METODE DE UPGRADE IN KUBERNETES

- APLICATIA -

Eusebiu Rizescu

1. Introducere

Semestrul trecut am prezentat diverse metode de upgrade in Kuberenetes. Atunci, am folosit o aplicatie simpla, pentru a testa aceste metode de upgrade (recreate, rolling-update, blue/green, canary, A/B testing). In acest semestru am lucrat la o aplicatie reala, practica, mai complexa, pe care sa aplic acelasi metode de upgrade. In acest document va fi prezentata aplicatia, urmand ca pana la prezentarea dizertatiei, sa deployez aceasta aplicatie intr-un cluster de Kubernetes si sa analizez metodele de upgrade direct pe ea.

Aplicatia nu are un scop clar, este mai degraba un sablon pentru un magazin online. Pe pagina principala avem produsele aflate in vanzare, iar cumparatorii isi pot adauga in cosul de cumparaturi produsele dorite. Pentru a lansa o comanda, cumparatorul trebuie sa isi creeze un cont, sa se autentifice, sa isi adauge produsele dorite in cosul de cumparaturi, si sa lanseze comanda.

2. Tehnologiile folosite

Aplicatia este scrisa in Python cu ajutorul framework-ului Flask. Am folosit si Bootstrap, care este un framework de CCS pentru a nu reinventa roata. Aplicatia foloseste ca stocare persistenta o baza de date relationala, si anume MySQL. Pentru a interactiona cu baza de date am folosit SQLAlchemy. SQLAlchemy este un toolkit si ORM care functioneaza ca un wrapper peste majoritatea bazelor de date relationale. Aceasta functioneaza prin maparea tabelelor din baza de date in clase din Python. Totodata, SQLAlchemy are si mecanisme ce previn SQL injection (acesta traduce comenzile in limbajul bazei de date si nu le executa direct).

In prezent, aplicatia ruleaza in Docker (un container cu aplicatia, si un container cu baza de date). Imaginea pentru aplicatie este construita plecand de la Apline, si continand toate dependintele necesare (Python, pip, pachetele de python necesare, etc). Imaginea de MySQL este una oficiala. Pornirea containerelor este facuta cu ajutorul docker-compose. In final, aplicatia va fi deployata pe un cluster de Kubernetes, pentru a testa metodele de upgrade prezentate in semestrul anterior.

3. Diagrama aplicatiei

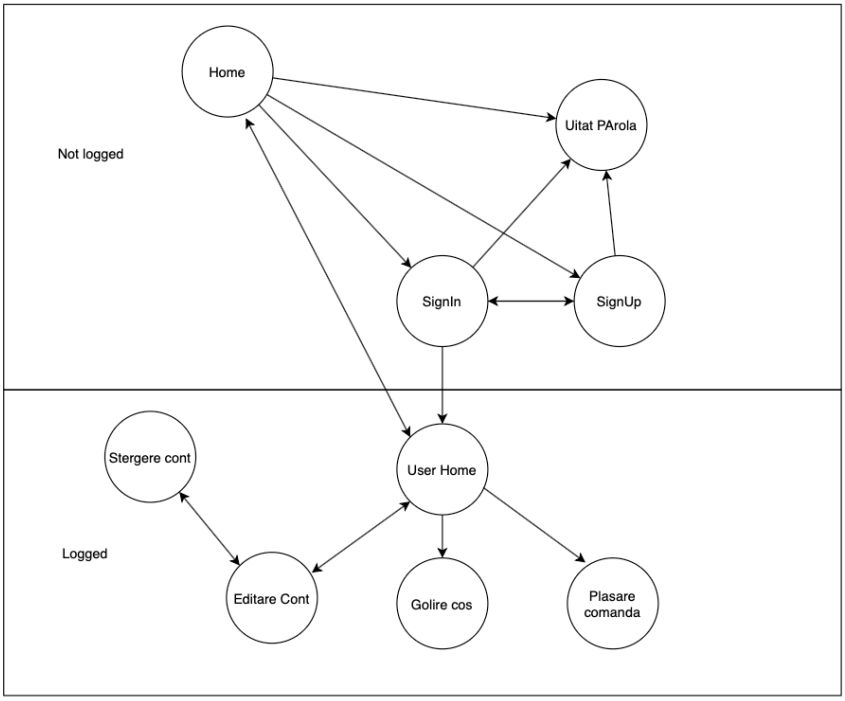

Fig. 1

Paginile se impart in doua categorii:

a. Care nu necesita ca utilizatorul sa fie autentificat:

- Home
  - Pagina principala. Aici se pot vedea produsele puse in vanzare de catre magazin. Pentru a putea adauga in cos produse, utilizatorul trebuie sa aiba cont si sa fie logat.
- SignUp
  - Pagina de inscriere. De aici un utilizator nou poate completa un formular pentru a isi crea un cont nou. Acesta va primi pe mail un link de activare.
- SignIn
  - Pagina de autentificare. De aici un utilizator se poate autentifica pentru a putea adauga produse in cosul de cumparaturi, goli cosul de cumparaturi, sau lansa o comanda.
- Uitat parola
  - Pagina in care se poate reseta parola contului. Se completeaza un formular care contine adresa de email, iar daca exista un cont

asociat acelei adrese de email, se va trimite un link de resetare a parolei.

b. Care necesita ca utilizatorul sa fie autentificat:

- User Home
  - Pagina pe care un utilizator o vede dupa ce se logheaza. Aici se poate vedea continutul cosului de cumparaturi, goli cosul de cumparaturi, lansa o comanda.
- Editare cont
  - Pagina de editare cont. De aici utilizatorul isi poate actualiza informatiile sale: numele, adresa de email, adresa fizica, parola.
- Stergere cont
  - Pagina de stergere cont. De aici un utilizator isi poate sterge contul. Aceasta operatiune este ireversibila, deoarece contul va fi sters permanent din baza de date
- Golire cos de cumparaturi
  - Cosul de cumparaturi va fi golit. Nu va fi trimisa nici o comanda..
- Plasare comanda
  - De aici utilizatorul poate plasa o comanda cu produsele aflate in cosul de cumparturi.

Pagina principala. Deoarece utilizatorul este autentificat (in dreapta sus apare butonul “logout” ceea ce inseamna ca utilizatorul este autentificat), utilizatorul poate sa adauge in cos produse. Daca nu ar fi fost autentificat, butonul de adaugat in cosul de conparaturi ar fi fost greyed out, si nu ar fi putut sa adauge in cos produsul dorit.

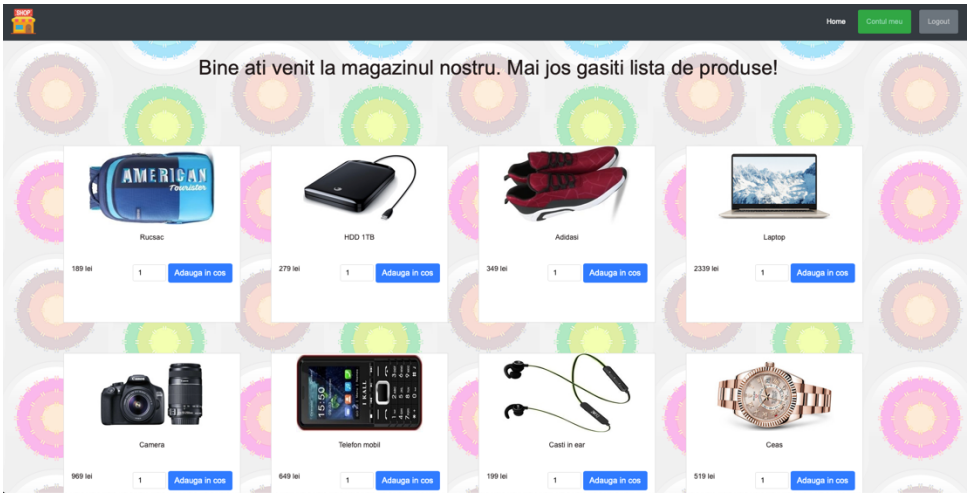

Fig. 2

User home - Pagina utilizatorului. Aceasta este pagina catre care utilizatorul va fi rutat dupa autentificare. De aici utilizatorul isi poate edita si sterge contul, isi poate goli cosul de cumparaturi, si poate lansa o comanda. Tot din aceasta pagina, prin apasarea butonului “Editare cont” utilizatorul va fi rutat catre pagina de editare a contului.

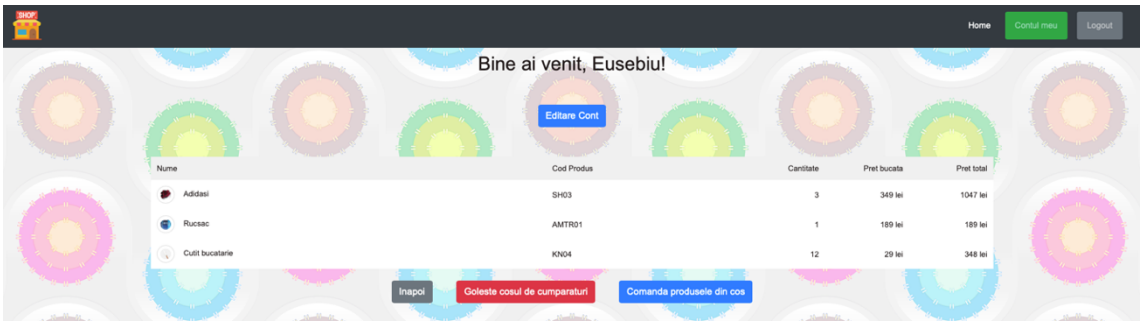

Fig. 3

Pagina editare cont. De aici utilizatorul isi poate actualiza numele, adresa de email, parola si adresa fizica, sau isi poate sterge contul. Pentru a isi actualiza datele personale, utilizatorul trebuie sa isi completeze campurile pe care doreste sa le actualizeze si sa apese butonul “Actualizeaza”. De mentionat ca stergerea contului este o operatiune ireversibila, deoarece toate datele asociate cu acest cont vor fi sterse din baza de date. Prin apasarea butonului “Inapoi” utilizatorul se poate intoarge la pagina “User Home”.

Fig. 4

4. Diagrama bazei de date

Baza de date contine doua tabele:

- tabela useri care contine userii
- tabela produs care contine produsele

Legatura dintre cele doua este facuta prin coloana “cosDeCumparaturi” a tabelei useri. Aceasta este un string care contine produsele (si cantitatea acestora) adaugate de utilizator in cos. Majoritatea campurilor sunt destul de usor de inteles, cu exceptia campurilor “activated” si “hash” care vor fi explicate in capitolul urmator. Acestea au legatura cu securitatea aplicatiei, si anume sunt folosite la crearea unui cont nou, si la accesarea functionalitatii de “Uitat Parola”.

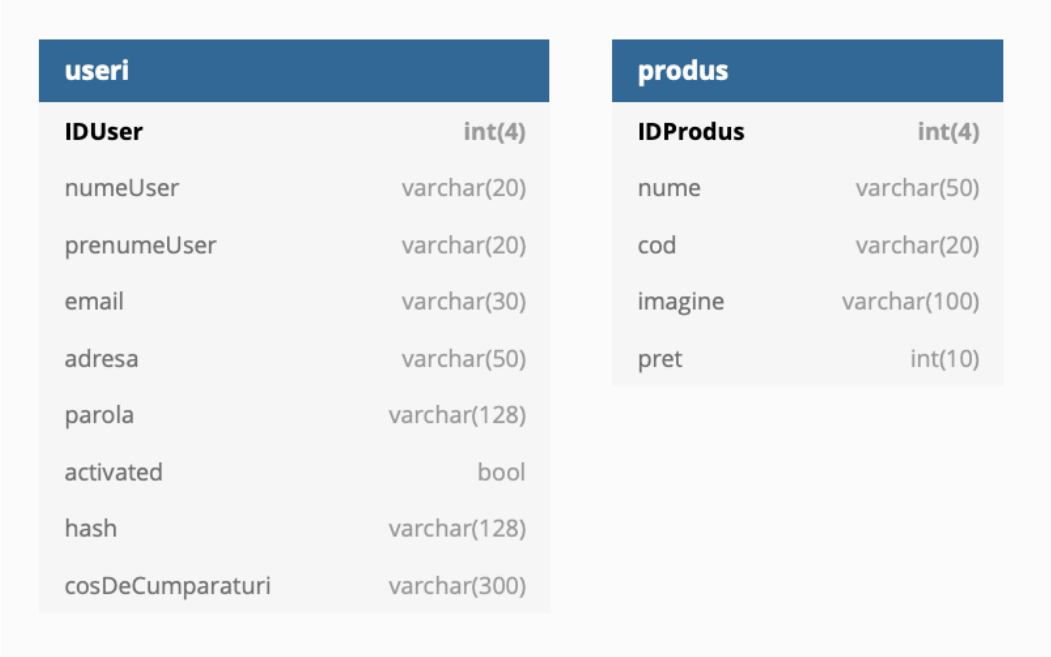

Fig. 5

5. Inscrierea si Autentificarea

Pentru ca un utilizator sa poata adauga produse in cosul de cumparaturi si implicit sa lanseze o comanda, aceasta trebuie sa fie autentificat. Pentru a se autentifica, mai intai are nevoie de un cont. Pentru a isi crea un cont (a se inscrie), utilizatorul trebuie sa acceseze pagina “Sign Up”, pagina dispobila de la pagina initial, prin apasarea butonului din dreapta sus numit “Sign Up”. Pentru a isi face un cont nou, utilizatorul trebuie sa completeze campurile de mai jos (Email, Nume, Prenume, Adresa, Parola, Confirmare Parola). Utilizatorul va primi bineinteles mesaje de eroare daca mailul nu este valid (spre exemplu nu contine “@”), sau daca mailul este deja existent in sistem, sau daca cele doua parole introduse in campurile “Parola” si “Confirmare Parola” nu corespund.

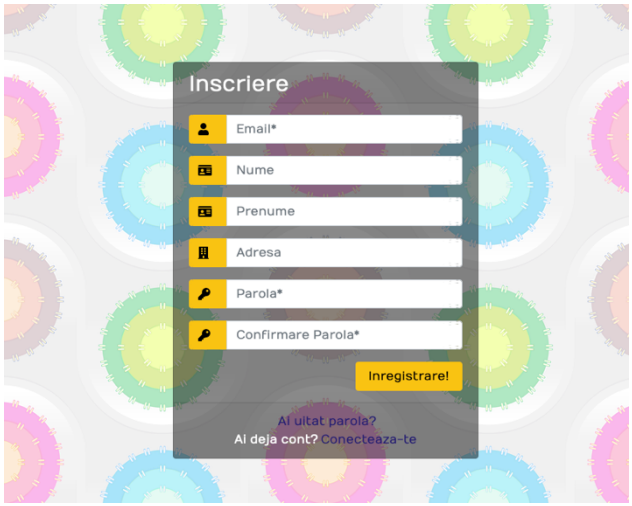

Fig. 6

Dupa introducerea campurilor si apasarea butonului “Inregistrare!”, utilizatorul va primi un mail cu un link de activare. Acel link de activare contine in el campul “hash” corespunator utilizatorului. Acel hash este format din email si timestamp-ul de cand a fost creat contul. Pana ce utilizatorul nu va accesa acel link, utilizatorul va avea campul “activated” setat pe fals, si nu se va putea autentifica. Acest pas de activare pe email, este pentru a verifica ca persoana care doreste sa isi faca un cont nou, detine acest email.

Pagina autentificare – pagina de unde un utilizator se poate autentifica:

Fig. 7

In baza de date parola utilizatorului nu este stocata in clar (din punct de vedere al securitatii). In schimb, este tinut un hash al acesteia (dupa cum se poate observa in figura 8). Acest hash este creat cvu ajutorul functiei “generate_password_hash” din catrul pachetului “wekzeug.security”, pachet care este folosit des in ecosistemul Python datorita gradului mare de securitate pe care il ofera.

```
mysql>
mysql> select * from useri where email="rizescueusebiu@gmail.com" \G
*************************** 1. row ***************************
     IDUser: 2
   numeUser: Rizescu
prenumeUser: Eusebiu
      email: rizescueusebiu@gmail.com
     parola: pbkdf2:sha256:50000$seGFleoM$c78987a943a81ff40a61c892e5c0e87f8c3972e2ebb00e4531b713f68b1fe378
     adresa: Str. Opulentei, nr. 1
   shopping: SH03=3;AMTR01=1;KN04=12
  activated: 1
       hash: 02bf5a0d67f55b90cb28cdaaffec5814ae9ab068
1 row in set (0.00 sec)

mysql>
```

Fig. 8

6. Bibliografie

[1] https://www.udemy.com/course/python-and-flask-bootcamp-create-websites-using-flask

[2] https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

[3] https://werkzeug.palletsprojects.com/en/2.0.x/utils/

7. Anexe

7.1 Baza de date

Baza de date este formata din doua tabele, “useri” si “produse”. Ambele tabele folosesc ca cheie primara un ID, care este incrementat automat la adaugarea unei noi intrari in cadrul tabelei.

```
mysql>
mysql> show tables;
+--------------------------+
| Tables_in_webapplication |
+--------------------------+
| produse                  |
| useri                    |
+--------------------------+
2 rows in set (0.00 sec)

mysql>
mysql> desc useri;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| IDUser      | int(11)      | NO   | PRI | NULL    | auto_increment |
| numeUser    | varchar(30)  | NO   |     | NULL    |                |
| prenumeUser | varchar(30)  | NO   |     | NULL    |                |
| email       | varchar(30)  | NO   |     | NULL    |                |
| parola      | varchar(128) | YES  |     | NULL    |                |
| adresa      | varchar(30)  | YES  |     | NULL    |                |
| shopping    | varchar(300) | YES  |     | NULL    |                |
| activated   | tinyint(1)   | YES  |     | NULL    |                |
| hash        | varchar(50)  | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql>
mysql> desc produse;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| IDProdus | int(11)      | NO   | PRI | NULL    | auto_increment |
| nume     | varchar(300) | NO   |     | NULL    |                |
| cod      | varchar(300) | NO   |     | NULL    |                |
| imagine  | varchar(300) | NO   |     | NULL    |                |
| pret     | int(11)      | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)

mysql>
```

Fig. 9

7.2 Docker compose

Pentru deploy-ul aplicatiei am folosit docker-compose pentru o mai usoara gestiune a containerelor. Stackul actual este format din doua servicii, si anume baza de date (care ruleaza si ea in Docker) si aplicatia. Pentru ambele servicii am configurat implicit volume cu nume, deoarece durata spatiului de stocare in docker este efemera (la oprirea containerului, datele nesalvate intr-un volum sunt pierdute).

```
version: '3.2'
services:
  # mysql -u root -p webapplication
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_DATABASE: 'webapplication'
      MYSQL_ROOT_PASSWORD: 'mypassword'
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
    volumes:
      -
/Users/eusebiu.rizescu/dizertatie/mysql:/var/lib/mysq
l
      - /Users/eusebiu.rizescu/dizertatie/mysql-
conf/mycustom.cnf:/etc/mysql/conf.d/mycustom.cnf
    networks:
      - webnet

  flask:
    image: webapplication
    container_name: flask
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
    ports:
      - '80:5000'
    depends_on:
      - mysql
    volumes:
      - /Users/eusebiu.rizescu/dizertatie/flask:/app
    networks:
      - webnet
```

```
networks:
  webnet:
    driver: overlay
    internal: false
```

7.3 Scriptul de deploy

Pentru o mai usoara dezvoltare a codului, si pentru un deploy rapid dupa o modificare, am folosit un script bash. Acest script bash, cand este executat cu parametrul “start” (sau fara parametru, “start” fiind cel default), codul sursa este copiat in volume, iar containerele sunt restartate.

```
#! /bin/bash

if [ "$1" == "help" ]; then
  echo "./build.sh help          <-  help"
  echo "./build.sh start         <-  start docker
compose"
  echo "./build.sh stop          <-  stop docker
compose"
  echo "./build.sh populate      <-  populate
database tables"
  exit
fi

if [ "$1" == "populate" ]; then
  docker exec $(docker ps | grep flask | cut -d "
" -f1 | head -1) python3 populateTables.py
  exit
fi

echo Exit Docker Swarm
sudo docker swarm leave --force
sleep 1
if [ "$1" != "stop" ]; then
  echo Init Docker Swarm
  sudo docker swarm init
fi
sleep 1

if [ "$1" != "stop" ]; then
  echo "Removing
/Users/eusebiu.rizescu/dizertatie/flask/*"
  sudo rm -rf
```

```
/Users/eusebiu.rizescu/dizertatie/flask/*
  echo "Removing
/Users/eusebiu.rizescu/dizertatie/mysql-conf/*"
  sudo rm -rf
/Users/eusebiu.rizescu/dizertatie/mysql-conf/*

  echo "Copying to
/Users/eusebiu.rizescu/dizertatie/flask/*"
  sudo cp -R ./WebApplicationCode/*
/Users/eusebiu.rizescu/dizertatie/flask/
  echo "Copying to
/Users/eusebiu.rizescu/dizertatie/mysql-conf/*"
  sudo cp -R ./MySQL/*
/Users/eusebiu.rizescu/dizertatie/mysql-conf/

  sleep 1
  sudo docker stack deploy -c docker-compose.yml
webapplication
fi

if [ "$1" == "stop" ]; then
  # delete all containers
  echo "Delete all containers"
  sudo docker rm -f $(docker ps -a -q)
fi
```