

Имена: Георги Киряков

фн: 81110

Имена: Даниел Шушков

фн: 81102

Начална година: 2014

Програма: бакалавър, (СИ)Курс: 4

Тема: Управление на конфигурациите на уеб приложение

Дата: 2018-01-23

Предмет: wwwTech2017_18_9ed_KN_winter

github: <https://github.com/g-ki/capitan>

преподавател: доц. д-р Милен Петров

Capitan

1. Условие - управление на конфигурациите на уеб приложение

Да се направи уеб-базирана система, която позволява да се конфигурира друго уеб приложение - добавяне на приложение, неговите настройки - Дали е локална, дали е достъпна онлайн и входно-изходни услуги.

2. Въведение

Приложението е базирано на клиент-сървър модела. То е направено с цел да се опрости управлението и следенето на натовареността на Docker инстанцията, на която е вдигнат сървърът на приложението. Приложението позволява лесно свързване на още машини с няколко клика и разпределянето на задачи между тях. Приложението също ни дава и таблици със статистики, които са взети в момента на запитването. Дава ни се възможността да създадем Digital Ocean машина с натискането на един бутон, която да се включи автоматично в Swarm-a, създаден от приложението. Чрез приложението може лесно да се създават нови задачи и да се разпределят в swarm-a.

3. Теория

За комуникацията между сървъра и клиента се използва HTTP протокол. Повечето от съдържанието, сервирано от сървъра, е статично, като html-a се генерира от него и просто се праща за рендериране на клиента. За комуникацията между различните представители на Docker Swarm-a се използва интернет връзка и HTTP протокол. Заявките на сървъра се обработват от помощна библиотека за изграждането на сървър на Python. Вземането на нужната ни информация се извършва, чрез правенето на call-ове към docker демона.

4. Използвани технологии

Използвани технологии за сървър са - Python, Docker Compose, Docker Swarm, Flask, SQLite3, Jinja. За реализацията на проекта ни бе нужна docker-py, което е open source библиотека, която дава възможност за получаване на информация от Docker демона и свързването на нови машини и задаването на задачи към тях. Това е и основната причина да изберем Python за реализирането на сървър частта. За по-лесното използване и билдване на проекта използваме Docker Compose, а за организацията и оркестрацията на различните машини Docker Swarm. За база използваме олекотената SQLite, защото базата ни е малка и нямаме нужда от скалируемост и много връзки между таблици. За избягване на копиране на html използваме темплейтите на Jinja, с които генерираме нашия html. За стилизирането на проекта използваме CSS, а за динамичното съдържание използваме Javascript. За да използваме повече от една машина използваме облачната услуга - Digital Ocean, където пускаме отдалечени машини, които връзваме в Swarm мрежа.

5. Инсталация и настройки

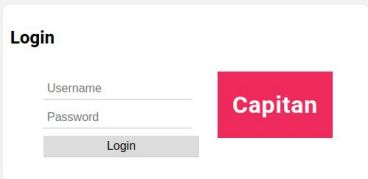
За инсталирането на проекта трябва да има на машината docker-compose и връзка към интернет. Изпълняването на скрипт, който да направи базата и да създаде потребител с администраторска роля. С командата docker-compose up web се стартира нашето приложение. Може да се използва и **docker build <github-repo>**, за да се стартира контейнер с нашето приложение на машината. След пускането на контейнера трябва да се влезе в него и да се изпълнят последователно командите **export FLASK_APP=/capitan/capitan/autoapp.py** и **flask initdb**, които създават базата и потребител с потребителско име admin и парола admin. При изпълнението на същата команда базата на приложението се затрива. За да успеем да се вържем към Digital Ocean и да правим и изтриваме нови машини ни е нужен **API key**, който да ни аутентикира пред тяхното API. За тази цел се генерира ключ от техния сайт и се поставя във файл instance/settings.py с DOCEAN_TOKEN = 'YOUR_API_KEY'. Други настройки на приложението не са необходими. За функционирането на приложението е необходимо Docker версията на машината да е над **1.12.0**.

6. Кратко ръководство на потребителя

Потребителя може да се аутентикира пред системата със потребителско име и парола, което съответно дава ограничен достъп до системата. Може да следи натовареността на самата машина и да следи статистики за нея. Дава се възможност и да се следят всички машини и информация свързана с тях, които

са свързани към нашия swarm. Потребителя може сам да добавя машини към мрежата, като просто използва генерираните от системата token-и в зависимост от ролята, която иска да даде на машината. Позволява се добавянето на възли от мрежата, чрез вдигане на инстанция в Digital Ocean, като се пуска автоматично скрипт за сваляне на Docker и прикачане към swarm-а. Потребителят може да следи наличните в мрежата сървиси и съответно да създава такива с попълване на нужната за това форма. Потребителят има възможност да вижда всички регистрирани в системата потребители и да добавя нови с име и парола. За да използваме SSH ключове на потребителя, той може да вижда собствените си ключове и да си добавя нови. SSH ключовете се използват за създаването на Digital Ocean инстанцията, като чрез тях на по-късен етап, потребителят може да се свързва за собствените си инстанции.

7. Примерни данни



The image shows a login form titled "Login" centered on a light gray background. The form is a white rectangle with a thin gray border. It contains the following elements: the title "Login" in bold black text at the top left; a "Username" label followed by a text input field; a "Password" label followed by a text input field; a "Login" button with a gray gradient; and a red rectangular button with the white text "Capitan" positioned to the right of the password field.

Фиг.1>Login формата на системата

Capitan

DASHBOARD
NODES
SERVICES
USERS
KEYS

Dashboard

log out

Metrics

Metric	Value
Containers	71
ContainersRunning	4
ContainersPaused	0
ContainersStopped	67
Images	230
DockerRootDir	/var/lib/docker
Name	dani
Driver	aufs
SwarmAddr	192.168.0.105

Фиг. 2 Статус на системата на сървъра

Capitan

DASHBOARD
NODES
SERVICES
USERS
KEYS

Nodes

log out

Swarm

+ Create

Host	Role	State	ID	EngineVersion	IP	Manager IP
dani	manager	ready	kjv8cz3pyvildo49o9ipfb2ef	17.09.1-ce	192.168.0.105	192.168.0.105:2377

Cloud

Name	Region	Power	Status	IP
test-master	Frankfurt 1	1/1024	active	159.89.8.221

Join tokens

Worker Token

docker swarm join --token SWMTKN-1-17ned1wx7snbp4uqlazktrd0dy0o95l8crywckgwo3klunq0i-511947kapi0z0ctsba33o6w8q 192.168.0.105:2377

Manager Token

docker swarm join --token SWMTKN-1-17ned1wx7snbp4uqlazktrd0dy0o95l8crywckgwo3klunq0i-6iaxxnv6fktkylhaebb1a3nfa 192.168.0.105:2377

Фиг. 3 Статус на инстанциите и линкове за връзване към Swarm-а

Capitan

DASHBOARD
NODES
SERVICES
USERS
KEYS

Services

log out

+ Create

Name	Image	Replicas
helloworld	alpine	1
helloworld1	alpine	1

Фиг. 4 Списък с активните сървиси в Swarm-a

Capitan

DASHBOARD
NODES
SERVICES
USERS
KEYS

Create wroker

log out

DigitalOcean Droplet

User data

```
#!/bin/bash
wget -qO- https://get.docker.com/ | sh
sudo usermod -aG docker $(whoami)
docker swarm join --token SWMTKN-1-17ned1wx7snbp4uqlazktrd0dy0o95l8crywckgwo3klunq0i-5i1947kpioz0ctsba33o6w8q 192.168.0.105:2377
```

Create

Фиг. 5 Форма за добавяне на работник към мрежата. Скриптът се изпълнява при създаването на Digital Ocean машината

The screenshot shows a web application interface for 'Capitan'. On the left is a dark sidebar with a red header containing the word 'Capitan'. The sidebar menu includes 'DASHBOARD', 'NODES', 'SERVICES', 'USERS', and 'KEYS'. The main content area is titled 'Add Service' and contains a form with the following fields: 'Service name:', 'Image:', 'Command:', 'Arguments:', 'Constraint: (ex. node.hostname==node-2)', and 'Replicas:'. Each field has a corresponding text input. At the bottom of the form is a green button labeled 'Add'. In the top right corner of the main area, there is a 'log out' link.

Фиг. 6 Форма за стартиране на сървис

8. Описание на програмния код

Структурата на проекта е разделена спрямо логическата типизация на файловете. В самото репо се две папки за конфигурация, папка с програмния код `capitan` и конфигурационни файлове за билдване и нужните библиотеки на приложението. В папката `blueprints` се помещават файлове са всяка груба от `endpoint`-и, който връщат отговор в клиент-сървърната ни архитектура. В папката `static` са поместени файловете за CSS и JavaScript. Папката `templates` съдържа целия html код на приложението, като е разделен на модулите в самото приложение, които отговарят на `blueprint` разделението на пътищата. В папката `capitan` се помещават и най-важните файлове на проекта, а именно `app.py` и `schema.sql`, които съответно зареждат приложението и съдържат схемата на базата.

9. Ограничения и възможности за бъдещо разширение

Приложението не може да показва в реално време и да се правят повече неща с контейнерите и сървисите. Може да се направят нещата като бъдещо развитие и да се направи като докер имидж като се сетъпва на различни инстанции. Бихме го качили в Docker Hub, където ще може да се сваля свободно като `docker image`. Въвеждане на повече роли и пазене на интеракцията на потребителите с приложението в базата от данни.

10. Какво научих

Научихме се да се справяме бързо с проблемите. Да използваме нови технологии за нас и да правим статични сайтове, което не сме правили от доста време. Идеята на проекта ни запозна с нова и иновативна за нас технология като Docker Swarm, което бихме могли да използваме в по-големи проекти.

11. Използвани източници

1. <https://docs.docker.com/engine/swarm/> - Docker Swarm Docs
2. <https://docker-py.readthedocs.io/en/stable/client.html> - Docker-py Docs
3. <http://flask.pocoo.org/docs/0.12/> - Flask Docs
4. <http://jinja.pocoo.org/docs/2.10/> - Jinja Docs
5. <https://developers.digitalocean.com/documentation/> - Digital Ocean API Docs

Предал:
/81102, Д.Шушков, КН 6гр/

Предал:
/81110, Г.Киряков, КН 5гр/

Приел:
/доц.д-р *Милен Петров*/