

Деплой и управление конфигурацией с Ansible

Развитие проекта `infra`

В прошлых ДЗ вы создали инфраструктурный репозиторий `infra` на GitHub. Убедитесь что данный проект находится у вас на локальной машине.

Если у вас нет репозитория `infra` на GitHub, выполните сначала предыдущие ДЗ.

Проект *infra* и проверка ДЗ

Создайте новую ветку в вашем локальном репозитории для выполнения данного ДЗ. Т.к. это второе задание, посвященное работе с **Ansible**, то ветку можно назвать **ansible-2**.

Проверка данного ДЗ, как и многих последующих, будет производиться через Pull Request ветки с ДЗ к ветке мастер и добавлению в Reviewers пользователей **Artemmkin** и **Nklya**.

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



Обратите внимание!

Начиная с текущего ДЗ помогать проверять задания будет еще один инженер из компании "Экспресс 42" Николай Анциферов. Пожалуйста, добавьте пользователя **Nklya** в collaborators в настройках своего репозитория **infra** на GitHub. И при выполнении следующих заданий добавляйте этого пользователя в reviewers.

Пользователя **serjs** больше добавлять в reviewers **не** нужно.



Развитие проекта *infra*

В директории **ansible**, которую создали в прошлом ДЗ **переименуйте** плейбуки `reddit_app.yml` и `reddit_db.yml` в `packer_reddit_app.yml` и `packer_reddit_db.yml` соответственно и поправьте нужным образом шаблоны пакера.

Также добавьте в файл **.gitignore** следующую строку:

**.retry*

Принципы работы Ansible

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

Для управления хостами при помощи Ansible на них также должен быть установлен Python 2.X

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**.

Работа с inventory

Inventory file

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

```
appserver ansible_ssh_host=35.195.186.154 ansible_ssh_user=appuser  
ansible_ssh_private_key_file=~/.ssh/appuser
```

Где **appserver** - краткое имя, которое идентифицирует данный хост. Обратите внимание, что это должна быть одна строка в файле `ansible/hosts`, 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 ./hosts -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/KLzBULgMeoLZ08awefRbMNUVQ.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

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

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

```
$ ansible appserver -i ./hosts -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/KLzBULgMeoLZ08awefRbMNUVQ.
Are you sure you want to continue connecting (yes/no)? yes
```

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

Имя хоста, которое
указали в инвентори,
откуда Ansible узнает,
как подключаться к
хосту

Путь до инвентори

Вызываемый модуль

Повторим для инстанса БД

Повторите такую же процедуру для инстанса БД. В инвентори файле `ansible/hosts` укажите название хоста `dbserver`.

Напомним, что можно определить выходную переменную для внешнего адреса инстанса БД в тераформе, чтобы облегчить себе поиск нужной информации, или использовать `terraform show`.

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

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

ansible.cfg

Для того чтобы управлять инстансами нам придется вписывать много данных в наш инвентори файл. К тому же, чтобы использовать данный инвентори, нам придется каждый раз указывать его явно, как опцию команды ansible. Многие 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 `infra/ansible`.

ansible.cfg

Укажем значения по умолчанию для работы Ansible.
Скопировать можно из данного [gist](#).

ansible/ansible.cfg

```
[defaults]  
hostfile = hosts  
remote_user = appuser  
private_key_file = ~/.ssh/appuser  
host_key_checking = False
```



Изменим инвентори

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

ansible/hosts

```
appserver ansible_ssh_host=35.195.74.54  
dbserver ansible_ssh_host=35.195.162.174
```

Проверим работу

Используем модуль `command`, который позволяет запускать команды на удаленном хосте. Выполним команду `uptime` для проверки времени работы инстанса. Команду передадим как аргумент для данного модуля, используя 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

Управление 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 несколькими хостами.

Определим группы хостов в инвентори файле.

Изменим инвентори файл следующим образом (не забывайте использовать свои IP):

ansible/hosts

Название группы

[app]

appserver ansible_ssh_host=35.195.74.54

Список хостов в
данной группе

[db]

dbserver ansible_ssh_host=35.195.162.174

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.

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"  
}
```

Имя сервера в
этой группе

Название группы

Группа всех хостов

Ansible также автоматически определяет специальную группу **all**, в которую входят все хосты из инвентори файла:

```
$ ansible all -m ping

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

Работа с playbooks

Плейбуки

Запуск отдельных команд из терминала на удаленном хосте при помощи Ansible не раскрывает преимуществ этого инструмента. Основная ценность Ansible заключается в том, что данный инструмент позволяет нам применять практику IaC, давая возможность декларативно описывать желаемое состояние наших систем в виде кода, который хранится в YAML файлах, называемых **плейбуками (playbooks)** в терминологии Ansible.



Создание плейбука

Создадим плейбук для управления конфигурацией и деплоя нашего приложения. Для этого создайте файл `reddit_app.yml` в директории `ansible`.



Сценарий плейбука

Плейбук может состоять из одного или нескольких **сценариев (plays)**. Сценарий позволяет группировать набор заданий (tasks), который Ansible должен выполнить на конкретном хосте (или группе). В нашем плейбуке мы будем использовать один сценарий для управления конфигурацией обоих хостов (приложения и бд).

ansible/reddit_app.yml

Словесное
описание
сценария

Для каких хостов
будут выполняться
описанные ниже
задачи

```
- name: Configure hosts & deploy application  
  hosts: all  
  tasks:
```

Блок задач
(заданий),
которые будут
выполняться для
данных хостов

Управление MongoDB

По умолчанию MongoDB слушает на локалхосте (127.0.0.1) и наше тестовое приложение работало без дополнительных настроек БД, когда приложение и БД находились на одном инстансе. Т.к. мы вынесли MongoDB на отдельный инстанс, то нам потребуется изменить конфигурацию MongoDB, указав ей слушать на имеющемся сетевом интерфейсе доступном для инстанса приложения, в противном случае наше приложение не сможет подключиться к БД.

Используем модуль **template**, чтобы скопировать параметризированный локальный конфиг файл монги на удаленный хост по указанному пути. Добавим task:

ansible/reddit_app.yml

Выполнить
задание от root

```
---  
- name: Configure hosts & deploy application  
  hosts: all  
  tasks:
```

Путь до
локального
файла

```
  - name: Change mongo config file  
    become: true  
    template:  
      src: templates/mongod.conf.j2  
      dest: /etc/mongod.conf  
      mode: 0644
```

Путь на
удаленном
хосте

права на файл,
которые нужно
установить

ansible/reddit_app.yml

```
---  
- name: Configure hosts & deploy application  
  hosts: all  
  tasks:  
    - name: Change mongo config file  
      become: true  
      template:  
        src: templates/mongod.conf.j2  
        dest: /etc/mongod.conf  
        mode: 0644  
      tags: db-tag
```

Для каждого из наших задач будем определять **тег**, чтобы иметь возможность запускать отдельные задачи, имеющие определенный тег, а не запускать задачи все сразу.

Шаблон конфига MongoDB

Создадим директорию `templates` внутри директории `ansible`. В директории `ansible/templates` создадим файл `mongod.conf.j2` (расширение `j2` будет напоминать нам, что данный файл является шаблоном).

Вставим в данный шаблон параметризованный конфиг для MongoDB (см.след слайд). Т.к. в нашем случае нас интересует возможность управления адресом и портом, на котором слушает БД, то мы параметризуем именно эти параметры конфигурации.

templates/mongod.conf.j2.

[Содержимое можно скопировать из данного [gist](#).]

```
# Where and how to store data.
```

```
storage:
```

```
  dbPath: /var/lib/mongodb
```

```
  journal:
```

```
    enabled: true
```

```
# where to write logging data.
```

```
systemLog:
```

```
  destination: file
```

```
  logAppend: true
```

```
  path: /var/log/mongodb/mongod.log
```

```
# network interfaces
```

```
net:
```

```
  port: {{ mongo_port | default('27017') }}
```

```
  bindIp: {{ mongo_bind_ip }}
```

default - один из фильтров Jinja2, который позволяет задавать значение по умолчанию, если переменная неопределенна

Получение
значения
переменной

Пробный прогон

Применение описания плейбука к хостам осуществляется при помощи команды `ansible-playbook`.

У этой команды есть опция (`--check`) которая позволяет произвести "пробный прогон" плейбука. Пробный прогон позволяет посмотреть, какие изменения произойдут на хосте(ах) в случае применения плейбука (похоже на `terraform plan`), а также указывает на ошибки синтаксиса, если они есть.



```
$ ansible-playbook reddit_app.yml --check --limit db
```

```
...
```

Ограничиваем, группу хостов для которых применить плейбук

```
TASK [Change mongo config file]
```

```
*****
```

```
fatal: [dbserver]: FAILED! => {"changed": false, "failed": true, "msg":  
"AnsibleUndefinedVariable: 'mongo_bind_ip' is undefined"}  
to retry, use: --limit @/Users/alexemkin/hw11/ansible/reddit_app.retry
```

```
PLAY RECAP
```

```
*****
```

```
dbserver : ok=1 changed=0 unreachable=0  
failed=1
```

Видим ошибку: переменная,
которая используется в шаблоне
не определена

Определим значения переменных в нашем плейбуке. Определять переменную для порта не будем, т.к. нас устраивает значение по умолчанию, которое мы задали в шаблоне.

```
---
- name: Configure hosts & deploy application
  hosts: all
  vars:
    mongo_bind_ip: 0.0.0.0
  tasks:
    - name: Change mongo config file
      become: true
      template:
        src: templates/mongod.conf.j2
        dest: /etc/mongod.conf
        mode: 0644
      tags: db-tag
```



Повторим проверку плейбука

```
$ ansible-playbook reddit_app.yml --check --limit db
```

```
PLAY [Configure hosts & deploy application] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [dbserver]
```

```
TASK [Change mongo config file] *****
```

```
changed: [dbserver]
```

```
PLAY RECAP *****
```

```
dbserver : ok=2 changed=1 unreachable=0 failed=0
```

Теперь проверка должна пройти успешно. Пробный прогон показывает нам, что task с описанием "Change mongo config file" изменит свое состояние для хоста dbserver, что означает, что на этом хосте произойдут изменения относительно его текущего состояния.



Handlers

Handlers похожи на задачи, однако запускаются только по оповещению других задач. Task шлет оповещение handler-у в случае, когда он меняет свое состояние. По этой причине handlers удобно использовать для перезапуска сервисов. Это, например, позволяет перезапускать сервис, только в случае если поменялся его конфиг файл.



Handler

Изменение конфигурационного файла MongoDB требует от нас перезапуска БД для применения конфигурации. Используем для этой задачи handler. Определим handler для рестарта БД и добавим оповещение handler-а в созданный нами task.

ansible/reddit_app.yml

- name: Configure hosts & deploy application
hosts: all
vars:
 mongo_bind_ip: 0.0.0.0
tasks:
 - name: Change mongo config file
become: true
template:
 src: templates/mongod.conf.j2
 dest: /etc/mongod.conf
 mode: 0644
tags: db-tag
notify: restart mongod

handlers:

- name: restart mongod
become: true
service: name=mongod state=restarted



Применим плейбук

Для начала используем пробный прогон, и убедимся, что нет ошибок:

```
$ ansible-playbook reddit_app.yml --check --limit db
```

Применим наш плейбук:

```
$ ansible-playbook reddit_app.yml --limit db
```

```
PLAY [Configure hosts & deploy application] *****
```

```
TASK [Gathering Facts] *****  
ok: [dbserver]
```

```
TASK [Change mongo config file] *****  
changed: [dbserver]
```

```
RUNNING HANDLER [restart mongod] *****  
changed: [dbserver]
```

```
PLAY RECAP *****  
dbserver          : ok=3    changed=2    unreachable=0    failed=0
```


Настройка инстанса приложения

Unit для вебсервера

Помним, как на предыдущих занятиях мы уже копировали unit файл для сервера Puma, чтобы иметь возможность управления сервером и добавлением его в автозапуск. Скопируем unit файл на инстанс приложения, используя Ansible.

Создайте директорию `files` внутри директории `ansible` и добавьте туда файл `puma.service` файл. Обратите внимание, что unit файл изменился.

Добавим в наш сценарий task для копирования unit файла на хост приложения. Для копирования простого файла на удаленный хост, используем модуль **copy**, а для настройка автостарта Puma сервера используем модуль **systemd**:

```
tasks:
  - name: Change mongo config file
    ...

  - name: Add unit file for Puma
    become: true
    copy:
      src: files/puma.service
      dest: /etc/systemd/system/puma.service
    tags: app-tag
    notify: reload puma

  - name: enable puma
    become: true
    systemd: name=puma enabled=yes
    tags: app-tag
```



Не забудем добавить новый handler, который указывает systemd, что unit для сервиса изменился и его следует перечитать.

handlers:

- name: restart mongod
become: true
service: name=mongod state=restarted
- name: reload puma
become: true
systemd: name=puma state=reloaded

Unit для вебсервера

Как уже упоминалось, unit файл для вебсервера изменился. В него добавилась строка чтения переменных окружения из файла:

```
EnvironmentFile=/home/appuser/db_config
```

Через переменную окружения мы будем передавать адрес инстанса БД, чтобы приложение знало, куда ему обращаться для хранения данных.



Создадим шаблон в директории `templates/db_config.j2` куда добавим следующую строку:

```
DATABASE_URL={{ db_host }}
```

Как видим, данный шаблон содержит присвоение переменной `DATABASE_URL` значения, которое мы передаем через Ansible переменную `db_host`.

Добавим таск для копирования созданного шаблона:

- name: Add unit file for Puma
...
- name: Add config for DB connection
template:
 src: templates/db_config.j2
 dest: /home/appuser/db_config
tags: app-tag
- name: enable puma
become: true
systemd: name=puma enabled=yes
tags: app-tag

И не забудем определить переменную:

```
---  
- name: Configure hosts & deploy application  
  hosts: all  
  vars:  
    mongo_bind_ip: 0.0.0.0  
    db_host: 10.132.0.2  
  tasks:  
    ...
```

Переменной `db_host` присваиваем значения внутреннего IP адреса инстанса базы данных. Этот адрес можно посмотреть в консоли GCP, используя `terraform show` или `gcloud` команду. Неплохо также было бы вынести эту информацию в `output` переменную в `terraform-e`



Настройка инстанса приложения

Пробный прогон:

```
$ ansible-playbook reddit_app.yml --check --limit app --tags app-tag
```

Применим наши задачи плейбука с тегом app-tag для группы хостов app:

```
$ ansible-playbook reddit_app.yml --limit app --tags app-tag
```

```
$ ansible-playbook reddit_app.yml --limit app --tags app-tag
```

```
...
```

```
TASK [Add unit file for Puma]
```

```
*****
```

```
changed: [appserver]
```

```
TASK [Add config for DB connection]
```

```
*****
```

```
changed: [appserver]
```

```
TASK [enable puma]
```

```
*****
```

```
changed: [appserver]
```

```
RUNNING HANDLER [reload puma]
```

```
*****
```

```
changed: [appserver]
```

```
PLAY RECAP
```

```
*****
```

```
****
```

```
appserver : ok=5 changed=4 unreachable=0 failed=0
```

Деплой

Деплой

Добавим еще несколько задач в сценарий нашего плейбука. Используем модули **git** и **bundle** для клонирования последней версии кода нашего приложения и установки зависимых гемов через bundle.

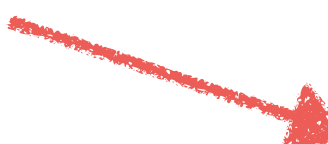
ansible/reddit_app.yml

tasks:

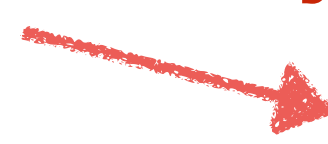
...

- **name:** Fetch the latest version of application code
git:
 - repo:** 'https://github.com/Artemmkin/reddit.git'
 - dest:** /home/appuser/reddit
 - version:** monolith
 - tags:** deploy-tag
 - notify:** restart puma
- **name:** Bundle install
bundler:
 - state:** present
 - chdir:** /home/appuser/reddit
 - tags:** deploy-tag

Указываем нужную ветку



В какой директории
выполнить команду
bundle



Не забудем добавить указанный в одном из тасков handler для перезагрузки сервера приложения при обновлении кода:

```
handlers:
```

- name: restart mongod
become: true
service: name=mongod state=restarted
- name: reload puma
become: true
systemd: name=puma state=reloaded
- name: restart puma
become: true
systemd: name=puma state=restarted

Выполняем деплой

```
$ ansible-playbook reddit_app.yml --check --limit app --tags deploy-tag
$ ansible-playbook reddit_app.yml --limit app --tags deploy-tag
```

...

```
TASK [Fetch the latest version of application code]
```

```
*****
```

```
changed: [appserver]
```

```
TASK [bundle install]
```

```
*****
```

```
changed: [appserver]
```

```
RUNNING HANDLER [restart puma]
```

```
*****
```

```
changed: [appserver]
```

```
PLAY RECAP
```

```
*****
```

```
appserver                : ok=4    changed=3    unreachable=0    failed=0
```

Проверяем работу приложения

Добавьте один пост, чтобы убедиться в отсутствии проблем с БД и работоспособности приложения

35.189.243.19:9292

Monolith Reddit

Post successfully published

^

0

v

We just deployed our application! Congrats!

[Go to the link](#)

Задание со звездочкой

Исследовать возможности использования dynamic inventory для GCE. Выбрать, на ваш взгляд оптимальное решение. Решение добавить в пул реквест к основному заданию.

