

Rapport du projet Docker:

Context:

Considérons une application web 3-tiers composée de frontend (Angular), backend (Java/Spring) et une base de données (MySQL).

Le projet comporte 2 parties principaux :

Partie 1 :

-Dans cette partie, il est demandé de réaliser les manipulations suivantes en utilisant un outil de conteneurisation comme Docker.

Partie 2 :

-Dans cette partie, il est permis d'utiliser n'importe quel outil permettant de créer un cluster Kubernetes en local (ex : Minikube, MicroK8s, kind, k3d, k3s, etc.).

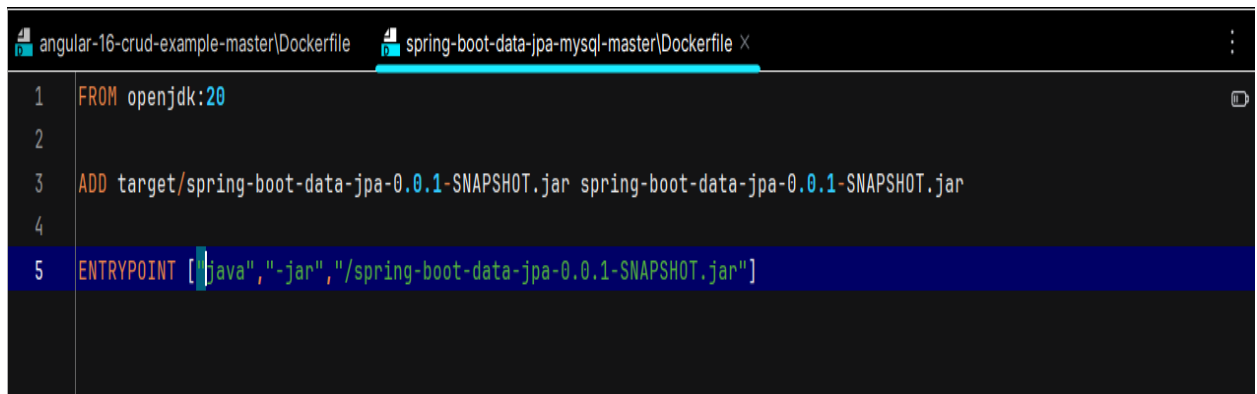
Réalisé par : **EL IDRISSI Imad**

2023-2024

Partie 1 :

Dans cette partie, il est demandé de réaliser les manipulations suivantes en utilisant un outil de conteneurisation comme Docker.

1. Pour créer un conteneur pour la base de données MySQL en instanciant l'image officielle mysql j'ai utilisé les commandes suivantes :
 - a. `# docker pull mysql`
 - b. `# docker run -p 3307:3306 --name mysqldb -e MYSQL_ROOT_PASSWORD=123456 -e MYSQL_DATABASE=testdb mysql:latest`
2. Créer un réseau Docker (tout en lui précisant un driver convenable). Puis connecter le conteneur de la base de données mysql avec ce réseau :
 - a. `# docker network create spring-network`
 - b. `# docker network connect spring-network mysqldb`
3. Créer un fichier Dockerfile pour le projet Backend (essayez d'appliquer le maximum de bonnes pratiques).



```
1 FROM openjdk:20
2
3 ADD target/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar spring-boot-data-jpa-0.0.1-SNAPSHOT.jar
4
5 ENTRYPOINT ["java", "-jar", "/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar"]
```

4. Ce Dockerfile suit généralement les bonnes pratiques pour créer une image Docker pour une application Spring Boot. Voici une explication détaillée des différentes parties de ce Dockerfile :
 - a. `FROM openjdk:20`

Explication : Utilise une image de base officielle OpenJDK version 20. C'est une bonne pratique d'utiliser des images officielles provenant de sources fiables, car elles sont généralement bien maintenues et mises à jour.
 - b. `ADD target/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar spring-boot-data-jpa-0.0.1-SNAPSHOT.jar`

Explication : Ajoute (copie) le fichier JAR de l'application Spring Boot dans l'image Docker. L'utilisation de l'instruction **ADD** est courante pour copier des fichiers depuis le système de fichiers local dans l'image Docker.
 - c. `ENTRYPOINT ["java", "-jar", "/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar"]`

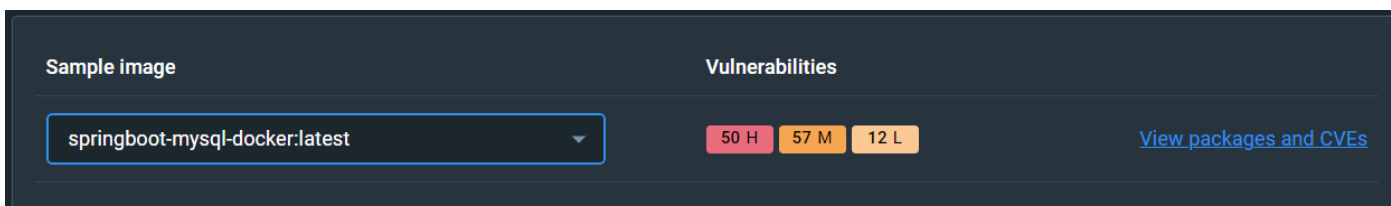
Explication : Définit le point d'entrée de l'image Docker lorsqu'un conteneur basé sur cette image est lancé. L'instruction **ENTRYPOINT** spécifie la commande qui sera exécutée lorsque le conteneur démarre. Dans ce cas, il exécute la commande **java -jar /spring-boot-data-jpa-0.0.1-SNAPSHOT.jar**, ce qui est typique pour une application Spring Boot.

5.

a. Créer une image docker en local à partir de ce fichier Dockerfile

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers> cd .\spring-boot-data-jpa-mysql-master\  
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\spring-boot-data-jpa-mysql-master> docker network connect spring-network mysqlldb  
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\spring-boot-data-jpa-mysql-master> docker build -t springboot-mysql-docker .
```

b. Scanner l'image des vulnérabilités qu'elle peut contenir.



d. Instancier cette image en créant un conteneur s'exécutant en local et partageant le même réseau avec le conteneur de la base de données.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\spring-boot-data-jpa-mysql-master> docker run -p 3307:3306 --name mysqlldb -e MYSQL_ROOT_PASSWORD=123456 -e MYSQL_DAT  
ABASE=testdb mysql:latest
```

e. Inspecter ce conteneur du backend.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\spring-boot-data-jpa-mysql-master> docker inspect springboot-mysql-docker  
[  
  {  
    "Id": "5257182f5f72d77901faddab3ad0f0f16e0c210d8786ac96ccbb273026bc002",  
    "Created": "2023-11-27T09:29:36.11184498Z",  
    "Path": "java",  
    "Args": [  
      "-jar",  
      "/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar"  
    ],  
    "State": {  
      "Status": "running",  
      "Running": true,  
      "Paused": false,  
      "Restarting": false,  
      "OOMKilled": false,  
      "Dead": false,  
      "Pid": 21936,  
      "ExitCode": 0,  
      "Error": "",  
      "StartedAt": "2023-11-27T09:29:40.694525082Z",  
      "FinishedAt": "0001-01-01T00:00:00Z"  
    }  
  }  
]
```

f. Afficher les logs liés à ce conteneur du backend.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\spring-boot-data-jpa-mysql-master> docker logs springboot-mysql-docker  
  
:: Spring Boot :: (v3.1.0)  
  
2023-11-27T09:29:42.484Z INFO 1 --- [main] c.b.s.d.SpringBootDataJpaApplication : Starting SpringBootDataJpaApplication v0.0.1-SNAPSHOT using Java 20 with PID 1 (/spr  
ing-boot-data-jpa-0.0.1-SNAPSHOT.jar started by root in /)  
2023-11-27T09:29:42.488Z INFO 1 --- [main] c.b.s.d.SpringBootDataJpaApplication : No active profile set, falling back to 1 default profile: "default"  
2023-11-27T09:29:44.036Z INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.  
2023-11-27T09:29:44.152Z INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 101 ms. Found 1 JPA repository interface  
s.
```

6.

a. Partie Dockerfile frontend :

```
angular-16-crud-example-master\Dockefile × spring-boot-data-jpa-mysql-master\Dockefile
1  # Use the official Node image as the base image
2  FROM node:latest as build
3
4  # Set the working directory in the container
5  WORKDIR /app
6
7  # Copy package.json and package-lock.json to the container
8  COPY package*.json ./
9
10 # Install Angular dependencies
11 RUN npm install
12
13 # Copy the rest of the application code to the container
14 COPY . .
15
16 # Build the Angular application
17 RUN npm run build --prod
18
19 # Use a smaller base image for the final image
20 FROM nginx:alpine
21
22 # Copy the built Angular app to the Nginx HTML directory
23 COPY --from=build app/dist/angular-16-crud /usr/share/nginx/html
24
25 # Expose the port that Nginx will run on
26 EXPOSE 80
27
28 # Start Nginx
29 CMD ["nginx", "-g","daemon off;"]
```

b. Ce Dockerfile semble bien structuré pour construire et déployer une application Angular avec Nginx. Voici une explication détaillée des différentes parties du fichier :

- i. # Use the official Node image as the base image
FROM node:latest as build*

Explication : Utilise l'image officielle Node comme image de base pour la construction de l'application Angular. C'est une bonne pratique d'utiliser des images officielles provenant de sources fiables.

ii. **# Set the working directory in the container**

WORKDIR /app

Explication : Définit le répertoire de travail à l'intérieur du conteneur. C'est le répertoire dans lequel les commandes suivantes seront exécutées.

iii. **# Copy package.json and package-lock.json to the container**

COPY package*.json ./

Explication : Copie les fichiers **package.json** et **package-lock.json** dans le conteneur. Ceci est souvent fait séparément avant de copier le reste des fichiers pour tirer parti de la mise en cache des dépendances lors de la construction.

iv. **# Install Angular dependencies**

RUN npm install

Explication : Installe les dépendances Angular. C'est une étape importante pour s'assurer que toutes les dépendances sont installées avant de copier le reste du code source

v. **# Copy the rest of the application code to the container**

COPY . .

Explication : Copie le reste du code source de l'application dans le conteneur.

vi. **# Build the Angular application**

RUN npm run build --prod

Explication : Construit l'application Angular en utilisant la commande **npm run build --prod**. Ceci génère les fichiers optimisés prêts pour la production.

vii. **# Use a smaller base image for the final image**

FROM nginx:alpine

Explication : Utilise une image Nginx plus légère (alpine) comme image de base pour l'image finale. Ceci réduit la taille de l'image finale.

viii. **# Copy the built Angular app to the Nginx HTML directory**

COPY --from=build app/dist/angular-16-crud /usr/share/nginx/html

Explication : Copie les fichiers construits de l'application Angular depuis l'image de construction vers le répertoire HTML de Nginx.

ix. # Expose the port that Nginx will run on
EXPOSE 80

Explication : Expose le port 80, le port par défaut sur lequel Nginx écoute.

x. # Start Nginx

CMD ["nginx", "-g", "daemon off;"]

Explication : Démarre Nginx en utilisant la commande **nginx -g "daemon off;"**. Cela permet d'exécuter Nginx en mode avant-plan dans le conteneur.

c.

ii. Créer une image docker en local à partir de ce fichier Dockerfile

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers> cd .\angular-16-crud-example-master\  
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master>  
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master> docker build -t angular-image .  
[+] Building 184.9s (16/16) FINISHED
```

docker:default

ii. Scanner l'image des vulnérabilités qu'elle peut contenir.

Sample image

angular-image:latest

Vulnerabilities

1 H 3 M 0 L

[View packages and CVEs](#)

iii. Instancier cette image en créant un conteneur s'exécutant en local et partageant le même réseau avec le conteneur de la base de données.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master> docker run -d --name angular-container --net spring-network -p 8081:80 angular-image  
a52719e23d1e93b91681391b43811256c1eaaa2056156b0979fbbcfbc6405c7  
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master>
```

iv. Inspecter ce conteneur du frontend.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master> docker inspect angular-container  
[  
  {  
    "Id": "a52719e23d1e93b91681391b43811256c1eaaa2056156b0979fbbcfbc6405c7",  
    "Created": "2023-11-27T12:52:23.344614482Z",  
    "Path": "/docker-entrypoint.sh",  
    "Args": [  
      "nginx",  
      "-g",  
    ]  
  }  
]
```

v. Afficher les logs liés à ce conteneur du frontend.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master> docker logs angular-container
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/11/27 12:52:24 [notice] 1#1: using the "epoll" event method
2023/11/27 12:52:24 [notice] 1#1: nginx/1.25.3
2023/11/27 12:52:24 [notice] 1#1: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r10)
2023/11/27 12:52:24 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2023/11/27 12:52:24 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
```

7. L'application a été bien conteneurisée et qu'elle fonctionne correctement :

bezKoder Tutorials Add

Tutorials List

imad

Remove All

Tutorial

Title: imad

Description: test

Status: Pending

Edit

8. Supprimer les 3 conteneurs en exécution :

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master> docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED        STATUS        PORTS                               NAMES
a52719e23d1e   angular-image            "/docker-entrypoint..." 4 hours ago   Up 4 hours   0.0.0.0:8081->80/tcp               angular-container
5257182f5f72   springboot-mysql-docker  "java -jar /spring-b..." 7 hours ago   Up 7 hours   0.0.0.0:8080->8080/tcp             springboot-mysql-docker
699baba38250   mysql:latest            "docker-entrypoint.s..." 7 hours ago   Up 7 hours   33060/tcp, 0.0.0.0:3307->3306/tcp   mysqlldb
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers\angular-16-crud-example-master> docker rm angular-container springboot-mysql-docker mysqlldb
```

9.

a. Fichier docker-compose.yml :

```
docker-compose.yml x
1 version: '3.8'
2
3 services:
4   # Frontend service
5   frontend:
6     build:
7       context: ../angular-16-crud-example-master # Chemin vers le répertoire contenant le Dockerfile du frontend
8     ports:
9       - "80:80" # Mappe le port 80 du conteneur vers le port 80 de l'hôte
10
11
12   # Backend service
13   backend:
14     build:
15       context: ../spring-boot-data-jpa-mysql-master # Chemin vers le répertoire contenant le Dockerfile du backend
16     depends_on:
17       - mysql
18
19   # Database service
20   mysql:
21     image: mysql:latest
22     environment:
23       MYSQL_DATABASE: testdb
24       MYSQL_ROOT_PASSWORD: 123456
25     ports:
26       - "3307:3306" # Mappe le port 3306 du conteneur MySQL vers le port 3306 de l'hôte
27
28
```

b. Commande d'exécution :

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers> docker-compose up -d
[+] Running 3/3
✔ Container web3-tiers-frontend-1 Started 0.1s
✔ Container web3-tiers-mysqldb-1 Started 0.1s
✔ Container web3-tiers-backend-1 Started 0.1s
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers>
```

c. S'assurer que l'application fonctionne correctement:

bezKoder Tutorials Add

Tutorials List

10.

```
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers> docker stop $(docker ps -aq) && docker rm $(docker ps -aq)
At line:1 char:30
+ docker stop $(docker ps -aq) && docker rm $(docker ps -aq)
+
+ ~~~
The token '&&' is not a valid statement separator in this version.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : InvalidEndOfLine

Deleted: sha256:a6316a7b5ef7408c536c06c5f20e6138cd8725c7ed3fc895ec2540d28f596d87
Deleted: sha256:ce9f13233387d62a08d318ab3756f8acca152ed67e54537ef94ceca1e899fd4
Deleted: sha256:92ad3da2e834e507138e715ff516025fea7019725b49bf34ff62b97a2cfa8bfd
Deleted: sha256:1b03c6666623e2577b638c0f64223da6c323c0afd6f71dc5b7c5f337fb1c9a3a
Deleted: sha256:e09381dd9cb61b8167503ef95fbaed7170e65d18a9f35c0750b8919d76d0982f
Deleted: sha256:7537eb8f37e6a0dfada19f135ad86b82f995bfb10fdc2ac31bfff5e911128be
Deleted: sha256:9af05b7bb2b5f7844eb2582d4211b77737a57b1f8087a728be40de0dc0eb542f
Deleted: sha256:d317a7939697b640434bdb25cb045b83b076cbb159baee7f9e198b1a72b8b6b4
Deleted: sha256:6437c74f97a878ba0770e4c41fdfb73df394605adf6cca9f9bea813dd43f08c43
Deleted: sha256:d6fe63e8be63d078aeef9739f2c7ea101e6cc1a3f998d179af63a10e7f0f959d
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\User\Desktop\Workspace\INE2 aseds\projet Docker\web 3-tiers>
```

11.

- a. Stocker les images que vous avez construites.

```
# Créez un réseau Docker pour le registre privé (facultatif)
docker network create my-registry-net

# Démarrez le registre privé Docker
docker run -d -p 5000:5000 --name registry --network my-registry-net registry:2
```

Partie 2 :

Dans cette partie, il est permis d'utiliser n'importe quel outil permettant de créer un cluster Kubernetes en local Minikube.

1.

a. Deployment

Frontend :

```
18 - apiVersion: apps/v1
19   kind: Deployment
20   metadata:
21     name: frontend-deployment
22     namespace: exam
23   spec:
24     replicas: 2
25     selector:
26       matchLabels:
27         app: frontend
28     template:
29       metadata:
30         labels:
31           app: frontend
32       spec:
33         containers:
34           - name: frontend
35             image: imad17/angular-image:latest
36             ports:
37               - containerPort: 80
38                 name: http-port
39             livenessProbe: Probe
46             readinessProbe: Probe
53             startupProbe: Probe
```

Backend :

```
15   apiVersion: apps/v1
16   kind: Deployment
17   metadata:
18     name: backend-deployment
19     namespace: exam
20   spec:
21     replicas: 2
22     selector:
23       matchLabels:
24         app: backend
25     template:
26       metadata:
27         labels:
28           app: backend
29       spec:
30         containers:
31         - name: backend
32           image: imad17/springboot-docker:latest
33           ports:
34             - containerPort: 8080
35           env:
36             - name: MYSQL_ROOT_PASSWORD
37               valueFrom:
38                 secretKeyRef:
39                   name: mysql-secret
40                   key: MYSQL_ROOT_PASSWORD
41             - name: SPRING_DATASOURCE_URL
42               valueFrom:
43                 configMapKeyRef:
44                   name: backend-config
45                   key: SPRING_DATASOURCE_URL
46             - name: SPRING_DATASOURCE_USERNAME
47               valueFrom:
48                 configMapKeyRef:
49                   name: backend-config
50                   key: SPRING_DATASOURCE_USERNAME
51           readinessProbe: Probe
52           livenessProbe: Probe
53           startupProbe: Probe
```

b. Service

Frontent :

```
1   apiVersion: v1
2   kind: List
3   items:
4     - apiVersion: v1
5       kind: Service
6       metadata:
7         name: frontend-service
8         namespace: exam
9       spec:
10        selector:
11          app: frontend
12        ports:
13          - protocol: TCP
14            port: 8081
15            targetPort: 80
16        type: ClusterIP
```

Backend :

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

Database :

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: mysql-service
5    namespace: exam
6  spec:
7    selector:
8      app: mysql
9    ports:
10     - protocol: TCP
11       port: 3306
12       targetPort: 3306
13
```

C. ConfigMap

```
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: backend-config
5    namespace: exam
6  data:
7    SPRING_DATASOURCE_URL: "jdbc:mysql://mysql-service:3306/testdb"
8    SPRING_DATASOURCE_USERNAME: "root"
9
```

d. Secret

```
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: mysql-secret
5    namespace: exam
6  type: Opaque
7  data:
8    MYSQL_ROOT_USERNAME: cm9vdA==
9    MYSQL_ROOT_PASSWORD: MTIzNDU2
```

e. StatefulSet

```
16  apiVersion: apps/v1
17  kind: StatefulSet
18  metadata:
19    name: mysql-statefulset
20    namespace: exam
21  spec:
22    serviceName: mysql-service
23    replicas: 2
24    selector:
25      matchLabels:
26        app: mysql
27    template:
28      metadata:
29        labels:
30          app: mysql
31      spec:
32        containers:
33          - name: mysql
34            image: mysql:latest
35            env:
36              - name: MYSQL_ROOT_PASSWORD
37                valueFrom:
38                  secretKeyRef:
39                    name: mysql-secret
40                    key: MYSQL_ROOT_PASSWORD
41              - name: MYSQL_DATABASE
42                value: testdb
43            ports:
44              - containerPort: 3306
45                name: db-port
46            livenessProbe:
47              exec:
48                command:
49                  - cat
50                  - /tmp/healthy
51              initialDelaySeconds: 3
52              periodSeconds: 5
```

2. Définition des quotas pour la consommation des ressources Mémoire et CPU de chaque Pod exécuté dans le namespace « exam » :

```
1  # Fichier : resource-quota.yaml
2  apiVersion: v1
3  kind: ResourceQuota
4  metadata:
5    name: compute-resources
6    namespace: exam
7  spec:
8    hard:
9      pods: "10"
10     limits.cpu: "6"
11     limits.memory: 10Gi
12
```

3. Contrôler l'accès aux ressources existantes dans le namespace « exam » par un rôle RBAC :

```
1  apiVersion: rbac.authorization.k8s.io/v1
2  kind: Role
3  metadata:
4    namespace: exam
5    name: resource-access-role
6  rules:
7    - apiGroups: [""]
8      resources: ["pods", "services", "deployments", "configmaps"]
9      verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
10
```

5.

a. Liveness prob:


Frontend :

```
39      livenessProbe:
40        exec:
41          command:
42            - cat
43            - /tmp/healthy
44          initialDelaySeconds: 10
45          periodSeconds: 20
```

Backend :

```
58      livenessProbe:
59        exec:
60          command:
61            - cat
62            - /tmp/healthy
63          initialDelaySeconds: 10
64          periodSeconds: 15
```

Database :

```
46       livenessProbe:
47        exec:
48          command:
49            - cat
50            - /tmp/healthy
51          initialDelaySeconds: 3
52          periodSeconds: 5
```

b. Readiness prob :

Frontend :

```
46      readinessProbe:
47        exec:
48          command:
49            - cat
50            - /tmp/healthy
51        initialDelaySeconds: 3
52        periodSeconds: 5
```

Backend :

```
      readinessProbe:
        exec:
          command:
            - cat
            - /tmp/healthy
        initialDelaySeconds: 5
        periodSeconds: 10
```

Database :

```
53      ✓      readinessProbe:
54        ✓      exec:
55          ✓      command:
56            - cat
57            - /tmp/healthy
58        initialDelaySeconds: 3
59        periodSeconds: 5
```

c. Startup prob :

Frontend :

```
53      ✓      startupProbe:
54        ✓      exec:
55          ✓      command:
56            - touch
57            - /tmp/healthy
58        failureThreshold: 30
59        periodSeconds: 5
```

Backend :

```
65 startupProbe:
66   exec:
67     command:
68       - touch
69       - /tmp/healthy # Créer le fichier /tmp/healthy lors de la sonde de démarrage
70   failureThreshold: 30
```

Database :

```
startupProbe:
  exec:
    command:
      - touch
      - /tmp/healthy
  failureThreshold: 30
  periodSeconds: 5
```

6. Création un Ingress pour accéder à l'application depuis un navigateur moyennant un nom de domaine

```
61 - apiVersion: networking.k8s.io/v1
62   kind: Ingress
63   metadata:
64     name: frontend-ingress
65     namespace: exam
66     annotations:
67       nginx.ingress.kubernetes.io/rewrite-target: /
68   spec:
69     rules:
70       - host: aseds-ine2-docker.com
71         http:
72           paths:
73             - path: /
74               pathType: Prefix
75             backend:
76               service:
77                 name: frontend-service
78                 port:
79                   number: 8081
80
```


7. Suite à une erreur technique la méthode ne marche pas donc j'ai procédé par la méthode **kubectl port-forward** :

Steps of the method :

1# Identify the Service Port:

```
$kubectl get service/backend-service -n exam
```

2# Run kubectl port-forward:

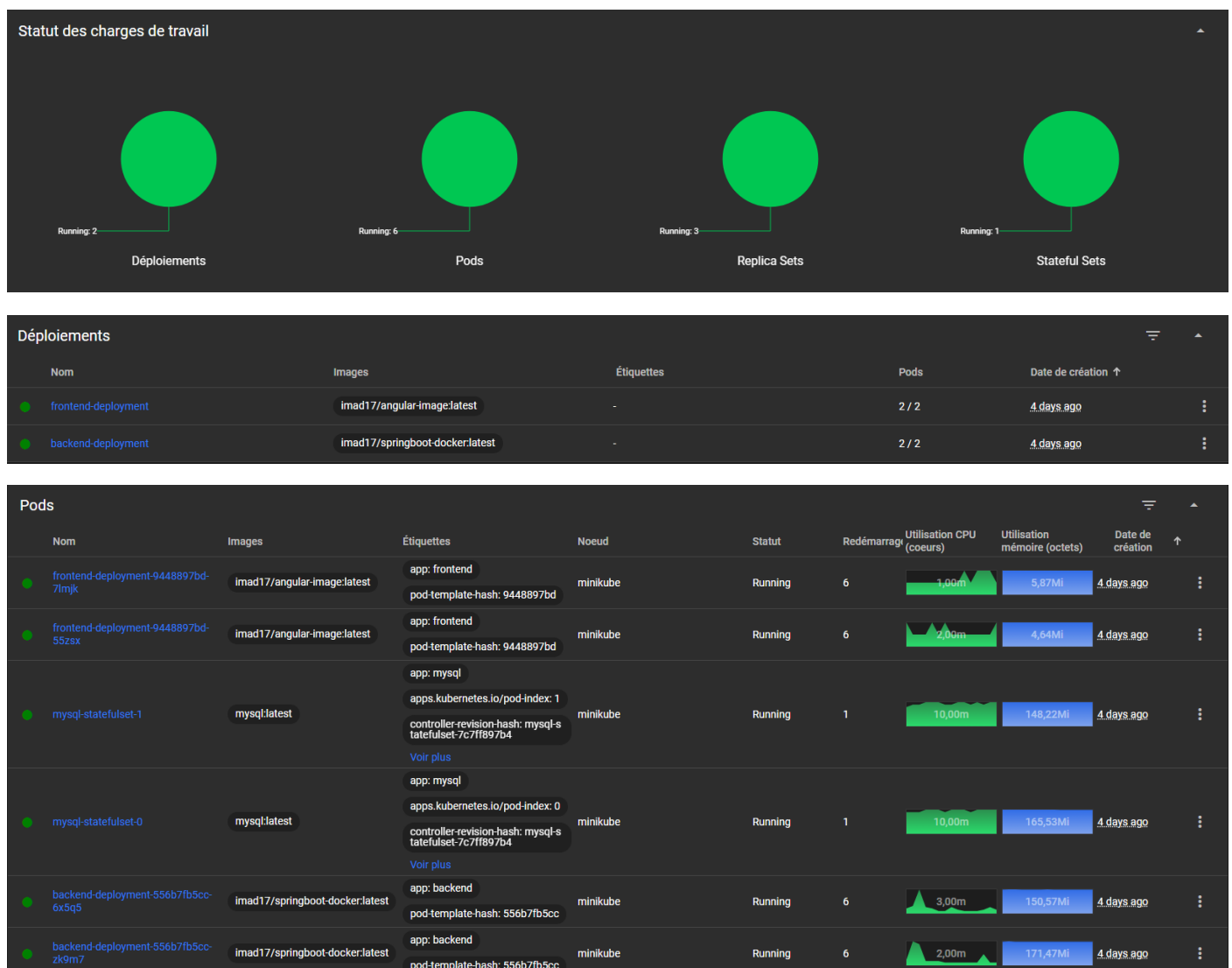
```
$kubectl port-forward service/backend-service -n exam 8080:8080
```

NB : Cette commande transfère votre port local 8080 vers le port 8080 du service backend dans l'espace de noms de l'examen.

3# Access the Application:

```
$kubectl port-forward service/frontend-service -n exam 8081:8081
```

NB : Once the port-forwarding is established, you can access your application by opening a web browser and navigating to <http://localhost:8081> . If your frontend service is on a different port, adjust the local port accordingly



Replica Sets						
Nom	Images	Étiquettes		Pods	Date de création ↑	
frontend-deployment-9448897bd	imad17/angular-image:latest	app: frontend	pod-template-hash: 9448897bd	2 / 2	4 days ago	
frontend-deployment-848d4c97b5	imad17/angular-image:latest	app: frontend	pod-template-hash: 848d4c97b5	0 / 0	4 days ago	
backend-deployment-556b7fb5cc	imad17/springboot-docker:latest	app: backend	pod-template-hash: 556b7fb5cc	2 / 2	4 days ago	

Stateful Sets						
Nom	Images	Étiquettes		Pods	Date de création ↑	
mysql-statefulset	mysql:latest			2 / 2	4 days ago	

← → ↺

localhost:8081/tutorials

☆ ABP

🔗 📄 🌐 🗑

Apps YouTube Participation | Hackt... Boîte de réception (...) UBI

bezKoder Tutorials Add

Search by title

Search

Tutorials List

test

imad

Remove All

Tutorial

Title: imad

Description: 123

Status: Published

Edit

Github Repo link : https://github.com/iei-imad/Docker_Kubernetes.git