Udviklingsmiljøer

Gruppe DELTA

Søren Raben, Laura Emilie Jacobsen, Peter Rud Frankild, Noel Pedersen

Docker

Dockerfile

docker-compose.yml

Python image, env settings, apk software, kopier base app og angiver work dir.

Installer pip software og angiv entrypoint filen.

Dockerfile > ...

```
1   FROM python:3.10-alpine
2
3   ENV PYTHONUNBUFFERED=1
4   ENV PYTHONDONTWRITEBYTECODE=1
5
6   RUN apk add python3-dev postgresql-client postgresql-dev musl-dev build-base
7
8   COPY . /app
9   WORKDIR /app
10
11  RUN pip install -r /app/requirements.txt
12
13  ENTRYPOINT ["sh", "entrypoint.sh"]
```

Docker

Dockerfile

docker-compose.yml

Docker-compose format, services, volumes.

Containere til db, app, nginx.

Request til host machine 8000, nginx.

Nginx reverse proxy, klient til server.

```yaml
---
version: '3.9'

services:

  db:
    image: postgres:14-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data/
    env_file:
      - "env-${RTE:-dev}"

  app:
#    image: djapp:latest
    image: registry.gitlab.com/delta9961510/environments_project_2/djapp:latest

    volumes:
      - media:/media/
      - static:/static/
      - .:/app/
    depends_on:
      - db
    env_file:
      - "env-${RTE:-dev}"

  nginx:
    image: nginx:latest
    volumes:
      - media:/media/
      - static:/static/
      - ./nginx/:/etc/nginx/conf.d/
    ports:
      - 8000:8000

    depends_on:
      - app

volumes:
  postgres_data:
  media:
  static:
```

Continuous integration

Static

Build

Deploy

deploy.sh

Konfigurations fil til udføre af CI pipeline.

stages, scripts.

```yaml
stages:
  - static
  - build
  - deploy

static:
  stage: static
  image: python:3.10-alpine
  before_script:
    - pip install -r sqa-requirements.txt
    - apk update && apk add shellcheck
  script:
    - pylama .
    - djlint .
    - pip-audit
    - yamllint *.yml
    - shellcheck *.sh
```

Continuous integration

Static

Build

Deploy

deploy.sh

Der bygges et image, ved hjælp af en existerende docker-container som indeholder forskellige package.

```
19  build:
20    stage: build
21    image: registry.gitlab.com/delta9961510/composer:latest
22    services:
23      - docker:dind #docker in docker
24    variables:
25      DOCKER_DRIVER: overlay2
26    before_script:
27      - docker login -u gitlab -p "$GITLAB_CI_TOKEN" "$CI_REGISTRY"
28
29    script:
30      - docker build --pull -t "$CI_REGISTRY_IMAGE/djapp:latest" .
31      - docker push "$CI_REGISTRY_IMAGE/djapp:latest"
32      - RTE=test docker-compose up --abort-on-container-exit --exit-code-from app
```

Continuous integration

Static

Build

Deploy

deploy.sh

Deploy stage der gennem et before_script og CI pipeline kan deploy fra VPS.

```
33
34    deploy:
35      stage: deploy
36      before_script:
37        - 'command -v ssh-agent >/dev/null || ( apk add --update openssh )'
38        - eval $(ssh-agent -s)
39        - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
40        - mkdir -p ~/.ssh
41        - chmod 700 ~/.ssh
42        - ssh-keyscan $VM_IPADDRESS >> ~/.ssh/known_hosts
43        - chmod 644 ~/.ssh/known_hosts
44      script:
45        - ssh $SSH_USER@$VM_IPADDRESS sh "~/2023.03.10/environments_project_2/deploy.sh"
46      only:
47        - master
```

Continuous integration

Static

Build

Deploy

deploy.sh

Ved at give login-rettigheder til et Linode-drev med CI-variabler, kan vi køre disse kommandoer fra deploy.sh. Dette vil få docker til at genstarte med det nye docker image. Autoriser runners to do the job with GITLAB_CI_TOKEN.

↑ Key	Value
GITLAB_CI_TOKEN	*****
SSH_PRIVATE_KEY	*****
SSH_USER	*****
VM_IPADDRESS	*****

```
#!/bin/sh

echo "starting deploying"
cd ~/2023.03.10/environments_project_2 || exit
docker-compose down
docker-compose up -d --scale app=2
```

Django setup & DB

databases

nginx.conf

entrypoint.sh

Ignore-filer

Her ændrer vi standart Sqlite til Postgres. og kalder env filen.

```
67  DATABASES = {
68      'default': {
69          # 'ENGINE': 'django.db.backends.sqlite3',
70          # 'NAME': BASE_DIR / 'db.sqlite3',
71          'ENGINE': 'django.db.backends.postgresql_psycopg2',
72          'NAME': os.environ['POSTGRES_DB'],
73          'USER': os.environ['POSTGRES_USER'],
74          'PASSWORD': os.environ['POSTGRES_PASSWORD'],
75          'HOST': 'db',
76          'PORT': '5432',
77      }
78  }
```

Django setup & DB

databases

nginx.conf

entrypoint.sh

Ignore-filer

upstream fra nginx til app_1 og app_2.

proxy_set host til proxy_pass app_upstream server.

```
upstream app_upstream {
    server environments_project_2_app_1:8000;
    server environments_project_2_app_2:8000;
}

server {
    server_name localhost;
    listen 8000;

    location / {
        proxy_set_header Host $host;
        proxy_pass http://app_upstream;
    }
}
```

Django setup & DB

databases

nginx.conf

entrypoint.sh

Ignore-filer

Chek, makemigration, migrate og Runtime Environment.

```sh
#!/bin/sh

python manage.py check
python manage.py makemigrations
python manage.py migrate

case "$RTE" in
    dev )
        echo "** Development mode."
        pip-audit
        coverage run --source="." --omit=manage.py manage.py test --verbosity 2
        coverage report -m
        python manage.py runserver 0.0.0.0:8000
        ;;
    test )
        echo "** Test mode."
        pip-audit || exit 1
        coverage run --source="." --omit=manage.py manage.py test --verbosity 2
        coverage report -m --fail-under=75
        ;;
    prod )
        echo "** Production mode."
        pip-audit || exit 1
        python manage.py check --deploy
#        gunicorn project.asgi:application -b 0.0.0.0:8080 -k uvicorn.workers.UvicornWorker
        ;;
esac
```

Django setup & DB

databases

nginx.conf

entrypoint.sh

Ignore-filer

.gitignore

```
*.swp
.DS_Store
*.pyc
__pycache__/
db.sqlite3
.coverage
_site
.sass-cache
.jekyll-metadata
Gemfile.lock
```

.dockerignore

```
README.md
.gitignore
.git
Dockerfile
*.swp
.DS_Store
*.pyc
__pycache__/
db.sqlite3
.coverage
```

Udvikling og miljøer

Krav specifikationer

VPS

Gitlab

Stakeholders	System	Moscow
Bruger	Functional	Strategi
Business	Non functional	Prioritering

id	prioritet	beskrivelse	Kravtype	Type
1	Must have	Vores udviklingsmiljø skal være stabilt og ikke gå ned når der er mange på	Ikke-funktionelt	System requirement
2	Must have	Udviklingsmiljøet skal være funktionelt på tre operativsystemer Windows, Linux og Mac os	Ikke-funktionelt	System requirement

Udvikling og miljøer

Krav specifikationer

VPS

Gitlab

Miljøet	Framework	Editor
Linode drev	Django	Vim, Tmux
Linux Alpine	Startproject	CLI
Python 3	MVT	VS code

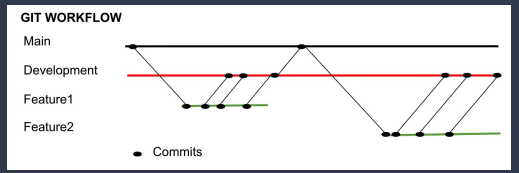
Udvikling og miljøer

Krav
specifikationer

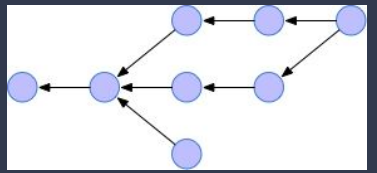
VPS

Gitlab

Version control systems



Direct acylic graph



Issues & Milestones