Ansible: работа с ролями и окружениями

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 третье задание, посвященое работе с **Ansible**, то ветку можно назвать **ansible-3.**

Проверка данного ДЗ 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.

Множественные сценарии

На прошлом занятии мы с вами познакомились с плейбуками. **Плейбуки** позволяют нам описывать различные **сценарии** (plays), которые в свою очередь представляют собой наборы **заданий** (tasks), необходимые для выполнения на заданном хосте (или группе хостов). В предыдущем ДЗ мы создали один плейбук, в котором определили один сценарий 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.

Несколько сценариев

Очевидна проблема такого подхода, которая видится в том, что мы должны помнить при каждом запуске плейбука, на каком хосте какие таски мы хотим применить, и передавать это в опциях команды. Давайте попробуем разбить наш сценарий на несколько и посмотрим, как это изменит ситуацию.

Создание инфраструктуры

Поднимите инфрастурктуру окружения **stage**:

```
$ terraform apply -auto-approve=false
...
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

app_external_ip = 35.195.155.173
db_external_ip = 35.189.243.19
db_internal_ip = 10.132.0.2
```

Сценарий для MongoDB

Создадим новый файл **reddit_app2.yml** в директории **ansible**. Определим в нем несколько сценариев, в которые объединим задачи, относящиеся к используемым в плейбуке тегам. Определим отдельный сценарий для управления конфигурацией MongoDB. Будем при этом использовать уже имеющиеся наработки в reddit_app.yml плейбуке.

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.

ansible/reddit_app2.yml *ссылка на gist*

```
---
- 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
```

т.к. данный сценарий мы составляем только для MongoDB, то словесное описание мы захотим поменять

Применять данный сценарий мы хотим только к серверам группы db, описанным в инвентори, а не ко всем

нужен ли нам здесь тег?

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 вынесем become: true на уровень сценария.

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
```

Аналогичным образом определим еще один сценарий для настройки инстанса приложения.
Скопируем еще раз определение сценария из reddit_app.yml и всю информацию относящуюся к настройки 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.

ansible/reddit_app2.yml

ссылка на gist

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

  handlers:
  - name: reload puma
    become: true
    systemd: name=puma state=reloaded
```

Обозначим красным цветом части, которые хотим поменять

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 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 и не требует команды sudo, явно укажем пользователя и владельца файла.

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
    - 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=reloaded
```

Проверим работу сценариев

Перед проверкой не забудьте изменить внешние IP адреса инстансов в инвентори файле ansible/hosts 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]
...
```

Обратите внимание, что теперь при вызове команд нам не нужно указывать явно, на каких хостах запускать плейбук. При запуске команды мы укываем тег, который ссылается на конкретный сценарий.

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

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]

...
```

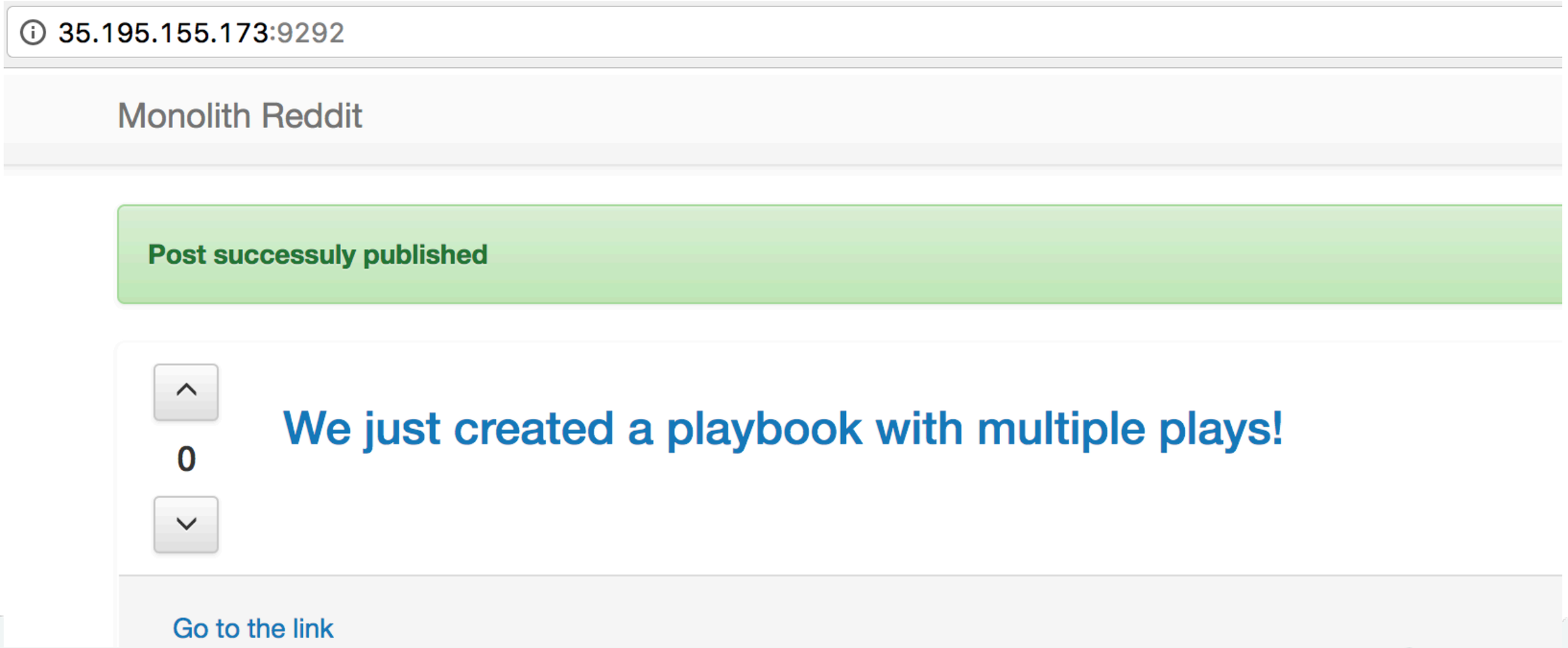
Сценарий для деплоя

Самостоятельно по аналогии с предыдущими заданиями добавьте сценарий для деплоя приложения в плейбук reddit_app2.yml. Проверьте, что при его выполнении происходит деплой приложения и оно вам 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
```



Плейбуки

Описав несколько сценариев для управления конфигурацией инстансов, а также 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 видится проблема: с ростом числа управляемых сервисов, будет возрастать количество различных сценариев и, как результат, увеличится объем плейбука. Это приведет к тому, что в плейбуке, будет сложно разобраться. Поэтому следующим шагом попытаемся разбить наш плейбук на несколько.

Несколько плейбуков

В директории **ansible** создадим три новых файла **app.yml**, **db.yml**, **deploy.yml**.

Заодно переименуем наши предыдущие плейбуки: изменим название файла reddit_app.yml на reddit_app_one_play.yml, а файл reddit_app2.yml на reddit_app_multiple_plays.yml.

db.yml

Из файла reddit_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, и поэтому тег нам больше не понадобится.

ansible/db.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
```

app.yml

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

ansible/app.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
   - 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=reloaded
```

deploy.yml

Создайте по аналогии плейбук для деплоя.

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:

ansible/site.yml

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

Пересоздадим инфраструктуру

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

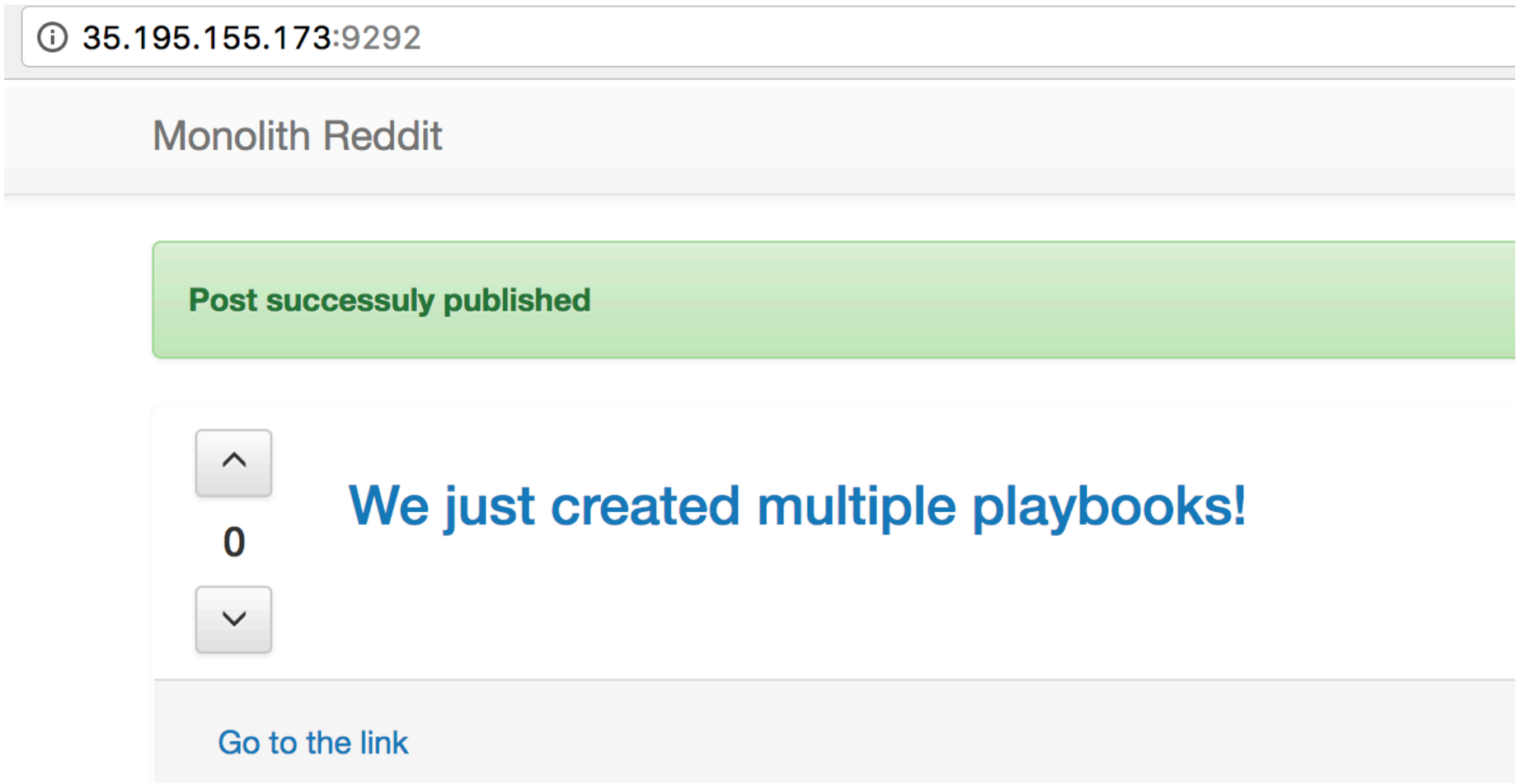
```
$ terraform destroy
$ terraform apply -auto-approve=false
```

Проверим работу плейбуков

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

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

Проверим работу приложения



Роли

Организация нашего конфигурационного кода стала выглядеть лучше, после того как мы ввели несколько плейбуков, но у нас все равно виднеются проблемы. Во-первых, наши шаблоны и файлы хранятся в одних и тех же директориях для всех плейбуков. Как результат, становится сложно понять, что к чему относится, особенно если у нас возрастет количество плейбуков.

Во-вторых, т.к. переменных в плейбуке может быть большое количество их не очень удобно определять в самом плейбуке. Нам определенно хотелось бы вынести их в отдельный файл.

Роли

Роли представляют собой основной механизм группировки и переиспользования конфигурационного кода в Ansible.
Роли позволяют сгруппировать в единое целое описание конфигурацию отдельных сервисов и компонент системы (таски, хендлеры, файлы, шаблоны, переменные). Роли можно затем переиспользовать при настройке окружений, тем самым избежав копирование кода.
Ролями можно также делиться и брать у сообщества (community).

Ansible Galaxy

Ansible Galaxy - это централизованное место, где хранится информация о ролях, созданных сообществом (community roles).

Ansible имеет специальную команду для работы с Galaxy. Получить справку по этой команде можно на сайте Galaxy или использовав команду:

```
$ ansible-galaxy -h
```

ansible-galaxy init

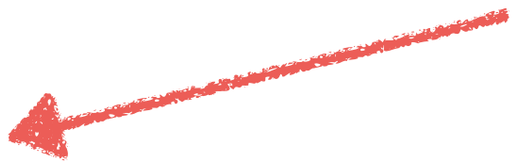
Команда ansible-galaxy init позволяет нам создать структуру роли в соответсвии с принятым на Galaxy форматом.
Мы не будем делиться созданными нами ролями на Galaxy, однако используем эту команду для создания заготовки ролей.
В директории ansible создайте директорию **roles** и выполните следующие команды для создания заготовки ролей для конфигурации нашего приложения и БД:

```
$ ansible-galaxy init app
$ ansible-galaxy init db
```

Посмотрим структуру созданных заготовок и выделим непонятные части.

```
$ ansible-galaxy init db
- db was created successfully

$ tree db
```

```
db
├── README.md
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

6 directories, 8 files
```



Директория для переменных по умолчанию

Информация о роли и его создателе

Директория для тестов

Директория переменных, которые не должны переопределяться

db role

Создадим роль для конфигурации MongoDB.
Скопируем секцию tasks в сценарии плейбука ansible/db.yml и вставим ее в файл для тасков роли db.

ansible/roles/db/tasks/main.yml

```
---
# tasks file for db

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

Создадим директорию для шаблонов templates в директории роли ansble/roles/db и скопируем туда шаблонизированный конфиг для MongoDB из директории ansible/templates.

Особенностью ролей также является, что модули для модулей **template** и **copy**, которые используются в тасках роли, Ansible будет по умолчанию проверять наличие шаблонов и файлов в директориях роли templates и files соответсвенно. Поэтому укажем в таске только имя шаблона в качестве источника.

ansible/roles/db/tasks/main.yml

```
---
# tasks file for db

- name: Change mongo config file
  template:
    src:  mongod.conf.j2
    dest: /etc/mongod.conf
    mode: 0644
  notify: restart mongod
```

Не забудем определить используемый хендлер в отдельной директории роли

ansible/roles/db/handlers/main.yml

```
---
# handlers file for db
- name: restart mongod
  service: name=mongod state=restarted
```

Определим используемые в шаблоне переменные в секции переменных по умолчанию:

ansible/roles/db/defaults/main.yml

```
---
# defaults file for db

mongo_port: 27017
mongo_bind_ip: 127.0.0.1
```

app role

Создадим роль для управления конфигурацией инстанса приложения. Скопируем секцию tasks в сценарии плейбука ansible/app.yml и вставим ее в файл для тасков роли app.

Не забудем при этом заменить src в модулях copy и template для указания только имени файлов.

ansible/roles/app/tasks/main.yml

```
---
# tasks file for app
- name: Add unit file for Puma
  copy:
    src: puma.service
    dest: /etc/systemd/system/puma.service
  notify: reload puma

- name: Add config for DB connection
  template:
    src: db_config.j2
    dest: /home/appuser/db_config
    owner: appuser
    group: appuser

- name: enable puma
  systemd: name=puma enabled=yes
```

Создадим директорию для шаблонов templates и директорию для файлов files в директории роли ansible/roles/app. Скопируйте файл db_config.j2 из директории ansible/templates в директорию ansible/roles/app/templates, файл ansible/files/puma.service скопируем в ansible/roles/app/files.

Не забудем определить используемый хендлер в отдельную директории роли app:

ansible/roles/app/handlers/main.yml

```
---
# handlers file for app
- name: reload puma
  systemd: name=puma state=reloaded
```

Не забудем также определить переменную по умолчанию для задания адреса подключения к MongoDB.

ansible/roles/app/defaults/main.yml

```
---
# defaults file for app

db_host: 127.0.0.1
```

Вызов ролей

Используем роли в созданных ранее плейбуках. Удалим определение тасков и хендлеров в плейбуке ansible/app.yml и заменим на вызов роли:

ansible/app.yml

```
---
- name: Configure App
  hosts: app
  become: true

  vars:
   db_host: 10.132.0.2

  roles:
    - app
```

ansible/db.yml

```
---
- name: Configure MongoDB
  hosts: db
  become: true

  vars:
    mongo_bind_ip: 0.0.0.0

  roles:
    - db
```

Проверка ролей

Для проверки роли пересоздадим инфраструктуру окружения **stage**, используя команды:

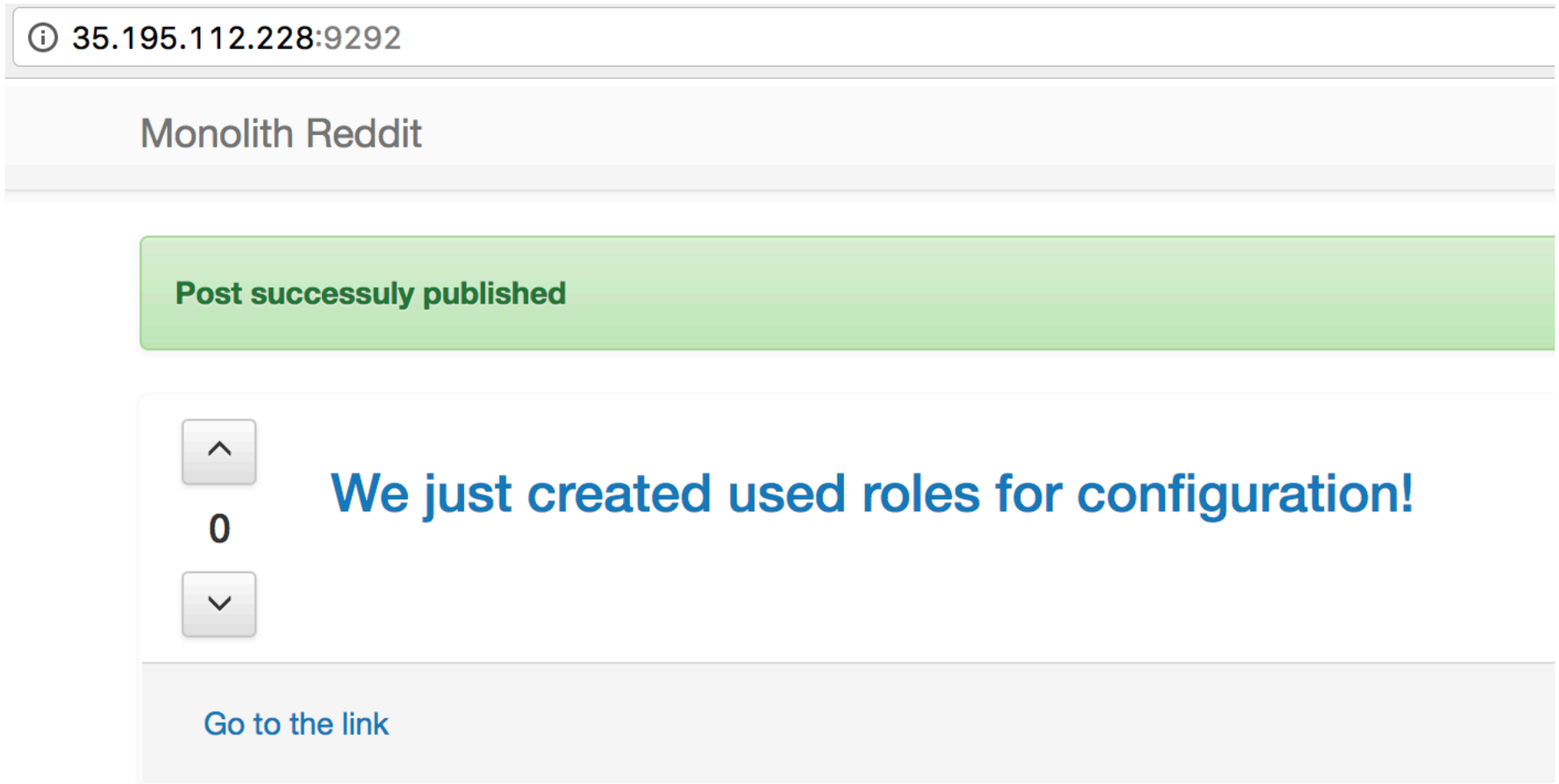
```
$ terraform destroy
$ terraform apply -auto-approve=false
```

Проверка ролей

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

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

Проверим работу приложения



Окружения

Окружения

Как мы помним из занятий по тераформу, обычно наша инфраструктура состоит из нескольких окружений. Эти окружения могут иметь небольшие отличия в настройках инфраструктуры и конфигурации управляемых хостов. Мы уже описали инфраструктуру тераформом для тестового и боевого окружения (продакшена).Теперь используем Ansible для управления каждым из них.

environments

Создадим директорию **environments** в директории ansible для определения настроек окружения. В директории ansible/environments создадим две директории для наших окружений **stage** и **prod**.

Inventory file

Так как мы управляем разными хостами на разных окружениях, то нам необходим свой инвентори файл для каждого из окружений.

Скопируем инвентори файл ansible/hosts в каждую из директорий окружения environtents/prod и environments/stage.

Сам файл ansible/hosts при этом удалим.

Окружение по умолчанию

Теперь, когда у нас два инвентори файла, то чтобы управлять хостами окружения нам необходимо явно передавать команде, какой инвентори мы хотим использовать. Например, чтобы задеплоить на prod мы должны теперь написать:

```
$ ansible-playbook -i environments/prod/hosts deploy.yml
```

Это всегда нам дает осознать, с каким окружением мы работаем, перед тем как применить изменения. В нашем случае, мы также определим окружение по умолчанию (stage), что упростит команду для тестового окружения.

Определим окружение по умолчанию в конфиге Ansible:

ansible/ansible.cfg

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

Переменные групп хостов

Параметризация конфигурации ролей за счет переменных дает нам возможность изменять настройки конфигурации, задавая нужные значения переменных. Ansible позволяет задавать переменные для групп хостов, определенных в инвентори файле. Воспользуемся этим для управления настройками окружений.

group_vars

Директория **group_vars**, созданная в директории плейбука или инвентори файла, позволяет создавать файлы (имена, которых должны соответствовать названиям групп в инвентори файле) для определения переменных для группы хостов.

Создадим директорию group_vars в директориях наших окружений environments/prod и environments/stage.

Конфигурация stage

Зададим настройки окружения stage, используя групповые переменные.
Создадим файлы stage/group_vars/app для определения переменных для группы хостов app, описанных в инвентори файле stage/hosts

Скопируем в этот файл переменные, определенные в плейбуке ansible/app.yml. Определение переменных из самого плейбука ansible/app.yml удалим:

ansible/environments/stage/group_vars/app

```
db_host: 10.132.0.2
```

Аналогичным образом определим переменные для группы хостов БД на окружении stage. Создадим файл stage/group_vars/db и скопируем в него содержимое переменные из плейбука ansible/db.yml. Секцию определения переменных из самого плейбука ansible/db.yml удалим.

ansible/environments/stage/group_vars/db

```
mongo_bind_ip: 0.0.0.0
```

Помним, что по умолчанию Ansible создает группу all для всех хостов указанных в инвентори файле. Создадим переменную, которую будут иметь все хосты окружения. Создайте файл stage/group_vars/all со следующим содержимым:

ansible/environments/stage/group_vars/all

```
env: stage
```

Конфигурация prod

Конфигурация окружения prod будет идентичной, за исключением переменной env, определенной для группы all. Для настройки окружения prod скопируйте файлы app, db, all из директории stage/group_vars в директорию prod/group_vars.

В файле prod/group_vars/all измените значение env переменной на prod:

```
env: prod
```

Вывод информации об окружении

Для хостов из каждого окружения мы определили переменную env, которая содержит название окружения. Определим вывод информацию об окружении, с которым мы работаем, при применении плейбука.

Определим переменную по умолчанию env в используемых ролях:

ansible/roles/app/defaults/main.yml

```
---
# defaults file for app
db_host: 127.0.0.1
env: local
```

ansible/roles/db/defaults/main.yml

```
---
# defaults file for db
mongo_port: 27017
mongo_bind_ip: 127.0.0.1
env: local
```

Вывод названия окружения

Будем выводит информацию о том, на каком окружении находится конфигурируемый хост. Воспользуемся модулем **debug** для вывода значения переменной. Добавим следующий таск в начало наших ролей:

ansible/roles/app/tasks/main.yml

```
---
# tasks file for app

- name: Show info about the env this host belongs to
  debug:
    msg: "This host is in {{ env }} environment!!!"
```

Добавим такой же таск в роль db.

ansible/roles/db/tasks/main.yml

```
---
# tasks file for db

- name: Show info about the env this host belongs to
  debug:
    msg: "This host is in {{ env }} environment!!!"
```

Проверка работы с окружениями

Для проверки пересоздадим инфраструктуру окружения **stage**, используя команды:

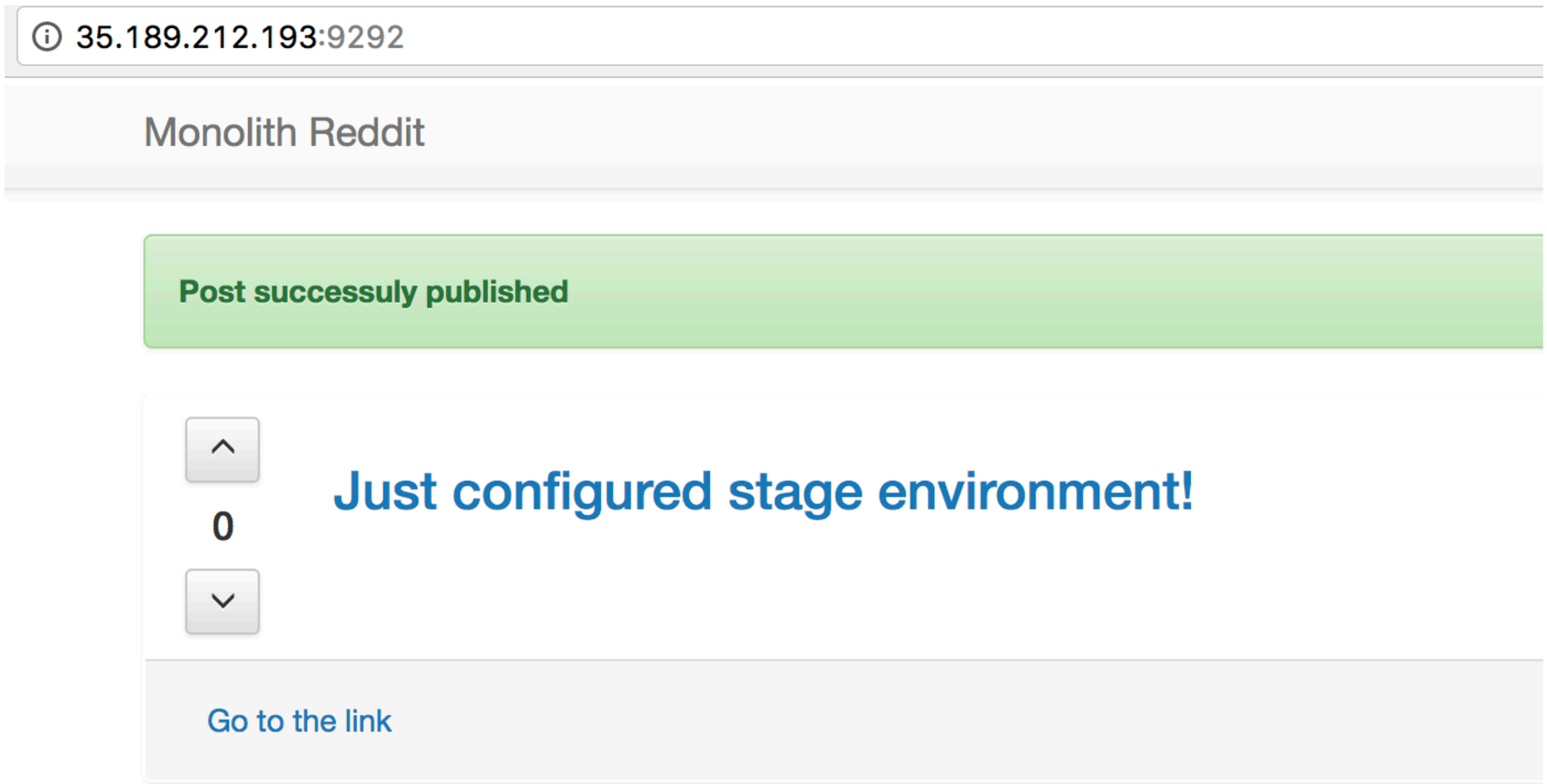
```
$ terraform destroy
$ terraform apply -auto-approve=false
```

Настройка stage окружения

Перед проверкой не забудьте изменить внешние IP адреса инстансов в инвентори файле ansible/environments/stage/hosts и переменную db_host в stage/group_vars/app:

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

...

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

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

TASK [db : Show info about the env this host belongs to]
********************************************
ok: [dbserver] => {
    "msg": "This host is in stage environment!!!"
}

...
```

Проверим работу приложения



Проверим настройку prod окружения

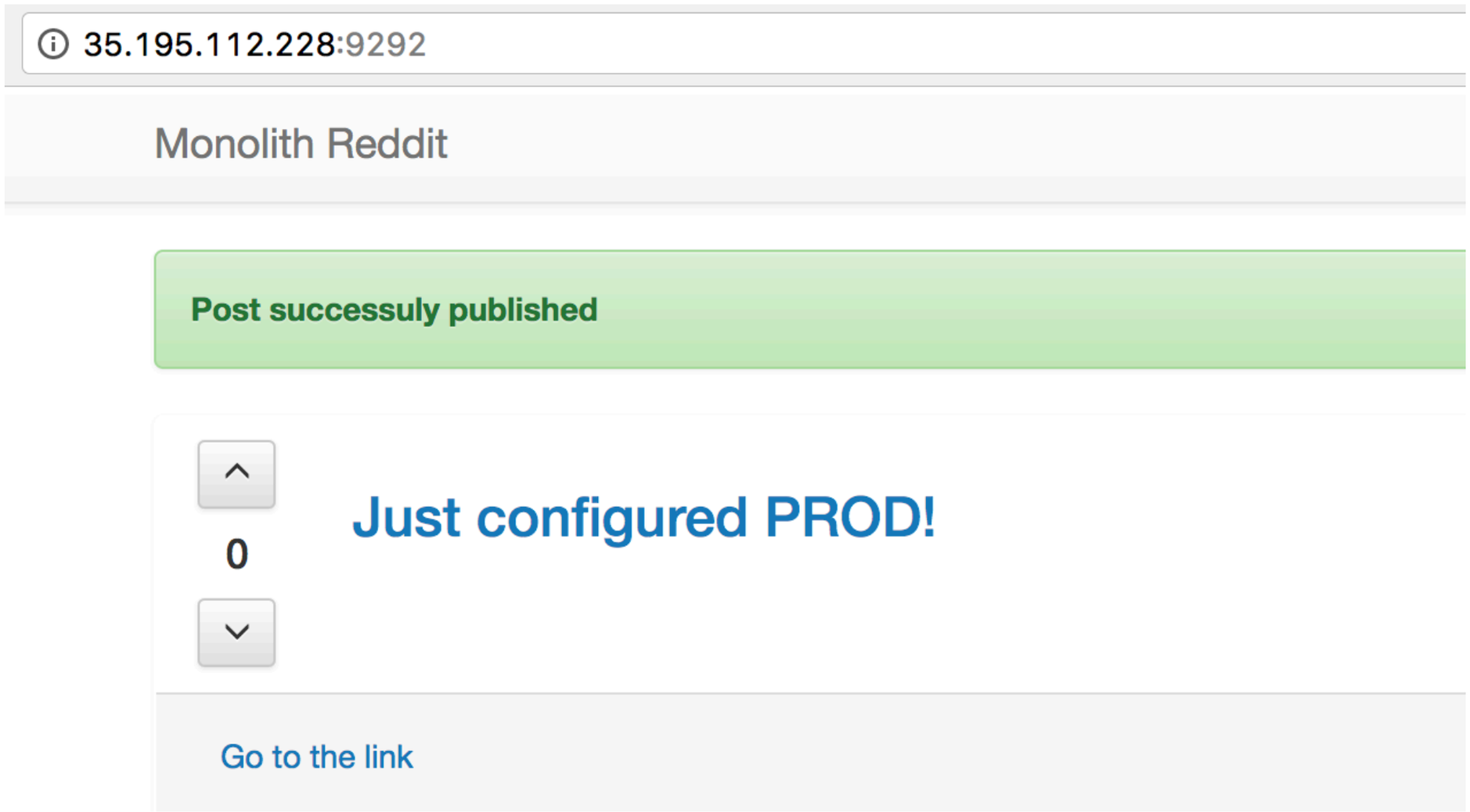
Для проверки настройки **prod** окружения сначала удалим инфраструктуру окружения stage. Затем поднимем инфраструктуру для prod окружения.

Настройка stage окружения

Перед проверкой не забудьте изменить внешние IP адреса инстансов в инвентори файле ansible/environments/prod/hosts и переменную db_host в prod/group_vars/app:

```
$ ansible-playbook -i environments/prod/hosts site.yml --check
$ ansible-playbook -i environments/prod/hosts site.yml

...

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

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

TASK [db : Show info about the env this host belongs to]
*******************************************
ok: [dbserver] => {
    "msg": "This host is in prod environment!!!"
}

...
```

Проверим работу приложения



Задание со звездочкой

В качестве дополнительного задания настройте работу приложения на 80 порту, используя nginx как обратный прокси. Используйте при этом одну из комьюнити ролей. Например, можно воспользоваться ролью jdauphant.nginx