Факултет за Информатички Науки и
Компјутерско Инженерство

Елаборат за проект

Проект: MoviesApi

Предмет: Континуирана интеграција и испорака

Ментори:
проф. д-р. Милош Јовановиќ
проф. д-р. Панче Рибарски

Изработил:
Иван Цекиќ (211146)

Скопје, 06.22.2024

Содржина

1. Вовед

MoviesApi е проектна задача изработена за предметот „Континуирана интеграција и испорака“ на Факултетот за Информатички Науки и Компјутерско Инженерство. Изработен е во тек на месецот јуни во 2024 год.

Проектот се стреми да примени концепти од полето DevOps (Development Operations) врз ASP.NET Core Web Api апликација, која користи SQL Server база на податоци. Концептите вклучуваат: верзиска контрола, контејнеризација, локална контејнерска оркестрација, CI/CD (Continuous Integration / Continuous Delivery) цевководи и контејнерска оркестрација во девелопмент околина. За концептите соодветно се применуваат следните технологии: Git/GitHub, Docker, Docker Compose, GitHub Actions и Kubernetes.

Во овој документ е даден детален опис за апликацијата, базата на податоци и употребените DevOps технологии. Изворниот код од проектот е достапен на следниот линк: https://github.com/ivancekikj/MoviesApi.

2. Апликација и база на податоци

Апликацијата MoviesApi е направена како ASP.NET Core Web Api со програмскиот јазик C#. Интегрирана развојна околина во која беше кодирана апликацијата е Microsoft Visual Studio. Изворниот код од апликацијата е достапна на следниот линк: https://github.com/ivancekikj/MoviesApi/tree/master/MoviesApi.

Доменот на апликацијата е за филмови. Ентитетите се: книги, режисери и жанрови. Една книга ги има атрбутите: Title (string), ReleaseDate (DateTime), DurationInMinutes (int), PosterUrl (string) и Description (string). Еден режисер ги има атрибутите: FirstName (string) и LastName (string). Еден жанр има атрибут Name (string). Секоја ентитет има атрибут Id (Guid) за уникатно идентификување на нови инстанци (наследено од BaseEntity). Помеѓу книга и режисер постои една many-to-many релација, исто како и помеѓу книга и жанр.

Функционалностите на апликацијата се CRUD (Create, Read, Update, Delete) операции врз атрибутите на ентитетите и релациите. Секоја функционалност е имплементирана како акција во API (Application Programming Interface) контролери. За да ги пристапиме функционалностите треба да користиме API повици. Во кодот, над секоја акција е даден URL за пристап.

За DBMS (Database Management System) се користи Microsoft SQL Server. Името на базата на податоци е MoviesApiDb. Со помош на Entity Framework Core библиотеката се врши објектно-релационо мапирање меѓу ентитетите на апликацијата и табелите на

базата на податоци. За пристапување на базата на податоци се користи LINQ (Language Integrated Query), кое се преведува во SQL прашалници. За одржување на структурата на базата на податоци се користат code-first миграции.

3. Верзиска контрола

За верзиска контрола на изворниот код се користи Git, а за централизирано хостирање на кодот се користи GitHub. Преку GitHub лесно може да се постигне интеграција со другите DevOps технологии.

4. Контејнеризација

Апликацијата, како и нејзините зависности, се контејнеризираат со помош на Docker. На следниот линк е даден Dockerfile за градење на Docker слика од апликацијата: https://github.com/ivancekikj/MoviesApi/blob/master/MoviesApi/Dockerfile. Dockerfile-от е генериран автоматски преку Visual Studio со одбирање да се додаде Docker поддршка. Потребно е градењето на сликата да се врши со контекстот поставен на root директориумот на репозиториумот.

Сликата се гради во четири фази. Во првата фаза, се поставува .NET работна околина и се алоцираат портите 8080 (HTTP) и 8081 (HTTPS). Во втората фаза, се гради проектот. Се вчитуваат зависностите од MoviesApi.csproj, се копираат сите фајлови и се компајлира целиот код. Во третата фаза, се врши публикување на компајлираната апликацијата. Во четвртата и последна фаза, се создава крајната Docker слика со сите потребни апликациски и извршувачки фајлови.

5. Локална оркестрација

Извршување на контејнеризираната апликација заедно со нејзината база е возможно преку оркестрација на двете во посебни контејнери со Docker compose. На следниот линк е даден фајлот за оркестрацијата: https://github.com/ivancekikj/MoviesApi/blob/master/docker-compose.yml. Фајлот за оркестрацијата е генериран од Visual Studio.

SQL Server се извршува во посебен контејнер и е изградено со официјалната слика на Microsoft. За да се перзистираат податоците се користи локален волумен. Податоци за комуникација со базата (Database Name, Database User и Database Password) се вчитуваат од .env фајл (мора да се наоѓа во root директориумот) и се поставуваат како environment променливи на соодвените контејнери. Името на контејнерот за базата е и име на Database Server, па се поставува како environment променлива на апликацијата.

Апликацијата ги форматира тие променливи во connection string и после тоа го користи за поврзување до базата во посебниот контејнер.

Наместо рачно извршување на оркестрацијата во терминал, Visual Studio автоматски се приспособува за извршување на оркестрацијата. Со кликање на копчето, Visual Studio ја гради сликата на апликацијата локално, ја започнува оркестрацијата и го вчитува swagger во веб прелистувач за полесно работење со API повиците.

6. CI/CD цевковод

Со помош на GitHub Actions се врши континуирана интеграција на кодот. Секој пат кога се прави push на master гранката, се извршува CI цевковод кој: ја гради сликата на апликацијата (дефинирана во Dockerfile-от), се логира на Dockerhub и ја поставува сликата на јавен регистар. Корисничкото име и токенот потребни за логирање во Dockerhub се поставени како Secrets во GitHub репозиториумот за да не се јавно достапни во кодот. Линк до GitHub акцијата: https://github.com/ivancekikj/MoviesApi/blob/master/.github/workflows/docker-image.yml.

7. Девелопмент оркестрација

Последниот и насложен дел од проектот е девелопмент оркестрацијата. За таа цел се корсити Kubernetes, кој ни нуди многу можности за склалирање, оправување и ажурирање на апликацијата. Оркестрацијата во Kubernetes е слично како во Docker compose, само што има повеќе за конфигурирање. Верзијата на Kubernetes којашто е употребена е k3d, а за управување со објектите во k8s кластерот се користи kubectl. За целите на проектот се создадени 5 манифести: namespace.yaml, deployment.yaml, service.yaml, ingress.yaml и storage.yaml. Сите фајлови се достапни на следниот линк: https://github.com/ivancekikj/MoviesApi/tree/master/Kubernetes.

Во namespace.yaml е дефиниран именскиот простор main. Во него ќе бидат додадени објектите од сите други манифести.

Во deployment.yaml е даден deployment објект за апликацијата, како и configmap и secrets објекти. Во configmap-ата се дефинирани Database Server, Database Name и Database User како key-value парови (исто како во docker-compose.yml). Во Secrets е дефиниран key-value пар Database Password и е енкриптиран во base64. Овие key-value парови се инјектираат соодветно во deployment-от како environment променливи. Deployment-от moviesapi ја користи Docker сликата создадена од GitHub акцијата, создава 3 реплика множества (подови), се поврзува со портата 8080 од докер сликата и користи соодветни селектори и лабели за да биде откриен од неговиот сервис.

Сервисот moviesapi-service во service.yaml доделува постојани мрежни својства (IP адреса, DNS, порти) на подовите кои ќе бидат создадени од moviesapi deployment-от. Со соодветните лабели и селектори ги идентификува подовите и управува со точно тие. Типот на сервисот е LoadBalancer. Се поврзува со портата 8000 на секој под и ја користи портата 8081 на кластерот за барања кои додаѓаат надвор од кластерот.

Во ingress.yaml има ingress контролер кој слуша на патеката "/" на порта 8081 на кластерот и ги пренесува барањата кон сервисот moviesapi-service.

Во storage.yaml се дефинирани statefulset, configmap, secrets и service објекти. Statefulset-от се користи за контејнеризираната база. Слично како во docker compose, се користи официјалната слика на Microsoft и се поставува volume за перзистирање на податоци (овој пат со повеќе можности за ограничувања). Се инјектираат key-value парови за базата како environment променливи од configmap-а и secrets објект (дефинирани во истиот фајл). Statefulset-от е откриен од сервисот (дефиниран во истиот фајл) на порта 1433, користејќи соодветни селектори и лабели. Базата може да се пристапи само во рамки на кластерот.

За да се постават базата и апликацијата во k8s кластер, преку unix/wsl terminal, треба да се навигира во Kubernetes папката (кадешто се наоѓаат манифестите), и да се извршат следните наредби:

1. k3d cluster create moviesapi -p "6969:8081@loadbalancer"
2. kubectl apply -f namespace.yaml -f deployment.yaml -f service.yaml -f ingress.yaml -f storage.yaml

Бидејќи swagger не е достапен надвор од Visual Studio, најдобро е да се тестираат API повиците со Postman.