



Универзитет „Св. Кирил и Методиј“ во Скопје  
**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И  
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

## **КОНТИНУИРАНА ИНТЕГРАЦИЈА И ИСПОРАКА**

*проект на тема*

### **ДОКЕРИЗАЦИЈА И ОРКЕСТРАЦИЈА НА SPRING BOOT АПЛИКАЦИЈА СО CI/CD И KUBERNETES**

**Изработил: Бојан Ристов (211151)**

**Ментори: Проф. д-р Милош Јовановиќ и Проф. д-р Панче Рибарски**

**Линк до GitHub repository: [https://github.com/ristov663/Project\\_KIII](https://github.com/ristov663/Project_KIII)**

**Линк до Docker hub: <https://hub.docker.com/repositories/ristov211151>**

**13.02.2025, Скопје**

# Вовед

Во овој елаборат накратко е опишан целиот процес на развој и автоматизација на деплојмент на едноставна **Spring Boot апликација со PostgreSQL база на податоци**, користејќи DevOps методологии и алатки кои беа дел од предметот Континуирана интеграција и испорака.

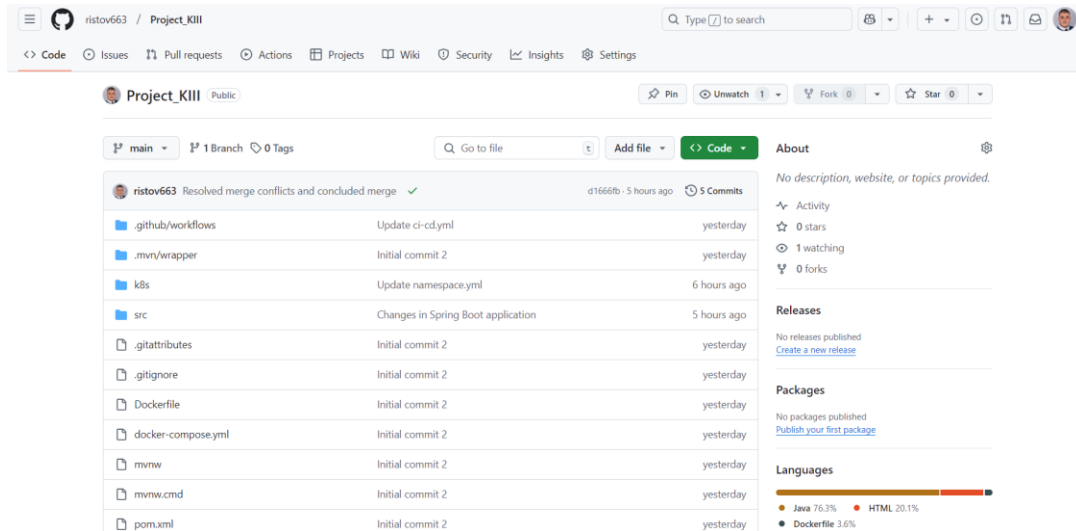
Главната цел на проектот беше да се постигне целосно автоматизиран процес на континуирана интеграција и испорака (CI/CD), како и безбедна и скалабилна инфраструктура за хостирање на апликацијата.

Оваа апликација, базирана на **Spring Boot** и **PostgreSQL**, претставува е само еден пример за тоа како една веб апликација може да биде подготвена за продукциско работење преку имплементација на DevOps практиките.

Овој проект опфаќа повеќе клучни компоненти:

- **Контејнеризација со Docker** – обезбедување на конзистентна извршна средина за апликацијата.
- **Оркестрација со Kubernetes** – автоматско управување со подигнување, скалирање и одржување на апликацијата.
- **Автоматизирано градење и деплојмент преку CI/CD pipeline** – осигурување дека секоја нова верзија на апликацијата се тестира и автоматски се испорачува во продукциско окружување.
- **Конфигурација и менаџмент на околините** – дефинирање на параметри и безбедно управување со тајни и крденцијали.

## Поставување на кодот на јавен Git репозиториум



```
git init
git remote add origin
https://github.com/ristov663/Project_KIII.git
git add .
git commit -m "Initial commit"
git push -u origin main
```

## Докеризација на апликацијата

Dockerfile-то изгледа вака:

```
Dockerfile x
1 FROM eclipse-temurin:18-jdk AS builder
2 WORKDIR /app
3 COPY . .
4 RUN chmod +x mvnw
5 RUN ./mvnw clean package -DskipTests
6 FROM eclipse-temurin:18-jdk
7 WORKDIR /app
8 COPY --from=builder /app/target/app-0.0.1-SNAPSHOT.jar /app/app.jar
9 EXPOSE 8080
10 CMD ["java", "-jar", "/app/app.jar"]
```

```
docker build -t ristov211151/app:latest .
docker push ristov211151/app:latest
```

## Оркестрација со Docker compose

За полесно управување со апликацијата и базата, користевме **Docker Compose**. Конфигурацијата во `docker-compose.yml` овозможува подигнување на апликацијата заедно со PostgreSQL база на податоци.

```
docker-compose.yml
1  version: '3.8'
2  services:
3    app:
4      build: .
5      ports:
6        - "8080:8080"
7      environment:
8        SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/kii
9        SPRING_DATASOURCE_USERNAME: postgres
10       SPRING_DATASOURCE_PASSWORD: password
11      depends_on:
12        - db
13
14    db:
15      image: postgres:15
16      environment:
17        POSTGRES_DB: kii
18        POSTGRES_USER: postgres
19        POSTGRES_PASSWORD: password
20      ports:
21        - "5432:5432"
```

**`docker-compose up -d`**

## CI/CD Pipeline со GitHub Actions

Целта на CI/CD е автоматизација на градењето и деплојментот. Фајлот `.github/workflows/ci-cd.yml` содржи автоматизирано градење и push на Docker image, како и деплојмент на Kubernetes.

The screenshot shows the GitHub Actions interface for a repository named 'Project\_Kill' by user 'ristov663'. The 'Actions' tab is selected, displaying a list of workflow runs. The 'CI/CD Pipeline' workflow is highlighted in the left sidebar. The main area shows three workflow runs, all with a green status indicating success. The first run is 'Resolved merge conflicts and concluded merge' (CI/CD Pipeline #3) and the second is 'Update namespace.yml' (CI/CD Pipeline #2). Both runs were triggered by a push to the 'main' branch by the user 'ristov663'.

Event	Status	Branch	Actor	Time
Resolved merge conflicts and concluded merge	Success	main	ristov663	5 hours ago
Update namespace.yml	Success	main	ristov663	6 hours ago

ristov663 / Project\_Kill

Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

CI/CD Pipeline

✓ Resolved merge conflicts and concluded merge #3 Re-run all jobs

**Summary**

Jobs

- ✓ build

Run details

- Usage
- Workflow file

Triggered via push 5 hours ago	Status	Total duration	Artifacts
ristov663 pushed · d1666fb main	Success	37s	—

**ci-cd.yml**  
on: push

✓ build 29s

```
ci-cd.yml x
1  name: CI/CD Pipeline
2
3  on:
4    push:
5      branches:
6        - main
7
8  jobs:
9    build:
10     runs-on: ubuntu-latest
11
12     steps:
13       - name: Checkout code
14         uses: actions/checkout@v3
15
16       - name: Set up JDK 18
17         uses: actions/setup-java@v3
18         with:
19           distribution: 'temurin'
20           java-version: '18'
21
22       - name: Log in to Docker Hub
23         uses: docker/login-action@v2
24         with:
25           username: ${ secrets.DOCKER_USERNAME }
26           password: ${ secrets.DOCKER_PASSWORD }
27
28       - name: Build Docker image
29         run: docker build -t ristov211151/app:latest .
30
31       - name: Push Docker image to Docker Hub
32         run: docker push ristov211151/app:latest
```

## Оркестрација со Kubernetes

За деплојмент на Kubernetes, креирав YAML манифести за **Deployment**, **Service** и **Ingress**, кои овозможуваат автоматско подигнување и достапност на апликацијата.

```
app-deployment.yml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: app
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: myapp
10   template:
11     metadata:
12       labels:
13         app: myapp
14     spec:
15       containers:
16         - name: myapp
17           image: ristov211151/app:latest
18           env:
19             - name: SPRING_DATASOURCE_URL
20               value: "jdbc:postgresql://db-statefulset.myapp-namespace.svc.cluster.local:5432/kii"
21             - name: SPRING_DATASOURCE_USERNAME
22               value: "postgres"
23             - name: SPRING_DATASOURCE_PASSWORD
24               value: "password"
25       ports:
26         - containerPort: 8080
27
```

```
app-ingress.yml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: app-ingress
5    namespace: myapp-namespace
6    annotations:
7      nginx.ingress.kubernetes.io/rewrite-target: /
8  spec:
9    rules:
10     - host: localhost.nip.io
11       http:
12         paths:
13           - path: /
14             pathType: Prefix
15             backend:
16               service:
17                 name: app-service
18                 port:
19                   number: 8080
20
```

db-statefulset.yml x

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: db-statefulset
5    namespace: myapp-namespace
6  spec:
7    selector:
8      app: db-statefulset
9    ports:
10     - protocol: TCP
11       port: 5432
12       targetPort: 5432
13     clusterIP: None
14
15  ---
16  apiVersion: apps/v1
17  kind: StatefulSet
18  metadata:
19    name: db-statefulset
20    namespace: myapp-namespace
21  spec:
22    serviceName: db-statefulset
23    replicas: 1
24    selector:
25      matchLabels:
26        app: db-statefulset
27    template:
28      metadata:
29        labels:
30          app: db-statefulset
31      spec:
32        containers:
33         - name: postgres
34           image: postgres:15
35           ports:
36             - containerPort: 5432
37           env:
38             - name: POSTGRES_USER
39               value: postgres
40             - name: POSTGRES_PASSWORD
41               value: password
42             - name: POSTGRES_DB
43               value: kii
44           volumeMounts:
45             - name: postgres-data
46               mountPath: /var/lib/postgresql/data
47        volumeClaimTemplates:
48         - metadata:
49             name: postgres-data
50         spec:
51             accessModes: ["ReadWriteOnce"]
52             resources:
53               requests:
54                 storage: 16Gi
```

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: app-service
5    namespace: myapp-namespace
6  spec:
7    selector:
8      app: app
9    ports:
10     - protocol: TCP
11       port: 80
12       targetPort: 8080
13    type: ClusterIP
14

```

```

kubectl create namespace myapp-namespace
kubectl apply -f db-statefulset.yml -n myapp-namespace
kubectl apply -f app-deployment.yml -n myapp-namespace
kubectl apply -f app-service.yml -n myapp-namespace
kubectl apply -f app-ingress.yml -n myapp-namespace
kubectl port-forward pod/app-5fc646cbb6-6d771 8080:8080 -n
myapp-namespace

```

## Docker desktop и Docker Hub

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and includes a search bar and a toggle for 'Only show running containers'. Below this is a table of running containers.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
k8s_myapp_app	6ba51332894a	ristov2111!		0.16%	5 hours ago	[Stop] [Restart] [Delete]
k8s_myapp_app	79f9b524cd6b	ristov2111!		0.18%	5 hours ago	[Stop] [Restart] [Delete]
k8s_postgres_d	a71e470ae58f	d609c3005		0%	5 hours ago	[Stop] [Restart] [Delete]
k8s_controller_i	7f346c63c490	e6b8de175		0.12%	5 hours ago	[Stop] [Restart] [Delete]
project_kitl	-	-	-	0%	24 hours ago	[Stop] [Restart] [Delete]

At the top of the main area, there are statistics: Container CPU usage is 0.46% / 800% (8 CPUs available) and Container memory usage is 745.24MB / 3.66GB. A 'Show charts' link is also present.




ristov211151 / [Repositories](#) / [app](#) / [Tags](#) / latest



## ristov211151/app:latest

MANIFEST DIGEST sha256:772606a96299ba9dfaed16fb7cd8913e95b85b983ed0ad798dc29171392fa601 

OS/ARCH	COMPRESSED SIZE	LAST PUSHED	TYPE	MANIFEST DIGEST
linux/amd64	277.55 MB	6 hours ago by <a href="#">ristov211151</a>	Image	sha256:772606a9... 

```
Command Prompt - kubectl | X + v
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Angela>cd Project_KIII

C:\Users\Angela\Project_KIII>cd k8s

C:\Users\Angela\Project_KIII\k8s>kubectl get pods -n myapp-namespace
NAME                READY   STATUS    RESTARTS   AGE
app-d47f57546-74mfg  1/1     Running   1 (5h20m ago)    23h
app-d47f57546-8q8xd  1/1     Running   1 (5h20m ago)    23h
db-statefulset-0     1/1     Running   1 (5h20m ago)    23h

C:\Users\Angela\Project_KIII\k8s>kubectl get services -n myapp-namespace
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
app-service         ClusterIP   10.101.174.156 <none>        80/TCP     23h
db-statefulset      ClusterIP   None         <none>        5432/TCP   23h

C:\Users\Angela\Project_KIII\k8s>kubectl get deployments -n myapp-namespace
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
app    2/2     2            2           23h

C:\Users\Angela\Project_KIII\k8s>kubectl get ingress -n myapp-namespace
NAME        CLASS    HOSTS          ADDRESS    PORTS    AGE
app-ingress <none>   localhost.nip.io 80         23h

C:\Users\Angela\Project_KIII\k8s>kubectl port-forward pod/app-d47f57546-74mfg 8080:8080 -n myapp-namespace
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

Categories and Events

Browse through categories and the events associated with them.

Categories

category0

Events:

event0-0

category0

event0-1

category0

event0-2

category0

category1

Events:

event1-0

category1

event1-1

category1

event1-2

category1

category2

Events:

event2-0

category2

event2-1

category2

event2-2

category2

category3

Events:

event3-0

category3

event3-1

category3

event3-2

category3

category4

Events:

event4-0

category4

event4-1

category4

event4-2

category4

category0

Events:

event0-0

category0

event0-1

category0

event0-2

category0

## Заклучок

Овој проект ја демонстрира моќта на DevOps методологијата во автоматизирање и оптимизирање на процесот на испорака на софтвер. Со успешната имплементација на Docker, Kubernetes и CI/CD, постигнавме:

- **Брз и сигурен деплојмент** на нови верзии без рачно интервенирање.
- **Лесно менаџирање** и скалирање на апликацијата.
- **Автоматско откривање на грешки** и стабилен развојен циклус.

Овој пристап овозможува стабилна, сигурна и скалабилна инфраструктура, подготвена за реална употреба во продукциска околина.

## Користени ресурси

1. **Материјали од курсот по Континуирана интеграција и испорака**
2. **Spring Boot** (Развој на апликацијата) - <https://spring.io/projects/spring-boot>
3. **PostgreSQL** (Релациона база на податоци) - <https://www.postgresql.org>
4. **Docker** - <https://www.docker.com>
5. **Docker Compose** (Оркестрација на апликацијата и базата на податоци) - <https://docs.docker.com/compose/>
6. **Docker Desktop** - <https://www.docker.com/products/docker-desktop/>
7. **GitHub Actions** (CI/CD платформа за автоматизација на build, testing и deployment) - <https://github.com/features/actions>
8. **Kubernetes** (Оркестрација на апликацијата во кластер) - <https://kubernetes.io>