition

AMFT
: Astra Modules For Terraform

ARFA
: Astra Roles For Ansible

CIDR
: Classless Inter-Domain Routing (бесклассовая междоменная маршрутизация) – метод IP-адресации, который позволяет экономить адресное пространство путем использования сетевых масок переменной длины.

Cloud-Init
: Промышленный стандарт, позволяющий настраивать ОС виртуальных машин в облачных сервисах с помощью специального файла. Особенностью использования Cloud-Init является то, что настройка виртуальной машины выполняется только один раз – во время первой загрузки. (Источник[179])

DAG
: Directed Acyclic graph (направленный ациклический граф, ориентированный ациклический граф) – орграф, в котором отсутствуют направленные циклы, но могут быть «параллельные» пути, выходящие из одного узла и разными путями приходящие в конечный узел. Направленный ациклический граф является обобщением дерева, точнее, их объединения – леса. (Источник[180])

EE
: Execution Environment (среда исполнения) – контейнер с образом на базе Astra Linux Special Edition, который рекомендуется использовать для развертывания и тестирования кода, запускаемого с помощью Ansible и Terraform.

[179] https://cloud-init.io/
[180] https://ru.wikipedia.org/wiki/Ориентированный_ациклический_граф

FQCN
Fully Qualified Content Name, полностью определенное название контента.

FQDN
Fully Qualified Domain Name, полностью определенное имя домена

HA
High Availiability (высокая доступность) – подход к развертыванию веб-сайтов или приложений, обеспечивающий их доступность для пользователей даже при отказе отдельных частей инфраструктуры. (Источник[181])

HCL
HashiCorp Configuration Language – декларативный язык описания настроек программного обеспечения, разработанный для Terraform и используемый также другими системами. (Источник[182])

IaC
Infrastructure as Code – способ (подход) автоматического развертывания сетевой инфраструктуры от одиночных серверов до центров обработки данных и облачных структур. Способ основан на определении требуемой инфраструктуры в файлах, которые интерпретирует и реализует система развертывания инфраструктур. (Источник[183])

Данный подход реализован в Ansible, Terraform, Astra Automation и других системах автоматизации.

IPA
Identity, Policy, Audit – три основных компонента системы безопасности, положенные в основу FreeIPA. (Источник[184])

OCFS2
Система управления файлами Oracle Cluster File System второй версии, обеспечивающая доступ к единому пространству файлов. Используется в основном для виртуализации (Oracle VM), кластерных баз данных (Oracle RAC), кластеров на промежуточном ПО (Oracle E-Business Suite) и подобных системах. (Источник[185])

RBAC
Role Based Access Control (управление доступом на основе ролей) – политика избирального управления доступом, при которой права доступа субъектов системы на объекты группируются с учетом специфики их применения, образуя роли. (Источник[186])

VCS
Version Control System (система контроля версий) – программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. (Источник[187])

YC CLI
Yandex Cloud Command Line Interface – это интерфейс для управления ресурсами

[181] https://ru.wikipedia.org/wiki/Высокая_доступность
[182] https://octopus.com/blog/introduction-to-hcl-and-hcl-tooling
[183] https://en.wikipedia.org/wiki/Infrastructure_as_code
[184] https://www.freeipa.org/page/About
[185] https://ru.wikipedia.org/wiki/OCFS
[186] https://ru.wikipedia.org/wiki/Управление_доступом_на_основе_ролей
[187] https://ru.wikipedia.org/wiki/Система_управления_версиями

Yandex Cloud с помощью утилиты командной строки. (Источник[188])

ГИС

Государственная информационная система.

СЗИ

Средство защиты информации – техническое, программное, программно-техническое средство, вещество и (или) материал, предназначенные или используемые для защиты информации. (ГОСТ Р 50922-2006. Защита информации. Основные термины и определения)

СЗИ Astra Linux Special Edition

Средства защиты информации в Astra Linux Special Edition – комплекс программных средств защиты информации, реализующих функции безопасности Astra Linux.

[188] https://yandex.cloud/ru/docs/cli/

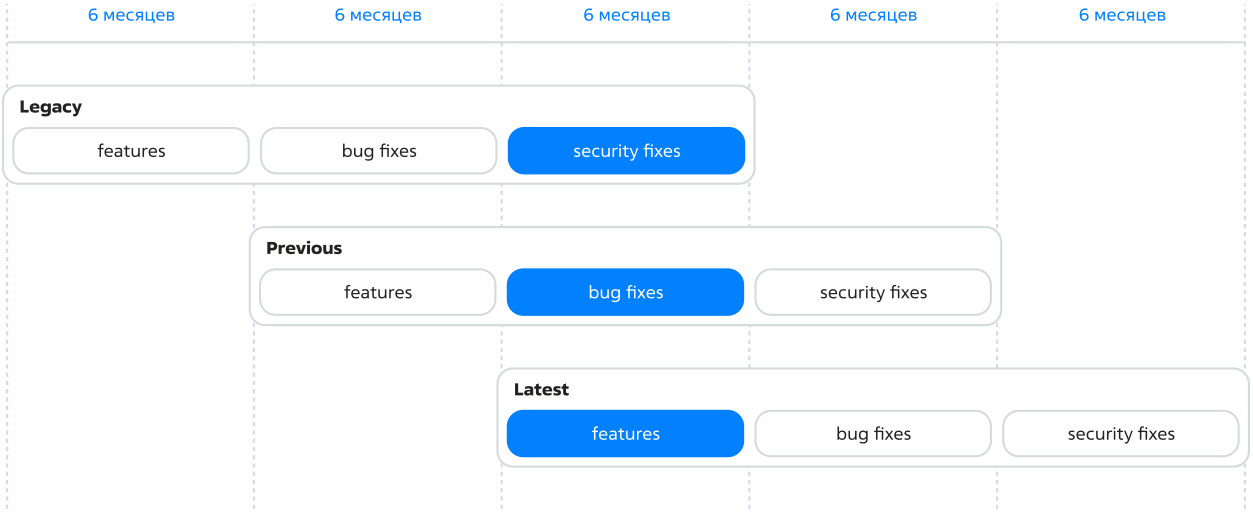
ГЛАВА 15

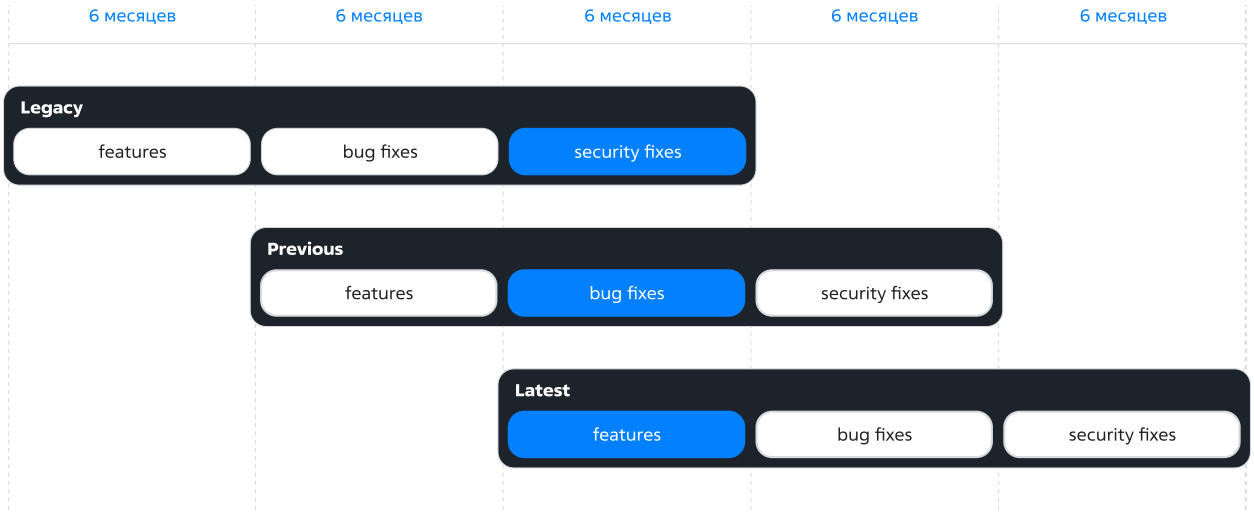
Обновления

Раздел содержит объявления о выпусках новых версий продукта Astra Automation, обновлениях его характеристик и исправлениях программного обеспечения.

15.1 Жизненный цикл продукта

Выпуск новой стабильной версии продукта происходит каждые 6 месяцев согласно следующей диаграмме:





Особенности:

- Одновременно поддерживаются три стабильные версии.
- В течение жизненного цикла каждой стабильной версии проводятся регулярные обновления.
- Жизненный цикл каждой стабильной версии содержит три стадии:
  - **features** – допустимы обновления функциональности, исправления ошибок и повышение безопасности.
  - **bug fixes** – допустимы только исправления ошибок и повышение безопасности.
  - **security fixes** – допустимы только обновления, связанные с повышением безопасности.

15.2 Версионирование

Версия выпуска Astra Automation представляется в соответствии со следующей схемой:

```
<field1>.<field2>-<update>
```

где:

- <field1> и <field2> – целые числа, вместе составляющие номер стабильной версии.

  Примечание: Обновления не основаны на семантическом версионировании.

- <update> – строка вида upd1, upd2 и так далее, означающая внесение изменений и исправление ошибок. При выпуске новой стабильной версии это поле отсутствует.

15.3 Цикл разработки документации

Разработка документации синхронизирована с разработкой продукта и имеет следующие особенности:

- Версии документации совпадают со стабильными версиями продукта, то есть имеют значения 1.0, 1.1 и так далее с периодом выпуска очередной версии 6 месяцев.
- Одновременно поддерживаются до трех последних версий документации с возможностью переключения между ними.
- С выходом минорного обновления продукта, например, 1.0-upd1 версия документации не изменяется. Новые изменения, которые появляются в документации в связи с выходом такого обновления, помечаются в виде короткой заметки в тексте сразу после соответствующего заголовка или параграфа.

15.4 Состав платформы

Следующая таблица содержит состав Astra Automation и зависимость от операционной системы и ее версии. Данные представлены в обратном хронологическом порядке.

Версия	Состав	Операционная система
1.0-upd2	Контроллер 1.0.2 Execution Environment (aa-base-ee) 0.5.1	Astra Linux Special Edition 1.7.4 Astra Linux Special Edition 1.7.5
1.0-upd1	Контроллер 1.0.1 Execution Environment (aa-base-ee) 0.2.1	Astra Linux Special Edition 1.7.4 Astra Linux Special Edition 1.7.5
1.0	Контроллер 1.0.0 Execution Environment (aa-base-ee) 0.2.1	Astra Linux Special Edition 1.7.4 (*см. особенности*) Astra Linux Special Edition 1.7.5 (*см. особенности*)

15.5 История обновлений

Следующая таблица содержит перечень обновлений в обратном хронологическом порядке и основные данные по ним.

Версия	Дата выпуска	Обновление характеристик (главное)	Исправления (главные)
1.0-upd2	22.04.202	Обновление кода из проекта AWX. Улучшена функция обновления версии платформы с помощью утилиты `aa-setup`.	Исключен излишний процесс сканирования всего репозитория при развертывании или обновлении платформы.
1.0-upd1	14.03.202	Добавлена возможность обновления версии платформы с помощью утилиты `aa-setup`.	Исправлен процесс восстановления базы данных кластера, состоящего из узлов типа `hybrid`.
1.0	15.02.202		

15.6 Стабильные версии

15.6.1 Версия 1.0

Дата выпуска: 15.02.2024
Тип выпуска: стабильная версия

Примечание: Первый выпуск продукта.

Основные характеристики

Astra Automation является комплексной платформой для автоматизации задач по развертыванию и управлению инфраструктур различной сложности.

Состав

Платформа Astra Automation состоит из следующих основных компонентов:

- Astra Automation Hub – облачный сервис, предоставляющий доступ к коллекциям Ansible и модулям Terraform, составляющим основу инфраструктурного кода для выполнения задач Astra Automation.
- Astra Automation Controller – контроллер (плоскость управления), предоставляющий удобный графический интерфейс и API для управления выполнением задач Astra Automation. Этот компонент наследует характеристики проекта AWX[189], адаптирует и развивает их для удовлетворения нужд заказчиков ПАО Группа Астра.
- Execution plane – сеть исполнительных узлов (плоскость исполнения), расширяющая возможности Astra Automation Controller по масштабированию и повышению уровня защиты структуры управления. Этот компонент позволяет создавать распределенную систему управления большими и сверхбольшими инфраструктурами, разделенными по географическому или другим принципам.

Функциональные возможности

Платформа позволяет развертывать инфраструктуру заказчика, устанавливать и настраивать программное обеспечение согласно *инструкции*.

Примеры сценариев:

- *Развертывание инфраструктуры в Yandex Cloud*
- *Установка сервера NGINX*
- *Установка ALD Pro*
- *Начало работы с контроллером*

Развертывание платформы

Развертыванию на стороне заказчика подлежат следующие компоненты:

- Astra Automation Controller можно установить в различных конфигурациях:
  - на одном узле для ознакомления и тестирования или для управления небольшой инфраструктурой;
  - в кластере, как основной рекомендуемый вариант.
- (Дополнительно) Компонент Execution plane необходим для создания распределенной структуры управления. Его устанавливают в едином процессе развертывания вместе с контроллером или при последующем обновлении.
- СУБД PostgreSQL необходимо установить на каком либо сервере в процессе развертывания платформы или использовать готовую базу данных.

[189] https://github.com/ansible/awx

Особенности установки и ограничения:

1. Установка первой версии платформы протестирована в Astra Linux Special Edition с использованием следующих версий:

Версия операционной системы	Версия ядра
1.7.4	5.15.0-generic
1.7.5	6.1.50-generic

2. При использовании Astra Linux Special Edition 1.7.5 необходимо исключить установку графической системы Fly и средств работы с графикой. Для развертывании кластера на виртуальных машинах рекомендуется использовать *готовые образы от производителя*, в названии которых отсутствует поле `gui`, например образ alse-vanilla-1.7.5-virtualbox-adv-mg12.1.2.ova[190] для VirtualBox.

   Примечание: Исправлено в обновлении *1.0-upd1*.

Преимущества

Astra Automation предоставляет ряд преимуществ по сравнению с решениями, доступными от свободно распространяемых продуктов:

- Безопасный проверенный код инструментария.
- Использование надежной операционной системы, оснащенной средствами защиты информации в соответствии с требованиями официальных регуляторов.
- Доступ к уникальному проверенному инфраструктурному коду, предназначенному для установки на операционной системе Astra Linux.
- Регулярные плановые обновления стабильных версий и внеплановые обновления текущих версий по запросам при обнаружении ошибок или уязвимостей.
- Резервное копирование и восстановление данных и настроек.
- Использование более совершенной системы журналирования, основанной на syslog-ng.
- Использование защищенного протокола SSL/TLS при работе с PostgreSQL.
- Документация на русском языке.
- Техническая поддержка.
- Обучение.

[190] https://download.astralinux.ru/artifactory/mg-generic/alse/virtualbox/alse-vanilla-1.7.5-virtualbox-adv-mg12.1.2.ova

Обновления стабильной версии

1.0-upd2

Дата выпуска: 22.04.2024
Тип выпуска: обновление стабильной версии

Главное

Следующие обновления представляются наиболее важными:

- Astra Automation Controller теперь базируется на AWX версии 23.7.0.
- Улучшена функция обновления версии платформы с помощью утилиты `aa-setup`.
- Утилита `aa-setup` позволяет указать образ среды исполнения при развертывании или обновлении платформы.
- Устранено ограничение на количество узлов в кластере Astra Automation.
- Обновлена версия образа среды исполнения.

Новые возможности и улучшения

Следующие секции содержат описание новых возможностей и улучшение существующих характеристик.

Переход на новую версию проекта AWX

Astra Automation Controller теперь базируется на новой версии upstream-проекта AWX. Произведен переход с версии 22.1.0[191] на версию 23.7.0[192]. Из большого количества обновлений наиболее заметными и значимыми являются следующие:

- В панели навигации пункт просмотра топологии доступен только для системных администраторов и аудиторов[193]. Ранее он был виден также для администраторов организаций, но, поскольку сама топология им недоступна, это приводило к появлению сообщения об ошибке.
- Обновлено расширение (plugin) credential_plugin[194] так, что теперь при работе с хранилищем vault поддерживается аутентификация через LDAP.
- Поддержка webhooks от Bitbucket Data Center[195] с отправлением обратно состояния соответствующего запроса.

Выполнен перевод на русский язык новых и обновленных строк в графическом интерфейсе контроллера.

[191] https://github.com/ansible/awx/releases/tag/22.1.0
[192] https://github.com/ansible/awx/releases/tag/23.7.0
[193] https://github.com/ansible/awx/pull/13905
[194] https://github.com/ansible/awx/pull/14654
[195] https://github.com/ansible/awx/pull/14674

Пакет astra-automation-setup

Выпущена версия 1.0.2 пакета, содержащая новые возможности:

- Улучшена функция обновления версии платформы с помощью утилиты aa-setup:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Новая функция автоматически определяет текущую версию платформы на всех компонентах кластера с помощью файла /etc/astra_automation.version, присутствующего в каждом узле кластера.
  - Пользователь, у которого есть административные полномочия на узлы кластера с доступом по SSH (используя приватный и публичный ключи SSH), может запускать процесс обновления, указав аргумент --upgrade. Утилиту можно запускать с узла, прошедшего *процесс подготовки*. Ранее процесс обновления необходимо было запускать с того узла, который использовался для первоначального развертывания платформы.
  - Функция доступна сразу после обновления пакета, но применить ее можно только с выпуском следующей версии платформы, предположительно при переходе с версии 1.0-upd2 на 1.0-upd3. Обновление на рассматриваемую версию подробно описано в секции *Обновление платформы до версии 1.0-upd2*.
- Утилита aa-setup позволяет указать образ среды исполнения, используемой по умолчанию для выполнения заданий:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Добавлен аргумент --default-job-ee, указывающий адрес требуемого образа.
  - Данная функция позволяет использовать требуемый образ среды исполнения вместо поставляемого в составе пакета.
  - Опция доступна сразу после установки или обновления пакета astra-automation-setup.

  Пример запуска команды:

```
sudo ./aa-setup --default-job-ee registry.astralinux.ru/aa/aa-base-ee:0.5.1
```

- Устранено ограничение на количество узлов в кластере Astra Automation:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Устранено ограничение на количество устанавливаемых узлов.
  - Позволяет создавать более сложную топологию платформы.
  - Доступно сразу после установки или обновления пакета astra-automation-setup.
- Утилита aa-setup проверяет корректность установки параметра pg_host:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Проверяет, что при отсутствии в описании инвентаря группы database (установка базы данных на локальном узле), параметру pg_host не назначено какое-либо значение. Допускается назначать адрес локального узла, то есть,

127.0.0.1. При нарушении этого условия утилита завершается аварийно с выводом сообщения об ошибке вида:

```
Параметр pg_host должен содержать пустое значение или '127.0.0.1' при␣
↪выполнении установки с внутренней базой данных (незаполненной секцией␣
↪[database]).
```

  - Эта функция повышает надежность развертывания платформы и сокращает время на выяснение возможных проблем.
  - Доступно сразу после установки или обновления пакета `astra-automation-setup`.

- Утилита `aa-setup` позволяет удалить ранее установленные компоненты платформы:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Добавлена функция удаления, вызываемая с помощью аргумента `-u` или `--uninstall`. Она удаляет все данные, установленные в процессе развертывания Astra Automation на узлах, входящих в состав файла инвентаризации.
  - Новая функция упрощает процесс удаления компонентов платформы.
  - Доступна сразу после установки или обновления пакета `astra-automation-setup`.

  Пример полного удаления всех компонентов Astra Automation из узлов, перечисленных в файле `inventory`:

```
aa-setup -i inventory -u --force-postgres-removal
```

Обновление спецификации API

Произведено обновление спецификации API на версию `1.0.2+23.7.0`.

Образ среды исполнения

Образ среды исполнения (EE, Execution Environment) `aa-base-ee`, используемый в платформе по умолчанию, обновлен до `0.5.1`:

- Предназначено для пользователей Astra Automation и специалистов по установке и настройке.
- Выполнены следующие обновления:
  - В основу положен образ Astra Linux Special Edition 1.7.5uu1 (registry.astralinux.ru/library/alse:1.7.5uu1-mg12.5.0).
  - Python и необходимые пакеты установлены из публичного репозитория Astra Automation.
  - Обновлены пакеты `ansible` и `ansible-core`.
  - Terraform заменен на OpenTofu[196].

[196] https://opentofu.org/

- Новая версия расширяет возможности EE по использованию необходимого программного обеспечения и новых возможностей Ansible. Она также позволяет отказаться от Terraform, на использование которого наложены ограничения от производителя.
- Образ доступен с момента выпуска данной версии Astra Automation.

Обновления коллекций Ansible

Произведено обновление следующих коллекций:

- Коллекция ceph[197] обновлена до версии `2.1.5` для обеспечения совместимости с EE версии `0.5.x`.
- Коллекция ald_pro[198] обновлена до версии `0.8.0` для поддержания ALD Pro версии `2.3.0`.
- Коллекция postgresql[199] обновлена до версии `2.2.1` для совместимости с операционной системой Astra Linux Special Edition версии `1.8`.

Исправление ошибок

В новой версии исправлены следующие ошибки:

- Исключен излишний процесс сканирования версий платформы в репозитории, когда версия платформы явно указана в качестве аргумента при запуске утилиты `aa-setup`.
- Устранена неисправность резервного копирования базы данных (dump), выражавшаяся в том, что прерывание процесса dump по какой-либо причине приводило к невозможности повторного запуска этого процесса.
- В структуре LOG_AGGREGATOR_SYSLOGNG_DEBUG, используемой для записи строк в журнал, исправлено название сервиса с `rsyslogd` на `syslog-ng`.

Обновление платформы до версии 1.0-upd2

Обновление с помощью утилиты `aa-setup`, описанное ранее, доступно уже в этой версии, но применить новые возможности можно будет только при переходе на будущую версию, то есть, предположительно, на `1.0-upd3` и последующие.

Поэтому переход от одной из предыдущих версий необходимо производить, используя возможности той версии:

- Переход с версии `1.0` на `1.0-upd2` необходимо выполнять так же, как и при *переходе с 1.0 на 1.0-upd1*.
- Переход с версии `1.0-upd1` на `1.0-upd2` позволяет использовать утилиту `aa-setup` из установочного пакета для версии `1.0-upd1`:

```
sudo ./aa-setup --upgrade --product-version 1.0-upd2
```

[197] https://hub.astra-automation.ru/aa-gca/ARFA/ceph
[198] https://hub.astra-automation.ru/aa-gca/ARFA/ald_pro
[199] https://hub.astra-automation.ru/aa-gca/ARFA/postgresql

Примечание: Есть временные ограничения:

- Вместе с аргументом `--upgrade` требуется использовать `--product-version` для явного указания версии Astra Automation.
- Обновление необходимо производить с того установочного узла, который использовался при первоначальном развертывании платформы. Вместо такого узла можно воспользоваться другим, на котором установлен пакет для развертывания Astra Automation и восстановлены необходимые настройки установочного узла.

При любом из перечисленных вариантов перехода необходимо использовать файл, описывающий действующий инвентарь платформы. По умолчанию он находится по адресу `/opt/rbta/aa/astra-automation-setup/inventory`.

1.0-upd1

Дата выпуска: 14.03.2024
Тип выпуска: обновление стабильной версии

Главное

Следующие обновления представляются наиболее важными:

- Добавлена функция автоматического обновления версии платформы с помощью утилиты `aa-setup`.
- Исправлен процесс восстановления базы данных кластера, состоящего из узлов типа `hybrid`.
- Исправлен процесс записи сообщений в журнал в формате JSON.

Новые возможности и улучшения

Следующие секции содержат описание новых возможностей и улучшение существующих характеристик.

Пакет astra-automation-setup

Выпущена версия 1.0.1 пакета, которая содержит новые возможности:

- Утилиту `aa-setup` теперь можно использовать для *обновления версии платформы*:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Добавлена опция `--upgrade` для обновления версии кластера.
  - Новая функция автоматически определяет текущую версию и обновляет ее до новейшей версии, доступной в репозитории Astra Automation.

  - Функция доступна сразу после обновления пакета, но применить ее можно только с выпуском следующей версии платформы, предположительно, `1.0-upd2`. Обновление на рассматриваемую версию подробно описано в секции *Обновление платформы до версии 1.0-upd1*.

- С помощью утилиты `aa-setup` можно узнать список версий платформы, доступных в репозиториях APT:
  - Предназначено для специалистов по установке и обслуживанию Astra Automation.
  - Добавлена опция `--check-releases`.
  - Данная опция позволяет инженерам узнать список версий платформы, доступных в главном репозитории (dl.astralinux.ru) или в репозитории, указанном с помощью аргумента `--repo-url`. Таким образом, они могут выбрать конкретную версию продукта для первичного развертывания или обновления.
  - Доступно сразу после установки или обновления пакета `astra-automation-setup`.

  Пример запуска команды:

```
sudo ./aa-setup --check-releases
```

  Пример вывода:

```
INFO: Astra Automation Setup has been started
INFO:

Following Astra Automation releases available:
mvp, mvp-upd1, mvp-upd2, 1.0, 1.0-upd1

To upgrade to the latest version, use:
./aa-setup --upgrade

Information about Astra Automation releases:
https://docs.astra-automation.ru/latest/releases/

INFO: Available releases checked
```

Исправление ошибок

В новой версии исправлены следующие ошибки:

- Исправлен процесс восстановления базы данных кластера, состоящего из узлов типа `hybrid`. Ранее при восстановлении данные не передавались из архива в базу данных.
- Исправлен процесс записи сообщений в журнал в формате JSON. Ранее формат записи JSON был искажен. Это, в частности, приводило к ошибкам интерпретации записей с помощью внешних обработчиков журналов, например, `logstash`.
- Исправлен счетчик затраченного времени при обработке задания. Ранее, при определенных условиях, он мог начинать отсчет от некоторого отрицательного значения.
- Устранено ограничение на наличие графических средств в узлах, на которых устанавливаются компоненты кластера платформы.

Обновление платформы до версии 1.0-upd1

Обновление с помощью утилиты aa-setup доступно уже в этой версии, но применить эту функцию можно будет только при переходе на будущую версию, то есть, предположительно, на 1.0-upd2 и последующие.

Поэтому обновление с версии 1.0 до 1.0-upd1 требует особой подготовки и последовательности.

Подготовка

Необходимо обеспечить выполнение следующих условий:

- Для обновления используйте тот же узел, с помощью которого происходило развертывание платформы версии 1.0, без внесения каких-либо изменений на нем.
- Убедитесь, что на этом узле остался в неизменном виде каталог /opt/rbta/aa/astra-automation-setup/, в частности файл инвентаря, описывающий состав платформы:

  /opt/rbta/aa/astra-automation-setup/inventory – описание структуры платформы.

Обновление

Обновление с версии 1.0 до 1.0-upd1 выполняйте в следующем порядке:

1. Добавьте ссылку на репозиторий, содержащий файлы Astra Automation, в файл /etc/apt/sources.list или же в любой файл в каталоге /etc/apt/sources.d/:

   ```
   deb https://dl.astralinux.ru/aa/aa-debs-for-alse-1.7 1.0-upd1 main
   ```

   Примечание: Убедитесь, что нет конфликтов, то есть, что этот репозиторий не объявлен в каком-либо другом из перечисленных файлов.

2. Обновите индекс пакетов с помощью следующей команды:

   ```
   sudo apt update
   ```

3. Обновите пакет astra-automation-setup с помощью следующей команды:

   ```
   sudo apt install --only-upgrade astra-automation-setup
   ```

4. Запустите процесс обновления таким же образом, как вы изначально развертывали платформу, то есть одним из следующих способов:

   - Запуск с использованием привилегий пользователя root:

     ```
     sudo ./aa-setup
     ```

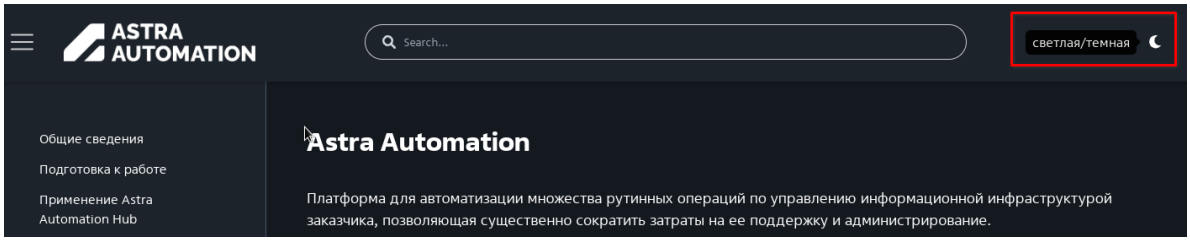
   - Запуск с привилегиями обычного пользователя:

     ```
     ./aa-setup --log-path=<log_path>
     ```

Изменения в документации

Добавлены новые возможности в документации:

- Добавлена поддержка темной темы:



- Изменен стиль для более заметного разграничения заголовков второго и третьего уровней.

ГЛАВА 16

Дополнительные материалы

Раздел содержит дополнительные материалы по продуктам, сервисам и инструментам для работы с Astra Automation.

16.1 Приложение 1. Устанавливаемые продукты

Важно: Эта часть документации находится в стадии разработки.

В данном разделе вы найдете общее описание продуктов, поддерживаемых платформой Astra Automation.

16.1.1 Образы виртуальных машин

Система управления образами операционных систем и программ предоставляет доступ к реестру, содержащему эти образы.

Классификация образов

Решения Astra Automation базируются на образах ВМ, в основе которых лежит операционная система Astra Linux. Все образы классифицируются по следующим признакам:

- название дистрибутива Astra Linux, на основе которого создан образ;
- состав пакетов и дополнительные настройки;
- поддержка графического интерфейса пользователя (GUI);
- версия ОС, на основе которой создан образ;
- целевая система виртуализации или облачный сервис;

- уровень защищенности ОС (только для Astra Linux Special Edition 1.7).

Дистрибутивы Astra Linux

При именовании образа для обозначения дистрибутива, на основе которого он собран, используются следующие обозначения:

- `smolensk` – Astra Linux Special Edition 1.6[200];
- `alse` – Astra Linux Special Edition 1.7[201];
- `orel` – Astra Linux Common Ediiton 2.12[202].

Состав пакетов и дополнительные настройки

По составу пакетов и дополнительным настройкам образы делятся на ванильный (`vanilla`) и базовый (`base`).

Ванильный образ имеет следующие особенности:

- Содержит только те пакеты и ПО, которые минимально необходимы для запуска в выбранном окружении.
- В конфигурации системы произведены только те настройки, которые необходимы для запуска.
- Включен сетевой интерфейс `eth0`, который настроен на получение сетевых параметров от сервера DHCP.

Базовый образ основан на ванильном, но содержит дополнительные настройки и пакеты:

- Подключен дополнительный репозиторий, включающий в себя пакеты, не вошедшие в дистрибутив.
- Время ожидания загрузчика GRUB равно `0` (по умолчанию – 5 секунд).
- Автодополнение команд в оболочке Bash по умолчанию включено.
- При входе в систему через терминал отображается расширенная информация о системе.
- В результатах работы команды **`history`** выводятся даты выполнения команд.

Поддержка графического интерфейса пользователя

Образ может включать пакеты, обеспечивающие поддержку графического интерфейса пользователя. В этом случае в названии образа присутствует слово `gui`.

[200] https://wiki.astralinux.ru/pages/viewpage.action?pageId=41192827
[201] https://wiki.astralinux.ru/pages/viewpage.action?pageId=158612043
[202] https://wiki.astralinux.ru/pages/viewpage.action?pageId=37290417

Версия ОС

Версия ОС, на основе которой собран образ, указывается в его названии. В номере версии ОС может быть указан номер установленного очередного оперативного обновления.

Система виртуализации или облачный сервис

Образы делятся на предназначенные для облачных сервисов и для локальных систем виртуализации. Образы для облачных сервисов уже загружены в их реестры образов. Образы для локальных систем виртуализации доступны для загрузки по ссылкам:

- https://download.astralinux.ru/images/
- https://dl.astralinux.ru/images/
- https://registry.astralinux.ru/

Поддерживаемые облачные сервисы:

- Cloud.ru;
- Selectel;
- VK Cloud;
- Yandex Cloud.

Поддерживаемые локальные системы виртуализации:

- Docker;
- QEMU
- VMmanager;
- VMware vSphere;
- Vagrant;
- ПК СВ «Брест».

Уровень защищенности ОС

Для образов на основе Astra Linux Special Edition 1.7 поддерживается дополнительный параметр – уровень защищенности ОС. В названии образа уровень защищенности обозначается следующим образом:

- base – базовый («Орел»);
- adv – продвинутый («Воронеж»);
- max – максимальный («Смоленск»).

Правила наименования

Все образы именуются по следующей схеме:

```
<distro>-<image_type>[-gui]-<version>-<image_env>-<security_level>-mg<builder>.<ext>
```

где:

- <distro> – название дистрибутива;
- <image_type> – тип образа по составу пакетов и количеству дополнительных настроек, `vanilla` или `base`;
- gui – если присутствует, то означает наличие пакетов, поддерживающих графическую оболочку;
- <version> – версия дистрибутива ОС;
- <image_env> – система виртуализации или облачный сервис;
- <security_level> – уровень защищенности ОС;
- mg<builder> – версия образа;
- <ext> – расширение файла.

Примеры:

- `alse-vanilla-gui-1.7.3uu2-virtualbox-base-mg9.2.0.ova`:
  - ОС – Astra Linux Special Edition 1.7;
  - тип образа – `vanilla`;
  - есть поддержка графического режима работы;
  - версия установленного оперативного обновления – `1.7.3uu2` (Бюллетень № 2023-0303SE17MD);
  - образ предназначен для системы виртуализации VirtualBox;
  - режим защищенности – базовый («Орел»);
  - версия образа – `9.2.0`.
- `orel-base-2.12.46-qemu-mg11.1.1.0.qcow2`:
  - ОС – Astra Linux Common Edition;
  - тип образа – `base`;
  - отсутствуют пакеты для поддержки графического режима;
  - версия установленного оперативного обновления – `2.12.46`;
  - образ предназначен для системы виртуализации QEMU;
  - версия образа – `11.1.1.0`.

Особенности образов

Все образы создаются на основе образов типа `base` и `vanilla`. При этом некоторые образы содержат изменения, обусловленные требованиями используемой системы виртуализации или облачного сервиса.

ПК СВ «Брест»

В образах для ПК СВ «Брест» дополнительно установлен пакет `one-context`.

Cloud.ru

Особенности образов для Cloud.ru[203]:

- Включена поддержка ACPI Power Event.
- В качестве серийной консоли используется `tty0`.
- Дополнительно установленные пакеты:
  - `cloud-guest-utils`;
  - `cloud-init`;
  - `dbus`;
  - `qemu-guest-agent`.
- По умолчанию используется локаль `en_US.UTF-8`.
- Время ожидания загрузчика GRUB равно `0` (по умолчанию – 5 секунд).
- cloud-init использует конфигурационный файл, подготовленный для окружения.
- Пользователю `root` разрешен вход по SSH.

В Cloud.ru доступны следующие образы Astra Linux:

- Astra Linux Special Edition 1.7 (режим защищенности «Воронеж»);
- Astra Linux Special Edition 1.7 (режим защищенности «Орел»).

Selectel

Особенности образов для Selectel[204].

- Включена поддержка ACPI Power Event.
- Дополнительно установлены пакеты:
  - `cloud-guest--utils`;
  - `cloud-init`;
  - `crontab-randomizer`;
  - `dbus`;

[203] https://cloud.ru/ru/
[204] https://selectel.ru/

  - `fstrim-blocks`;
  - `qemu-guest-agent`;
  - `set-root-pw`.
- Конфигурационный файл `/etc/network/interfaces` пуст, настройки сети определяются файлом `50-cloud-init.cfg`.
- По умолчанию используется локаль `en_US.UTF-8`.
- Время ожидания загрузчика GRUB равно `0` (по умолчанию – 5 секунд).
- cloud-init использует конфигурационный файл, подготовленный для окружения.
- Пользователю `root` разрешен вход по SSH.
- Размер диска равен 5 ГБ.

QEMU

В образах для QEMU дополнительно установлен пакет `qemu-guest-agent`.

VK Cloud

Особенности образов для VK Cloud[205]:

- Не используется GUI.
- Включена поддержка ACPI Power Event.
- В качестве серийной консоли используется `tty0`.
- Дополнительно установлены пакеты:
  - `cloud-guest-utils`;
  - `cloud-init`;
  - `qemu-guest-agent`.
- По умолчанию используется локаль `en_US.UTF-8`.
- Время ожидания загрузчика GRUB равно `0` (по умолчанию – 5 секунд).

VMware vSphere

Особенности образов для VMware vSphere:

- Добавлены утилиты из пакета OpenVM Tools.
- Значение параметра `Hardware Version` равно 11.

[205] https://cloud.vk.com/

VMmanager

Особенности образов для VMmanager[206].

- Дополнительное установлен пакет `qemu-guest-agent`.
- Образ распространяется в виде архива в формате `raw.xz`.

Vagrant

Образы для Vagrant предназначены только для использования совместно с VirtualBox.

В образах дополнительно установлен пакет расширений Virtual Box Guest Additions версии 6.1.38.

Пошаговые инструкции по настройке и использованию Vagrant в Astra Automation приведены в *соответствующем разделе*.

Yandex Cloud

Особенности образов для Yandex Cloud[207]:

- Включена поддержка ACPI Power Event.
- В качестве серийной консоли используется `tty0`.
- Дополнительно установлены пакеты:
  - `cloud-guest-utils`;
  - `cloud-init`;
  - `dbus`;
  - `qemu-guest-agent`.
- По умолчанию используется локаль `en_US.UTF-8`.
- Время ожидания загрузчика GRUB равно `0` (по умолчанию – 5 секунд).

Для получения списка образов, доступных вашему аккаунту, воспользуйтесь командой *Yandex CLI*:

```
yc compute image list
```

Список может выглядеть следующим образом:

```
+----------------------+----------------------------------------+--------+-------------
↪+--------+
|          ID          |                NAME                    | FAMILY | PRODUCT IDS␣
↪| STATUS |
+----------------------+----------------------------------------+--------+-------------
↪+--------+
| fd80b7qbr8kgkcmcvk4b | alse-vanilla-172-cloud-zero            |        |             ␣
↪| READY  |
| fd84mpqkpv8i4fa3h8qd | alse-vanilla-171-cloud-zero            |        |             ␣
```

(continues on next page)

[206] https://www.ispsystem.ru/
[207] https://yandex.cloud/ru/

(продолжение с предыдущей страницы)

```
↪| READY  |
| fd8anpqfr96a9maq4c92 | alse-vanilla-1-7-3-cloud-base-mg9-0-0 |        |           ␣
↪| READY  |
| fd8l2tbptcr5qpspikl2 | alse-vanilla-1-7-3-base-cloud-mg8-0-0 |        |           ␣
↪| READY  |
| fd8m51f8rpd9dobd1n21 | orel-vanilla-2-12-44-cloud-mg7-0-0    |        |           ␣
↪| READY  |
| fd8sghnc3ft7j4jso4tk | alse-vanilla-172-cloud-two            |        |           ␣
↪| READY  |
| fd8t2dhe0374sjn164c8 | alse-vanilla-172-cloud-one            |        |           ␣
↪| READY  |
+----------------------+---------------------------------------+--------+-------------
↪+--------+
```

Для получения более подробной информации об определенном образе введите следующую команду с указанием названия (name) или идентификатора (ID) образа:

```
yc compute image get alse-vanilla-1-7-3-cloud-base-mg9-0-0
```

Результат выполнения команды выглядит следующим образом (часть вывода опущена с целью сокращения):

```
id: fd8an...maq4c92
folder_id: b1gqs...evtdpu7
created_at: "2023-01-26T17:58:34Z"
name: alse-vanilla-1-7-3-cloud-base-mg9-0-0
description: alse-vanilla-1.7.3-cloud-base-mg9.0.0
storage_size: "1790967808"
min_disk_size: "3145728000"
status: READY
os:
type: LINUX
```

Подробности о командах YC CLI для работы с образами см. в документации Yandex Cloud[208].

В Yandex Cloud Marketplace доступны следующие образы Astra Linux:

- Astra Linux Special Edition 1.7 (режим защищенности «Воронеж»)[209];
- Astra Linux Special Edition 1.7 (режим защищенности «Орел»)[210].

16.1.2 Контейнеры

Образы контейнеров доступны в реестре, размещенном по адресу https://registry.astralinux.ru/.

Содержимое каталогов:

- `alse/` – образы Astra Linux Special Edition 1.7;
- `astra/` – универсальные базовые образы Astra Linux Special Edition 1.7 с набором предустановленного программного обеспечения для различных языков программирования и системы инициализации systemd;

[208] https://yandex.cloud/ru/docs/cli/cli-ref/managed-services/compute/image/
[209] https://yandex.cloud/ru/marketplace/products/astralinux/alse
[210] https://yandex.cloud/ru/marketplace/products/astralinux/alse-orel

- `orel/` – образы Astra Linux Common Edition 2.12.

16.1.3 ALD Pro

Важно: Эта часть документации находится в стадии разработки.

Структура

Параметры настройки

Методы проверки работоспособности

16.1.4 RuPost

Важно: Эта часть документации находится в стадии разработки.

RuPost[211] представляет собой систему управления электронной почтой.

Структура

Параметры настройки

Методы проверки работоспособности

16.2 Приложение 2. Облачные сервисы

16.2.1 ПК СВ «Брест»

Astra Automation предоставляет множество коллекций и модулей для автоматизации установки и настройки продуктов ПАО Группа Астра в ПК СВ «Брест»[212]. Перечень ПО, которое можно установить в этом комплексе средств виртуализации, представлен в разделе *Приложение 1. Устанавливаемые продукты*.

Ресурсная модель

В рамках Astra Automation чаще всего используются следующие ресурсы ПК СВ «Брест»:

- виртуальные сети;
- образы дисков;
- шаблоны виртуальных машин;
- экземпляры виртуальных машин.

Все создаваемые ресурсы размещаются внутри кластера ПК СВ «Брест».

[211] https://www.rupost.ru
[212] https://astragroup.ru/software-services/application-software-astra-group/brest/

Способы управления

В Astra Automation используются следующие способы управления ресурсами ПК СВ «Брест»:

- Веб-интерфейс.

  Этот способ удобен для получения информации о существующих ресурсах ПК СВ «Брест» и управления ими вручную.

- CLI.

  Этот способ удобен для создания сценариев автоматизации, однако, требует доступа к фронтальной машине от имени учетной записи администратора в сессии с высоким уровнем целостности.

- Terraform.

  ПК СВ «Брест» поддерживает провайдер Terraform для OpenNebula[213], позволяющий реализовать подход IaC (Infrastructure as Code, инфраструктура как код). Этот способ взаимодействия с ПК СВ «Брест» является рекомендуемым.

Необходимые привилегии

Для использования ПК СВ «Брест» при работе с Astra Automation необходимо иметь привилегии, позволяющие управлять следующими объектами:

- Виртуальные сети.
- Образы дисков.
- Шаблоны виртуальных машин.
- Экземпляры виртуальных машин.

Необходимые ресурсы

Для использования Astra Automation совместно с ПК СВ «Брест» необходимы:

- URI точки доступа (endpoint) к API;
- образы Astra Linux Special Edition;
- шаблоны ВМ;
- идентификатор виртуальной сети;
- идентификатор группы ВМ;
- токен входа;
- ключи SSH для доступа к стендам.

Пошаговые инструкции по получению данных и созданию необходимых ресурсов приведены далее.

[213] https://github.com/OpenNebula/terraform-provider-opennebula

Настройка ПК СВ «Брест»

Для работы с ресурсами ПК СВ «Брест» необходимо подготовить следующие данные:

1. URI точки доступа к API.
2. Образы Astra Linux Special Edition.
3. Шаблоны ВМ.
4. Идентификатор виртуальной сети.
5. Токен входа.
6. Ключи SSH для доступа к создаваемым стендам.

Все описанные ниже действия следует выполнять в веб-интерфейсе ПК СВ «Брест».

Предполагается, что комплекс средств виртуализации настроен согласно инструкций:

- РДЦП.10001-02 95 01-1 Программный комплекс «Средства виртуализации «БРЕСТ». Руководство администратора. Часть 1»
- РДЦП.10001-02 95 01-2 Программный комплекс «Средства виртуализации «БРЕСТ». Руководство администратора. Часть 2»

Определение точки доступа к API

URI точки доступа к API формируется следующим образом:

```
https://<brest>:8443/RPC2/
```

где <brest> – FQDN или IP-адрес фронтальной машины ПК СВ «Брест».

Загрузка установочных образов в хранилище ПК СВ «Брест»

Если у вас нет подготовленных образов Astra Linux Special Edition, выполните следующие действия для загрузки в ПК СВ «Брест» установочных образов Astra Linux Special Edition:

1. Из Личного кабинета[214] загрузите на фронтальную машину установочные образы Astra Linux Special Edition в формате ISO.

   Примечание: Образы доступны для загрузки при наличии действующей лицензии на продукт.

2. На панели навигации ПК СВ «Брест» выберите *Хранилище* ▸ *Образы*.
3. Нажмите кнопку + и в выпадающем меню выберите **Создать**.
4. Заполните форму **Образ**:
   - **Название**: произвольное название образа.

     Допускается использование букв латинского алфавита, цифр и дефиса.
   - **Тип**: **CD-ROM только для чтения**.
   - **Расположение образа**:

[214] https://lk-new.astralinux.ru

   a. Выберите значение **Закачать**.
   b. Нажмите кнопку *Browse* и укажите путь к файлу образа.
5. Нажмите кнопку *Создать*.
6. Дождитесь загрузки образа в хранилище. Загрузка считается выполненной, когда статус образа принимает значение `ГОТОВО`.

Создание образа диска

Образ диска необходим для создания шаблона ВМ. Чтобы создать образ диска, выполните следующие действия:

1. На панели навигации выберите *Хранилище* ‣ *Образы*.
2. Нажмите кнопку + и в выпадающем меню выберите **Создать**.
3. Заполните форму **Образ**:
   a. **Название**: произвольное название образа.

      Допускается использование букв латинского алфавита, цифр и дефиса.
   b. **Тип**: **Общий блок данных хранилища**.
   c. **Этот образ является постоянным**: **Да**.
   d. **Расположение образа**: **Пустой образ диска**.
   e. **Размер**: размер диска по умолчанию.

      Рекомендуемое значение – не менее 20 ГБ.
4. Нажмите кнопку *Создать*.

Дождитесь создания образа. Создание считается завершенным, когда статус образа принимает значение `ГОТОВО`.

Создание шаблона ВМ

Чтобы создать шаблон ВМ, выполните следующие действия:

1. На панели навигации выберите *Шаблоны* ‣ *ВМ*.
2. Нажмите кнопку + и в выпадающем меню выберите **Создать**.
3. Заполните форму **Создать шаблон ВМ**:
   a. Заполните вкладку **Общие**:
      - **Название**: произвольное название шаблона ВМ.

        Допускается использование букв латинского алфавита, цифр и дефиса.
      - **Гипервизор**: `KVM`.
      - **Память**: объем памяти, доступный создаваемым ВМ.

        Рекомендуемое значение – не менее 2 ГБ.
      - **Физический CPU**: укажите значение не ниже `0.5`.

- **Виртуальный CPU**: количество ядер, доступных ВМ.

  Рекомендуемое значение – не менее 2.

b. Заполните вкладку **Хранилище**:

- **ДИСК0**:
  1. Выберите значение **Образ**.
  2. Из списка образов выберите созданный ранее образ диска типа «Блок данных».
- В списке дисков нажмите кнопку +.
- **ДИСК1**:
  1. Выберите значение **Образ**.
  2. Из списка образов выберите образ установочного диска Astra Linux Special Edition.

4. Нажмите кнопку *Создать*.

Определение идентификатора виртуальной сети

По умолчанию при развертывании ПК СВ «Брест» создается виртуальная сеть `virtnetwork` с идентификатором `0`. Если для Astra Automation планируется использовать другую виртуальную сеть, для получения ее идентификатора на панели навигации выберите *Сеть ▸ Вирт. сети*. Идентификаторы виртуальных сетей указаны в колонке **ID**.

Создание токена входа

Чтобы сформировать токен входа, выполните следующие действия:

1. Нажмите кнопку с именем активного пользователя и в выпадающем меню выберите **Настройки**.
2. Выберите вкладку **Аутентификация**.
3. Нажмите кнопку *Управление токенами входа*.
4. В форме **Токен входа** заполните поле **Истечение, в секундах**, в котором укажите срок действия токена.
5. Нажмите кнопку *Получить новый токен*.
6. Значение из колонки **Токен** сохраните в надежном месте.
7. Закройте форму **Токен входа**.

Настройка SSH

ПК СВ «Брест» позволяет подключаться к созданным ВМ по протоколам RDP и VNC, однако, для использования Astra Automation требуется доступ к ВМ по SSH. Чтобы настроить его, выполните следующие действия:

1. Нажмите кнопку с именем активного пользователя и в выпадающем меню выберите **Настройки**.
2. Выберите вкладку **Аутентификация**.
3. В разделе **Публичный SSH ключ** нажмите кнопку редактирования.
4. В открывшемся поле введите содержимое публичного ключа, *используемого для доступа к стендам*.
5. Нажмите кнопку редактирования еще раз.

16.2.2 Cloud.ru

Astra Automation позволяет развертывать продукты ПАО Группа Астра в Cloud.ru. Перечень ПО, поддерживающего развертывание в Cloud.ru, представлен в разделе *Приложение 1. Устанавливаемые продукты*.

Ресурсная модель

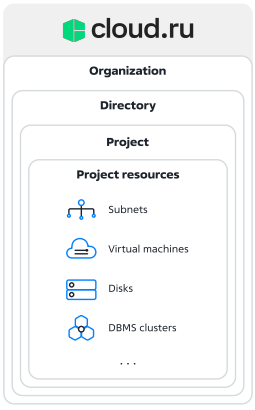
В рамках Astra Automation чаще всего используются следующие ресурсы Cloud.ru:

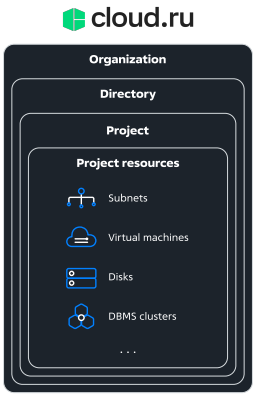
- виртуальные машины;
- сети;
- подсети.

Все создаваемые ресурсы размещаются внутри проектов. Каждый проект размещается в определенном каталоге. Каждый каталог принадлежит определенной организации.

Примечание: При необходимости можно создать дополнительные проекты.

Упрощенная структура Cloud.ru показана на схеме:





Способы управления

В Astra Automation используются следующие способы управления ресурсами Cloud.ru:

- Личный кабинет.

  Личный кабинет доступен по адресу https://console.cloud.ru/ и предоставляет веб-интерфейс для управления ресурсами облака. Этот способ удобен для получения информации о существующих ресурсах Cloud.ru и управления ими вручную.

- Terraform.

  Cloud.ru поддерживает провайдер Terraform, позволяющий реализовать подход IaC (Infrastructure as Code, инфраструктура как код). Этот способ взаимодействия с Cloud.ru является рекомендуемым.

Необходимые привилегии

Для использования Cloud.ru при работе с Astra Automation необходимо иметь роли, назначенные на организацию или проект и обеспечивающие выполнение следующих операций:

- Создание сервисных аккаунтов и назначение им ролей.
- Создание ключей доступа.

  Ключи доступа используются для управления ресурсами Cloud.ru от имени сервисного аккаунта.

- Управление сетями и подсетями.
- Управление виртуальными машинами.
- Создание бакетов в Объектном хранилище, загрузка и выгрузка объектов из бакетов.
- Импорт ключей SSH.

Предупреждение: Если в организации или проекте не существует сервисный аккаунт с необходимыми ролями, для его создания и назначения ему ролей потребуется учетная запись администратора или пользователя, входящего в группу `admin`.

Необходимые ресурсы

Для использования Astra Automation совместно с Cloud.ru необходимы:

- Права на управление ресурсами Cloud.ru от имени любой из указанных учетных записей:
  - собственная учетная запись;
  - учетная запись организации;
  - сервисный аккаунт[215].
- Организация и ее идентификатор.
- Каталог и его идентификатор.
- Проект и его идентификатор.

  В этом проекте размещаются ВМ, используемые для развертывания необходимого ПО.
- (опционально) Бакет Объектного хранилища S3 для хранения объектов состояния (state) инфраструктуры, развернутой с помощью Terraform.

  Если планируете использовать бакет Объектного хранилища S3, необходимы также:
  - адрес для подключения к бакету (endpoint);
  - Access Key ID;
  - Secret Key.

Пошаговые инструкции по получению данных и созданию необходимых ресурсов приведены далее.

Настройка сервисного аккаунта

Для работы с ресурсами Cloud.ru можно использовать учетную запись обычного пользователя, однако, для средств автоматизации лучше подходит сервисный аккаунт.

Для подготовки сервисного аккаунта Cloud.ru к работе с Astra Automation выполните следующие действия:

1. Убедитесь, что сервисный аккаунт существует и ему назначены необходимые роли.
2. Назначьте созданному сервисному аккаунту необходимые роли.
3. Создайте ключ доступа к ресурсам проекта.

[215] https://cloud.ru/ru/docs/console/ug/topics/guides__service_accounts.html

Проверка наличия необходимых ролей

Для управления ресурсами сервисному аккаунту должны быть назначены роли, дающие необходимые привилегии.

Примечание: Роль Администратор дает привилегии на создание, изменение и удаление любых типов ресурсов в проекте, однако, в большинстве случае столь широкие полномочия избыточны. Для более тонкой настройки ознакомьтесь со списком стандартных ролей[216], предоставляемых Cloud.ru, и назначьте сервисному аккаунту только те роли, которые действительно необходимы.

Если сервисный аккаунт уже существует, для проверки наличия у него необходимых ролей используйте Личный кабинет пользователя Cloud.ru.

Создание сервисного аккаунта

Если сервисный аккаунт не существует, создайте его согласно инструкции[217].

Назначение ролей сервисному аккаунту

Чтобы назначить сервисному аккаунту роль на проект, воспользуйтесь инструкцией[218].

Создание ключа доступа

Чтобы создать для сервисного аккаунта ключ доступа, воспользуйтесь инструкцией[219].

16.2.3 Yandex Cloud

Astra Automation позволяет развертывать продукты ПАО Группа Астра в Yandex Cloud. Перечень ПО, поддерживающего развертывание в Yandex Cloud, представлен в разделе *Приложение 1. Устанавливаемые продукты*.

Ресурсная модель

В рамках Astra Automation чаще всего используются следующие ресурсы Yandex Cloud:

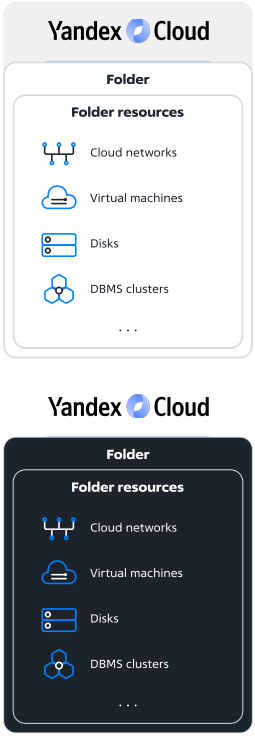
- облачные сети и подсети[220];
- виртуальные машины Yandex Compute Cloud[221];
- бакеты Yandex Object Storage[222].

[216] https://cloud.ru/ru/docs/iam/ug/topics/overview__roles-and-policies-list.html
[217] https://cloud.ru/ru/docs/console/ug/topics/guides__service_accounts_create.html
[218] https://cloud.ru/ru/docs/console/ug/topics/guides__employees_edit.html
[219] https://cloud.ru/ru/docs/console/ug/topics/guides__service_accounts_key.html
[220] https://yandex.cloud/ru/docs/vpc/concepts/network/
[221] https://yandex.cloud/ru/docs/compute/concepts/vm/
[222] https://yandex.cloud/ru/docs/storage/

Все создаваемые ресурсы размещаются внутри облачных каталогов. Каждый облачный каталог является частью облака. При создании облака в нем автоматически создается облачный каталог с именем `default`.

Примечание: При необходимости можно создать дополнительные облачные каталоги.

Упрощенная структура Yandex Cloud показана на схеме:



Способы управления

В Astra Automation используются следующие способы управления ресурсами Yandex Cloud:

- Консоль управления.

  Консоль управления доступна по адресу https://console.yandex.cloud/ru/ и предоставляет веб-интерфейс для управления ресурсами облака. Этот способ удобен для получения информации о существующих ресурсах Yandex Cloud и управления ими вручную.

- Клиент командной строки.

  YC CLI позволяет управлять облачными ресурсами с помощью утилиты командной строки. Этот способ удобен при использовании сценариев оболочки командной строки.

  Все дальнейшие инструкции предполагают использование YC CLI для выполнения необходимых настроек.

- Terraform.

Yandex Cloud поддерживает провайдер Terraform, позволяющий реализовать подход IaC. Этот способ взаимодействия с Yandex Cloud является рекомендуемым.

Необходимые привилегии

Для использования Yandex Cloud при работе с Astra Automation необходимо иметь роль editor[223] на облачный каталог, либо роли, обеспечивающие выполнение следующих операций:

- Создание сервисных аккаунтов и назначение им ролей.

  Эта привилегия необходима, если сервисные аккаунты с нужными ролями на каталог не существуют.

- Создание авторизованных и AWS-совместимых статических ключей доступа.

  Эта привилегия необходима для настройки доступа сервисного аккаунта к ресурсам Yandex Cloud и бакетам Yandex Object Storage.

- Управление облачными сетями и подсетями.
- Управление виртуальными машинами Yandex Compute Cloud.
- Создание бакетов Yandex Object Storage, загрузка и выгрузка объектов из них.

Предупреждение: Если в облачном каталоге не существует сервисный аккаунт с необходимыми ролями, для его создания и назначения ролей потребуется роль admin[224].

Необходимые ресурсы

Для использования Astra Automation совместно с Yandex Cloud необходимы:

- Права на управление ресурсами Yandex Cloud от имени любой из указанных учетных записей:
  - собственная учетная запись;
  - учетная запись организации;
  - сервисный аккаунт[225].
- Облако и его идентификатор.
- Облачный каталог и его идентификатор. В этом каталоге размещаются ВМ, используемые для развертывания необходимого ПО.
- Сервисный аккаунт Yandex Cloud. Этот аккаунт позволяет средствам Astra Automation иметь доступ к Yandex Cloud.
- Файл формата JSON, содержащий авторизованные ключи. Используется при управлении облачными ресурсами с использованием сервисного аккаунта.

[223] https://yandex.cloud/ru/docs/iam/concepts/access-control/roles#editor
[224] https://yandex.cloud/ru/docs/iam/concepts/access-control/roles#admin
[225] https://yandex.cloud/ru/docs/iam/concepts/users/service-accounts

- (опционально) Бакет Yandex Object Storage для хранения объектов состояния (state) инфраструктуры, развернутой с помощью Terraform.

  Если планируете использовать бакет Yandex Object Storage, необходимы также:

  - имя бакета;
  - статический ключ доступа для сервисного аккаунта.

Пошаговые инструкции по получению данных и созданию необходимых ресурсов приведены далее.

Настройка интерфейса командной строки (YC CLI)

Платформа Yandex Cloud предоставляет возможность управления облачными ресурсами с помощью *YC CLI*. Основу клиентской части составляет утилита yc. Все дальнейшие инструкции по использованию Yandex Cloud совместно с Astra Automation предполагают использование этого инструмента для работы с облаком.

Установка и настройка интерфейса командной строки YC CLI

Детальная информация о начале работы в командной строке приведена в документации Yandex Cloud[226].

Чтобы начать пользоваться YC CLI, выполните следующие действия:

1. Установите клиентскую часть:

   ```
   curl -sSL https://storage.yandexcloud.net/yandexcloud-yc/install.sh | bash
   ```

   Эта команда выполняет следующие действия:

   - создает каталог `yandex-cloud/` в домашнем каталоге пользователя;
   - дополняет файл инициализации `.bashrc` вызовом скриптов настройки клиента – `~/yandex-cloud/path.bash.inc` и `~/yandex-cloud/completion.bash.inc`.

2. Перезапустите терминал или выполните команду:

   ```
   source ~/.bashrc
   ```

3. Создайте профиль пользователя для работы с облачным каталогом.

   Примечание: Профиль определяет набор параметров окружения, в котором пользователь работает в каждый конкретный момент.

   При использовании федеративного метода идентификации на базе SAML выполните команду:

   ```
   yc init --federation-id=<federation_ID>
   ```

   где <federation_ID> – идентификатор федерации[227] для вашей организации или подразделения.

[226] https://yandex.cloud/ru/docs/cli/
[227] https://yandex.cloud/ru/docs/organization/concepts/add-federation#saml-authentication

Необходимо выбрать – использовать существующий каталог или создать новый.

4. (Опционально) Настройте зону доступности по умолчанию.
5. Проверьте полученные настройки с помощью команды:

```
yc config list
```

В терминал выводятся идентификаторы федерации, облака и каталога, используемых в текущий момент.

6. Проверьте список профилей:

```
yc config profile list
```

В терминал выводится список существующих профилей YC CLI. Активный профиль будет отмечен меткой ACTIVE.

Управление профилями

Дополнительные профили YC CLI могут понадобиться, например:

- для работы с разными облаками и каталогами Yandex Cloud;
- для управления ресурсами одного и того же каталога от имени разных пользователей, в том числе сервисных аккаунтов.

Чтобы создать дополнительный профиль YC CLI, выполните следующие действия:

1. Получите идентификаторы нужного облака и облачного каталога любым удобным способом. Чтобы получить сведения об облаке и каталоге, к которым привязан активный профиль YC CLI, выполните команду:

```
yc config profile get <profile>
```

В терминал выводится следующая информация о профиле:

- `cloud-id` – идентификатор облака;
- `folder-id` – идентификатор облачного каталога;
- `compute-default-zone` – зона доступности по умолчанию (если была указана при настройке профиля).

2. Выполните команду создания нового профиля:

```
yc config profile create <profile_name>
```

Профиль автоматически активируется после создания.

3. Привяжите профиль к нужному облаку и каталогу:

```
yc config set cloud-id <cloud_ID>
yc config set folder-id <folder_ID>
```

где <cloud_ID> и <folder_ID> – полученные ранее идентификаторы облака и облачного каталога соответственно.

Совет: Полный перечень команд YC CLI приведен в документации Yandex Cloud[228].

Примеры использования

Получение списка доступных профилей:

```
yc config profile list
```

Переключение на другой профиль:

```
yc config profile activate <profile>
```

Назначение зоны доступности по умолчанию:

```
yc config set compute-default-zone <zone>
```

где <zone> – название зоны доступности, например:

```
yc config set compute-default-zone ru-central1-a
```

Подробности о зонах доступности см. в документации Yandex Cloud[229].

Настройка сервисного аккаунта

Для работы с ресурсами Yandex Cloud можно использовать учетную запись обычного пользователя, однако, для средств автоматизации лучше подходит сервисный аккаунт.

Для подготовки сервисного аккаунта Yandex Cloud к работе с Astra Automation выполните следующие действия:

1. Если клиент командной строки Yandex Cloud не настроен, следуйте инструкциям из раздела *Установка и настройка интерфейса командной строки YC CLI*.
2. Убедитесь, что сервисный аккаунт существует и ему назначены необходимые роли.
3. Создайте ключ доступа к ресурсам каталога.
4. (Опционально) Создайте статический ключ доступа к бакету Yandex Object Storage.

Проверка наличия необходимых ролей

Для управления ресурсами сервисному аккаунту должны быть назначены роли, дающие необходимые привилегии.

Примечание: Роль `editor` дает привилегии на создание, изменение и удаление любых типов ресурсов в каталоге, однако, в большинстве случаев столь широкие полномочия

[228] https://yandex.cloud/ru/docs/cli/cli-ref/
[229] https://yandex.cloud/ru/docs/overview/concepts/geo-scope

избыточны. Для более тонкой настройки ознакомьтесь со списком всех ролей[230], предоставляемых Yandex Cloud, и назначьте сервисному аккаунту только те роли, которые действительно необходимы.

Если сервисный аккаунт уже существует, убедитесь, что ему назначены роли, позволяющие управлять необходимыми типами ресурсов:

```
yc resource-manager folder list-access-bindings <folder_name|folder_ID> | grep <sa_ID>
```

В терминал выводится таблица, отображающая список ролей, назначенных сервисному аккаунту с указанным идентификатором <sa_ID>.

```
+------------------+----------------+----------------------+
|     ROLE ID      |  SUBJECT TYPE  |      SUBJECT ID      |
+------------------+----------------+----------------------+
| storage.uploader | serviceAccount | a..................1 |
| editor           | serviceAccount | a..................1 |
+------------------+----------------+----------------------+
```

Создание сервисного аккаунта

Если сервисный аккаунт не существует, создайте его:

```
yc iam service-account create \
   --name=<sa_name> \
   --description="<description>"
```

где

- <sa_name> – имя сервисного аккаунта. Должно удовлетворять следующим требованиям:
  - длина от 3 до 63 символов;
  - может содержать только строчные буквы латинского алфавита, цифры и дефисы;
  - первый символ – буква;
  - последний символ – не дефис;
  - имя должно быть уникальным в рамках облака.
- <description> – опциональное описание сервисного аккаунта.

Подробности о создании сервисных аккаунтов см. в документации Yandex Cloud[231].

[230] https://yandex.cloud/ru/docs/iam/concepts/access-control/roles
[231] https://yandex.cloud/ru/docs/iam/quickstart-sa#create-sa

Назначение ролей сервисному аккаунту

Чтобы назначить сервисному аккаунту роль на каталог, выполните команду:

```
yc resource-manager folder add-access-binding \
   --name <folder_name> \
   --role <role_name> \
   --service-account-name <sa_name>
```

где

- <folder_name> – имя облачного каталога;
- <role_name> – имя роли;
- <sa_name> – имя сервисного аккаунта.

Подробности о назначении ролей сервисным аккаунтам см. в документации Yandex Cloud[232].

Создание пары авторизованных ключей доступа

Чтобы создать пару авторизованных ключей доступа к облачному каталогу и сохранить ее в файл формата JSON, выполните команду:

```
yc iam key create \
   --service-account-name <sa_name> \
   --output key.json
```

где

- <sa_name> – имя сервисного аккаунта;
- `key.json` – имя файла для сохранения пары ключей.

При успешном выполнении команды в терминал выводится информация о созданной паре ключей, например:

```
id: ajonetwothreefoursp63
service_account_id: aj**************o3
created_at: "2023-03-21T08:13:19.002945823Z"
key_algorithm: RSA_2048
```

Пример содержимого файла `key.json` (ключи представлены в сокращенном виде):

```
{
   "id": "ajonetwothreefoursp63",
   "service_account_id": "aj**************o3",
   "created_at": "2023-03-21T08:13:19.002945823Z",
   "key_algorithm": "RSA_2048",
   "public_key": "-----BEGIN PUBLIC KEY-----\nMIIBI...DAQAB\n-----END PUBLIC KEY-----\
↪n",
   "private_key": "-----BEGIN...END PRIVATE KEY-----\n"
}
```

Подробности о создании авторизованных ключей доступа см. в документации Yandex Cloud[233].

[232] https://yandex.cloud/ru/docs/iam/operations/sa/assign-role-for-sa
[233] https://yandex.cloud/ru/docs/iam/operations/authorized-key/create

Создание AWS-совместимого статического ключа доступа

Примечание: Создавайте AWS-совместимый статический ключ доступа в том случае, если планируете использовать бакет Yandex Object Storage для хранения объектов состояния Terraform.

Чтобы создать AWS-совместимый статический ключ доступа для сервисного аккаунта, выполните команду:

```
yc iam access-key create --service-account-name <sa_name>
```

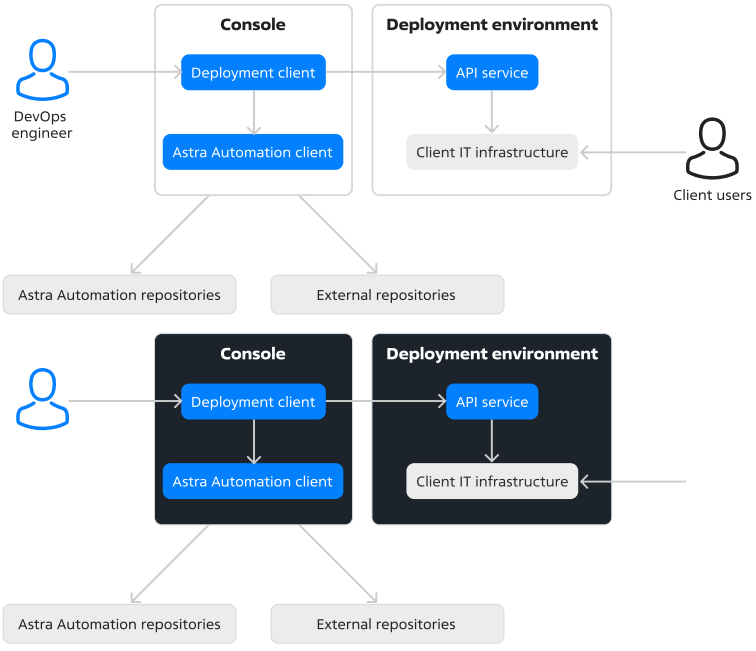
При успешном выполнении команды в терминал выводится информация о созданном ключе, например:

```
access_key:
  id: aj**************ml
  service_account_id: aj**************o3
  created_at: "2023-08-16T10:52:39.234764687Z"
  key_id: YC********************-
secret: YC************************s
```

Сохраните значения полей `key_id` и `secret` – они понадобятся в дальнейшем.

Подробности о создании AWS-совместимых статических ключей доступа для сервисных аккаунтов см. в документации Yandex Cloud[234].

Типовая схема развертывания облачной инфраструктуры:



[234] https://yandex.cloud/ru/docs/iam/operations/sa/create-access-key

16.3 Приложение 3. Инструментарий

Для реализации своих функций Astra Automation использует различные методы, инструменты и системы, описания которых приводятся в следующих разделах.

16.3.1 Ansible

Astra Automation использует Ansible как основной инструмент для установки и настройки систем различной сложности.

Назначение

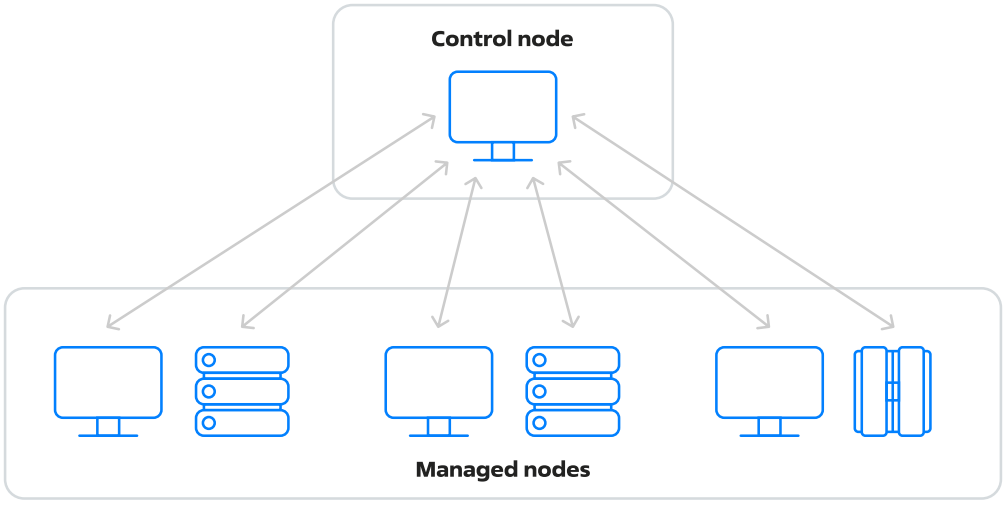
Ansible[235] – это система управления развертыванием и настройкой программного обеспечения. Система выполняет заданные директивы на указанных серверах, подключаясь к ним по протоколу SSH. Для этого необходимы только ключи SSH. Установка агентов на управляемые узлы не требуется.

Базовая функциональность Ansible обеспечивается большим количеством встроенных модулей. При этом возможности Ansible можно расширять с помощью следующих компонентов:

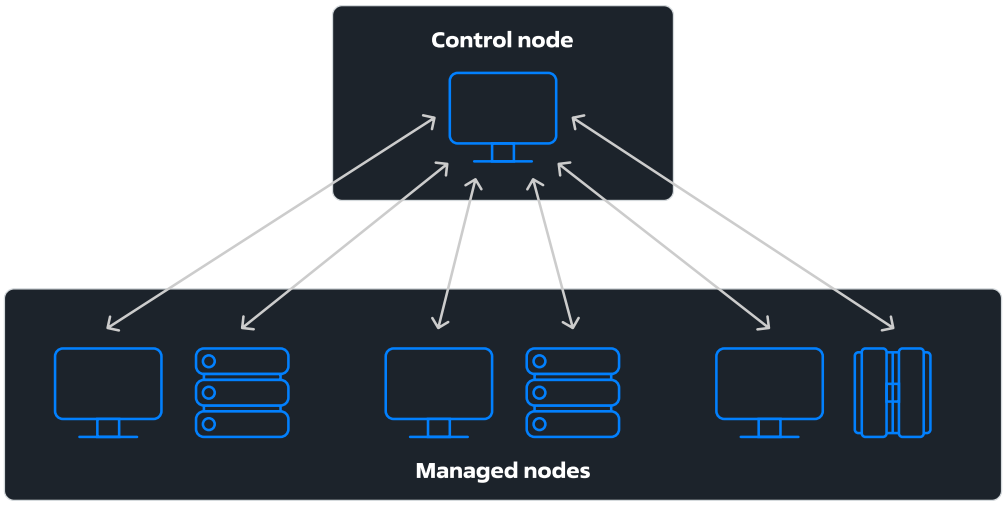
- расширения (plugins);
- модули (modules);
- роли (roles);
- коллекции (collections).

Структура управления

При использовании Ansible управляющий узел подключается к управляемым узлам и запускает на них команды настройки.



[235] https://www.ansible.com/



Управляющий узел

Управляющим узлом (control node) называется компьютер, на который устанавливают компоненты Ansible и который используют для запуска заданий по настройке управляемых узлов.

Подробности об управляющем узле см. в документации Ansible[236].

Управляемые узлы

Управляемым узлом (managed node) называется компьютер, настраиваемый с помощью Ansible.

Подробности об управляемых узлах см. в документации Ansible[237].

Инвентарь

Инвентарь – это файл формата `INI` или `YAML`, в котором хранятся данные об управляемых узлах.

В самом простом случае достаточно указать для каждого управляемого узла только его IP-адрес:

```
192.168.56.11
192.168.56.12
192.168.56.13
```

Также допускается использовать доменные имена, например:

```
dc01.example.com
dc02.example.com
dc03.example.com
```

[236] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#control-node
[237] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#managed-nodes

В этом случае разрешение доменного имени каждого узла в IP-адрес происходит путем обращения к системной службе разрешения имен. В Astra Linux настройки этой службы хранятся в файле `/etc/nsswitch.conf`. По умолчанию в нем указан следующий порядок разрешения имен:

1. Проверка записей, хранящихся в файле `/etc/hosts`.
2. Обращение к серверу DNS, указанному в настройках сети.

Чтобы сделать инвентарь не зависящим от службы разрешения имен, для каждого узла всегда явно указывайте его IP-адрес в поле `ansible_host`, например:

```
dc01.example.com   ansible_host=192.168.56.11
dc02.example.com   ansible_host=192.168.56.12
dc03.example.com   ansible_host=192.168.56.13
```

Подробности о файлах инвентаря см. в документации Ansible[238].

Иерархия компонентов playbook

Иерархия исполняемых компонентов в скриптах Ansible:

- *Playbook* (набор сценариев) состоит из plays (сценариев), использующих инвентарный список, который указывается при запуске playbook с помощью параметра `-i` или в конфигурационном файле `ansible.cfg`. Утилита `ansible-playbook` начинает выполнение playbook.
- *Play* (сценарий) состоит из множества задач, выполняемых для указанных узлов и групп узлов из определенного инвентарного списка.
- *Task* (задача) использует один из модулей Ansible для выполнения отдельной операции по настройке управляемого узла.
- *Module* (модуль) реализует требуемую функциональность. Например, существуют модули для копирования файлов, управления пакетами и службами и т. д. Код модулей выполняется на управляемых узлах.
- *Plugin* (расширение) расширяет функциональность Ansible. В отличие от модулей, код расширений выполняется на управляющем узле.

Задача

Задача (task) – это минимальный шаг по настройке управляемого узла.

Подробности о задачах см. в документации Ansible[239].

[238] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#inventory
[239] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#tasks

Сценарий

Сценарий (play) – это одна или несколько задач, выполняемых на указанных управляемых узлах.

Подробности о сценариях см. в документации Ansible[240].

Playbook

Playbook – это файл формата YAML, содержащий перечень сценариев по настройке управляемых узлов.

Подробности о playbook см. в документации Ansible[241].

Роль

Роли используют для группирования задач при развертывании и настройке сложных систем. Роль объединяет несколько задач и может быть использована в различных playbook.

Пример создания структуры каталогов для роли:

```
ansible-galaxy init httpd
```

где `httpd` – название каталога, который будет реализовывать требуемую роль.

Файлы playbook содержат список подключаемых ролей как в следующем примере:

```
- hosts: all
  user: root
  port: 22
  gather_facts: True
  roles:
    - { role: selinux, tags: selinux }
    - { role: httpd, tags: httpd }
    - { role: resolver, tags: resolver }
```

Если необходимо выполнить какую-то одну роль, то можно выполнить этот playbook, указав конкретную роль с помощью аргумента `-t`, например `-t selinux`.

Подробности о ролях см. в документации Ansible[242].

Коллекция

Коллекцией (collection) называется распространяемый как единое целое набор компонентов, расширяющих возможности Ansible.

Подробности о коллекциях см. в документации Ansible[243].

Особенности коллекций, доступных на портале Astra Automation Hub, описаны в разделе *Коллекции Ansible*.

[240] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#plays
[241] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#playbooks
[242] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#roles
[243] https://docs.ansible.com/ansible/latest/getting_started/basic_concepts.html#collections

Факты

Факты (facts) – это набор сведений об управляемом узле, например, архитектура и версия установленной ОС, версия BIOS, наличие в системе определенных устройств и тому подобное.

Для просмотра всех доступных фактов об узле необходимо выполнить команду:

```
- name: Show all available facts
  ansible_builtin.debug:
    var: ansible_facts
```

Пример списка фактов об управляемом узле


```
{
    "ansible_facts": {
        "all_ipv4_addresses": [
            "10.0.2.15",
            "192.168.56.101"
        ],
        "all_ipv6_addresses": [
            "fe80::a00:27ff:fe0d:82ad",
            "fe80::a00:27ff:feac:2f5c"
        ],
        "ansible_local": {},
        "apparmor": {
            "status": "disabled"
        },
        "architecture": "x86_64",
        "bios_date": "12/01/2006",
        "bios_vendor": "innotek GmbH",
        "bios_version": "VirtualBox",
        "board_asset_tag": "NA",
        "board_name": "VirtualBox",
        "board_serial": "NA",
        "board_vendor": "Oracle Corporation",
        "board_version": "1.2",
        "chassis_asset_tag": "NA",
        "chassis_serial": "NA",
        "chassis_vendor": "Oracle Corporation",
        "chassis_version": "NA",
        "cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-6.1.50-1-generic",
            "net.ifnames": "0",
            "parsec.max_ilev": "63",
            "quiet": true,
            "ro": true,
            "root": "UUID=17007b23-97d0-46fd-90b6-2c359be9c2b0"
        },
        "date_time": {
            "date": "2024-01-29",
            "day": "29",
            "epoch": "1706532112",
            "epoch_int": "1706532112",
            "hour": "15",
            "iso8601": "2024-01-29T12:41:52Z",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "iso8601_basic": "20240129T154152867152",
            "iso8601_basic_short": "20240129T154152",
            "iso8601_micro": "2024-01-29T12:41:52.867152Z",
            "minute": "41",
            "month": "01",
            "second": "52",
            "time": "15:41:52",
            "tz": "MSK",
            "tz_dst": "MSK",
            "tz_offset": "+0300",
            "weekday": "Понедельник",
            "weekday_number": "1",
            "weeknumber": "05",
            "year": "2024"
        },
        "default_ipv4": {
            "address": "10.0.2.15",
            "alias": "eth0",
            "broadcast": "10.0.2.255",
            "gateway": "10.0.2.2",
            "interface": "eth0",
            "macaddress": "08:00:27:0d:82:ad",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "10.0.2.0",
            "prefix": "24",
            "type": "ether"
        },
        "default_ipv6": {},
        "device_links": {
            "ids": {
                "sda": [
                    "ata-VBOX_HARDDISK_VBcf71a1d6-26264a12"
                ],
                "sda1": [
                    "ata-VBOX_HARDDISK_VBcf71a1d6-26264a12-part1"
                ]
            },
            "labels": {},
            "masters": {},
            "uuids": {
                "sda1": [
                    "17007b23-97d0-46fd-90b6-2c359be9c2b0"
                ]
            }
        },
        "devices": {
            "loop0": {
                "holders": [],
                "host": "",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "none",
                "sectors": "0",
                "sectorsize": "512",
                "size": "0.00 Bytes",
                "support_discard": "0",
                "vendor": null,
                "virtual": 1
            },
            "loop1": {
                "holders": [],
                "host": "",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "none",
                "sectors": "0",
                "sectorsize": "512",
                "size": "0.00 Bytes",
                "support_discard": "0",
                "vendor": null,
                "virtual": 1
            },
            "loop2": {
                "holders": [],
                "host": "",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "none",
                "sectors": "0",
                "sectorsize": "512",
                "size": "0.00 Bytes",
                "support_discard": "0",
                "vendor": null,
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "virtual": 1
        },
        "loop3": {
            "holders": [],
            "host": "",
            "links": {
                "ids": [],
                "labels": [],
                "masters": [],
                "uuids": []
            },
            "model": null,
            "partitions": {},
            "removable": "0",
            "rotational": "1",
            "sas_address": null,
            "sas_device_handle": null,
            "scheduler_mode": "none",
            "sectors": "0",
            "sectorsize": "512",
            "size": "0.00 Bytes",
            "support_discard": "0",
            "vendor": null,
            "virtual": 1
        },
        "loop4": {
            "holders": [],
            "host": "",
            "links": {
                "ids": [],
                "labels": [],
                "masters": [],
                "uuids": []
            },
            "model": null,
            "partitions": {},
            "removable": "0",
            "rotational": "1",
            "sas_address": null,
            "sas_device_handle": null,
            "scheduler_mode": "none",
            "sectors": "0",
            "sectorsize": "512",
            "size": "0.00 Bytes",
            "support_discard": "0",
            "vendor": null,
            "virtual": 1
        },
        "loop5": {
            "holders": [],
            "host": "",
            "links": {
                "ids": [],
                "labels": [],
                "masters": [],
                "uuids": []
            },
```


(continues on next page)

(продолжение с предыдущей страницы)


```
                "model": null,
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "none",
                "sectors": "0",
                "sectorsize": "512",
                "size": "0.00 Bytes",
                "support_discard": "0",
                "vendor": null,
                "virtual": 1
            },
            "loop6": {
                "holders": [],
                "host": "",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "none",
                "sectors": "0",
                "sectorsize": "512",
                "size": "0.00 Bytes",
                "support_discard": "0",
                "vendor": null,
                "virtual": 1
            },
            "loop7": {
                "holders": [],
                "host": "",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "none",
                "sectors": "0",
                "sectorsize": "512",
                "size": "0.00 Bytes",
                "support_discard": "0",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
                "vendor": null,
                "virtual": 1
            },
            "sda": {
                "holders": [],
                "host": "SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-
↪E) SATA Controller [AHCI mode] (rev 02)",
                "links": {
                    "ids": [
                        "ata-VBOX_HARDDISK_VBcf71a1d6-26264a12"
                    ],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": "VBOX HARDDISK",
                "partitions": {
                    "sda1": {
                        "holders": [],
                        "links": {
                            "ids": [
                                "ata-VBOX_HARDDISK_VBcf71a1d6-26264a12-part1"
                            ],
                            "labels": [],
                            "masters": [],
                            "uuids": [
                                "17007b23-97d0-46fd-90b6-2c359be9c2b0"
                            ]
                        },
                        "sectors": "61435904",
                        "sectorsize": 512,
                        "size": "29.29 GB",
                        "start": "2048",
                        "uuid": "17007b23-97d0-46fd-90b6-2c359be9c2b0"
                    }
                },
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "mq-deadline",
                "sectors": "61440000",
                "sectorsize": "512",
                "size": "29.30 GB",
                "support_discard": "0",
                "vendor": "ATA",
                "virtual": 1
            }
        },
        "distribution": "Astra Linux",
        "distribution_file_parsed": true,
        "distribution_file_path": "/etc/os-release",
        "distribution_file_variety": "NA",
        "distribution_major_version": "1",
        "distribution_release": "1.7_x86-64",
        "distribution_version": "1.7_x86-64",
        "dns": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "domain": "astralinux.ru",
            "nameservers": [
                "10.0.2.3"
            ],
            "search": [
                "astralinux.ru"
            ]
        },
        "domain": "",
        "effective_group_id": 1000,
        "effective_user_id": 1000,
        "env": {
            "HOME": "/home/vagrant",
            "LANG": "ru_RU.UTF-8",
            "LC_CTYPE": "C.UTF-8",
            "LOGNAME": "vagrant",
            "PATH": "/usr/local/bin:/usr/bin:/bin:/usr/games",
            "PWD": "/home/vagrant",
            "SHELL": "/bin/bash",
            "SHLVL": "1",
            "SSH_CLIENT": "192.168.56.11 37284 22",
            "SSH_CONNECTION": "192.168.56.11 37284 192.168.56.101 22",
            "SSH_TTY": "/dev/pts/0",
            "TERM": "xterm",
            "USER": "vagrant",
            "XDG_RUNTIME_DIR": "/run/user/1000",
            "XDG_SESSION_CLASS": "user",
            "XDG_SESSION_ID": "31",
            "XDG_SESSION_TYPE": "tty",
            "_": "/usr/bin/python3"
        },
        "eth0": {
            "active": true,
            "device": "eth0",
            "ipv4": {
                "address": "10.0.2.15",
                "broadcast": "10.0.2.255",
                "netmask": "255.255.255.0",
                "network": "10.0.2.0",
                "prefix": "24"
            },
            "ipv6": [
                {
                    "address": "fe80::a00:27ff:fe0d:82ad",
                    "prefix": "64",
                    "scope": "link"
                }
            ],
            "macaddress": "08:00:27:0d:82:ad",
            "module": "e1000",
            "mtu": 1500,
            "pciid": "0000:00:03.0",
            "promisc": false,
            "speed": 1000,
            "type": "ether"
        },
        "eth1": {
```


(continues on next page)

(продолжение с предыдущей страницы)


```
            "active": true,
            "device": "eth1",
            "ipv4": {
                "address": "192.168.56.101",
                "broadcast": "192.168.56.255",
                "netmask": "255.255.255.0",
                "network": "192.168.56.0",
                "prefix": "24"
            },
            "ipv6": [
                {
                    "address": "fe80::a00:27ff:feac:2f5c",
                    "prefix": "64",
                    "scope": "link"
                }
            ],
            "macaddress": "08:00:27:ac:2f:5c",
            "module": "e1000",
            "mtu": 1500,
            "pciid": "0000:00:08.0",
            "promisc": false,
            "speed": 1000,
            "type": "ether"
        },
        "fibre_channel_wwn": [],
        "fips": false,
        "form_factor": "Other",
        "fqdn": "node-1",
        "gather_subset": [
            "all"
        ],
        "hostname": "node-1",
        "hostnqn": "",
        "interfaces": [
            "eth0",
            "eth1",
            "lo"
        ],
        "is_chroot": false,
        "iscsi_iqn": "",
        "kernel": "6.1.50-1-generic",
        "kernel_version": "#astra2+ci6 SMP PREEMPT_DYNAMIC Fri Oct  6 14:38:42 UTC␣
↪2023",
        "lo": {
            "active": true,
            "device": "lo",
            "ipv4": {
                "address": "127.0.0.1",
                "broadcast": "",
                "netmask": "255.0.0.0",
                "network": "127.0.0.0",
                "prefix": "8"
            },
            "ipv6": [
                {
                    "address": "::1",
                    "prefix": "128",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
                    "scope": "host"
                }
            ],
            "mtu": 65536,
            "promisc": false,
            "type": "loopback"
        },
        "lsb": {
            "codename": "1.7_x86-64",
            "description": "Astra Linux 1.7 x86-64",
            "id": "AstraLinux",
            "major_release": "1",
            "release": "1.7_x86-64"
        },
        "machine": "x86_64",
        "machine_id": "d383adfed45648cdbd75a02273b61314",
        "memfree_mb": 1639,
        "memory_mb": {
            "nocache": {
                "free": 1805,
                "used": 160
            },
            "real": {
                "free": 1639,
                "total": 1965,
                "used": 326
            },
            "swap": {
                "cached": 0,
                "free": 0,
                "total": 0,
                "used": 0
            }
        },
        "memtotal_mb": 1965,
        "module_setup": true,
        "mounts": [
            {
                "block_available": 6732778,
                "block_size": 4096,
                "block_total": 7514846,
                "block_used": 782068,
                "device": "/dev/sda1",
                "fstype": "ext4",
                "inode_available": 1884545,
                "inode_total": 1921360,
                "inode_used": 36815,
                "mount": "/",
                "options": "rw,relatime,errors=remount-ro",
                "size_available": 27577458688,
                "size_total": 30780809216,
                "uuid": "17007b23-97d0-46fd-90b6-2c359be9c2b0"
            }
        ],
        "nodename": "node-1",
        "os_family": "Astra Linux",
        "pkg_mgr": "apt",
```


(continues on next page)

(продолжение с предыдущей страницы)


```
        "proc_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-6.1.50-1-generic",
            "net.ifnames": "0",
            "parsec.max_ilev": "63",
            "quiet": true,
            "ro": true,
            "root": "UUID=17007b23-97d0-46fd-90b6-2c359be9c2b0"
        },
        "processor": [
            "0",
            "GenuineIntel",
            "Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz",
            "1",
            "GenuineIntel",
            "Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz"
        ],
        "processor_cores": 2,
        "processor_count": 1,
        "processor_nproc": 2,
        "processor_threads_per_core": 1,
        "processor_vcpus": 2,
        "product_name": "VirtualBox",
        "product_serial": "NA",
        "product_uuid": "NA",
        "product_version": "1.2",
        "python": {
            "executable": "/usr/bin/python3",
            "has_sslcontext": true,
            "type": "cpython",
            "version": {
                "major": 3,
                "micro": 3,
                "minor": 7,
                "releaselevel": "final",
                "serial": 0
            },
            "version_info": [
                3,
                7,
                3,
                "final",
                0
            ]
        },
        "python_version": "3.7.3",
        "real_group_id": 1000,
        "real_user_id": 1000,
        "selinux": {
            "status": "disabled"
        },
        "selinux_python_present": true,
        "service_mgr": "systemd",
        "ssh_host_key_ecdsa_public":
↪"AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJe9D5fI+xDzdcOfpQ/
↪JzxooISS7wskk7RzuX1Qe2eB+yIitXmNBkzRUTvTGsokhQCGPzl8rRlPEOcWCkwA37O4=",
        "ssh_host_key_ecdsa_public_keytype": "ecdsa-sha2-nistp256",
        "ssh_host_key_ed25519_public":
```


(continues on next page)

(продолжение с предыдущей страницы)


```
↪"AAAAC3NzaC1lZDI1NTE5AAAAILjVQqxasGBvwxxIFoAV+b37omFihnDLe/AW1XT0raVF",
        "ssh_host_key_ed25519_public_keytype": "ssh-ed25519",
        "ssh_host_key_rsa_public":
↪"AAAAB3NzaC1yc2EAAAADAQABAAABgQC5SLc9hewzm7M9prRWaR+v5ds0Qjj41Bzb2rZz0ITwHrEAIcdSXXvlRkq2wQgGxdcVL
↪xEvX2iwJffJ2yyILr+yXlBQehYbqCJhZt7zkF7KyLwC/
↪QESIRG416U5KQuviMStXCbLBOoTo1BmNHNW8jX2NE4/8hY/M94/naC86wJ805/l4Dc2+FKXTua8/
↪yq53Abc8I9RBP7qMPOx7nAcbCmiKxJd1IED2Bj9avwhsYKCdWqBlKMJ3EUccrVYM20Jhp5j/
↪EqTRYWtkpda2XSkFkXTYgU0nmyboav0JDd18PmnxcG3/Jzzm5m/
↪EDM3vulRqWoqIYTCmAh5F32tleJllhBsMHsGOmYWgXUF0zEZob1kkn5/
↪mUVMHxN3lLdxkjY7WRuTXjaplW71fgV6Mq3mJIaHHVuBnpUkwFwWLZdPkLEQyma0lrK5O1W2Uenxmed5Y72vMq2Mvj+X5P3Tng
↪",
        "ssh_host_key_rsa_public_keytype": "ssh-rsa",
        "swapfree_mb": 0,
        "swaptotal_mb": 0,
        "system": "Linux",
        "system_capabilities": [
            ""
        ],
        "system_capabilities_enforced": "True",
        "system_vendor": "innotek GmbH",
        "uptime_seconds": 58762,
        "user_dir": "/home/vagrant",
        "user_gecos": "vagrant,,,",
        "user_gid": 1000,
        "user_id": "vagrant",
        "user_shell": "/bin/bash",
        "user_uid": 1000,
        "userspace_architecture": "x86_64",
        "userspace_bits": "64",
        "virtualization_role": "guest",
        "virtualization_tech_guest": [
            "virtualbox"
        ],
        "virtualization_tech_host": [],
        "virtualization_type": "virtualbox"
    }
}
```


Подробности об использовании фактов см. в документации Ansible[244].

[244] https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html#ansible-facts

16.3.2 Docker

Docker[245] – это инструмент для создания и запуска контейнеризованных приложений.

Назначение

В Astra Automation утилиты, входящие в состав ПО Docker, используются для следующих задач:

- запуск заданий Ansible в *среде исполнения*;
- *сборка собственного образа* среды исполнения Astra Automation Controller.

Установка

Для установки Docker выполните следующие действия:

1. Обновите список доступных пакетов:

```
sudo apt update
```

2. Установите пакеты:

```
sudo apt install docker.io docker-compose containerd runc --yes
```

3. Добавьте активного пользователя в группу `docker`:

```
sudo gpasswd --add ${USER} docker
```

4. Для применения привилегий новой группы без перезапуска сессии выполните команду:

```
newgrp docker
```

16.3.3 Podman

Podman[246] – это инструмент с открытым исходным, используемый для создания и запуска контейнеризованных приложений. В отличие от *Docker*, по умолчанию Podman запускает контейнеры в непривилегированном режиме.

[245] https://www.docker.com/
[246] https://podman.io

Назначение

В Astra Automation утилиты, входящие в состав ПО Podman, используются для следующих задач:

- запуск заданий Ansible в *среде исполнения*;
- использование конструктора кода для управления инфраструктурой;
- сборка собственного образа среды исполнения Astra Automation Controller.

Установка

Важно: Пакеты Podman доступны в репозиториях Astra Linux Special Edition начиная с оперативного обновления 1.7.2 (БЮЛЛЕТЕНЬ № 2022-0819SE17). Инструкции по обновлению ОС приведены в разделе Справочного центра Оперативные обновления для Astra Linux Special Edition РУСБ.10015-01 (очередное обновление 1.7), РУСБ.10015-10, РУСБ.10015-37[247].

Для установки пакетов Podman выполните следующие действия:

1. Получите номер установленного оперативного обновления ОС:

   ```
   cat /etc/astra_version
   ```

   Команда выводит в терминал строку вида:

   ```
   1.7.5
   ```

2. Добавьте в файл `/etc/apt/sources.list` ссылку на расширенный (extended) репозиторий Astra Linux Special Edition 1.7:

   ```
   deb https://download.astralinux.ru/astra/frozen/1.7_x86-64/<version>/repository-
   ↪extended/ 1.7_x86-64 main contrib non-free
   ```

   где <version> – версия установленного оперативного обновления Astra Linux Special Edition. Для примера выше указанная строка имеет вид:

   ```
   deb https://download.astralinux.ru/astra/frozen/1.7_x86-64/1.7.5/repository-
   ↪extended/ 1.7_x86-64 main contrib non-free
   ```

3. Обновите список доступных пакетов:

   ```
   sudo apt update
   ```

4. Установите пакеты Podman:

   ```
   sudo apt install podman --yes
   ```

[247] https://wiki.astralinux.ru/pages/viewpage.action?pageId=158612043

Проверка корректности установки

Для проверки корректности установки Podman выполните следующие действия:

1. В любом каталоге создайте два файла:
   - `playbook.yml` – простейший playbook Ansible:

```
---
- name: Validate Podman setup
  hosts: all
  tasks:
    - name: Print Execution Environment OS info
      ansible.builtin.debug:
        msg:
          - OS version is {{ ansible_distribution_version }}
          - Python version is {{ ansible_python_version }}
```

   - `inventory` – инвентарь Ansible:

```
[all]
localhost  ansible_connection=local  ansible_python_interpreter=/usr/bin/
↪python3
```

2. Запустите выполнение playbook в контейнере Podman:

```
podman run \
    --rm \
    --volume "$(pwd):/app" \
    registry.astralinux.ru/aa/aa-base-ee:latest \
    bash -c 'ansible-playbook -i inventory playbook.yml'
```

   где:

   - `--rm` – автоматическое удаление контейнера после выполнения команды.
   - `--volume`, `-v` – монтирование объекта файловой системы управляющего узла в файловую систему контейнера. В данном случае каталог с файлами `inventory` и `playbook.yml` монтируется в каталог `/app`. Путь к каталогу определяется с помощью команды pwd.
   - `registry.astralinux.ru/aa/aa-base-ee:latest` – ссылка на образ.
   - `bash -c 'ansible-playbook -i inventory.yml playbook.yml'` – команда, которая должна быть выполнена в контейнере. В данном случае – запуск playbook с указанным инвентарем.

   Если Podman настроен корректно, в терминал выводятся строки следующего вида:

```
PLAY [Validate Podman setup]␣
↪*********************************************************************************

TASK [Gathering Facts]␣
↪*********************************************************************************
ok: [localhost]

TASK [Print Execution Environment␣
↪information]*****************************************************************
ok: [localhost] => {
    "msg": [
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "OS version is 1.7_x86-64",
        "Python version is 3.9.18"
    ]
}

PLAY RECAP␣
↪*********************************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0    ␣
↪skipped=0    rescued=0    ignored=0
```

16.3.4 Task

Task[248] – это инструмент, используемый для управления сборкой программ и запуском задач аналогично утилите Make.

Назначение

В Task задачей называют одну или несколько команд, выполняемых в определенном порядке.

В Astra Automation утилита Task используется для запуска следующих задач:

- подготовка конфигурационных файлов Terraform и Ansible с помощью конструктора кода;
- развертывание инфраструктуры с помощью Terraform;
- настройка управляемых узлов с помощью Ansible.

Установка

Важно: Утилита Task доступна в репозиториях Astra Linux Special Edition, начиная с оперативного обновления 1.7.4. Инструкции по обновлению ОС приведены в разделе Справочного центра Оперативные обновления для Astra Linux Special Edition РУСБ.10015-01 (очередное обновление 1.7), РУСБ.10015-10, РУСБ.10015-37[249].

Для установки утилиты Task выполните следующие действия:

1. Обновите список доступных пакетов:

   ```
   sudo apt update
   ```

2. Установите пакет task:

   ```
   sudo apt install task --yes
   ```

[248] https://taskfile.dev/
[249] https://wiki.astralinux.ru/pages/viewpage.action?pageId=158612043

Принципы работы

Task предоставляет широкие возможности для управления задачами:

- Описание зависимостей между задачами.
- Запуск задач в определенной последовательности.
- Параллельное выполнение задач.
- Запуск одних и тех же задач с различными параметрами.

Для использования утилиты Task необходимо выполнить следующие действия:

1. Создать в каталоге проекта файл `Taskfile.yml`.
2. Описать в файле `Taskfile.yml` необходимые задачи.
3. Запустить выполнение нужный задачи с помощью команды:

```
task <task_name>
```

Пример файла `Taskfile.yml`, в котором определена одна задача с названием `hello`:

```
---
version: '3'

tasks:
   hello:
      cmds:
         - echo 'Hello World from Task!'
```

Для выполнения задачи `hello` необходимо использовать следующую команду:

```
task hello
```

Зависимости (dependencies)

Задачи могут зависеть друг от друга – запуск одной приводит к автоматическому запуску других, указанных в списке зависимостей.

Для указания списка зависимостей используется блок deps:

```
---
version: '3'

tasks:
  task-1:
    cmds:
      - echo "Task 1"

  task-2:
    cmds:
      - echo "Task 2"

  task-3:
    deps: [task-1, task2]
    cmds:
      - echo "Task 3"
```

Команды, описанные в задаче task-3, будут выполнены только после того, как завершится выполнение задач task-1 и task-2.

Порядок выполнения зависимостей

В целях оптимизации Task запускает параллельно выполнение задач, указанных в блоке deps. Если задачи из списка зависимостей нужно выполнять в определенном порядке, вместо блока deps укажите названия задач в блоке cmds:

```
version: '3'

tasks:
  task-1:
    cmds:
      - echo "Task 1"

  task-2:
    cmds:
      - echo "Task 2"

  task-3:
    cmds:
      - task: task-1  # Выполнится первой
      - task: task-2  # Выполнится второй
      - echo "Task 3" # Выполнится последней
```

Внутренние задачи

По умолчанию для запуска из терминала доступны все задачи, описанные в Taskfile.yml. Если необходимо запретить запуск какой-либо задачи из терминала, отметьте ее как внутреннюю. Внутренними считаются задачи, к описанию которых добавлен блок internal со значением true, например:

```
---
version: '3'

tasks:
  update-apt-cache:
    cmds:
      - sudo apt update
    internal: true

  install-seahorse:
    cmds:
      - sudo apt install seahorse --yes
    deps: [update-apt-cache]
```

Задачу update-apt-cache нельзя вызвать напрямую, однако, ее можно использовать как зависимость в других задачах.

Переменные (variables)

Переменные позволяют упростить работу со значениями, которые многократно используются в различных частях `Taskfile.yml` или должны быть вычислены во время выполнения задач.

Task выполняет поиск переменной в следующем порядке:

1. В блоке `vars` в описании задачи.
2. В списке переменных, переданных при запуске задачи из другой задачи.
3. В списке переменных, импортированных из включенного `Taskfile.yml`.
4. В списке переменных, переданных во включенный `Taskfile.yml`.
5. В блоке `vars` в корне `Taskfile.yml`.
6. В списке переменных среды.

Объявление и использование

Переменные описываются в блоке `vars`, который размещается в корне `Taskfile.yml` или на уровне отдельных задач. В первом случае описанные переменные называются глобальными, так как становятся доступными во всех задачах.

Для доступа к значению переменной используется синтаксис с обрамляющими фигурными скобками и точкой перед именем.

Пример описания и использования глобальной переменной `VERSION`:

```
---
version: '3'

vars:
  VERSION: 1.0.0

tasks:
  default:
    cmds:
      - echo {{.VERSION}}
```

Значения переменных могут быть вычислены динамически. В этом примере значения переменных `CURRENT_DATE` и `CURRENT_TIME` будут вычисляться заново при каждом использовании утилиты Task:

```
---
version: '3'

vars:
  CURRENT_DATE:
    sh: date +"%Y-%m-%d"
  CURRENT_TIME:
    sh: date +"%H:%m:%S"

tasks:
  default:
    cmds:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      - echo "Current date is {{.CURRENT_DATE}}"
      - echo "Current time is {{.CURRENT_TIME}}"
```

Переменные среды (environment variables)

Для доступа к значениям переменных среды используется синтаксис с символом $ перед именем, например:

```
---
version: '3'

tasks:
  default:
    cmds:
      - echo $USER
```

Можно задать значения переменных среды в Taskfile или отдельном файле формата INI.

Описание

Чтобы задать значение переменной среды в Taskfile, добавьте в корневой раздел или описание задачи блок env.

Пример описания и использования переменной окружения TERMINAL:

```
---
version: '3'

env:
  TERMINAL: Bash

tasks:
  default:
    cmds:
      - echo "Terminal is $TERMINAL"
```

Загрузка из файлов

Чтобы использовать значения переменных среды, указанные в отдельном файле, следует:

1. Создать файл, в котором указать значения переменных в формате <NAME>=<VALUE>, например:

   ```
   USER_ID=af93jdsflkj2
   CLOUD_ID=cjdkl1490
   FOLDER_ID=u1093j1kalfop
   ```

2. Добавить в Taskfile.yml блок dotenv, в значении которого указать список имен файлов, из которых следует загрузить значения переменных окружения, например:

```yaml
---
version: '3'

tasks:
  prod-task:
    dotenv: ['.env', 'production/.env']
    cmds:
      # -

  test-task:
    dotenv: ['.env', 'test/.env']
    cmds:
      # -
```

В этом примере для задач `prod-task` и `test-task` указаны разные наборы файлов, из которых следует загрузить значения переменных окружения. В первом случае будет выполнена загрузка из файлов `.env` и `production/.env`, а во втором – `.env` и `test/.env`.

Совет: Для файлов с переменными среды рекомендуется использовать имя `.env`.

Специальные переменные среды

Во время выполнения задач доступны переменные среды, созданные Task:

Переменная	Описание
`{{.CHECKSUM}}`	Контрольные суммы файлов, созданных во время предыдущего выполнения задачи
`{{.TASK}}`	Название выполняемой задачи
`{{.TIMESTAMP}}`	Дата и время создания файлов, созданных во время предыдущего выполнения задачи
`{{.USER_WORKING_DIR}}`	Полный путь к текущему каталогу

Подключение других файлов настройки

Task позволяет включать одни `Taskfile.yml` в другие. Благодаря этому можно организовать задачи в виде иерархической структуры.

Пусть имеется такая структура файлов и каталогов:

```
├─ build
│  └─ Taskfile.yml
├─ deploy
│  └─ Taskfile.yml
└─ Taskfile.yml
```

Файл `build/Taskfile.yml` содержит задачу `ci`, выполняющую сборку приложения. Файл `deploy/Taskfile.yml` содержит задачу `cd`, выполняющую доставку приложения в рабочее окружение.

Файл `Taskfile` в каталоге проекта включает в себя содержимое этих двух файлов.

```
---
version: '3'

includes:
  build: build/Taskfile.yml
  deploy: deploy/Taskfile.yml

tasks:
  build-and-deploy:
    cmds:
      - task: build:ci
      - task: deploy:cd
      - echo "Task build-and-deploy is done"
```

Здесь:

- `includes` – блок, описывающий включаемые файлы;
- `build`, `deploy` – пространства имен, в которых размещаются задачи из включенных файлов;
- `build:ci` – задача `ci`, импортированная из файла `build/Taskfile.yml`;
- `deploy:cd` – задача `cd`, импортированная из файла `deploy/Taskfile.yml`.

Task позволяет передавать значения переменных в импортируемые `Taskfile`.

Пусть имеется файл `deploy/Taskfile.yml`, содержащий задачу `deploy`:

```
---
version: '3'

tasks:
  deploy:
    cmds:
      - echo "Target host is {{.TARGET_HOST}}"
```

Здесь `TARGET_HOST` – переменная, задающая FQDN целевого хоста, на котором нужно выполнить развертывание приложения. Чтобы использовать задачу `deploy` с разными значениями переменной `TARGET_HOST`, следует указать их при импорте `deploy/Taskfile.yml`:

```
---
version: '3'

includes:
  test:
    taskfile: deploy/Taskfile.yml
    vars:
      TARGET_HOST: testing.example.com

  prod:
    taskfile: deploy/Taskfile.yml
    vars:
      TARGET_HOST: production.example.com
```

Результат выполнения задач `test:deploy` и `prod:deploy`:

```
task: [test:deploy] echo "Target host is testing.example.com"
Target host is testing.example.com

task: [prod:deploy] echo "Target host is production.example.com"
Target host is production.example.com
```

16.3.5 Terraform

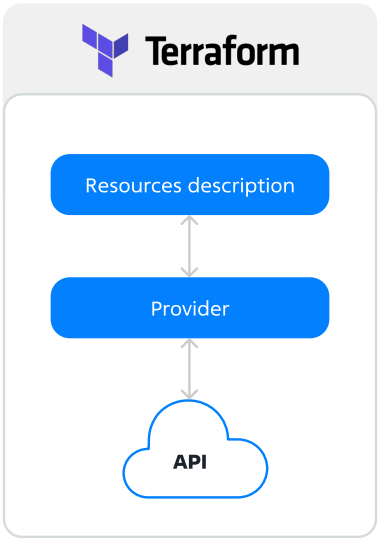
Terraform[250] представляет собой систему автоматизации развертывания и обновления информационной инфраструктуры с использованием доступного для организации оборудования и программного обеспечения. Для описания требуемой инфраструктуры Terraform предлагает декларативный язык на базе *HCL*.

Развертываемая инфраструктура может содержать самые разнообразные аппаратные и программные компоненты, включая серверы, балансировщики нагрузки, сетевое оборудование, системы хранения, компоненты безопасности и другие. Она может быть реализована как в приватном центре обработки данных, так и в одном из облачных сервисов.

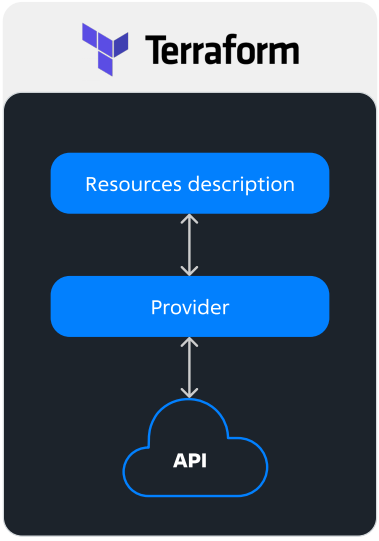
Terraform предоставляет средства разработки собственных модулей в составе корпоративных систем автоматизации. Astra Automation включает в себя модули Terraform для управления информационными инфраструктурами, развертываемыми в различных облачных средах.

Принципы управления

Принцип работы Terraform показан на схеме:



[250] https://www.terraform.io/



Terraform для управления ресурсами использует провайдеры – особые расширения, содержащие сведениях об API нужных сервисов. Как правило, файлы провайдеров размещаются в реестре провайдеров Terraform[251], однако, могут быть загружены и со сторонних ресурсов.

Используя данные, предоставленные провайдером, Terraform обращается к API нужного сервиса для получения сведений о состоянии ресурсов. Если описание ресурсов в конфигурационных файлах не совпадает с фактическим, Terraform изменяет состояние ресурсов сервиса, приводя их в соответствие описанию в конфигурационных файлах.

Управление инфраструктурой с помощью Terraform состоит из следующих шагов:

1. Описание ресурсов в конфигурационных файлах.
2. Инициализация модулей и провайдеров.

   ```
   terraform init
   ```

   На этом этапе Terraform загружает все необходимые файлы и сохраняет в каталоге проекта в следующие подкаталоги:

   - `.terraform/modules/` – модули;
   - `.terraform/providers/` – провайдеры.

3. Проверка плана инфраструктуры.

   Для проверки плана инфраструктуры используется следующая команда:

   ```
   terraform validate
   ```

   На этом этапе Terraform выполняет сверку описания инфраструктуры с формальными правилами, содержащимися в описаниях провайдера, модулей и переменных. Если в конфигурации инфраструктуры есть ошибки, Terraform на них укажет.

4. Планирование изменений.

   Для получения списка изменений инфраструктуры, которые будут выполнены Terraform, чтобы привести инфраструктуру в соответствие с ее описанием в файлах проекта, используется следующая команда:

[251] https://registry.terraform.io/

```
terraform plan
```

На этом этапе никаких фактических изменений инфраструктуры не происходит.

5. Применение изменений.

   Для применения изменений используется следующая команда:

```
terraform apply
```

Типовая структура проекта Terraform

В самом простом случае описание инфраструктуры с помощью Terraform хранится в одном или нескольких файлах с расширением `.tf`, содержащих описания следующих сущностей:

- общие настройки;
- описание ресурсов.

Общие настройки

К общим настройкам относятся:

- сведения о провайдерах;
- учетные данные для управления ресурсами выбранного сервиса.

Провайдер

Провайдер определяет возможности Terraform по работе с сервисом:

- параметры доступа к сервису;
- перечень доступных для управления типов ресурсов;
- перечень доступных для использования источников данных;
- правила работы с ресурсами и источниками данных.

Перечень используемых провайдеров задается в блоке `terraform.required_providers`, например:

```
terraform {
  required_providers {
    sbercloud = {
      source = "sbercloud-terraform/sbercloud"
    }
  }
}
```

Здесь:

- `sbercloud` – название провайдера;
- `sbercloud-terraform/sbercloud` – ссылка на источник для загрузки файлов провайдера.

Учетные данные для работы с ресурсами задаются в блоке `provider`:

```
provider "sbercloud" {
  # Настройки доступа к сервису
}
```

Здесь `sbercloud` – название провайдера, указанное ранее.

Подробности об использовании провайдеров см. в документации Terraform[252].

Ресурс

Ресурсы (виртуальные машины, сети, подсети, кластеры СУБД и т. п.) описываются в блоках `resource`:

```
resource "openstack_db_user_v1" "user-1" {
  # Описание ресурса
}
```

Здесь:

- `openstack_db_user_v1` – тип ресурса;
- `user-1` – название ресурса, используемое в Terraform.

По умолчанию Terraform использует следующий подход при применении изменений конфигурации:

- создает ресурсы, которые есть в описании инфраструктуры но отсутствуют в файле состояния;
- удаляет ресурсы, которые есть в файле состояния, но отсутствуют в описании инфраструктуры;
- обновляет ресурсы, параметры которых изменились;
- удаляет и создает заново ресурсы, параметры которых изменились таким образом, что иначе применить их невозможно.

Подробности о ресурсах см. в документации Terraform[253].

Источник данных

Источники данных (data sources) позволяют получать сведения о существующих ресурсах.

Источники данных описываются в блоках `data`, например:

```
data "yandex_mdb_mysql_cluster" "data-source" {
  name = "mysql-cluster-name"
}
```

где

- `yandex_mdb_mysql_cluster` – тип источника данных (кластер Yandex Managed Service for MySQL);

[252] https://developer.hashicorp.com/terraform/language/providers
[253] https://developer.hashicorp.com/terraform/language/resources

- `data-source` – название источника данных;
- `mysql-cluster-name` – название кластера.

Подробности об источниках данных см. в документации Terraform[254].

Состояние

Состояние (state) – это информация о фактических существующих в сервисе ресурсах и связи их идентификаторов с описаниями в конфигурационных файлах Terraform.

По умолчанию состояние хранится локально в файле `terraform.tfstate`. Terraform использует его для расчета изменений, которые нужно выполнить, чтобы привести фактическое состояние ресурсов к желаемому.

Подробности о состояниях см. в документации Terraform[255].

Переменные и значения

Terraform позволяет модулям хранить данные и обмениваться ими друг с другом следующими способами:

- с помощью локальных значений (Local Values);
- с помощью переменных, также называемых входными переменными (Input Variables);
- с помощью выходных значений (Output Values).

Локальные значения

Локальные значения позволяют многократно ссылаться на одно и то же значение без необходимости вводить его заново. Аналогом локальных значений в обычных языках программирования являются константы.

Локальные значения описываются в блоке `locals`, например:

```
locals {
  image_id     = "fd87qct47l4isk56gucq",
  storage_size = 40
}
```

где `image_id` и `storage_size` – названия, а `fd87qct47l4isk56gucq` и `40` – значения.

Область видимости локальных значений ограничена модулем, в котором они описаны.

Для обращения к локальному значению используется следующий синтаксис:

```
resource "vm" "vm-1" {
  image     = local.image_id
  disk_size = local.storage_size
  # ...
}
```

Подробности о локальных значениях см. в документации Terraform[256].

[254] https://developer.hashicorp.com/terraform/language/data-sources
[255] https://developer.hashicorp.com/terraform/language/state
[256] https://developer.hashicorp.com/terraform/language/values/locals

Входные переменные

Входные переменные (input variables), часто для краткости называемые просто «переменными», позволяют использовать одни и те же значения в разных модулях и `.tf`-файлах.

Каждая входная переменная описывается в отдельном блоке `variable`, например:

```
variable "image_id" {
  type = string
}
```

Здесь `image_id` – название переменной, `string` – ее тип.

Для входной переменной может быть указано значение по умолчанию:

```
variable "image_id" {
  type    = string
  default = "fd87qct47l4isk56gucq"
}
```

Для обращения к входной переменной используется следующий синтаксис:

```
resource "vm" "vm-1" {
  image = var.image_id
  # ...
}
```

Значения входных переменных, объявленных в корневом модуле, могут быть заданы следующими способами:

- Значение параметра `-var` в аргументах командной строки.
- Объявления в файле с расширением `.tfvars`.

  Примечание: Terraform автоматически загружает значения входных переменных из файлов `terraform.tfvars` и `terraform.tfvars.json`, а также из файлов, имена которых оканчиваются на `.auto.tfvars` и `.auto.tfvars.json`.

- Переменные среды.

Подробности о входных переменных см. в документации Terraform[257].

Выходные значения

Выходные значения (output values) используются для передачи данных за пределы модуля.

Каждое выходное значение описывается в отдельном блоке `output`, например:

```
output "bastion_ip" {
  value = vsphere_virtual_machine.virtual_machine.default_ip_address
}
```

[257] https://developer.hashicorp.com/terraform/language/values/variables

Для обращения к выходной переменной модуля из другого модуля используется следующий синтаксис:

```
module "vm" {
  # ...
  public_ip = module.bastion.bastion_ip
}
```

где:

- `bastion` – название модуля;
- `bastion_ip` – название выходного значения.

Подробности о выходных значениях см. в документации Terraform[258].

Мета-аргументы

Мета-аргументы (meta-arguments) используются для добавления в конфигурационные файлы дополнительной функциональности.

depends_on

Мета-аргумент depends_on используется для создания между ресурсами или модулями связей, которые не могут быть вычислены Terraform автоматически.

Пусть имеется следующее описание ресурсов, необходимых для запуска на ВМ приложения, работающего с базой данных PostgreSQL:

```
# Кластер Yandex Managed Service for PostgreSQL
resource "yandex_mdb_postgresql_cluster" "pg-cluster" {
  # ...
}

# Пользователь кластера "pg-cluster"
resource "yandex_mdb_postgresql_user" "db-owner" {
  cluster_id = yandex_mdb_postgresql_cluster.pg-cluster.id
  # ...
}

# База данных в кластере "pg-cluster"
resource "yandex_mdb_postgresql_database" "db1" {
  cluster_id = yandex_mdb_postgresql_cluster.pg-cluster.id
  name       = "db1"
  owner      = yandex_mdb_postgresql_user.db-owner.name
}

# Виртуальная машина
resource "yandex_compute_instance" "app-vm" {
  name = "app-vm"
  depends_on = [
    yandex_mdb_postgresql_cluster.pg-cluster,
    yandex_mdb_postgresql_user.db-owner,
    yandex_mdb_postgresql_database.db1
  ]
```

(continues on next page)

[258] https://developer.hashicorp.com/terraform/language/values/outputs

(продолжение с предыдущей страницы)

```
  # Прочие настройки ВМ
}
```

Между кластером СУБД, базой данных и пользователем есть прямая зависимость – поле `cluster_id` у базы данных и пользователя обязательно для заполнения. Порядок создания ресурсов в этом случае определяется логикой на стороне провайдера:

1. Кластер СУБД.
2. Пользователь СУБД.
3. База данных.

В то же время, ВМ не зависит от кластера СУБД, базы данных или ее пользователя. Однако, приложение, запущенное на ВМ, не сможет работать до тех пор, пока кластер СУБД, база данных и используемый для доступа к базе данных пользователь не будут созданы.

Блок depends_on в описании ВМ указывает, что она зависит от перечисленных ресурсов и должна быть создана после них.

Подробности об использовании мета-аргумента depends_on см. в документации Terraform[259].

count

Мета-аргумент `count` используется для создания однотипных ресурсов.

Пусть необходимо создать 4 одинаковых ВМ. Соответствующую конфигурацию можно описать без использования мета-аргумента `count` следующим образом:

```
resource "sbercloud_compute_instance" "vm-1" {
  name = "vm-1"
  # ...
}

resource "sbercloud_compute_instance" "vm-2" {
  name = "vm-2"
  # ...
}

resource "sbercloud_compute_instance" "vm-3" {
  name = "vm-3"
  # ...
}

resource "sbercloud_compute_instance" "vm-4" {
  name = "vm-4"
  # ...
}
```

При использовании мета-аргумента `count` эту запись можно сократить:

[259] https://developer.hashicorp.com/terraform/language/meta-arguments/depends_on

```
resource "sbercloud_compute_instance" "vms" {
  count = 4

  name = "vm-${count.index + 1}"
  # ...
}
```

Подробности об использовании мета-аргумента count см. в документации Terraform[260].

for_each

Мета-аргумент for_each позволяет обрабатывать списки путем последовательного перебора их значений. Это позволяет сократить описание однотипных ресурсов, имеющих различия в конфигурации.

Пусть необходимо создать в одной сети несколько подсетей. Описание инфраструктуры в этом случае может иметь следующий вид:

```
resource "sbercloud_vpc" "vpc-1" {
  name = "vpc-1"
  cidr = "10.33.0.0/24"
}

resource "sbercloud_vpc_subnet" "subnet-1" {
  vpc_id        = sbercloud_vpc.vpc-1.id
  name          = "subnet-1"
  cidr          = "10.33.10.0/24"
  primary_dns   = "100.125.13.59"
  secondary_dns = "8.8.8.8"
}

resource "sbercloud_vpc_subnet" "subnet-2" {
  vpc_id        = sbercloud_vpc.vpc-1.id
  name          = "subnet-2"
  cidr          = "10.33.20.0/24"
  primary_dns   = "100.125.13.59"
  secondary_dns = "8.8.8.8"
}

resource "sbercloud_vpc_subnet" "subnet-3" {
  vpc_id        = sbercloud_vpc.vpc-1.id
  name          = "subnet-3"
  cidr          = "10.33.30.0/24"
  primary_dns   = "100.125.13.59"
  secondary_dns = "8.8.8.8"
}
```

Из этой записи видно, что значения полей vpc_id, primary_dns и secondary_dns повторяются.

С помощью for_each описание ресурсов можно значительно сократить:

```
resource "sbercloud_vpc" "vpc-1" {
  name = "vpc-1"
```

(continues on next page)

[260] https://developer.hashicorp.com/terraform/language/meta-arguments/count

(продолжение с предыдущей страницы)

```
  cidr = "10.33.0.0/24"
}

locals {
  subnets = {
    "subnet-1" = { cidr = "10.33.10.0/24" },
    "subnet-2" = { cidr = "10.33.20.0/24" },
    "subnet-3" = { cidr = "10.33.30.0/24" }
  }
}

resource "sbercloud_vpc_subnet" "subnets" {
  for_each = local.subnets

  name          = each.key
  cidr          = each.value.cidr

  vpc_id        = sbercloud_vpc.vpc-1.id
  primary_dns   = "100.125.13.59"
  secondary_dns = "8.8.8.8"
}
```

Подробности об использовании мета-аргумента `for_each` см. в документации Terraform[261].

provider

По умолчанию Terraform определяет нужный тип провайдера по типу ресурса. Мета-аргумент `provider` позволяет переопределить это поведение, явно указав провайдер, который должен использоваться для управления ресурсами.

Для использования мета-аргумента `provider` выполните следующие действия:

1. Добавьте блок `provider` без поля `alias`. В этом блоке опишите настройки провайдера, используемого по умолчанию, например:

   ```
   provider "yandex" {
     # ...
     zone      = "ru-central1-a"
   }
   ```

2. Добавьте блок `provider`, содержащий поле `alias`. Опишите специфичные настройки провайдера, а в поле `alias` укажите псевдоним, который будет использоваться для обращения к нему, например:

   ```
   provider "yandex" {
     alias     = "zone-b"
     # ...
     zone      = "ru-central1-b"
   }
   ```

3. Добавьте поле `provider` к описанию ресурса. В значении укажите нужный провайдер в формате `<name>.<alias>`, где:

   - `<name>` – название провайдера;

[261] https://developer.hashicorp.com/terraform/language/meta-arguments/for_each

- <alias> – псевдоним.

```
resource "yandex_compute_instance" "vm-1" {
  provider = yandex.zone-b
  # ...
}
```

Подробности об использовании мета-аргумента `provider` см. в документации Terraform[262].

lifecycle

В разделе *Ресурс* описано стандартное поведение Terraform при применении конфигурации. Мета-аргумент `lifecycle` позволяет тонко настроить этот процесс.

В значении мета-аргумента `lifecycle` можно указать значения следующих полей:

- `create_before_destroy`

  Тип: `bool`

  По умолчанию, если Terraform не может обновить ресурс «на лету», происходит следующее:

  1. Существующий ресурс нужного типа удаляется.
  2. Создается новый ресурс, который используется вместо удаленного.

  Мета-аргумент `create_before_destroy` позволяет изменить этот порядок. При значении `true` новый ресурс будет создан *до того*, как удален существующий.

- `prevent_destroy`

  Тип: `bool`

  При значении `true` этот мета-аргумент блокирует изменения ресурса, для применения которых его необходимо удалить и создать заново. Он особенно полезен, например, для таких ресурсов, как кластеры СУБД, хранилища данных и т. д.

- `ignore_changes`

  Тип: список атрибутов

  Когда Terraform обнаруживает различия между описанием ресурса и его фактическим состоянием в сервисе, он пытается изменить состояние ресурса таким образом, чтобы оно соответствовало описанию. Чтобы Terraform игнорировал изменение отдельных атрибутов ресурса, сделанное в сервисе, следует указать названия атрибутов в значении параметра `ignore_changes`.

  Специальное значение `all` позволяет Terraform игнорировать любые изменения ресурса.

- `replace_triggered_by`

  Тип: список ресурсов или ссылок на атрибуты

  В значении этого мета-атрибута следует указать список ресурсов или атрибутов, при изменении которых ресурс должен быть пересоздан.

Подробнее об управлении жизненным циклом ресурсов см. в документации Terraform[263].

[262] https://developer.hashicorp.com/terraform/language/meta-arguments/resource-provider
[263] https://developer.hashicorp.com/terraform/language/meta-arguments/lifecycle

Подключение модулей

Terraform позволяет группировать в модули (modules) ресурсы, используемые вместе. Это позволяет переиспользовать существующий код при описании инфраструктуры в новом проекте.

По умолчанию описание всех ресурсов размещается в одном модуле, называемом корневым (root).

Модуль может принимать входящие параметры и возвращать какие-либо значения. Таким образом, модуль может быть настроен для использования в проекте в соответствии с необходимыми требованиями.

Для подключения модуля из реестра Astra Automation Hub следует указать в поле `source` ссылку на соответствующий репозиторий, например:

```
module "vpc" {
  source = "ssh://git@hub.astra-automation.ru:2222/aa-gca/AMFT/yandex-cloud-vpc.git"
  # ...
}
```

Общие сведения о модулях Terraform, доступных в реестре Astra Automation Hub, приведены в разделе *Модули Terraform*.

Описание существующих модулей Terraform, доступных в реестре Astra Automation Hub, приведено в разделе *Модули Terraform*.

Подробности об использовании модулей см. в документации Terraform[264].

16.3.6 Vagrant

Vagrant[265] относится к инструментам для создания ВМ и объединения их в инфраструктуру в некотором виртуальном окружении, используя такие известные технологии, как VMware, VirtualBox и AWS. К такому виртуальному окружению относится, например, сервер с установленной на нем одной из упомянутых систем виртуализации.

Vagrant предоставляет развитый API, необходимый для автоматизации развертывания систем. Следуя парадигме *IAC*, указания для Vagrant создают в файле `Vagrantfile` в виде директив, объединенных в общий поток последовательных действий.

Применение

Приложение Vagrant удобно использовать для отладки развертывания пилотной инфраструктуры в локальной среде. Типовая последовательность использования Vagrant состоит из следующих шагов:

1. Создание и отладка потока управляющих директив на локальном узле с помощью VMware или VirtualBox.
2. Перенос полученного потока в производственную среду на базе облачных технологий или на собственные серверы.

В этом разделе рассматривается использование Vagrant совместно с VirtualBox.

[264] https://developer.hashicorp.com/terraform/language/modules
[265] https://developer.hashicorp.com/vagrant

Примечание: Приведенные ниже инструкции проверены в ОС Astra Linux Special Edition 1.7.4, 1.7.5 и 1.7.5uu1.

Установка

Приведенные ниже инструкции по установке и настройке ПО предназначены для ОС Astra Linux Special Edition 1.7. Пошаговые инструкции для других ОС доступны на сайтах разработчиков соответствующего ПО:

- Vagrant[266]
- VirtualBox[267]

Подготовка к установке

Подготовьте ОС к установке и настройке Vagrant и VirtualBox.

1. Если установка производится на ВМ, включите в ее настройках вложенную виртуализацию (nested virtualization). Для получения инструкций обратитесь к производителю ПО, используемого для виртуализации.
2. Получите номер установленного оперативного обновления ОС:

   ```
   cat /etc/astra_version
   ```

   Команда выводит в терминал строку вида:

   ```
   1.7.5uu1
   ```

3. Добавьте в файл `/etc/apt/sources.list` ссылки на базовый (base) и расширенный (extended) репозитории Astra Linux Special Edition 1.7:

   ```
   deb https://download.astralinux.ru/astra/frozen/1.7_x86-64/<version>/repository-
   ↪base/     1.7_x86-64 main contrib non-free
   deb https://download.astralinux.ru/astra/frozen/1.7_x86-64/<version>/repository-
   ↪extended/ 1.7_x86-64 main contrib non-free
   ```

   где <version> – версия установленного оперативного обновления Astra Linux Special Edition. Для примера выше указанные строки имеют вид:

   ```
   deb https://download.astralinux.ru/astra/frozen/1.7_x86-64/1.7.5/uu/1/repository-
   ↪base/     1.7_x86-64 main contrib non-free
   deb https://download.astralinux.ru/astra/frozen/1.7_x86-64/1.7.5/uu/1/repository-
   ↪extended/ 1.7_x86-64 main contrib non-free
   ```

4. Обновите список доступных пакетов:

   ```
   sudo apt update
   ```

5. Получите номер версии и вариант сборки установленного ядра:

[266] https://developer.hashicorp.com/vagrant/install
[267] https://www.virtualbox.org/wiki/Downloads

```
uname -r
```

В терминал выводится строка вида:

```
6.1.50-1-generic
```

Здесь `6.1.50-1` – версия ядра, `generic` – вариант сборки.

6. Установите менеджер загрузок `wget`, заголовочные файлы используемого ядра и утилиты, необходимые для сборки модулей ядра:

```
sudo apt install build-essential linux-headers-<version>-<build_variant> make␣
↪wget --yes
```

где <version> и <build_variant> – версия ядра и вариант его сборки соответственно. Для примера выше нужная команда имеет вид:

```
sudo apt install build-essential linux-headers-6.1.50-1-generic make wget --yes
```

Совет: Шаги получения информации о версии ядра и установки его заголовочных файлов можно объединить в одну команду:

```
sudo apt install build-essential linux-headers-$(uname -r) make wget --yes
```

Установка VirtualBox

Для установки VirtualBox выполните следующие действия:

1. Загрузите пакет `libvpx5` из репозиториев Debian Linux, например:

```
wget http://security.debian.org/debian-security/pool/updates/main/libv/libvpx/
↪libvpx5_1.7.0-3+deb10u2_amd64.deb
```

Совет: Актуальная ссылка на загрузку доступна на странице пакета: https://packages.debian.org/buster/amd64/libvpx5/download

---

2. Установите загруженный пакет:

```
sudo dpkg -i libvpx5_1.7.0-3+deb10u2_amd64.deb
```

3. Импортируйте ключ репозитория VirtualBox:

```
wget https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- |\
    sudo gpg --dearmor --yes --output /etc/apt/trusted.gpg.d/oracle.gpg
```

4. Создайте в каталоге `/etc/apt/sources.list.d/` файл `virtualbox.list` со следующим содержимым:

```
deb [arch=amd64] https://download.virtualbox.org/virtualbox/debian buster contrib
```

5. Обновите список доступных пакетов:

```
sudo apt update
```

6. Установите пакет `virtualbox-7.0`:

```
sudo apt install virtualbox-7.0 --yes
```

7. Получите номер установленной версии VirtualBox:

```
apt policy virtualbox-7.0
```

В терминал выводятся строки вида:

```
virtualbox-7.0:
  Установлен: 7.0.16-162802~Debian~buster
  Кандидат:   7.0.16-162802~Debian~buster
  Таблица версий:
 *** 7.0.16-162802~Debian~buster 500
        500 https://download.virtualbox.org/virtualbox/debian buster/contrib␣
↪amd64 Packages
        100 /var/lib/dpkg/status
```

Здесь `7.0.16` – версия VirtualBox.

8. Загрузите Extension Pack[268] для установленной версии VirtualBox:

```
wget https://download.virtualbox.org/virtualbox/<version>/Oracle_VM_VirtualBox_
↪Extension_Pack-<version>.vbox-extpack
```

где <version> – версия VirtualBox.

Для примера выше указанная команда имеет вид:

```
wget https://download.virtualbox.org/virtualbox/<version>/Oracle_VM_VirtualBox_
↪Extension_Pack-7.0.16.vbox-extpack
```

9. Установите Extension Pack:

```
sudo vboxmanage extpack install --replace <path>
```

где <path> – путь к загруженному ранее файлу.

Для примера выше указанная команда имеет вид:

```
sudo vboxmanage extpack install --replace ./Oracle_VM_VirtualBox_Extension_Pack-7.
↪0.16.vbox-extpack
```

[268] https://www.virtualbox.org/wiki/Downloads

Установка Vagrant

Для установки Vagrant выполните следующие действия:

1. Загрузите DEB-пакет Vagrant с сайта разработчика, например:

```
wget https://releases.hashicorp.com/vagrant/2.4.1/vagrant_2.4.1-1_amd64.deb
```

Примечание: Если сайт разработчика по какой-либо причине недоступен, используйте для загрузки зеркало репозитория:

```
wget https://releases.comcloud.xyz/vagrant/2.4.1/vagrant_2.4.1-1_amd64.deb
```

---

2. Установите загруженный пакет:

```
sudo dpkg -i vagrant_2.4.1-1_amd64.deb
```

3. Определите путь к каталогу, в котором хранятся расширения Vagrant:

```
dpkg -L vagrant | egrep 'plugins/guests$'
```

В терминал выводится строка вида:

```
/opt/vagrant/embedded/gems/gems/vagrant-<version>/plugins/guests
```

где <version> – номер установленной версии Vagrant.

4. Загрузите архив с расширением для Vagrant, позволяющим ему работать с образами Astra Linux:

```
wget https://dl.astralinux.ru/files/astra-vagrant.tar.gz
```

5. Распакуйте загруженный архив в каталог с расширениями Vagrant, например:

```
sudo tar xf astra-vagrant.tar.gz -C <plugins_dir>
```

где <plugins_dir> – путь к каталогу с расширениями Vagrant.

Проверка корректности установки

Для проверки корректности установки Vagrant и VirtualBox выполните следующие действия:

1. В любом каталоге создайте `Vagrantfile` с простейшей конфигурацией ВМ:

```
# frozen_string_literal: true

Vagrant.configure("2") do |config|
  config.vm.box = "alse-vanilla-base/1.7.5uu1"
  config.vm.box_url = "https://dl.astralinux.ru/vagrant/alse-vanilla-base%2F1.7.
↪5uu1"

  config.vm.define "VG" do |conf|
    conf.vm.hostname = "VG"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  end
end
```

В этом файле описана ВМ с названием VG, использующая образ с Astra Linux Special Edition 1.7.5uu1.

Подробности об именовании и составе образов см. в разделе *Образы виртуальных машин*.

2. Для создания и запуска ВМ выполните команду:

```
vagrant up
```

3. Подключитесь к созданной ВМ по SSH:

```
vagrant ssh
```

При успешном подключении приглашение командной строки меняется на следующее:

```
vagrant@VG:~$
```

4. Для отключения от созданной ВМ выполните команду:

```
exit
```

Подготовка инфраструктуры

Приведенные ниже инструкции описывают типовую последовательность действий для создания инфраструктуры ВМ, управляемой с помощью Vagrant.

ВМ на локальной машине

Для создания ВМ с установленной операционной системой Astra Linux выполните следующие действия:

1. Создайте пару ключей SSH, используемых для подключения к ВМ:

```
ssh-keygen -C "<comment>" -f ~/.ssh/deployment -N ""
```

В результате выполнения команды в каталоге `~/.ssh/` создаются два файла, содержащие приватный (`deployment`) и публичный (`deployment.pub`) ключи SSH соответственно.

2. Создайте файл `Vagrantfile` со следующим содержимым:

```
# -*- frozen_string_literal: true -*-

Vagrant.configure('2') do |config|
  config.vm.box = 'alse-vanilla-base/1.7.5uu1' # Название бокса

  # Ссылка на образ
  config.vm.box_url = 'https://dl.astralinux.ru/vagrant/alse-vanilla-base%2F1.7.
↪5uu1'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  config.vm.provision 'file',
                      source: '~/.ssh/deployment.pub',
                      destination: '/home/vagrant/.ssh/id_rsa.pub'

  config.vm.provision 'shell',
                      inline: 'cat /home/vagrant/.ssh/id_rsa.pub >> /home/vagrant/
↪.ssh/authorized_keys'

  # Настройка параметров ВМ
  config.vm.define 'host01' do |node|
    node.vm.hostname = 'host01' # Короткое имя хоста

    # Настройки, специфичные для VirtualBox
    node.vm.provider 'virtualbox' do |vb|
      vb.cpus = 4 # Кол-во ядер CPU
      vb.memory = 4096 # Объем RAM, МБ
    end

    # Параметры сети
    node.vm.network 'private_network',
                    ip: '192.168.56.11', # Статический IP-адрес
                    netmask: '255.255.255.0', # Маска подсети
                    dhcp_enabled: false # Запрет использования DHCP
  end
end
```

Имя бокса Vagrant задается в строке:

```
config.vm.box = 'alse-vanilla-base/1.7.5uu1' # Название бокса
```

После этого указывается ссылка на образ:

```
config.vm.box_url = 'https://dl.astralinux.ru/vagrant/alse-vanilla-base%2F1.7.5uu1
↪'
```

Чтобы получить доступ к ВМ по SSH, необходимо в подкаталоге `~/.ssh/` нужного пользователя ВМ разместить публичный ключ SSH, а также добавить его в список доверенных ключей:

```
config.vm.provision 'file',
                    source: '~/.ssh/deployment.pub',
                    destination: '/home/vagrant/.ssh/id_rsa.pub'

config.vm.provision 'shell',
                    inline: 'cat /home/vagrant/.ssh/id_rsa.pub >> /home/vagrant/.
↪ssh/authorized_keys'
```

Короткое имя хоста (`vm`) задается в параметре `node.vm.hostname`:

```
node.vm.hostname = 'vm' # Короткое имя хоста
```

В этих строках задается количество доступных ВМ ядер CPU и МБ RAM:

```
vb.cpus = 4 # Кол-во ядер CPU
vb.memory = 4096 # Объем RAM, МБ
```

Настройки подключения ВМ к сети:

```
node.vm.network 'private_network',
                ip: '192.168.56.11', # Статический IP-адрес
                netmask: '255.255.255.0', # Маска подсети
                dhcp_enabled: false # Запрет использования DHCP
```

3. Для проверки корректности синтаксиса созданного `Vagrantfile` выполните команду:

```
vagrant validate
```

При отсутствии ошибок в терминал выводится строка:

```
Vagrantfile validated successfully.
```

В случае наличия ошибок исправьте их и выполните повторную проверку синтаксиса `Vagrantfile`.

4. Для создания и запуска VM выполните команду:

```
vagrant up
```

Дождитесь завершения выполнения команды, это может занять некоторое время.

5. Проверьте состояние ВМ:

```
vagrant status
```

Она должна быть активна (состояние `running`):

```
Current machine states:

vm               running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
susptend the virtual machive. In either case, to restart it again,
simply run `vagrant up`.
```

Если в процессе создания или запуска ВМ будут выведены сообщения об ошибках, выполните следующие действия:

1. Проверьте содержимое `Vagrantfile` на наличие ошибок в параметрах настройки ВМ.
2. Удалите существующую ВМ и создайте ее заново:

```
vagrant destroy --force && vagrant up
```

Инфраструктура из нескольких ВМ

Следующий пример файла Vagrantfile демонстрирует задание на создание инфраструктуры из двух ВМ:

Содержимое файла Vagrantfile

```
# frozen_string_literal: true

boxes = {
  'dc01' => {
    box: 'alse-vanilla-base/1.7.5uu1',
    box_url: 'https://dl.astralinux.ru/vagrant/alse-vanilla-base%2F1.7.5uu1',
    cpu_cores: '2',
    memory: '2048',
    ipv4: '192.168.56.10'
  },
  'dc02' => {
    box: 'alse-vanilla-base/1.7.5uu1',
    box_url: 'https://dl.astralinux.ru/vagrant/alse-vanilla-base%2F1.7.5uu1',
    cpu_cores: '2',
    memory: '2048',
    ipv4: '192.168.56.11'
  }
}

Vagrant.configure(2) do |config|
  boxes.each do |hostname, cfg|
    config.vm.define hostname do |host|
      host.vm.boot_timeout = 600
      host.vm.hostname = hostname
      host.vm.box = cfg[:box]
      host.vm.box_url = cfg[:box_url]
      host.vm.provider 'virtualbox' do |v|
        v.cpus = cfg[:cpu_cores]
        v.memory = cfg[:memory]
      end
      host.vm.network 'private_network', ip: cfg[:ipv4]
    end
  end
end
```

Настройка состоит из двух частей:

- Первая часть boxes задает параметры двух ВМ.
- Вторая часть описывает создание этих ВМ с использованием параметров, заданных в структуре boxes.

Удаление ВМ и образов

Для удаления неиспользуемых ресурсов выполните следующие действия:

1. Определите список существующих ВМ:

```
vagrant global-status
```

2. Удалите ненужные ВМ с помощью команды:

```
vagrant destroy <VM_name>
```

3. Получите список образов, загруженных на компьютер:

```
vagrant box list
```

4. Удалите ненужные образы:

```
vagrant box remove <box_name>
```

Г

ГИС, **407**

С

СЗИ, **407**
СЗИ ALSE, **407**

A

ALSE, **405**
AMFT, **405**
ARFA, **405**

B

-b
    command line option, 42
--backup
    command line option, 42

C

--check-releases
    command line option, 43
CIDR, **405**
Cloud-Init, **405**
command line option
    -b, 42
    --backup, 42
    --check-releases, 43
    --component, 43
    -d, 42
    --debug, 42
    --default-job-ee, 42
    --force-postgres-removal, 42
    --generate-key, 42
    -h, 42
    --help, 42
    -i, 42
    --inventory, 42
    -k, 42
    --log-path, 42
    -p, 42
    --plain, 42
    --product-version, 43
    -r, 42
    --repo-url, 42
    --restore, 42
    -u, 42
    --uninstall, 42
    --upgrade, 43
    -v, 43
    --version, 43
--component
    command line option, 43

D

-d
    command line option, 42
DAG, **405**
--debug
    command line option, 42
--default-job-ee
    command line option, 42

E

EE, **405**

F

--force-postgres-removal
    command line option, 42
FQCN, **406**
FQDN, **406**

G

--generate-key
    command line option, 42

H

-h
    command line option, 42

HA, **406**
HCL, **406**
--help
    command line option, 42

I

-i
    command line option, 42
IaC, **406**
--inventory
    command line option, 42
IPA, **406**

K

-k
    command line option, 42

L

--log-path
    command line option, 42

O

OCFS2, **406**

P

-p
    command line option, 42
--plain
    command line option, 42
--product-version
    command line option, 43

R

-r
    command line option, 42
RBAC, **406**
--repo-url
    command line option, 42
--restore
    command line option, 42

U

-u
    command line option, 42
--uninstall
    command line option, 42
--upgrade
    command line option, 43

V

-v
    command line option, 43
VCS, **406**
--version
    command line option, 43

Y

YC CLI, **406**