Деплой и управление конфигурацией с Ansible

Проект infra и проверка ДЗ

- Создайте новую ветку в вашем инфраструктурном репозитории для выполнения данного ДЗ. Т.к. это второе задание, посвященое работе с Ansible, то ветку назовите `ansible-2`.
- 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 можно смерджить.

План

- Используем плейбуки, хендлеры и шаблоны для конфигурации окружения и деплоя тестового приложения. Подход один плейбук, один сценарий (play)
- Аналогично один плейбук, но много сценариев
- И много плейбуков.
- Изменим провижн образов Packer на Ansible-плейбуки

Но сначала ...

Если вы настраивали в ДЗ по Terraform провижининг приложения (второе задание со ⭐), то сейчас необходимо:

- Либо выключить его с помощью переменной, если это было предусмотрено
- Либо закомментировать или удалить код провижининга для `app` и `db` модулей

Один playbook, один сценарий

Плейбуки

Запуск отдельных команд из терминала на удаленном хосте при помощи Ansible не раскрывает преимуществ этого инструмента.

Основное преимущество Ansible заключается в том, что данный инструмент позволяет нам применять практику IaC, давая возможность декларативно описывать желаемое состояние наших систем в виде кода.

Код Ansible 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`.

Чтобы не запушить в репу временные файлы Ansible, добавим в файл `.gitignore` следующую строку:

```
*.retry
```

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).

Сценарий для MongoDB

Файл `ansible/reddit_app.yml`:

```
---
- name: Configure hosts & deploy application  # <-- Словесное описание сценария (name)
  hosts: all  # <-- Для каких хостов будут выполняться описанные ниже таски (hosts)
  tasks:  # <-- Блок тасков (заданий), которые будут выполняться для данных хостов
```

Управление MongoDB

По умолчанию MongoDB слушает на *localhost* (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.

Сценарий для MongoDB

Используем модуль `template`, чтобы скопировать параметризированный локальный конфиг файл MongoDB на удаленный хост по указанному пути. Добавим *task* в файл `ansible/reddit_app.yml`:

```
---
- name: Configure hosts & deploy application
  hosts: all
  tasks:
    - name: Change mongo config file
      become: true  # <-- Выполнить задание от root
      template:
        src: templates/mongod.conf.j2  # <-- Путь до локального файла-шаблона
        dest: /etc/mongod.conf  # <-- Путь на удаленном хосте
        mode: 0644  # <-- Права на файл, которые нужно установить
```

Сценарий для MongoDB

Для каждого из наших тасков сразу будем определять тег, чтобы иметь возможность запускать отдельные таски, имеющие определенный тег, а не весь сценарий сразу.

Файл `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  # <-- Список тэгов для задачи
```

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. следующий слайд).

Т.к. в нашем случае нас интересует возможность управления адресом и портом, на котором слушает БД, то мы параметризуем именно эти параметры конфигурации.

Шаблон конфига MongoDB

Файл `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:
  # default - один из фильтров Jinja2, он задает значение по умолчанию,
  # если переменная слева не определена
  port: {{ mongo_port | default('27017') }}
  bindIp: {{ mongo_bind_ip }}  # <-- Подстановка значения переменной
```

Пробный прогон

Применение 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.

Пробный прогон

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

...

TASK [Change mongo config file] ***********************************************
fatal: [dbserver]: FAILED! => {"changed": false, "failed": true, "msg":
"AnsibleUndefinedVariable: 'mongo_bind_ip' is undefined"}
    to retry, use: --limit @/Users/user/hw11/ansible/reddit_app.retry

PLAY RECAP ********************************************************************
dbserver                   : ok=1    changed=0    unreachable=0    failed=1
```

`--limit` - ограничиваем группу хостов, для которых применить плейбук

`AnsibleUndefinedVariable` - Ошибка ! Переменная, которая используется в шаблоне не определена

Определение переменных

Определим значения переменных в нашем плейбуке. Задавать 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  # <-- Переменная задается в блоке vars
  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 таск 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 таски, однако запускаются только по оповещению от других задач.

Таск 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.

Handlers

Изменение конфигурационного файла MongoDB требует от нас перезапуска БД для применения конфигурации. Используем для этой задачи *handler*.

Определим handler для рестарта БД и добавим вызов handler-а в созданный нами таск.

Добавим handlers

Файл `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:  # <-- Добавим блок handlers и задачу
  - name: restart mongod
    become: true
    service: name=mongod state=restarted
```

Применим плейбук

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

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

Применим плейбук

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 приложения

Вспомним, как на предыдущих занятиях мы уже копировали unit-файл для сервера Puma, чтобы управлять сервисом и добавить его в автостарт. Теперь скопируем unit-файл на инстанс приложения, используя Ansible.

Создайте директорию `files` внутри директории `ansible` и добавьте туда файл `puma.service`. Файл доступен по этой ссылке

> Обратите внимание, что unit-файл изменился

Unit для приложения

Добавим в наш сценарий таск для копирования unit-файла на хост приложения. Для копирования простого файла на удаленный хост, используем модуль `copy`, а для настройки 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
```

Unit для приложения

Не забудем добавить новый *handler*, который укажет 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=restarted
```

Unit для приложения

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.

Добавим шаблон для приложения

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`.

Шаблон приложения

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
```

Зададим переменную

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  # <-- подставьте сюда ваш IP
  tasks:
...
```

Переменной `db_host` присваиваем значение внутреннего IP-адреса инстанса базы данных. Этот адрес можно посмотреть в консоли GCP, используя `terraform show` или команду `gcloud`.

> Неплохо бы вынести эту информацию в *output-переменную* в Terraform

Настройка инстанса приложения

Пробный прогон:

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

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

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

Настройка инстанса приложения

```
$ 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 тасков в сценарий нашего плейбука.

Используем модули `git` и `bundle` для клонирования последней версии кода нашего приложения и установки зависимых Ruby Gems через `bundle`.

Деплой кода и установка зависимостей

Файл `ansible/reddit_app.yml`:

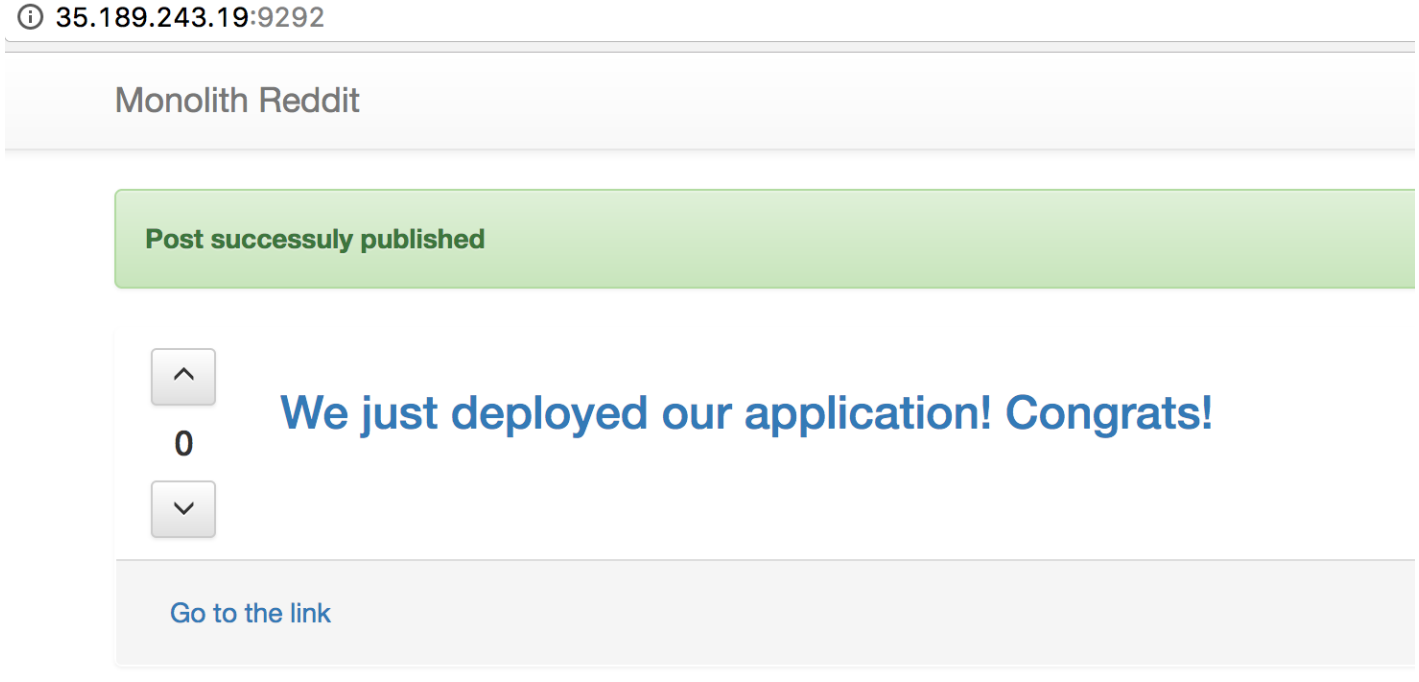
```
  tasks:
...
    - name: Fetch the latest version of application code
      git:
        repo: 'https://github.com/express42/reddit.git'
        dest: /home/appuser/reddit
        version: monolith  # <-- Указываем нужную ветку
      tags: deploy-tag
      notify: reload puma

    - name: Bundle install
      bundler:
        state: present
        chdir: /home/appuser/reddit  # <-- В какой директории выполнить команду
bundle
      tags: deploy-tag
```

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 [reload 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:



Один плейбук, несколько сценариев

Плейбуки

В предыдущей части мы создали один плейбук, в котором определили один сценарий (play) и, как помним, для запуска нужных тасков на заданной группе хостов мы использовали опцию `--limit` для указания группы хостов и `--tags` для указания нужных тасков.

Очевидна проблема такого подхода, которая состоит в том, что мы должны помнить при каждом запуске плейбука, на каком хосте какие таски мы хотим применить, и передавать это в опциях командной строки.

Давайте попробуем разбить наш сценарий на несколько и посмотрим, как это изменит ситуацию.

Сценарий для MongoDB

Создадим новый файл `reddit_app2.yml` в директории `ansible`. Определим в нем несколько сценариев (plays), в которые объединим задачи, относящиеся к используемым в плейбуке тегам.

Определим отдельный сценарий для управления конфигурацией MongoDB. Будем при этом использовать уже имеющиеся наработки из `reddit_app.yml` плейбука.

Сценарий для MongoDB

Скопируем определение сценария из `reddit_app.yml` и всю информацию, относящуюся к настройке MongoDB, которая будет включать в себя таски, хендлеры и переменные.

Помним, что таски для настройки MongoDB приложения мы помечали тегом `db-tag`.

Сценарий для MongoDB

Файл `ansible/reddit_app2.yml` (ссылка на gist):

```
# Данный сценарий мы составляем только для MongoDB, может стоит поменять
описание?
- name: Configure hosts & deploy application
  # Применять сценарий мы хотим только к серверам группы db или ко всем?
  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  # <-- нужен ли нам здесь тег?
      notify: restart mongod

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

Сценарий для MongoDB

Итак:

- Изменим словесное описание
- Укажем нужную группу хостов
- Уберем теги из тасков и определим тег на уровне сценария, чтобы мы могли запускать сценарий, используя тег.

Также заметим, что **все** наши таски требуют выполнения из-под пользователя `root`, поэтому нет смысла их указывать для каждого task.

- Вынесем `become: true` на уровень сценария.

Результат

Файл `ansible/reddit_app2.yml` (ссылка на gist)

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

  handlers:
  - name: restart mongod
    service: name=mongod state=restarted
```

Сценарий для App

Аналогичным образом определим еще один сценарий для настройки инстанса приложения.

Скопируем еще раз определение сценария из `reddit_app.yml` и всю информацию относящуюся к настройке инстанса приложения, которая будет включать в себя таски, хендлеры и переменные.

Помним, что таски для настройки инстанса приложения мы помечали тегом `app-tag`. Вставим скопированную информацию в `reddit_app2.yml` следом за сценарием для MongoDB.

Сценарий для App

Файл `ansible/reddit_app2.yml` (ссылка на gist):

```
...
- name: Configure hosts & deploy application  # <-- Изменить плейбук надо здесь
  hosts: all  # <-- Изменить плейбук надо здесь
  vars:
   db_host: 10.132.0.2
  tasks:
    - name: Add unit file for Puma
      become: true  # <-- Изменить плейбук надо здесь
      copy:
        src: files/puma.service
        dest: /etc/systemd/system/puma.service
      tags: app-tag  # <-- Изменить плейбук надо здесь
      notify: reload puma
...
# Продолжение на следующем слайде
```

Сценарий для App

```
# Начало на предыдущем слайде
...
    - name: Add config for DB connection
      template:
        src: templates/db_config.j2
        dest: /home/appuser/db_config
      tags: app-tag    # <-- Изменить плейбук надо здесь

    - name: enable puma
      become: true  # <-- Изменить плейбук надо здесь
      systemd: name=puma enabled=yes
      tags: app-tag  # <-- Изменить плейбук надо здесь

  handlers:
  - name: reload puma
    become: true  # <-- Изменить плейбук надо здесь
    systemd: name=puma state=restarted
```

Сценарий для App

- Изменим словесное описание
- Укажем нужную группу хостов
- Уберем теги из тасков и определим тег на уровне сценария, чтобы мы запускать сценарий, используя тег.
- Также заметим, что большинство из наших тасков требуют выполнения из-под пользователя `root`, поэтому вынесем `become: true` на уровень сценария.
- В таске, который копирует конфиг-файл в домашнюю директорию пользователя `appuser`, явно укажем пользователя и владельца файла.

Результат

Файл `ansible/reddit_app2.yml` (ссылка на gist):

```
- name: Configure App
  hosts: app
  tags: app-tag
  become: true
  vars:
   db_host: 10.132.0.2
  tasks:
    - name: Add unit file for Puma
      copy:
        src: files/puma.service
        dest: /etc/systemd/system/puma.service
      notify: reload puma
...
# Продолжение на следующем слайде
```

Результат

```
# Начало на предыдущем слайде
...
    - name: Add config for DB connection
      template:
        src: templates/db_config.j2
        dest: /home/appuser/db_config
        owner: appuser
        group: appuser
    - name: enable puma
      systemd: name=puma enabled=yes

  handlers:
  - name: reload puma
    systemd: name=puma state=restarted
```

Пересоздадим инфраструктуру

Для чистоты проверки наших плейбуков пересоздадим инфраструктуру окружения **stage**, используя команды:

```
$ terraform destroy
$ terraform apply -auto-approve=false
```

Проверим работу сценариев

Перед проверкой не забудьте изменить внешние IP-адреса инстансов в инвентори файле `ansible/inventory` и переменную `db_host` в сценарии приложения.

```
$ ansible-playbook reddit_app2.yml --tags db-tag --check
$ ansible-playbook reddit_app2.yml --tags db-tag

PLAY [Configure MongoDB] ********************************************************

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

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

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

Проверим работу сценариев

Обратите внимание, что теперь при вызове команд нам *не нужно указывать* явно, *на каких хостах* запускать плейбук.

При запуске команды мы *указываем тег*, который ссылается на конкретный сценарий.

```
$ ansible-playbook reddit_app2.yml --tags app-tag --check
$ ansible-playbook reddit_app2.yml --tags app-tag
```

Проверим работу сценариев

```
PLAY [Configure MongoDB] ***************************************************

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

PLAY [Configure App] *******************************************************

TASK [Gathering Facts] *****************************************************
ok: [appserver]

TASK [Add unit file for Puma] **********************************************
changed: [appserver]

TASK [Add config for DB connection] ****************************************
changed: [appserver]

TASK [enable puma] *********************************************************
changed: [appserver]
...
```

Сценарий для деплоя

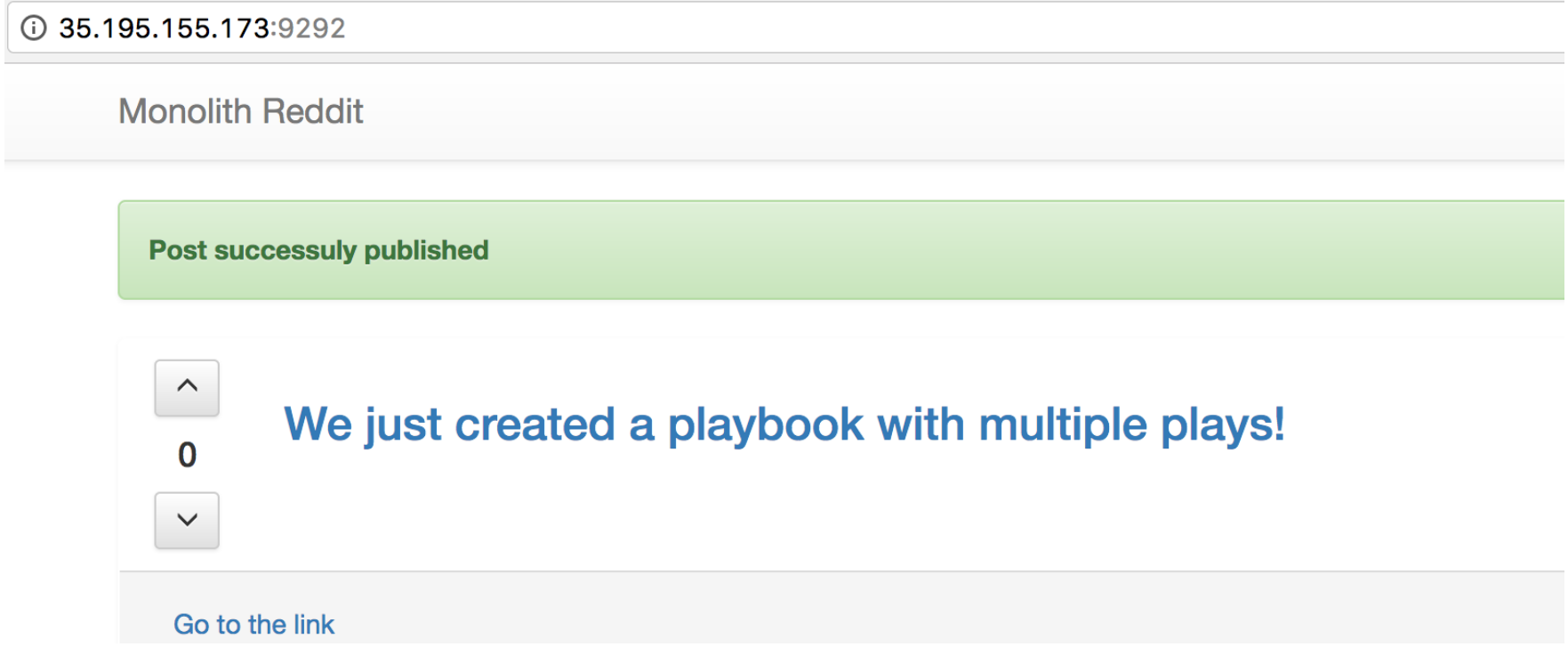
Самостоятельно, по аналогии с предыдущими заданиями:

- Добавьте сценарий для деплоя приложения в плейбук `reddit_app2.yml`
- Проверьте, что при его выполнении происходит деплой приложения
- Убедитесь, что приложение доступно по внешнему IP инстанса приложения.

Если возникнут трудности, то подсмотреть, как должен выглядеть конечный плейбук можно в данном gist

Проверка сценария

```
$ ansible-playbook reddit_app2.yml --tags deploy-tag --check
$ ansible-playbook reddit_app2.yml --tags deploy-tag
```



Несколько плейбуков

Несколько плейбуков

Описав несколько сценариев для управления конфигурацией инстансов и деплоя приложения, управлять хостами стало немного легче.

Теперь, для того чтобы *применить нужную часть* конфигурационного кода (сценарий) к нужной группе хостов достаточно лишь указать ссылку на эту часть кода, *используя тег*.

Однако, **есть проблема**:

с ростом числа управляемых сервисов, *будет расти количество* различных *сценариев* и, как результат, *увеличится объем плейбука*.

Это приведет к тому, что в плейбуке, будет сложно разобраться. Поэтому, следующим шагом попытаемся *разделить наш плейбук* на несколько.

Несколько плейбуков

В директории `ansible` создадим три новых файла:

- `app.yml`
- `db.yml`
- `deploy.yml`

Заодно переименуем наши предыдущие плейбуки:

- `reddit_app.yml` ➡ `reddit_app_one_play.yml`
- `reddit_app2.yml` ➡ `reddit_app_multiple_plays.yml`

db.yml

- Из файла r`eddit_app_multiple_plays.yml` скопируем сценарий, относящийся к настройке БД, в файл `db.yml`
- При этом, удалим тег определенный в сценарии.

Поскольку мы выносим наши сценарии в отдельные плейбуки, то для запуска нужного нам сценария *достаточно будет указать имя плейбука*, который его содержит. Значит, *тег* нам больше *не понадобится*.

db.yml

Файл `ansible/db.yml` (ссылка на gist):

```
- name: Configure MongoDB
  hosts: db
  # tags: db-tag    <-- Тэг нам больше не нужен
  become: true
  vars:
    mongo_bind_ip: 0.0.0.0
  tasks:
    - name: Change mongo config file
      template:
        src: templates/mongod.conf.j2
        dest: /etc/mongod.conf
        mode: 0644
      notify: restart mongod

  handlers:
  - name: restart mongod
    service: name=mongod state=restarted
```

app.yml

- Аналогично вынесем настройку хоста приложения из `reddit_app_multiple_plays.yml` в отдельный плейбук `app.yml`.
- Не забудем удалить тег, т.к. в нем теперь у нас нет необходимости.

app.yml

Файл `ansible/app.yml` (ссылка на gist):

```
- name: Configure App
  hosts: app
  # tags: app-tag  <-- Тэг нам больше не нужен
  become: true
  vars:
   db_host: 10.132.0.2
  tasks:
    - copy: src=files/puma.service dest=/etc/systemd/system/puma.service
      notify: reload puma
    - name: Add config for DB connection
      template:
        src: templates/db_config.j2
        dest: /home/appuser/db_config
        owner: appuser
        group: appuser
    - systemd: name=puma enabled=yes
  handlers:
    - systemd: name=puma state=restarted
```

deploy.yml

- Создайте по аналогии плейбук для деплоя приложения
- Очевидно, файл будет называться `deploy.yml`

> Если возникнут трудности, то плейбук можно посмотреть в данном gist

site.yml

Создадим файл `site.yml` в директории `ansible`, в котором опишем управление конфигурацией всей нашей инфраструктуры. Это будет нашим главным плейбуком, который будет включать в себя все остальные:

Файл `ansible/site.yml`:

```
---
- import_playbook: db.yml
- import_playbook: app.yml
- import_playbook: deploy.yml
```

Проверка результата

Для чистоты проверки наших плейбуков пересоздадим инфраструктуру окружения **stage**, используя команды:

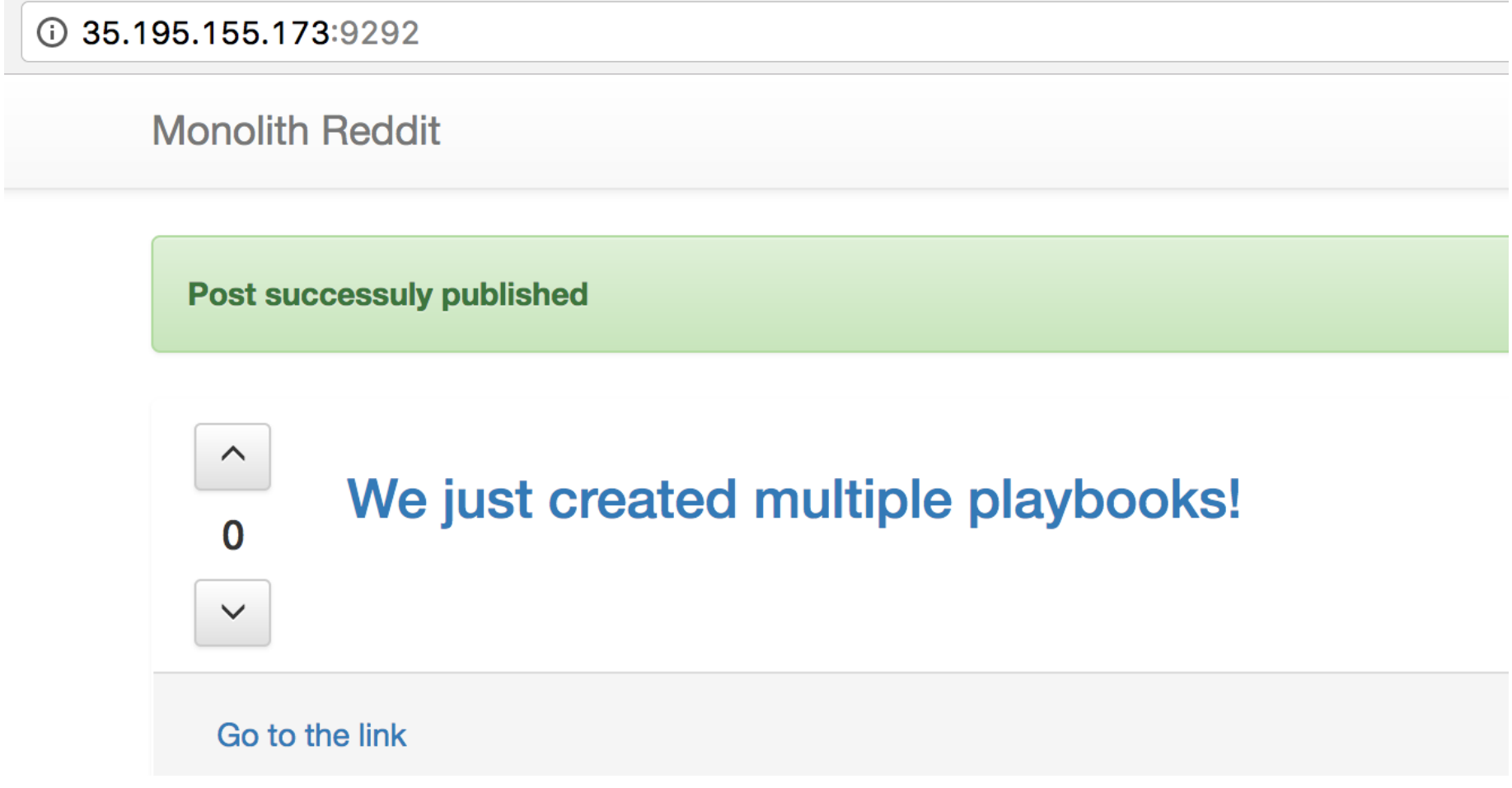
```
$ terraform destroy
$ terraform apply -auto-approve=false
```

и проверим работу плейбуков:

Перед проверкой не забудьте изменить внешние IP-адреса инстансов в инвентори файле `ansible/inventory` и переменную `db_host` в плейбуке `app.yml`:

```
$ ansible-playbook site.yml --check
$ ansible-playbook site.yml
```

Проверка результата



Задание со ⭐

- Исследуйте возможности использования dynamic inventory для GCP (для этого есть не только `gce.py` 😄).
- Нужно выбрать оптимальное, на ваш взгляд, решение. Решение добавить в PR к основному заданию.
- Использование динамического инвентори означает, что это **должно быть отражено в `ansible.cfg` и плейбуках** (т.е. они должны использовать выбранное решение)

> **!** Если вы решили использовать `gce.py`, убедитесь в том, что вы **НЕ закоммитите ключи** сервисного пользователя (файл `*.json`)

Провижининг в Packer

Изменим провижининг в Packer

В данной части мы изменим *provision* в Packer и заменим bash-скрипты на Ansible-плейбуки.

Мы уже создали плейбуки для конфигурации и деплоя приложения. Создадим теперь на их основе плейбуки `ansible/packer_app.yml` и `ansible/packer_db.yml`.

Каждый из них должен реализовывать функционал bash-скриптов, которые использовались в Packer ранее.

- `packer_app.yml` - устанавливает Ruby и Bundler
- `packer_db.yml` - добавляет репозиторий MongoDB, устанавливает ее и включает сервис.

Самостоятельное задание

Опишите с помощью модулей Ansible в плейбуках `ansible/packer_app.yml` и `ansible/packer_db.yml` действия, аналогичные bash-скриптам, которые сейчас используются в нашей конфигурации Packer.

Использовать модули `command` и `shell` нежелательно!

- Документация по модулям
- Документация по циклам в Ansible

> При возникновении трудностей, готовые плейбуки можно подсмотреть в этом gist

Интегрируем Ansible в Packer

Заменим секцию Provision в образе `packer/app.json` на Ansible:

```
"provisioners": [
    {
        "type": "ansible",
        "playbook_file": "ansible/packer_app.yml"
    }
]
```

Такие же изменения выполним и для `packer/db.json`:

```
"provisioners": [
    {
        "type": "ansible",
        "playbook_file": "ansible/packer_db.yml"
    }
]
```

Проверяем образы

- Выполните билд образов с использованием нового провижинера.
- На основе созданных `app` и `db` образов запустите **stage** окружение.
- Проверьте, что с помощью плейбука `site.yml` из предыдущего раздела окружение конфигурируется, а приложение деплоится и работает.

P.S. Пути в JSON-файлах **корректны**, билд образов **нужно** производить **из корня репозитория**

P.P.S. Для WSL может понадобиться задать еще пользователя `"user": "appuser"`

Проверка ДЗ

- Результаты вашей работы находятся в ветке `ansible-2` 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-2** 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, наличие пустой строки в конце файлов, а также валидирует шаблоны пакера.