"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Seminar work for the subject Continuous Integration and Delivery:

Dockerizing and Deploying a Spring Boot application

Professor: Milosh Jovanovikj

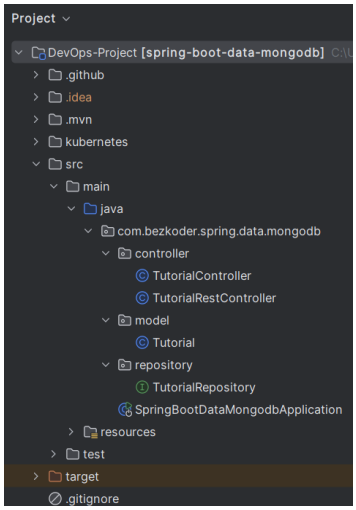
Student: Bojana Arizankovska 211018

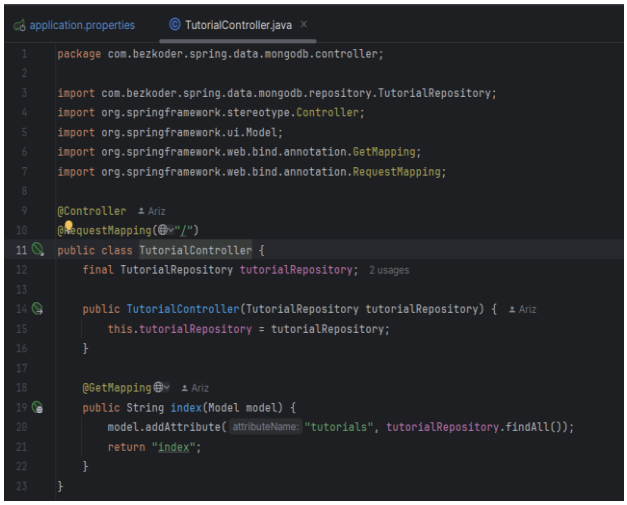
September 2024

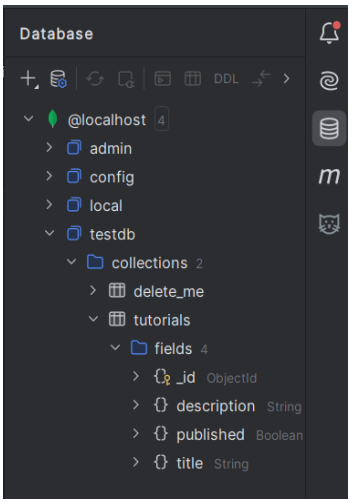
Intro to the application

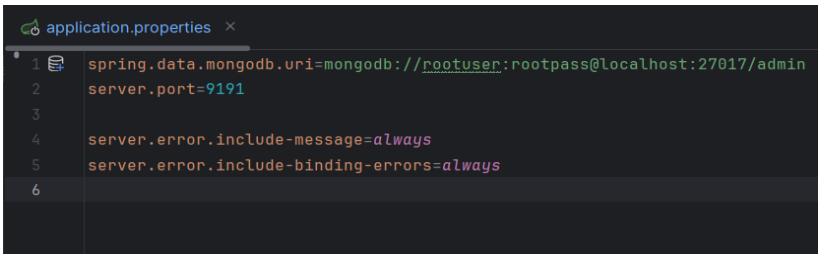
This Spring Boot Application is quite simple – it has one model class Tutorial, one repository TutorialRepository that extends MongoRepository<Tutorial, String> and two controllers – one normal and one RestController.

The purpose of this app is to keep created Tutorials and display them. Nothing complex.



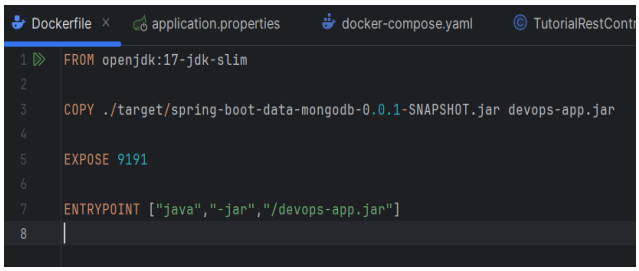






Dockerizing the application

The first step is to write a Dockerfile in order to create a Docker Image for the application. In summary, this Dockerfile sets up a minimal environment to run the Spring Boot application packaged as a JAR file, exposing the necessary port 9191 for external access – shown later when setting up Kubernetes.



The Dockerfile and how it is built.

Orchestrating the application and database

This Docker Compose setup creates a complete environment with MongoDB, a web-based admin interface (Mongo Express), and a Spring Boot application. The services are interconnected, with the Spring Boot app depending on MongoDB, and the configuration ensures that data persists even if the containers are stopped and restarted.

application.properties | TutorialController.java | Tutorial.java | docker-compose.yaml

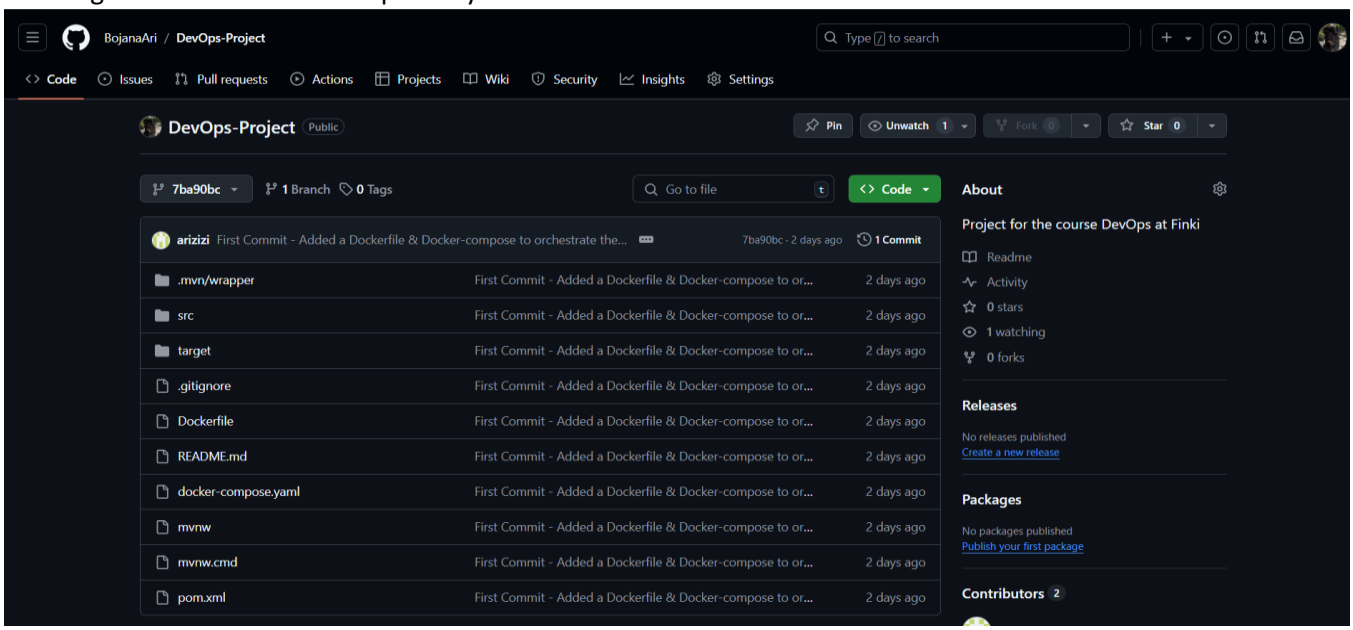
```yaml
version: "3.8"
services:
  mongodb:
    image: mongo
    container_name: mongodb
    ports:
      - 27017:27017
    volumes:
      - data:/data
    environment:
      - MONGO_INITDB_ROOT_USERNAME=rootuser
      - MONGO_INITDB_ROOT_PASSWORD=rootpass
  mongo-express:
    image: mongo-express
    container_name: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=rootuser
      - ME_CONFIG_MONGODB_ADMINPASSWORD=rootpass
      - ME_CONFIG_MONGODB_SERVER=mongodb
  spring-boot-app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: springboot-app
    ports:
      - 9191:9191
    environment:
      - SPRING_DATA_MONGODB_URI=mongodb://rootuser:rootpass@mongodb:27017/admin
    depends_on:
      - mongodb
volumes:
  data: {}

networks:
  default:
    name: mongodb_network
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project> docker-compose up --build
[+] Building 0.0s (0/0)  docker:default
[+] Building 1.1s (7/7) FINISHED                                                      docker:default
 => [spring-boot-app internal] load build definition from Dockerfile                            0.0s
 => => transferring dockerfile: 197B                                                             0.0s
 => [spring-boot-app internal] load metadata for docker.io/library/openjdk:17-jdk-slim          1.0s
 => [spring-boot-app internal] load .dockerignore                                                0.0s
 => => transferring context: 2B                                                                  0.0s
 => [spring-boot-app 1/2] FROM docker.io/library/openjdk:17-jdk-slim@sha256:aaa3b3cb27e3e520b8f116863d0580c438ed55ecfa0bc126b41f68c3f62f9774  0.0s
 => [spring-boot-app internal] load build context                                                0.0s
 => => transferring context: 100B                                                                0.0s
 => CACHED [spring-boot-app 2/2] COPY ./target/spring-boot-data-mongodb-0.0.1-SNAPSHOT.jar devops-app.jar  0.0s
 => [spring-boot-app] exporting to image                                                         0.0s
 => => exporting layers                                                                          0.0s
 => => writing image sha256:4cbdf055e061d77234403d995dc02aa5abc4c0cc7ad7625c34d2860ce281e01f     0.0s
 => => naming to docker.io/library/devops-project-spring-boot-app                                0.0s
[+] Running 3/3
 ✓ Container mongodb          Created                                                            0.0s
 ✓ Container mongo-express    Created                                                            0.0s
 ✓ Container springboot-app   Recreated                                                          0.2s
Attaching to mongo-express, mongodb, springboot-app
mongo-express   | Waiting for mongo:27017...
mongodb         | {"t":{"$date":"2024-09-21T13:59:09.761+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.
```

Creating a Github repository

Pushing all the files to a new repository.

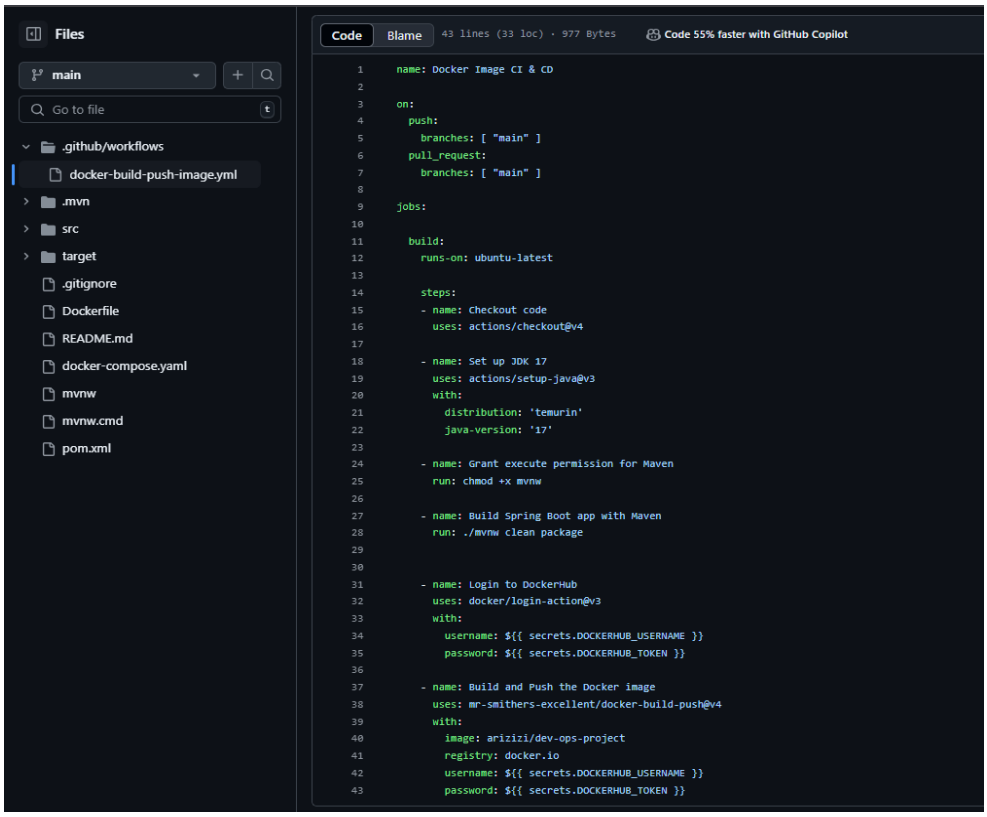


CI pipeline using GitHub Actions

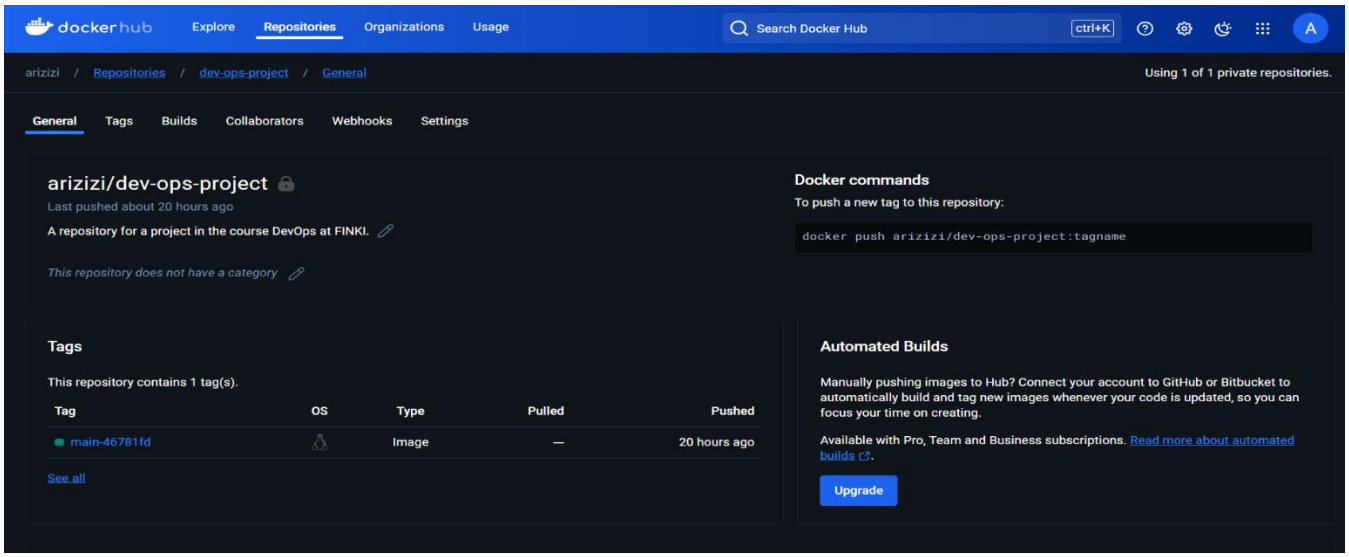
This GitHub Actions CI pipeline automates the process of building my Spring Boot application and pushing the Docker image to Docker Hub , when new commits are pushed or pull requests are opened on the main branch.

So the following workflow was created and it:

1) Checks out the repository code,
2) Sets up Java 17
3) Grants permissions to Maven, so it can build the Spring Boot application.
4) After the build, it logs into Docker Hub and builds a Docker image of the application.
5) The Docker image is then pushed to Docker Hub under the specified name (arizizi/dev-ops-project).



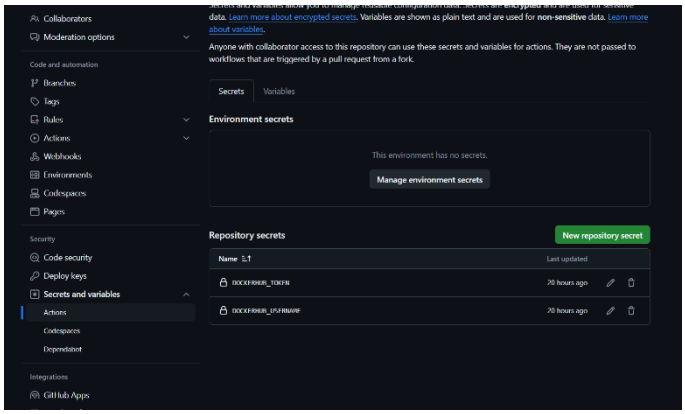
Created an account to DockerHub & Repository where the image will be pushed in .

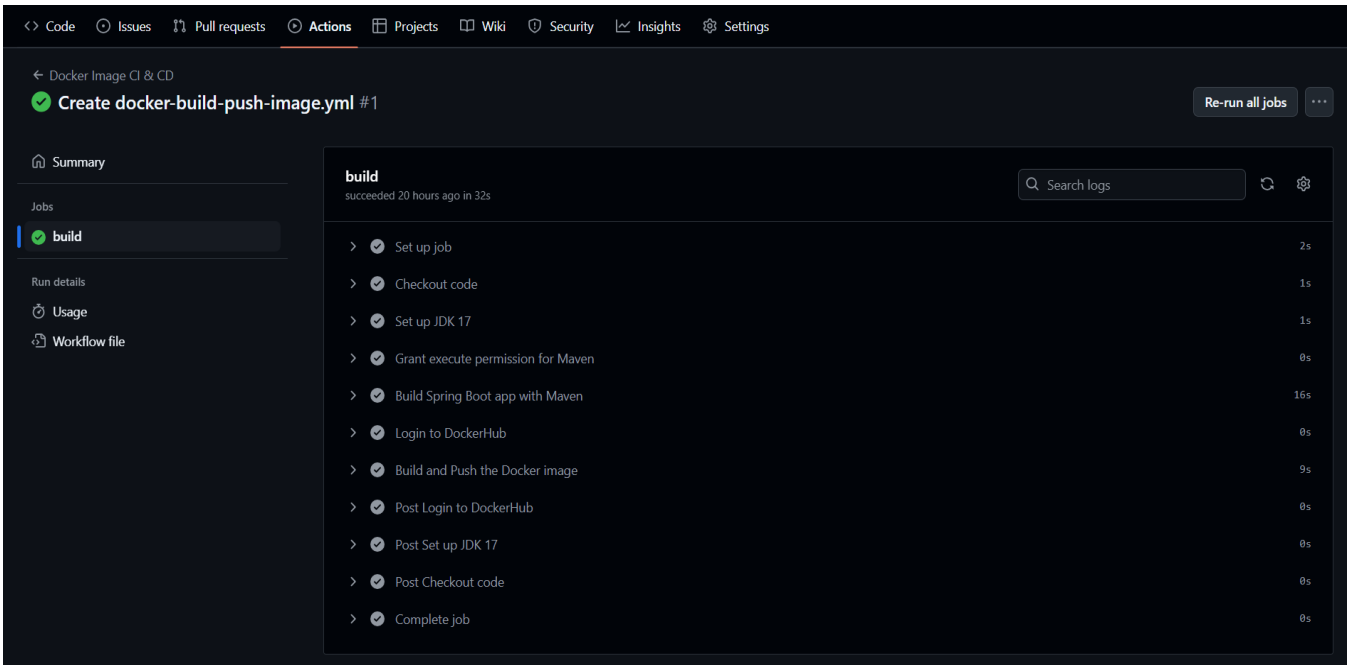


In order for the workflow to log into Docker Hub properly, secrets had to be created in the Github repository:

- DOCKERHUB_USERNAME -> my username arizizi
- DOCKERHUB_TOKEN -> personal access token created from DockerHub

The workflow will do the rest of the job for you 😊 & No tedious job pushing the image every time you create a PR.





KUBERNETES

Kubernetes is an open-source platform used for automating the deployment, scaling, and management of containerized applications. It helps you manage and orchestrate large numbers of containers, ensuring they run reliably and efficiently in production environments.

FIRST STEP – create your own cluster.

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> k3d cluster create dev
ops-project-cluster -p "80:80@loadbalancer" -s 1 -a 1
INFO[0000] portmapping '80:80' targets the loadbalancer: defaulting to [servers:*:proxy agents:*:proxy]
INFO[0000] Prep: Network
INFO[0000] Created network 'k3d-devops-project-cluster'
INFO[0000] Created image volume k3d-devops-project-cluster-images
INFO[0000] Starting new tools node...
INFO[0000] Starting node 'k3d-devops-project-cluster-tools'
INFO[0001] Creating node 'k3d-devops-project-cluster-server-0'
INFO[0001] Creating node 'k3d-devops-project-cluster-agent-0'
INFO[0001] Creating LoadBalancer 'k3d-devops-project-cluster-serverlb'
INFO[0001] Using the k3d-tools node to gather environment information
INFO[0001] Starting new tools node...
INFO[0001] Starting node 'k3d-devops-project-cluster-tools'
INFO[0003] Starting cluster 'devops-project-cluster'
INFO[0003] Starting servers...
INFO[0003] Starting node 'k3d-devops-project-cluster-server-0'
INFO[0007] Starting agents...
INFO[0007] Starting node 'k3d-devops-project-cluster-agent-0'
INFO[0011] Starting helpers...
                                                    INFO[0011] Starting node 'k3d-devops-project-cluster-serverlb'
                                                                                                                  I
NFO[0018] Injecting records for hostAliases (incl. host.k3d.internal) and for 4 network members into CoreDNS configmap..
.
  INFO[0020] Cluster 'devops-project-cluster' created successfully!
                                                          INFO[0020] You can now use it like this:

    kubectl cluster-info

PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

As you can see, the command created a cluster named devops-project-cluster with 1 server and 1 agent with port 80 exposed. This means the load balancer in my k3d cluster will forward external requests on port 80 to the services running inside the cluster. Servers are responsible for managing the cluster and coordinating activities like scheduling and scaling , while Agents is where the actual containerized applications run.

Create a namespace

Namespaces divide a Kubernetes cluster - their purpose is to easily apply quotas and policies to groups of objects. They're not designed for strong workload isolation. If you don't explicitly define a target Namespace when deploying a namespaced object, it'll be deployed to the default Namespace. I created the namespace devops-app-namespace.

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get namespaces
NAME              STATUS   AGE
kube-system       Active   13m
kube-public       Active   13m
kube-node-lease   Active   13m
default           Active   13m
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

```
Windows PowerShell
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get nodes
NAME                                STATUS   ROLES                  AGE    VERSION
k3d-devops-project-cluster-server-0   Ready    control-plane,master   8m7s   v1.28.8+k3s1
k3d-devops-project-cluster-agent-0    Ready    <none>                 8m2s   v1.28.8+k3s1
k3d-devops-project-cluster-agent-1    Ready    <none>                 8m1s   v1.28.8+k3s1
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type namespace-manifest.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: devops-app-namespace
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f namespace-manifes
t.yaml
namespace/devops-app-namespace created
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get namespaces
NAME                 STATUS   AGE
kube-system          Active   16m
kube-public          Active   16m
kube-node-lease      Active   16m
default              Active   16m
devops-app-namespace   Active   5s
```

Create the needed Configmaps and Secrets

ConfigMaps are typically used to store non-sensitive configuration data such as: Environment variables, configuration files , Hostnames , Service ports, … while Secrets are designed for sensitive data such as passwords, certificates, and OAuth tokens.

In my configmap-database-manifest.yaml I defined two key-value pairs that contained information for the database and in the next picture it is shown that the configmap was indeed applied.

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type configmap-database-manifest.y
aml
apiVersion: v1
kind: ConfigMap
metadata:
  name: devops-app-configmap
  namespace: devops-app-namespace
data:
  MONGO_SERVER: mongodb
  MONGO_PORT: "27017"
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f configmap-databas
e-manifest.yaml
configmap/devops-app-configmap created
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get cm -n devops-app-names
pace
NAME                  DATA   AGE
kube-root-ca.crt      1      32m
devops-app-configmap   2      3m2s
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl describe cm devops-app-con
figmap -n devops-app-namespace
Name:         devops-app-configmap
Namespace:    devops-app-namespace
Labels:       <none>
Annotations:  <none>

Data
====
MONGO_PORT:
----
27017
MONGO_SERVER:
----
mongodb

BinaryData
====

Events:  <none>
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

In my secrets-database-manifest.yaml I added the mongo user & password as secrets.

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type secrets-database-manifest.yam
l
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
  namespace: devops-app-namespace
type: Opaque
stringData:
  mongo-root-username: rootuser
  mongo-root-password: rootpass
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f secrets-database-
manifest.yaml
secret/mongodb-secret created
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>  kubectl get secret creds -o yaml
Error from server (NotFound): secrets "creds" not found
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl describe secret mongodb-se
cret -n devops-app-namespace
Name:         mongodb-secret
Namespace:    devops-app-namespace
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
mongo-root-password:  8 bytes
mongo-root-username:  8 bytes
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

Create a Statefulset for the database

StatefulSets are used to manage stateful applications, where the identity or state of each Pod is important. They are often used for databases and other services that require stable network identities and persistent storage.

This manifest defines a **StatefulSet** to run 3 MongoDB instances with persistent storage. Each instance (Pod) is given a stable identity and 1GB of dedicated storage. Secrets are used to securely manage credentials, and a ConfigMap is used for port configuration.

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type statefulset-manifest.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb-sts
  namespace: devops-app-namespace
spec:
  serviceName: mongodb-sts
  replicas: 3
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo
          ports:
            - containerPort: 27017
          env:
            - name: MONGO_ROOT_USERNAME
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo-root-username
            - name: MONGO_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo-root-password
            - name: MONGO_PORT
              valueFrom:
                configMapKeyRef:
                  name: devops-app-configmap
                  key: MONGO_PORT
          volumeMounts:
            - name: mongodb-data
              mountPath: /data/db
  volumeClaimTemplates:
    - metadata:
        name: mongodb-data
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 1Gi
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f statefulset-manif
est.yaml
statefulset.apps/mongodb-sts created
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get statefulsets -n devops
-app-namespace --watch
NAME          READY   AGE
mongodb-sts   3/3     2m45s
```

Deployment

Created a configmap with key value pairs needed for the deployment manfiest. Next, I created a deployment manifest and applied it to the cluster.

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type configmap-deployment-manifest
.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: devops-app-deployment-configmap
  namespace: devops-app-namespace
data:
  SPRING_BOOT_APP: springboot-app
  SPRING_BOOT_PORT: "9393"
  SPRING_DATA_MONGODB_URI: mongodb://rootuser:rootpass@mongodb:27017/admin
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f configmap-deploym
ent-manifest.yaml
configmap/devops-app-deployment-configmap created
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get cm -n devops-app-names
pace
NAME                              DATA   AGE
kube-root-ca.crt                  1      107m
devops-app-configmap              2      78m
devops-app-deployment-configmap   3      11s
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type deployment-manifest.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: springboot-app-deployment
  namespace: devops-app-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: springboot-app
  template:
    metadata:
      labels:
        app: springboot-app
    spec:
      containers:
        - name: springboot-app
          image: arizizi/springboot-app:latest
          ports:
            - containerPort: 9393
          env:
            - name: SPRING_BOOT_APP
              valueFrom:
                configMapKeyRef:
                  name: devops-app-deployment-configmap
                  key: SPRING_BOOT_APP
            - name: SPRING_BOOT_PORT
              valueFrom:
                configMapKeyRef:
                  name: devops-app-deployment-configmap
                  key: SPRING_BOOT_PORT
            - name: SPRING_DATA_MONGODB_URI
              valueFrom:
                configMapKeyRef:
                  name: devops-app-deployment-configmap
                  key: SPRING_DATA_MONGODB_URI
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl describe deploy springboot
-app-deployment -n devops-app-namespace
Name:                   springboot-app-deployment
Namespace:              devops-app-namespace
CreationTimestamp:      Sat, 21 Sep 2024 18:28:11 +0200
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=springboot-app
Replicas:               1 desired | 1 updated | 1 total | 0 available | 1 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=springboot-app
  Containers:
   springboot-app:
    Image:      arizizi/springboot-app:latest
    Port:       9393/TCP
    Host Port:  0/TCP
    Environment:
      SPRING_BOOT_APP:          <set to the key 'SPRING_BOOT_APP' of config map 'devops-app-deployment-configmap'>          Optional
: false
      SPRING_BOOT_PORT:         <set to the key 'SPRING_BOOT_PORT' of config map 'devops-app-deployment-configmap'>         Optional
: false
      SPRING_DATA_MONGODB_URI:  <set to the key 'SPRING_DATA_MONGODB_URI' of config map 'devops-app-deployment-configmap'>  Optional
: false
    Mounts:                     <none>
  Volumes:                      <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      False   MinimumReplicasUnavailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  <none>
NewReplicaSet:   springboot-app-deployment-666986dfb4 (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  2m18s  deployment-controller  Scaled up replica set springboot-app-deployment-666986dfb4 to 1
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes>
```

This Deployment configuration sets up a Spring Boot application in a specified namespace, pulling the application image from a private repository, and configuring the application using environment variables sourced from configMap-deployment. It ensures that there is always one instance of the application running.

- **apiVersion: apps/v1 -**Specifies the API version of the Kubernetes resource
- **kind: Deployment -** Indicates that this resource is a Deployment, which manages the deployment of applications by ensuring the specified number of replicas are running.
- **metadata** Contains information about the Deployment like name and namespace
- **spec -** Describes the desired state of the Deployment: **replicas**, how to identify the pods that belong to this Deployment..
- **template -**Describes the pod template that the Deployment uses to create new pods

Create a Service

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type service-manifest.
yaml
apiVersion: v1
kind: Service
metadata:
  name: springboot-app-service
  namespace: devops-app-namespace
spec:
  selector:
    app: springboot-app
  ports:
    - protocol: TCP
      port: 9191
      targetPort: 9191
  type: ClusterIP
```

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb
  namespace: devops-app-namespace
spec:
  ports:
    - port: 27017
      name: mongo
  clusterIP: None
  selector:
    app: mongodb
```

- The **MongoDB Headless Service** is set up as a headless Service to allow direct access to MongoDB pods without load balancing.
- **clusterIP: None**: This makes the Service a headless Service, meaning it does not get assigned a ClusterIP. It allows direct access to the individual pods' IPs instead of load-balancing traffic.
- The **Spring Boot app Service** provides a stable internal endpoint for accessing the Spring Boot application pods, allowing other services within the cluster to communicate with it over TCP.
-

Ingress Controller

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type ingress-manifest.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: springboot-app-ingress
  namespace: devops-app-namespace
  annotations:
    traefik.ingress.kubernetes.io/router.entrypoints: web
spec:
  ingressClassName: traefik
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: springboot-app-service
            port:
              number: 9191
  - host: devops-app.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: springboot-app-service
            port:
              number: 9191
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl describe ingre
ss springboot-app-ingress -n devops-app-namespace
Name:             springboot-app-ingress
Labels:           <none>
Namespace:        devops-app-namespace
Address:          172.20.0.3,172.20.0.4
Ingress Class:    traefik
Default backend:  <default>
Rules:
  Host                  Path  Backends
  ----                  ----  --------
  *
                        /   springboot-app-service:9191 (10.42.0.9:9191)
  devops-app.example.com
                        /   springboot-app-service:9191 (10.42.0.9:9191)
Annotations:            traefik.ingress.kubernetes.io/router.entrypoints: web
Events:                 <none>
```

The following Ingress configuration allows external HTTP traffic to access the Spring Boot application through the Traefik Ingress controller. It does the job by routing requests either to the root path or specifically to the host devops-app.example.com, forwarding them to the springboot-app-service on port 9191. This setup enables users to access the application via a user-friendly domain name while leveraging the capabilities of an Ingress controller.

A problem occurred !

Had to change the deployment controller because login credentials were needed for Docker Hub.

As you can see I created another secret named docker-registry-secret and applied it to my cluster. Then I proceeded to perform a rollout to the deployment controller.

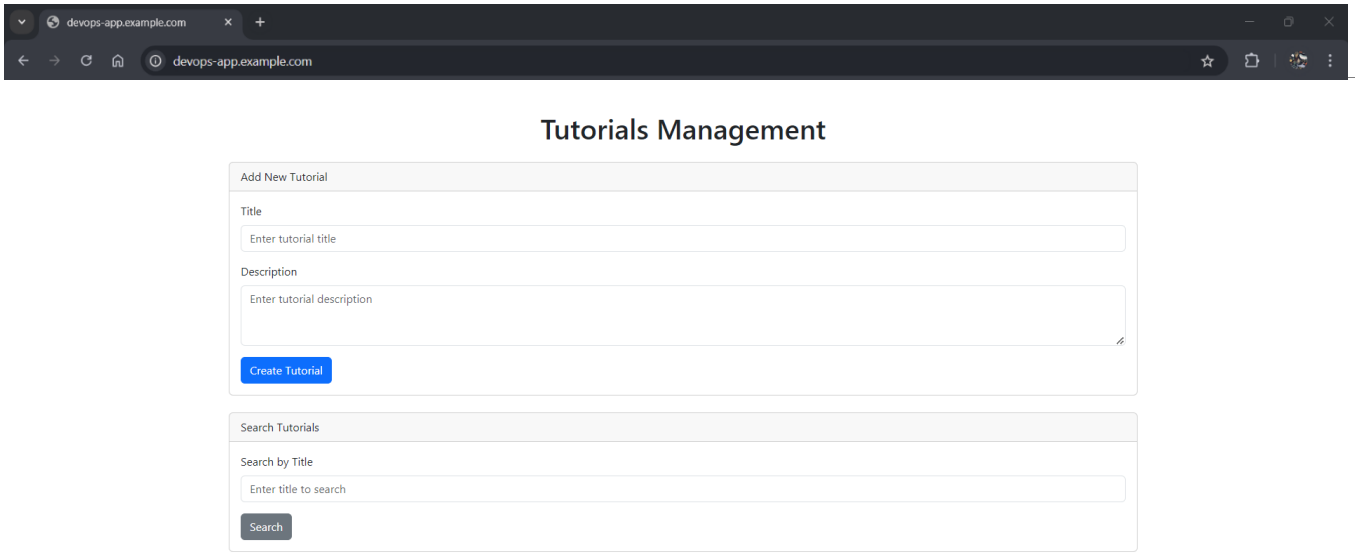
```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> [Convert]::ToBase64String([Text.En
coding]::UTF8.GetBytes((Get-Content 'C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes\d
ockerhub-registry.json' -Raw)))
ew0KICAgICJhdXRocyI6IHsNCiAgICAgICAgImh0dHBzOi8vaW5kZXguZG9ja2VyLmlvL3YxLyI6IHsNCiAgICAgICAgICAgICJ1c2VybmFtZSI6ICJhcml6aXppIiwNCiAg
ICAgICAgICAgICJwYXNzd29yZCI6ICJzd2FnZ3kxMjMiLA0KICAgICAgICAgICAgImVtYWlsIjogImFyaXpib2phbmFAZ21haWwuY29tIiwNCiAgICAgICAgICAgICJhdXRo
IjogIlllYSnBlbWw2YVRwemQyRm5aM2t4TWpNPSINCiAgICAgICAgfQ0KICAgIH0NCn0NCg==
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> type docker-registry-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: registry-credentials-secret
  namespace: devops-app-namespace
type: kubernetes.io/dockerconfigjson
data:
  .dockerconfigjson: ew0KICAgICJhdXRocyI6IHsNCiAgICAgICAgImh0dHBzOi8vaW5kZXguZG9ja2VyLmlvL3YxLyI6IHsNCiAgICAgICAgICAgICJ1c2VybmFtZSI
6ICJhcml6aXppIiwNCiAgICAgICAgICAgICJwYXNzd29yZCI6ICJzd2FnZ3kxMjMiLA0KICAgICAgICAgICAgImVtYWlsIjogImFyaXpib2phbmFAZ21haWwuY29tIiwNCiA
gICAgICAgICAgICJhdXRoIjogIlllYSnBlbWw2YVRwemQyRm5aM2t4TWpNPSINCiAgICAgICAgfQ0KICAgIH0NCn0NCg==
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f docker-registry-s
ecret.yaml
secret/registry-credentials-secret created
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl apply -f deployment-manife
st.yaml
deployment.apps/springboot-app-deployment configured
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl rollout restart deployment
/springboot-app-deployment -n devops-app-namespace
deployment.apps/springboot-app-deployment restarted
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get pods -n devops-app-nam
espace --watch
NAME                                   READY   STATUS              RESTARTS   AGE
mongodb-sts-0                          1/1     Running             0          90m
mongodb-sts-1                          1/1     Running             0          88m
mongodb-sts-2                          1/1     Running             0          88m
springboot-app-deployment-7b9b678fb5-fb6g5   0/1     ContainerCreating   0          35s
springboot-app-deployment-89d7777f6-lftwf    0/1     ContainerCreating   0          25s
springboot-app-deployment-7b9b678fb5-fb6g5   1/1     Running             0          61s
springboot-app-deployment-89d7777f6-lftwf    1/1     Running             0          51s
springboot-app-deployment-7b9b678fb5-fb6g5   1/1     Terminating         0          61s
springboot-app-deployment-7b9b678fb5-fb6g5   0/1     Terminating         0          64s
springboot-app-deployment-7b9b678fb5-fb6g5   0/1     Terminating         0          64s
springboot-app-deployment-7b9b678fb5-fb6g5   0/1     Terminating         0          64s
springboot-app-deployment-7b9b678fb5-fb6g5   0/1     Terminating         0          64s
```

Everything works great!

September 2024



```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:60487
CoreDNS is running at https://127.0.0.1:60487/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://127.0.0.1:60487/api/v1/namespaces/kube-system/services/https:metrics-server:https/proxy
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> docker ps
CONTAINER ID   IMAGE                            COMMAND                  CREATED             STATUS             PORTS
                                NAMES
12d12c06c526   ghcr.io/k3d-io/k3d-tools:5.6.3   "/app/k3d-tools noop"    16 minutes ago      Up 16 minutes
                                k3d-devops-project-cluster-tools
9e69e577599f   ghcr.io/k3d-io/k3d-proxy:5.6.3   "/bin/sh -c nginx-pr…"   About an hour ago   Up About an hour   0.0.0.0:
80->80/tcp, 0.0.0.0:60487->6443/tcp   k3d-devops-project-cluster-serverlb
6fa4820a1baa   rancher/k3s:v1.28.8-k3s1         "/bin/k3d-entrypoint…"   About an hour ago   Up About an hour
                                k3d-devops-project-cluster-agent-0
0434060dca4a   rancher/k3s:v1.28.8-k3s1         "/bin/k3d-entrypoint…"   About an hour ago   Up About an hour
                                k3d-devops-project-cluster-server-0
414715ef0e90   mongo-express                    "/sbin/tini -- /dock…"   2 days ago          Up 14 seconds      0.0.0.0:
8081->8081/tcp                  mongo-express
74e6da988825   postgres                         "docker-entrypoint.s…"   4 months ago        Up About an hour   0.0.0.0:
54322->5432/tcp                 finki_services_db
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> mkdir -p $HOME/.kube
```

```
PS C:\Users\Bojana\Desktop\finki\semestar 6\kiii-proekt\DevOps-project\DevOps-Project\kubernetes> kubectl get pods -n devops-app-namespace
NAME                                   READY   STATUS    RESTARTS   AGE
mongodb-sts-0                          1/1     Running   0          63m
mongodb-sts-1                          1/1     Running   0          55m
mongodb-sts-2                          1/1     Running   0          54m
springboot-app-deployment-645b7dbf9b-krxhc   1/1     Running   0          50m
```