# CI/CD pipeline

Zoia Holubkova

Relevance:

- I’ve chosen Django because its extremely scalable. Also web apps whose development are easy to manage and that can handle big amount of requests are in demand today

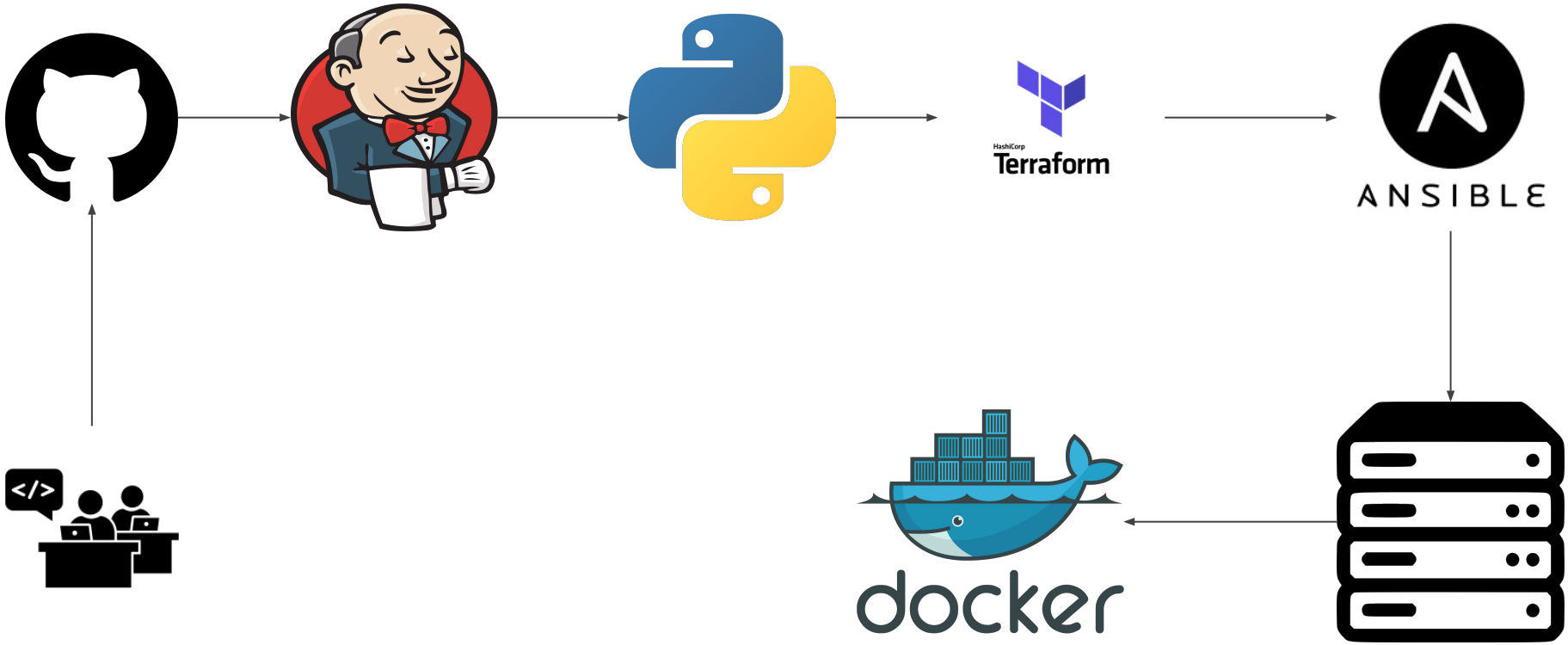
Goal:

- Automate process of test and deployment as much as possible
- Personally for me also is a goal to dive into CI/CD tools

# Technology stack

- Github
- AWS
- Jenkins
- Docker
- Terraform
- Ansible

# Project scheme



# Realization

At first, we make commit of our code and make push to Github:

```
PS D:\Project> git add .\random_animals\
PS D:\Project> git commit -m "change animal of the day"
[dev 9d5f091] change animal of the day
 2 files changed, 2 insertions(+), 2 deletions(-)
PS D:\Project> git push origin dev
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 596 bytes | 596.00 KiB/s, done.
Total 8 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/alieninochi/library-django.git
   32552bd..9d5f091  dev -> dev
PS D:\Project>
```

For Jenkins job I've used pipeline script. Build triggers were made via webhook, when we push some code to Github:

# Вывод на консоль

```
Started by GitHub push by alieninochi
Obtained jenkinsfile_config from git git@github.com:alieninochi/library-django.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/deploy_app
```

Next step is create virtual environment for app and run some unittests.

```
stages {
    stage('Testing app') {
        steps {
            dir('random_animals') {
                sh 'virtualenv venv'
                sh 'source ./venv/bin/activate'
                sh 'pip install -r requirements.txt'
                sh 'python3 manage.py test'
            }
            echo 'Done!'
        }
    }
```

```
+ python3 manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
https://media4.giphy.com/media/3fx3c2Xzxa22dlunbV/giphy.gif?cid=3ac49
.
----------------------------------------------------------------------
Ran 1 test in 0.047s

OK
Destroying test database for alias 'default'...
[Pipeline] }
[Pipeline] // dir
[Pipeline] echo
Done!
```

If tests were successful, Jenkins goes to next stage, where Terraform start to build infrastructure:

```
stage('Terraform Building Infrastructure') {
    steps {
        dir('terraform') {
            sh 'terraform init'
            sh 'terraform apply -auto-approve'
            sh 'terraform output | grep -E -o "([0-9]{1,3}[\\.]){3}[0-9]{1,3}" >> ../ansible/hosts'
        }
        echo 'Done!'
    }
}
```

When all necessary units is up, Ansible comes into play with its playbooks. With playbook we configure our instance – we need to install Docker:

```
stage('Ansible Config Servers') {
    steps {
        dir('ansible') {
            sh 'cat hosts'
            sh 'ansible -i hosts all --private-key $ANSIBLE_KEY -m ping'
            sh 'ansible-playbook -i hosts --private-key $ANSIBLE_KEY servcfg.yml'
        }
        echo 'Done!'
    }
}
```

Dockerfile for app:

```
FROM python:3.9

COPY ./random_animals /srv/www/random_animals
WORKDIR /srv/www/random_animals
RUN pip install -r requirements.txt
```

# docker-compose file:

```yaml
services:
  nginx:
    restart: always
    image: nginx:latest
    expose:
      - 8080
    ports:
      - "80:8080"
    volumes:
      - ./random_animals/static:/srv/www/random_animals/static
      - ./random_animals/media:/srv/www/random_animals/media
      - ./random_animals/logs:/srv/www/random_animals/logs
      - ./docker/nginx:/etc/nginx/conf.d
    depends_on:
      - python
```

```yaml
  python:
    restart: always
    build:
      context: .
      dockerfile: docker/python/Dockerfile
    volumes:
      - ./random_animals:/srv/www/random_animals
    expose:
      - 8000
    ports:
      - 8000:8000
    command: "gunicorn -c gunicorn.py random_animals.wsgi"
```

Last stage is deployment of app on the server. Here I also use Ansible playbook for copying files for app and Docker:

```
stage('Deploying app') {
    steps {
        dir('ansible') {
            sh 'echo "DEBUG = False" >> ../random_animals/random_animals/settings.py'
            sh 'ansible-playbook -i hosts --private-key $ANSIBLE_KEY push_app.yml'
        }
        echo 'Done!'
    }
}
```

# And finally we can access our app