



Realtime Chat - Continuous Integration/Continuous Delivery

Берат Ахметаж - 216130

CI/CD PIPELINE

Code Development



Dockerfile



K8 Manifests

Continuous Integration



Gradle build



Github Actions



Spring Boot

Continuous Deployment



Orchestration



Azure K8



Minikube

Фази на проектот:

Ф1: јавен git репозиториум	2
Ф2: Докеризација	3
Ф3: Docker-compose Оркестрација	5
Ф4: Pipeline CI/CD	7
Ф5: Deployment за апликацијата со потребните ConfigMaps/Secrets	8
Ф6: Service за апликацијата	10

Materials:

[Github Repository](#)

[DockerHub Repository](#)



Ф1: јавен git репозиториум

Цел на проектот:

Во оваа секција ќе објасниме како ја користиме апликацијата за систем за чат користејќи WebSocket, Spring Boot, Java како backend, PostgreSQL од Azure како база на податоци и Thymeleaf како фронтенд шаблонски мотор, и како сето ова е поставено во јавен репозиториум на GitHub.

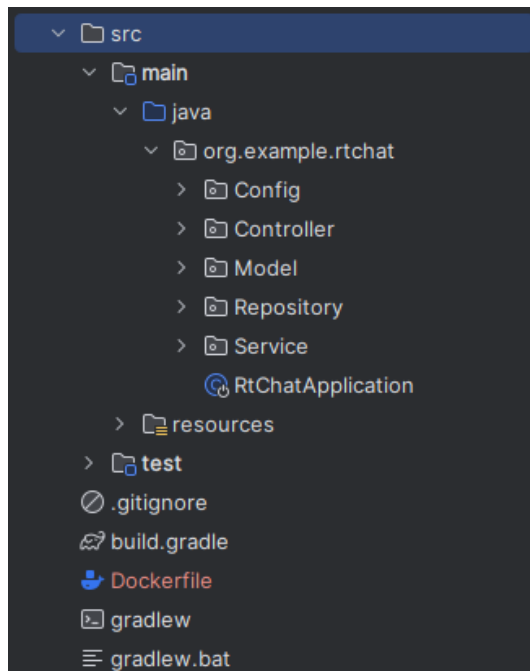
Иницијализација на локалниот Git репозиториум

```
git init  
git remote add origin https://github.com/BeratAhmetaj/RT-CHAT-KIII.git
```

Додавање на апликацијата во репозиториумот

```
git commit -m "Initial commit of chat application"  
git push -u origin master
```

Структура на апликацијата





Ф2: Докеризација

Пред да се направи докеризација, потребно е да добиеме .Jar фајл од секој Build, бидејќи користам Gradle потребно е да се додаваат овие линии во gradle.build

```
jar {  
    enabled = true  
}  
  
springBoot {  
    mainClass.set('org.example.rtchat.RtChatApplication') //Main App Class  
}
```

Build gradle + JAR command in cmd (windows)

```
./gradlew bootJar
```

Result:

```
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT> ./gradlew bootJar  
Starting a Gradle Daemon, 2 incompatible Daemons could not be reused, use --status for details  
  
BUILD SUCCESSFUL in 12s  
4 actionable tasks: 2 executed, 2 up-to-date  
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT> |
```

DOCKERFILE

```
FROM gradle:7.6.0-jdk17 AS build  
  
WORKDIR /app  
  
COPY . .  
  
RUN ./gradlew bootJar  
  
FROM openjdk:17-jdk-slim  
  
WORKDIR /app  
  
COPY --from=build /app/build/libs/RT-CHAT-0.0.1-SNAPSHOT.jar  
/app/RT-CHAT-0.0.1-SNAPSHOT.jar  
  
EXPOSE 8080  
  
CMD ["java", "-jar", "/app/RT-CHAT-0.0.1-SNAPSHOT.jar"]
```

Gradle Build команда за да земе Jar



Java SDK 17 kako base image

Docker Build:

```
docker build -t rt-chat-app .
```

Result:

```
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT> docker build -t rt-chat-app .
[+] Building 0.0s (0/0) docker:default
2024/06/24 19:55:32 http2: server: error reading preface from client //./pipe/[+] Building 43.3s (9/9) FINISHED
:default
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 441B                                              0.0s
=> [internal] load metadata for docker.io/library/openjdk:17-jdk-slim           2.3s
=> [auth] library/openjdk:pull token for registry-1.docker.io                  0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                    0.0s
=> [1/3] FROM docker.io/library/openjdk:17-jdk-slim@sha256:aaa3b3cb27          39.2s
=> => resolve docker.io/library/openjdk:17-jdk-slim@sha256:aaa3b3cb27e         0.0s
=> => sha256:44d3aa8d076675d49d85180b0ced9daef210fe4fd 1.58MB / 1.58MB        0.5s
=> => sha256:6ce99fdf16e86bd02f6ad66a0e1334878528 187.90MB / 187.90MB        36.4s
=> => sha256:aaa3b3cb27e3e520b8f116863d0580c438ed55ecfa0bc 547B / 547B       0.0s
=> => sha256:779635c0c3d23cc8dbab2d8c1ee4cf2a9202e198dfc8f 953B / 953B      0.0s
=> => sha256:37cb44321d0423bc57266a3bfff658daf00478e4cd 4.80kB / 4.80kB      0.0s
=> => sha256:1fe172e4850f03bb45d41a20174112bc119fbf 31.38MB / 31.38MB      12.2s
=> => extracting sha256:1fe172e4850f03bb45d41a20174112bc119fbfec42a650     2.3s
=> => extracting sha256:44d3aa8d076675d49d85180b0ced9daef210fe4fdff4bd      0.2s
=> => extracting sha256:6ce99fdf16e86bd02f6ad66a0e1334878528b5a4b54878     2.6s
=> [internal] load build context                                                2.1s
=> => transferring context: 50.39MB                                             2.1s
=> [2/3] WORKDIR /app                                                           1.1s
=> [3/3] COPY build/libs/RT-CHAT-0.0.1-SNAPSHOT.jar /app/RT-CHAT-0.0.1        0.2s
=> exporting to image                                                           0.3s
=> => exporting layers                                                         0.2s
=> => writing image sha256:2471bc43dd1a87d505b9b1be13ef3f1f6fead084d8d      0.0s
=> => naming to docker.io/library/rt-chat-app                                0.0s

View build details: docker-desktop://dashboard/build/default/default/tt41nbcfnzr3ln9k4ha7fbud

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT> |
```

Docker Run

```
docker run -p 8080:8080 rt-chat-app
```

Result:



Docker-compose.yml

```
version: '3.8'
```



```
- db-data:/var/opt/mssql
```

```
volumes:  
  db-data:
```

Користам .env фајл за на безбеден начин да пренесам информации за passwords и usernames во Dockercompose

.env е ставено во .gitignore

Run docker-compose and build the image

```
docker-compose up --build
```

Result:

The screenshot shows two windows. The left window is Docker Desktop, displaying a list of containers. The right window is a web browser showing a chat interface with messages from users @Berat and @Besam.

Name	Image	Status	CPU (%)	Port(s)	Actions
minikube	gcr.io/k8s-	Exited (130)	0%	0:22	Show all ports
sweet_jemison	rl-chat-app	Exited (130)	0%	8080:8080	
rl-chat		Running (2/2)	1.93%		

Additional commands:

```
docker-compose logs -f
```



docker-compose down

Ф4: Pipeline CI/CD

Github Action:

започнува со билд на Јава апликацијата на секој push на гранката **main**. Потоа се креира Docker слика од апликацијата и се врши нејзино push на DockerHub. Во последен чекор, се користи **kubect1** апликацијата на локалното Minikube Kubernetes, користејќи манифести за Deployment и Service. Ова е корисно за локално тестирање и развој на Kubernetes апликации во Minikube.

ci-cd-pipeline.yml

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Set up JDK 17
        uses: actions/setup-java@v2
        with:
          java-version: 17

      - name: Cache Gradle packages
        uses: actions/cache@v2

      - name: Log in DockerHub
        uses: docker/login-action@v1
        with:
          username: ${ secrets.DOCKER_USERNAME }
          password: ${ secrets.DOCKER_PASSWORD }

      - name: Build and push Docker image
        uses: docker/build-push-action@v2
        with:
          context: .
          push: true
          tags: ${ secrets.DOCKER_USERNAME }/rt-chat:latest

  deploy:
    runs-on: ubuntu-latest
    needs: build
    steps:
      - name: Install kubect1
        run: |
          curl -LO
          https://storage.googleapis.com/kubernetes-release/release/$(curl -s
          https://storage.googleapis.com/kubernetes-release/release/stable.tx
```



```
with:
  path: |
    ~/.gradle/caches
    ~/.gradle/wrapper
  key: ${ runner.os }}-gradle-${ hashFiles('**/*.gradle*',
'*/gradle-wrapper.properties') }}
  restore-keys: ${ runner.os }}-gradle

- name: Gradle Build
  run: ./gradlew bootJar

- name: Docker Buildx
  uses: docker/setup-buildx-action@v1
```

```
t)/bin/linux/amd64/kubect1
  chmod +x ./kubect1
  sudo mv ./kubect1 /usr/local/bin/kubect1

- name: Configure kubect1 to use Minikube
  run: minikube kubect1 -- get po -A

- name: Deploy to Minikube
  run: |
    kubect1 apply -f k8s/deployment.yaml -f k8s/service.yaml
```

Github Secrets:

Со цел pipeline-от безбедно да се завршува, користам Github Action Secrets, и .env за локално кога старувам оркестрација

```
SPRING_DATASOURCE_URL: ${SPRING_DATASOURCE_URL}
SPRING_DATASOURCE_USERNAME: beratahmetaj02@berat-finki
SPRING_DATASOURCE_PASSWORD: ${SPRING_DATASOURCE_PASSWORD}
```

Repository secrets

[New repository secret](#)

Name	Last updated	
DB_SA_PASSWORD	2 minutes ago	
SPRING_DATASOURCE_PASSWORD	3 minutes ago	
SPRING_DATASOURCE_URL	3 minutes ago	

Ф5: Deployment за апликацијата со потребните ConfigMaps/Secrets

Namespace

Првин треба да се креира Namespace

```
kubect1 create namespace rt-chat-local
```




Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
  namespace: rt-chat-local # NAMESPACE
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: rt-chat-application:latest # DOCKER IMAGE NAME
          ports:
            - containerPort: 8080
          env:
            - name: DB_URL
              value:
jdbc:sqlserver://berat-finki.database.windows.net:1433;databaseName=RT-CHAT;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;
            - name: SPRING_DATASOURCE_USERNAME
              value: beratahmetaj02@berat-finki
            - name: SPRING_DATASOURCE_PASSWORD # USE OF SECRETS
              valueFrom:
                secretKeyRef:
                  name: db-secrets
                  key: db-sa-password
            - name: SPRING_DATASOURCE_DRIVER_CLASS_NAME
              value: com.microsoft.sqlserver.jdbc.SQLServerDriver
            - name: SPRING_JPA_DATABASE_PLATFORM
              value: org.hibernate.dialect.SQLServerDialect
            - name: SPRING_JPA_HIBERNATE_DDL_AUTO
              value: update
          envFrom:
            - secretRef:
                name: my-app-secrets # GitHub secret name
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: rt-chat-local # Use your specific namespace here
data:
  app.properties: |
    app.name=RT-CHAT-APP
    app.version=1.0.0
```

Github Action Pipeline модификација (користејќи манифести)

```
deploy:
```



```
runs-on: ubuntu-latest
needs: build
steps:
  - name: Install kubectl
    run: |
      curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
      chmod +x ./kubectl
      sudo mv ./kubectl /usr/local/bin/kubectl

  - name: Configure kubectl to use Minikube
    run: minikube kubectl -- get po -A

  - name: Deploy to Minikube
    run: |
      kubectl apply -f k8s/deployment.yaml -f k8s/service.yaml
```

Ф6: Service за апликацијата

При правење на Service.yaml пажев на истите Labels да се користат како што се на Deployment, и точниот Namespace да се користи

Service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
  namespace: rt-chat-local # NAMESPACE SPECIFIED
spec:
  selector:
    app: my-app # label match from deployment.yaml
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer # nodeport za minikube
```

Apply Manifests:

```
kubectl apply -f deployment.yaml --validate=false
kubectl apply -f service.yaml --validate=false
```



Result:

```
★ Enabled addons: default-storageclass, storage-provisioner
🔔 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT\k8s> kubectl apply -f deployment.yaml --validate=false
deployment.apps/my-app-deployment created
configmap/app-config created
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT\k8s> kubectl apply -f service.yaml --validate=false
service/my-app-service created
PS C:\Users\berat\OneDrive\Documents\Github\RT-Chat-KIII\RT-CHAT\k8s> |
```

Dockerhub push

```
docker push berat Ahmetaj/rt-chat-application:latest
```

Run the docker image

```
docker run -p 8080:8080 berat Ahmetaj/rt-chat-application:latest
```

Get pods + Get Service

```
kubectl get pods -n rt-chat-cloud
kubectl get services -n rt-chat-cloud
```

Result:

▼	✓	Get Pods	0s
1	▶	Run kubectl get pods -n rt-chat-cloud	
4			
5		NAME	READY STATUS RESTARTS AGE
6		rt-chat-deployment-54f7bdc459-7mckn	0/1 ContainerCreating 0 2s
7		rt-chat-deployment-54f7bdc459-cssb9	0/1 ContainerCreating 0 2s
8		rt-chat-deployment-54f7bdc459-t14sg	0/1 ContainerCreating 0 2s
▼	✓	Get Services	0s
1	▶	Run kubectl get services -n rt-chat-cloud	
4			
5		NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE	
6		rt-chat-service LoadBalancer 10.0.188.62 <pending> 80:32688/TCP 1s	

Update ci-cd-pipeline.yml 6 minutes ago 1m 51s ...

CI/CD Pipeline #37: Commit 7926d1f pushed by BeratAhmetaj master

RTCHATCluster Kubernetes service

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Cost analysis (preview)

Kubernetes resources

Namespaces

Workloads

Services and ingresses

Storage

Configuration

Custom resources

Events

Run command

Essentials

Resource group: link

Status: Succeeded (Running)

Subscription: Azure for Students

Location: North Europe

Subscription ID: 70bce514-ae53-4dce-aef4-ecb964bffe

Tags (edit): Add tags

Get started

Properties

Monitoring

Capabilities (5)

Recommendations (3)

Tutorials

Kubernetes services

Encryption type: Encryption at rest with a platform-managed key

Virtual node pools: Not enabled

Node pools

Node pools: 1 node pool

Kubernetes versions: 1.28.9

Node sizes: Standard_DS2_v2

Networking

API server address: rtchatcluster-dns-rtjneyq7.hcp.northeurope.azuremk8.io

Network configuration: Azure CNI

Pod CIDR: -

Service CIDR: 10.0.0/16

DNS service IP: 10.0.0.10

Docker bridge CIDR: -

Network Policy: None