Управление конфигурацией. Основные DevOps инструменты. Знакомство с Ansible

Проект infra и проверка ДЗ

Создайте новую ветку в вашем инфраструктурном репозитории для выполнения данного ДЗ. Т.к. это первое задание, посвященое работе с Ansible, то ветку назовите `ansible-1`.

Проверка данного ДЗ будет производиться через Pull Request ветки с ДЗ к ветке `master`.

После того, как один из преподавателей сделает approve пул реквеста, ветку с ДЗ нужно смерджить.

План

- Установка Ansible
- Знакомство с базовыми функциями и инвентори
- Выполнение различных модулей на подготовленной в прошлых ДЗ инфраструктуре
- Пишем простой плейбук

Ansible, установка и настройка клиента на рабочую машину

Работа в Ansible гарантирована в Linux/Unix машинах и в подситеме WSL Windows 10.

Добейтесь установки Python 2.7 и/или проверьте что установлена нужная версия

```
$ python --version
```

Рекомендуется также поставить пакетный менеджер `pip` или `easy_install`

Ansible, установка и настройка клиента

Создайте в корне вашего инфраструктурного репозитория директорию `ansible`. Вся дальнейшая работа с Ansible будет производится в ней.

Создайте в директории `ansible` файл `requirements.txt` с содержимым, приведенным по ссылке

Любым из пакетных менеджеров установите ansible, примеры:

```
pip install -r requirements.txt
pip install ansible>=2.4
easy_install `cat requirements.txt`
```

Ansible, установка и настройка клиента

При невозможности установить Ansible из `pip`, можно попробовать установить его с помощью пакетного менеджера системы (`apt`, `yum`, ...).

Проверяем, что Ansible установлен:

```
$ ansible --version
ansible 2.4.x.x
```

Официальная документация по установке доступна по ссылке

Принципы работы Ansible

Ansible управляет инстансами виртуальных машин (с Linux ОС) используя SSH-соединение. Поэтому для управление инстансом при помощи Ansible нам нужно убедиться, что мы можем подключиться к нему по SSH.

Для управления хостами при помощи Ansible на них также *должен быть установлен Python >=2.7*

Запуск VMs

Поднимите инфраструктуру, описанную в окружении `stage`. Проверьте output-переменную для определения внешнего IP-адреса инстанса приложения и проверьте SSH доступ к нему.

```
$ terraform apply
...
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:
app_external_ip = 35.195.186.154
```

Inventory file

После того, как мы создали инстансы VM, мы можем использовать Ansible для выполнения различных команд на данных машинах.

Для этого нам нужно сказать Ansible, какими инстансами (хостами в терминологии Ansible) ему управлять.

Хосты и группы хостов, которыми Ansible должен управлять, описываются в инвентори-файле.

Inventory file

Создадим инвентори файл `ansible/inventory`, в котором укажем информацию о созданном инстансе приложения и параметры подключения к нему по SSH:

```
appserver ansible_host=35.195.186.154 ansible_user=appuser \
    ansible_private_key_file=~/.ssh/appuser
```

где **appserver** - краткое имя, которое идентифицирует данный хост.

Обратите внимание, что это должна быть *одна строка* в файле `ansible/inventory`, и не забудьте поменять IP.

Управление хостом при помощи Ansible

Убедимся, что Ansible может управлять нашим хостом. Используем команду `ansible` для вызова модуля `ping` из командной строки.

```
$ ansible appserver -i ./inventory -m ping

The authenticity of host '35.195.186.154 (35.195.186.154)' can't be established.
ECDSA key fingerprint is SHA256:UdoT2Rgc/hY+rBeCX/KLzBULgMeoLZO8awefRbMNUVQ.
Are you sure you want to continue connecting (yes/no)? yes

appserver | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Управление хостом при помощи Ansible

Ping-модуль позволяет протестировать SSH-соединение, при этом ничего не изменяя на самом хосте.

```
$ ansible appserver -i ./inventory -m ping
```

- `-m ping` - вызываемый модуль
- `-i ./inventory` - путь до файла инвентори
- `appserver` - Имя хоста, которое указали в инвентори, откуда Ansible узнает, как подключаться к хосту
- вывод команды:

```
appserver | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Повторим для инстанса БД

Повторите такую же процедуру для инстанса БД.

- В инвентори файле `ansible/inventory` укажите название хоста `dbserver`.
- Напомним, что можно определить выходную переменную для внешнего адреса инстанса БД в Terraform, чтобы облегчить себе поиск нужной информации. Или использовать `terraform show`.
- Запустим Ansible

```
$ ansible dbserver -i inventory -m ping

dbserver | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Параметры ansible.cfg

Для того чтобы управлять инстансами нам приходится вписывать много данных в наш инвентори файл.

К тому же, чтобы использовать данный инвентори, нам приходится каждый раз указывать его явно, как опцию команды `ansible`. Многое из этого мы можем определить в конфигурации Ansible.

Для того чтобы настроить Ansible под нашу работу, создадим конфигурационный файл для него `ansible.cfg` в директории `ansible`.

Параметры ansible.cfg

Укажем значения по умолчанию для работы Ansible. Скопировать можно из данного gist.

ansible/ansible.cfg:

```
[defaults]
inventory = ./inventory
remote_user = appuser
private_key_file = ~/.ssh/appuser
host_key_checking = False
retry_files_enabled = False
```

Изменим инвентори

Теперь мы можем удалить избыточную информацию из файла `inventory` и использовать значения по умолчанию:

ansible/inventory:

```
appserver ansible_host=35.195.74.54
dbserver ansible_host=35.195.162.174
```

Проверим работу

Используем модуль `command`, который позволяет запускать произвольные команды на удаленном хосте.

Выполним команду `uptime` для проверки времени работы инстанса. Команду передадим как аргумент для данного модуля, использовав опцию `-a`:

```
$ ansible dbserver -m command -a uptime

dbserver | SUCCESS | rc=0 >>
07:47:41 up 24 min,  1 user,  load average: 0.00, 0.00, 0.03
```

Работа с группами хостов

Управлять при помощи Ansible отдельными хостами становится неудобно, когда этих хостов становится более одного.

В инвентори файле мы можем определить группу хостов для управления конфигурацией сразу нескольких хостов.

Список хостов указывается под названием группы, каждый новый хост указывается в новой строке. В нашем случае, каждая группа будет включать в себя всего один хост.

Сейчас мы определим группы хостов в инвентори файле (см. следующий слайд)

> **!** Не забывайте подставлять в конфигурацию свои IP-адреса

Работа с группами хостов

Изменим инвентори файл следующим образом:

ansible/inventory:

```
[app]   # ⬅ Это название группы
appserver ansible_host=35.195.74.54   # ⬅ Список хостов в данной группе

[db]
dbserver ansible_host=35.195.162.174
```

Проверим работу

Теперь мы можем управлять не отдельными хостами, а целыми группами, ссылаясь на имя группы:

```
$ ansible app -m ping

appserver | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

- `app` - имя группы
- `-m ping` - имя модуля Ansible
- `appserver` - имя сервера в группе, для которого применился модуль

Использование YAML inventory

Начиная с Ansible 2.4 появилась возможность использовать YAML для inventory. Давайте попробуем его использовать.

Создадим файл `inventory.yml` и перенесем в него записи из имеющегося inventory.

Документация

Использование YAML inventory

Для проверки выполним например следующую команду.

Ключ `-i` переопределяет путь к инвентори файлу

```
$ ansible all -m ping -i inventory.yml
dbserver | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
appserver | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

P.S. если есть затруднения с созданием аналогичного инвентори, вот gist

Выполнение команд

Теперь, когда у нас установлен Ansible и настроен инвентори, мы можем конфигурировать хосты и деплоить туда приложение.

Мы опишем это в коде в следующих ДЗ, а сейчас рассмотрим подробнее исполнение модулей по отдельности на хостах.

Как мы помним, на **app-сервере** у нас пакером при билде установлен *ruby*. А на **db-сервер** установлена *MongoDB*.

Попробуем не заходя на хосты, проверить наличие необходимых компонентов в созданном окружении.

Выполнение команд

Проверим, что на app сервере установлены компоненты для работы приложения (`ruby` и `bundler`):

```
$ ansible app -m command -a 'ruby -v'
appserver | SUCCESS | rc=0 >>
ruby 2.3.1p112 (2016-04-26) [x86_64-linux-gnu]
$ ansible app -m command -a 'bundler -v'
appserver | SUCCESS | rc=0 >>
Bundler version 1.11.2
```

А теперь попробуем указать две команды модулю `command`:

```
$ ansible app -m command -a 'ruby -v; bundler -v'
appserver | FAILED | rc=1 >>
ruby: invalid option -;  (-h will show valid options) (RuntimeError)non-zero
return code
```

Выполнение команд

В то же время модуль `shell` успешно отработает:

```
$ ansible app -m shell -a 'ruby -v; bundler -v'
appserver | SUCCESS | rc=0 >>
ruby 2.3.1p112 (2016-04-26) [x86_64-linux-gnu]
Bundler version 1.11.2
```

Модуль `command` выполняет команды, не используя оболочку (`sh`, `bash`), поэтому в нем не работают перенаправления потоков и нет доступа к некоторым переменным окружения.

Выполнение команд

Проверим на хосте с БД статус сервиса MongoDB с помощью модуля `command` или `shell`. (Эта операция аналогична запуску на хосте команды `systemctl status mongod`):

```
$ ansible db -m command -a 'systemctl status mongod'
dbserver | SUCCESS | rc=0 >>
● mongod.service - High-performance, schema-free document-oriented database
$ ansible db -m shell -a 'systemctl status mongod'
dbserver | SUCCESS | rc=0 >>
● mongod.service - High-performance, schema-free document-oriented database
```

А можем выполнить ту же операцию используя модуль `systemd`, который предназначен для управления сервисами:

```
$ ansible db -m systemd -a name=mongod
dbserver | SUCCESS => {
    "changed": false,
    "name": "mongod",
    "status": {
        "ActiveState": "active", ...
```

Выполнение команд

Или еще лучше с помощью модуля `service`, который более универсален и будет работать и в более старых ОС с init.d-инициализацией:

```
$ ansible db -m service -a name=mongod
dbserver | SUCCESS => {
    "changed": false,
    "name": "mongod",
    "status": {
        "ActiveState": "active", ...
```

Выполнение команд

На предыдущем примере с проверкой состояния сервиса можно увидеть преимущества использования модуля вместо запуска shell-команд.

Модуль возвращает в качестве ответа набор переменных, которые можно легко использовать для проверки в дальнейшем коде.

Например `status.ActiveState` содержит состояние сервиса. А для shell-команд нужно будет реализовывать проверку с помощью регулярных выражений, кодов возврата и других сложных и ненадежных решений.

Выполнение команд

Используем модуль `git` для клонирования репозитория с приложением на app сервер:

```
$ ansible app -m git -a \
    'repo=https://github.com/express42/reddit.git dest=/home/appuser/reddit'
appserver | SUCCESS => {
    "after": "61a7f75b3d3e6f7a8f279896fb4e9f0556e1a70a",
    "before": null,
    "changed": true
}
$ ansible app -m git -a \
    'repo=https://github.com/express42/reddit.git dest=/home/appuser/reddit'
appserver | SUCCESS => {
    "after": "61a7f75b3d3e6f7a8f279896fb4e9f0556e1a70a",
    "before": "5c217c565c1122c5343dc0514c116ae816c17ca2",
    "changed": false,
    "remote_url_changed": false
}
```

*Как мы видим, повторное выполнение этой команды проходит успешно, только переменная `changed` будет `false` (**что значит, что изменения не произошли**)*

Выполнение команд

И попробуем сделать то же самое с модулем `command`:

```
$ ansible app -m command -a \
    'git clone https://github.com/express42/reddit.git /home/appuser/reddit'
appserver | SUCCESS | rc=0 >>
Cloning into '/home/appuser/reddit'...

$ ansible app -m command -a \
    'git clone https://github.com/express42/reddit.git /home/appuser/reddit'
appserver | FAILED | rc=128 >>
fatal: destination path '/home/appuser/reddit' already exists and is not
an empty directory. non-zero return code
```

*А в этом примере, повторное выполнение завершается **ошибкой**.*

Напишем простой плейбук

Реализуем простой плейбук, который выполняет аналогичные предыдущему слайду действия (клонирование репозитория).

Создайте файл **ansible/clone.yml** gist:

```
- name: Clone
  hosts: app
  tasks:
    - name: Clone repo
      git:
        repo: https://github.com/express42/reddit.git
        dest: /home/appuser/reddit
```

И выполните: `ansible-playbook clone.yml`

Напишем простой плейбук

Результат примерно такой:

```
PLAY RECAP
*********************************************************************
appserver                  : ok=2    changed=0    unreachable=0    failed=0
```

Теперь выполните `ansible app -m command -a 'rm -rf ~/reddit'` и проверьте еще раз выполнение плейбука. Что изменилось и почему? Добавьте информацию в `README.md`.

Задание со ⭐

Вводная

Для описания инвентори Ansible использует форматы файлов INI и YAML. Также поддерживается формат JSON. При этом, Ansible поддерживает две различных схемы JSON-inventory: одна является прямым отображением YAML-формата (можно сделать через конвертер YAML <-> JSON), а другая используется для динамического inventory.

С небольшими ухищрениями можно заставить Ansible использовать вторую схему и для статических JSON-файлов. Попробуем это сделать...

Задание со ⭐

Описание задания:

1. Ознакомьтесь с документацией на формат JSON для динамического инвентори.
2. Создайте файл `inventory.json` в **формате, описанном в п.1** для нашей GCP-инфраструктуры и скрипт для работы с ним.
3. Добейтесь успешного выполнения команды `ansible all -m ping` и опишите шаги в `README`.
4. Добавьте параметры в файл `ansible.cfg` для работы с инвентори в формате JSON.
5. Если вы разобрались с отличиями схем JSON для динамического и статического инвентори, также добавьте описание в `README`

Задание со ⭐

Условия сдачи:

1. `inventory.json` должен быть в формате, описанном в п.1 (вариант JSON-инвентори для статической конфигурации **НЕ** засчитывается!)
2. Соответственно, файла `inventory.json` недостаточно 😉: должен быть написан скрипт, позволяющий Ansible использовать данный файл (или генерирующий его "на лету")
3. В файле `ansible.cfg` сделаны настройки для работы с JSON-inventory
4. `gce.py` и другие готовые утилиты здесь **не рассматривается в качестве решения**

Проверка ДЗ

- Результаты вашей работы находятся в ветке `ansible-1` вашего инфраструктурного репозитория
- В README внесите описание того, что сделано
- Создайте Pull Request к ветке master (**описание PR нужно заполнять**)
- В ревьюверы можно никого не добавлять
- Добавьте "Labels" `ansible` и `ansible-1` к вашему Pull Request
- После того, как один из преподавателей сделает approve пул реквеста, ветку с ДЗ можно смерджить и закрыть PR.

P.S. TravisCI проверяет наличие файлов ansible и наличие пустой строки в конце файлов.