Работа с Ansible: шпаргалка

Специально для t.me/i_odmin_book

Ansible – это современный инструмент управления конфигурацией, который облегчает задачу настройки и обслуживания удаленных серверов.

Этот мануал написан в виде шпаргалки. По сути это краткий справочник по командам и методам, обычно используемым при работе с Ansible. Краткий обзор Ansible и инструкции по установке и настройке вы найдете в руководстве Установка и настройка Ansible в Ubuntu 18.04.

Как пользоваться этим мануалом:

- Найдите раздел, который описывает интересующую вас задачу.
- Выделенный текст в командах – условные данные, которые нужно заменить своими данными. Обычно это хосты, имена пользователей и IP-адреса из вашего инвентаря.

Краткий словарь терминов Ansible

В этом руководстве широко используются такие термины Ansible:

- Control Machine (или Node): ведущая система, в которой установлен Ansible и откуда он может подключаться к нодам и выполнять на них команды.
- Нода: сервер, управляемый Ansible.
- Файл инвентаря: файл, который содержит информацию о серверах, которыми управляет Ansible, обычно находится в /etc/ansible/hosts.
- Плейбук (Playbook): файл, содержащий серию задач, которые нужно выполнить на удаленном сервере.
- Роль: коллекция плейбуков и других файлов, которые имеют отношение к цели (например, к установке веб-сервера).
- Play: полный набор инструкций Ansible. В play может быть несколько плейбуков и ролей, включенных в один плейбук, который служит точкой входа.

Проверка подключения к нодам

Чтобы убедиться, что Ansible может подключаться к нодам и запускать команды и плейбуки, вы можете использовать следующую команду:

```
ansible all -m ping
```

Модуль ping проверит, есть ли у вас валидные учетные данные для подключения к нодам, определенным в файле инвентаря, и может ли Ansible запускать сценарии Python на удаленном сервере. Ответ pong означает, что Ansible готов запускать команды и плейбуки на этой ноде.

Подключение с помощью другого пользователя

По умолчанию Ansible пытается подключиться к нодам в качестве текущего пользователя системы через соответствующую пару ключей SSH. Чтобы подключиться как другой пользователь, добавьте команду с флагом -u и именем нового пользователя:

```
ansible all -m ping -u disnetern
```

То же самое относится и к ansible-playbook:

```
ansible-playbook myplaybook.yml -u disnetern
```

Настройка пользовательского ключа SSH

Если вы используете свой ключ SSH для подключения к удаленным серверам, вы можете предоставить его с помощью параметра –private-key:

```
ansible all -m ping --private-key=~/.ssh/custom_id
```

Эта опция также работает для ansible-playbook:

```
ansible-playbook myplaybook.yml --private-key=~/.ssh/custom_id
```

Настойка парольной аутентификации

Если вам нужно использовать парольную аутентификацию для подключения к нодам, добавьте опцию –ask-pass к команде Ansible.

Это заставит Ansible запросить у вас пароль пользователя на удаленном сервере, к которому вы пытаетесь подключиться:

```
ansible all -m ping --ask-pass
```

Эта опция также действительна для ansible-playbook:

```
ansible-playbook myplaybook.yml --ask-pass
```

Использование пароля sudo

Если удаленный пользователь должен предоставить пароль для запуска команд sudo, вы можете включить опцию –ask-become-pass в команду Ansible. Опция позволит вам ввести пароль sudo удаленного пользователя:

```
ansible all -m ping --ask-become-pass
```

Эта опция также действительна для ansible-playbook:

```
ansible-playbook myplaybook.yml --ask-become-pass
```

Пользовательский файл инвентаря

Файл инвентаря по умолчанию обычно находится в /etc/ansible/hosts, но вы можете использовать опцию -i для указания пользовательских файлов при запуске команд и плейбуков Ansible. Это удобный способ настройки индивидуального инвентаря для каждого проекта, который можно включить в системы контроля версий, такие как Git:

```
ansible all -m ping -i my_custom_inventory
```

Такая опция действительна и для ansible-playbook:

```
ansible-playbook myplaybook.yml -i my_custom_inventory
```

Динамический файл инвентаря

Ansible поддерживает сценарии инвентаризации для создания динамических файлов. Это полезно, если ваш инвентарь часто меняется, когда серверы создаются и уничтожаются.

Вы можете найти ряд скриптов с открытым исходным кодом в официальном репозитории Ansible GitHub. После загрузки требуемого сценария на Ansible control machine и настройки необходимых параметров (например, учетных данных API) вы можете запустить исполняемый файл в качестве пользовательского инвентаря с любой командой Ansible, которая поддерживает эту опцию.

Следующая команда использует скрипт инвентаря my_inventory.py с командой ping для проверки подключения ко всем текущим активным серверам:

```
ansible all -m ping -i my_inventory.py
```

За более подробной информацией о том, как использовать динамические файлы инвентаризации, пожалуйста, обратитесь к официальной документации Ansible.

Запуск специальных команд

Чтобы выполнить любую команду на ноде, используйте параметр -a, а затем укажите нужную команду в кавычках.

Например, следующая команда выполнит uname -a на всех нодах в вашем инвентаре:

```
ansible all -a "uname -a"
```

Также с помощью опции -m можно запускать модули Ansible. Следующая команда установит пакет vim на server1 из вашего инвентаря:

```
ansible server1 -m apt -a "name=vim"
```

Прежде чем вносить изменения на свои ноды, вы можете выполнить пробный прогон, чтобы увидеть, как ваша команда повлияет на серверы. Это можно сделать, включив параметр –check:

```
ansible server1 -m apt -a "name=vim" --check
```

Запуск плейбуков

Чтобы запустить плейбук и выполнить все определенные в нем задачи, используйте команду ansible-playbook:

```
ansible-playbook myplaybook.yml
```

Чтобы перезаписать в плейбуке опцию hosts по умолчанию и ограничить выполнение определенной группой или хостом, включите в команду опцию -l:

```
ansible-playbook -l server1 myplaybook.yml
```

Запрос информации о play

Опция –list-tasks используется для перечисления всех задач, которые будут выполнены в play, при этом не внося никаких изменений на удаленные серверы:

```
ansible-playbook myplaybook.yml --list-tasks
```

Точно так же можно запросить все хосты, которые будут затронуты выполнением play, без запуска каких-либо задач на удаленных серверах:

```
ansible-playbook myplaybook.yml --list-hosts
```

Вы можете использовать теги, чтобы ограничить выполнение play. Чтобы вывести список всех тегов, доступных в play, используйте параметр –list-tags:

```
ansible-playbook myplaybook.yml --list-tags
```

Управление выполнением плейбука

Вы можете использовать опцию –start-at-task, чтобы определить новую точку входа вашего плейбука. Затем Ansible пропустит все, что предшествует указанной задаче, выполнив оставшуюся часть play с заданного момента. Эта опция в качестве аргумента требует правильное имя задачи:

```
ansible-playbook myplaybook.yml --start-at-task="Set Up Nginx"
```

Чтобы выполнять только задачи, связанные с конкретными тегами, вы можете использовать опцию –tags. Например, если вы хотите выполнить только задачи, помеченные как nginx или mysql, вы можете использовать:

```
ansible-playbook myplaybook.yml --tags=mysql,nginx
```

Если вы хотите пропустить все задачи, которые находятся под определенными тегами, используйте –skip-tags. Следующая команда будет выполнять myplaybook.yml, пропуская все задачи, помеченные как mysql:

```
ansible-playbook myplaybook.yml --skip-tags=mysql
```

Ansible Vault для хранения конфиденциальных данных

Если ваши плейбуки Ansible содержат конфиденциальные данные, такие как пароли, ключи API и учетные данные, важно обеспечить их безопасность с помощью шифрования. Ansible предоставляет ansible-vault для шифрования файлов и переменных.

Несмотря на то, что любой файл данных Ansible, а также двоичные файлы, возможно зашифровать изначально, чаще для шифрования переменных файлов, содержащих конфиденциальные данные, используется ansible-vault. После шифрования файла с помощью этого инструмента вы сможете выполнять, редактировать или просматривать его, только предоставив соответствующий пароль, указанный при первом шифровании файла.

Создание нового зашифрованного файла

Вы можете создать новый зашифрованный файл Ansible с помощью:

```
ansible-vault create credentials.yml
```

Эта команда выполнит следующие действия:

1. Сначала вам будет предложено ввести новый пароль. Вам нужно будет указывать этот пароль при каждом доступе к содержимому файла, будь то редактирование, просмотр или просто запуск плейбука (или команд с использованием его значений).
2. Затем откроется редактор командной строки по умолчанию, чтобы вы могли заполнить файл требуемым содержимым.
3. Наконец, когда вы закончите редактирование, ansible-vault сохранит файл как зашифрованный.

Шифрование существующего файла Ansible

Чтобы зашифровать существующий файл Ansible, вы можете использовать следующую команду:

```
ansible-vault encrypt credentials.yml
```

Эта команда запросит у вас пароль, который вам нужно будет вводить при каждом доступе к файлу credentials.yml.

Просмотр содержимого зашифрованного файла

Если вы хотите просмотреть содержимое файла, который ранее был зашифрован с помощью ansible-vault, и вам не нужно изменять его содержимое, вы можете использовать команду:

```
ansible-vault view credentials.yml
```

Она предложит вам указать пароль, который вы выбрали при первом шифровании файла с помощью ansible-vault.

Редактирование зашифрованного файла

Чтобы изменить содержимое файла, который ранее был зашифрован с помощью Ansible Vault, выполните:

```
ansible-vault edit credentials.yml
```

Эта команда предложит вам указать пароль, который вы выбрали при первом шифровании файла credentials.yml. После проверки пароля откроется редактор командной строки по умолчанию с незашифрованным содержимым файла, что позволит вам внести нужные изменения. По завершении вы можете сохранить и закрыть файл, как обычно, и обновленное содержимое будет сохранено и зашифровано.

Расшифровка файлов

Если вы хотите навсегда расшифровать файл, ранее зашифрованный с помощью ansible-vault, вы можете сделать это с помощью следующего синтаксиса:

```
ansible-vault decrypt credentials.yml
```

Эта команда предложит вам ввести тот пароль, который использовался при первом шифровании файла. После проверки пароля содержимое файла будет сохранено на диск в виде незашифрованных данных.

Использование нескольких паролей

Ansible поддерживает для хранилища несколько паролей, сгруппированных по разным идентификаторам. Это полезно, если вы хотите иметь выделенные пароли хранилища для различных сред – для разработки, тестирования и производства.

Чтобы создать новый зашифрованный файл с пользовательским идентификатором хранилища, включите параметр –vault-id вместе с меткой и расположением, где ansible-vault может найти пароль для этого хранилища. Метка может быть любой, а расположение может быть либо prompt (что означает, что команда должна предложить вам ввести пароль), либо путь к файлу паролей.

```
ansible-vault create --vault-id dev@prompt credentials_dev.yml
```

Это создаст новый идентификатор по имени dev, который использует prompt для получения пароля. Комбинируя этот метод с файлами переменных группы, вы сможете создать отдельные хранилища для каждой среды приложения:

```
ansible-vault create --vault-id prod@prompt credentials_prod.yml
```

Мы использовали dev и prod в качестве идентификаторов хранилищ, чтобы продемонстрировать, как вы можете создавать отдельные хранилища для каждой среды. Самостоятельно вы можете создать столько хранилищ, сколько захотите, и использовать любой ID.

Теперь, чтобы просмотреть, отредактировать или расшифровать эти файлы, вам необходимо предоставить тот же ID хранилища и источник пароля вместе с командой ansible-vault:

```
ansible-vault edit credentials_dev.yml --vault-id dev@prompt
```

Использование файла паролей

Если вам нужно автоматизировать процесс инициализации серверов в Ansible с помощью стороннего инструмента, вам потребуется способ ввода пароля хранилища без его запроса. Вы можете сделать это, используя файл паролей через ansible-vault.

Файл паролей может быть простым текстовым файлом или исполняемым скриптом. Если файл является исполняемым, выходные данные, созданные ним, будут использоваться в качестве пароля хранилища. В противном случае в качестве пароля хранилища будет использоваться необработанное содержимое файла.

Чтобы применить файл паролей в ansible-vault, необходимо указать путь к файлу паролей при выполнении любой из команд vault:

```
ansible-vault create --vault-id dev@path/to/passfile credentials_dev.yml
```

Ansible не различает контент, который был зашифрован с помощью prompt, и простой файл пароля при условии, что входной пароль один и тот же. С практической точки зрения это означает, что файл можно зашифровать, используя prompt, а затем создать файл пароля для хранения того же пароля, который использовался в методе prompt. Также верно и обратное: вы можете зашифровать содержимое, используя файл паролей, а затем использовать метод prompt, предоставляя тот же пароль при запросе Ansible.

Для большей гибкости и безопасности, чтобы не хранить свой пароль в текстовом файле, вы можете использовать скрипт Python для получения пароля из других источников. Официальный репозиторий Ansible содержит несколько примеров сценариев, которые вы можете использовать для справки при создании своего скрипта под потребности вашего проекта.

Запуск плейбука с зашифрованными данными

Каждый раз, когда вы запускаете плейбук, в котором используются данные, ранее зашифрованные с помощью ansible-vault, вам нужно будет указывать пароль хранилища в команде playbook.

Если вы использовали параметры по умолчанию и prompt при шифровании данных плейбука, вы можете использовать опцию –ask-vault-pass, чтобы Ansible запрашивал пароль:

```
ansible-playbook myplaybook.yml --ask-vault-pass
```

Если вы использовали файл пароля вместо prompt, вы должны использовать опцию –vault-password-file:

```
ansible-playbook myplaybook.yml --vault-password-file my_vault_password.py
```

Если вы используете данные, зашифрованные с помощью ID, вам нужно указать тот же ID хранилища и источник пароля, который вы использовали при первом шифровании данных:

```
ansible-playbook myplaybook.yml --vault-id dev@prompt
```

Если вы используете файл пароля с ID, вы должны указать метку, а затем полный путь к файлу пароля в качестве источника:

```
ansible-playbook myplaybook.yml --vault-id dev@vault_password.py
```

Если ваш play использует несколько хранилищ, вы должны добавить параметр –vault-id для каждого из них в произвольном порядке:

```
ansible-playbook myplaybook.yml --vault-id dev@vault_password.py --vault-id test@prompt --vault-id ci@prompt
```

Устранение неполадок

Если вы сталкиваетесь с ошибками при выполнении команд и плейбуков, рекомендуется увеличить детализацию вывода, чтобы получить больше информации о проблеме. Вы можете сделать это, включив в команду параметр -v:

```
ansible-playbook myplaybook.yml -v
```

Если вам нужно больше деталей, вы можете использовать -vvv, и это увеличит детализацию вывода. Если вы не можете подключиться к удаленным нодам через Ansible, используйте -vvvv для получения информации об отладке соединения:

```
ansible-playbook myplaybook.yml -vvvv
```

Заключение

В этом мануале рассматриваются наиболее распространенные команды Ansible, которые вы можете использовать при подготовке серверов. Также вы узнали, как удаленно выполнять команды на ваших нодах, как запускать плейбуки и использовать различные пользовательские настройки.

Существует еще много вариантов команд и флагов, которые могут пригодиться вам в работе с Ansible. Чтобы получить обзор всех доступных опций, вы можете использовать команду help:

```
ansible  --help
```