UNIVERSITÉ ABDELMALEK ESSAADI FACULTÉ DES SCIENCES ET TECHNIQUES DE TANGER
DÉPARTEMENT GÉNIE INFORMATIQUE

Atelier11 : Conteneurisation et Orchestration d'une Application Web

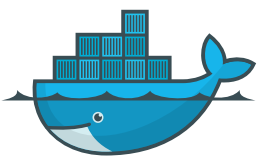
Module : Virtualisation
Master M1 : Sécurité IT et Big Data

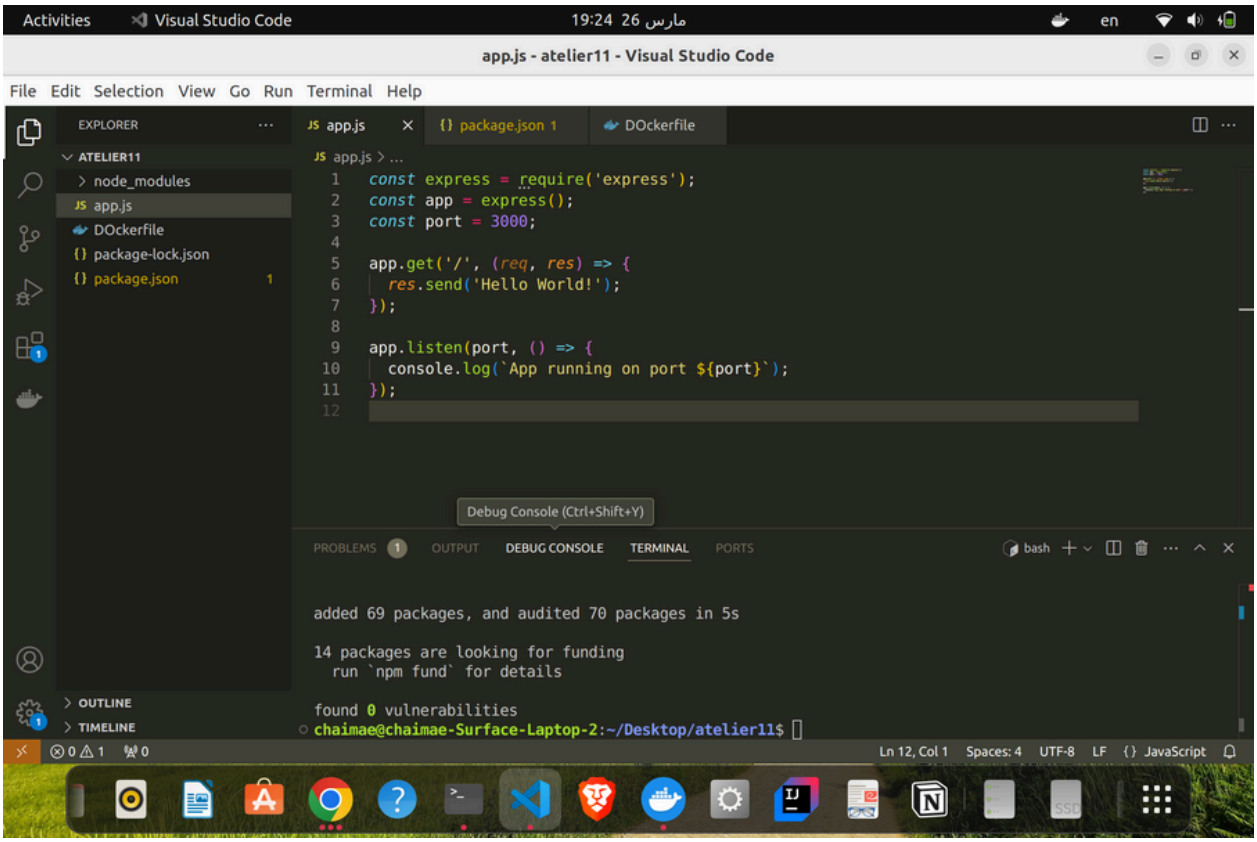
Préparé par :
Bouassab Chaimae
Encadré par :
Pr Mohamed BEN AHMED

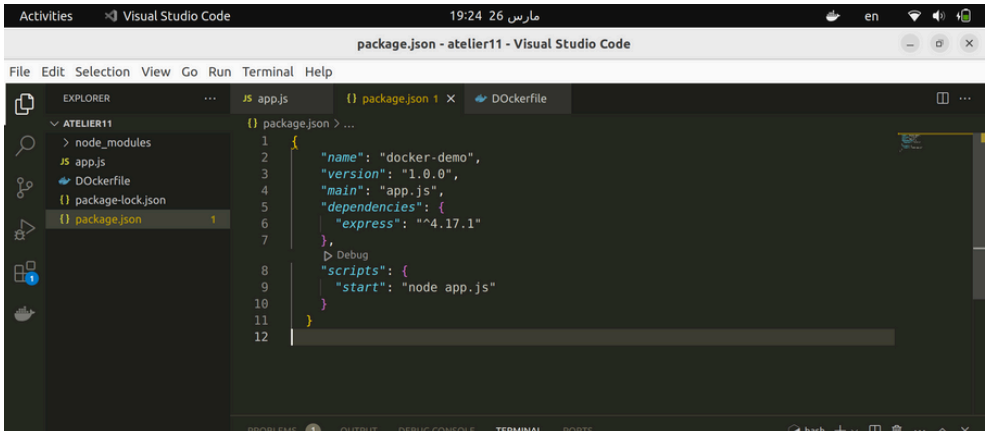
Année universitaire : 2024 - 2025

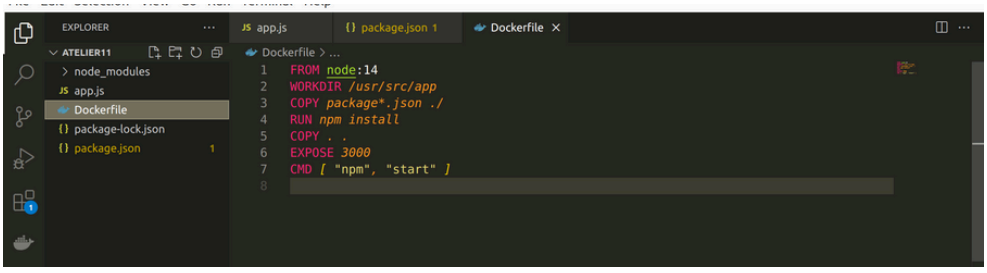
Atelier11 : Conteneurisation et Orchestration d'une Application Web

L'atelier sur la conteneurisation et l'orchestration d'une application web vise à introduire les concepts fondamentaux de la gestion des applications dans un environnement virtuel. Dans ce contexte, Docker et Kubernetes sont des technologies puissantes permettant de développer, déployer et gérer des applications de manière efficace et scalable. Ce TP est divisé en deux parties principales : dans la première, l'objectif est d'apprendre à conteneuriser une application web simple en utilisant Docker, tandis que la deuxième partie se concentre sur l'orchestration de cette application dans un environnement Kubernetes, en utilisant Minikube pour simuler un cluster local. Cette approche permet de maîtriser les bases du cycle de vie d'une application conteneurisée, de sa construction à son déploiement et son scaling dans un environnement de production.

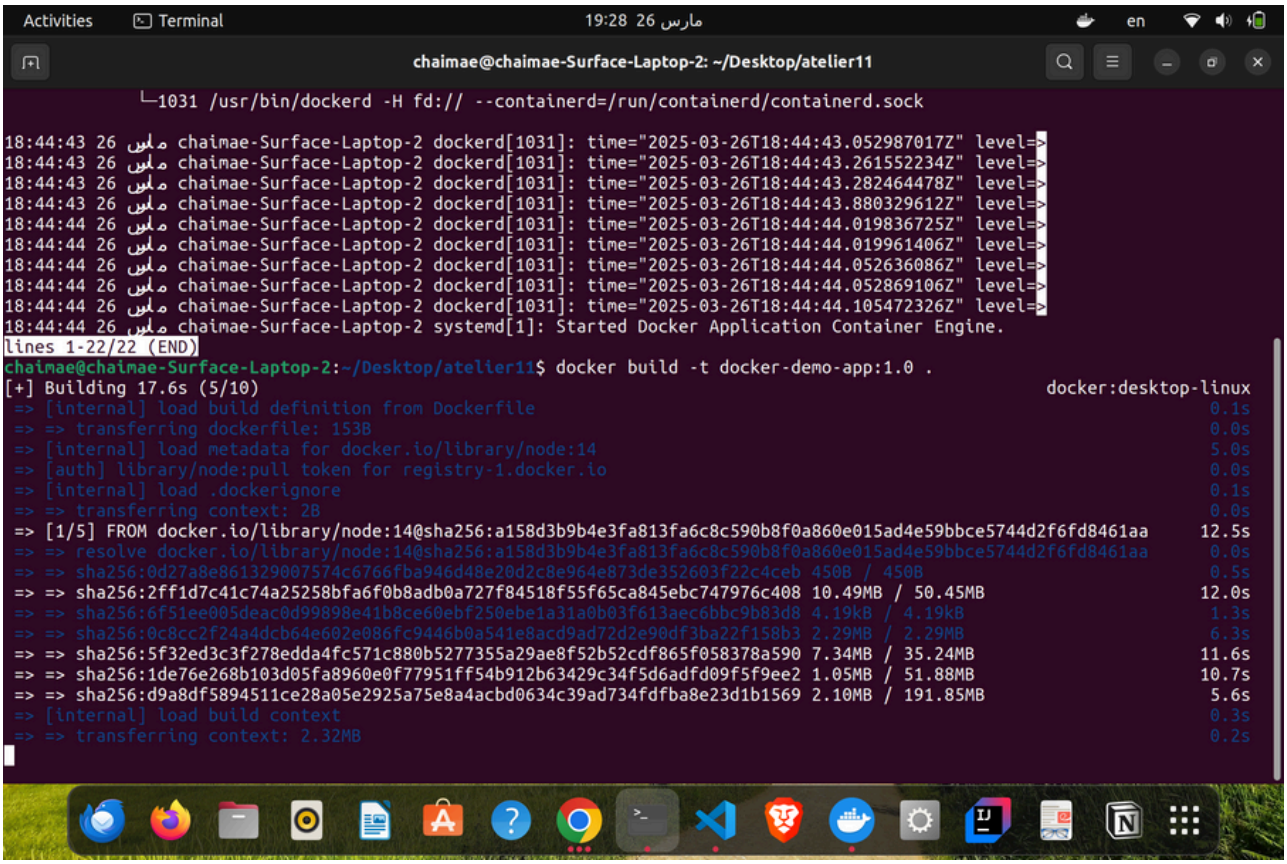



Activities
Visual Studio Code
19:24 26 مارس
en
app.js - atelier11 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
ATELIER11
node_modules
app.js
DOckerfile
package-lock.json
package.json 1
app.js
package.json 1
DOckerfile
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
res.send('Hello World!');
});
app.listen(port, () => {
console.log(`App running on port ${port}`);
});
Debug Console (Ctrl+Shift+Y)
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
bash
added 69 packages, and audited 70 packages in 5s
14 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
OUTLINE
TIMELINE
Ln 12, Col 1
Spaces: 4
UTF-8
LF
JavaScript

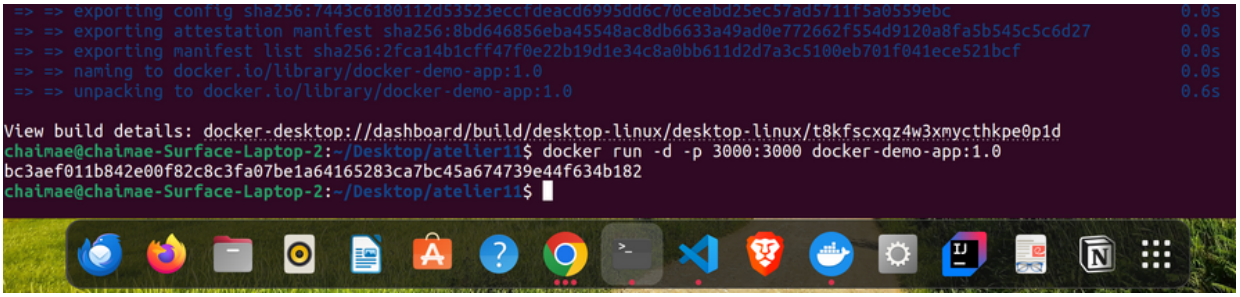

Activities
Visual Studio Code
19:24 26 مارس
en
package.json - atelier11 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
ATELIER11
node_modules
app.js
DOckerfile
package-lock.json
package.json 1
app.js
package.json 1
DOckerfile
{
"name": "docker-demo",
"version": "1.0.0",
"main": "app.js",
"dependencies": {
"express": "^4.17.1"
},
Debug
"scripts": {
"start": "node app.js"
}
}


EXPLORER
ATELIER11
node_modules
app.js
Dockerfile
package-lock.json
package.json 1
app.js
package.json 1
Dockerfile
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]

On construit l'image Docker en utilisant la commande suivante :

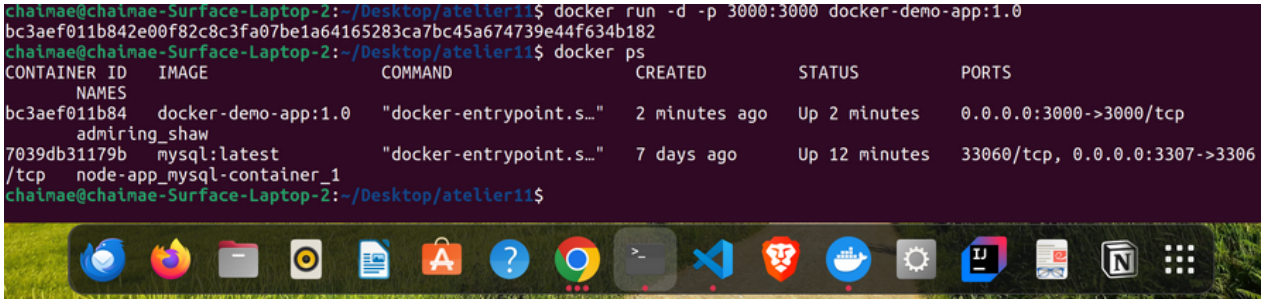


On execute l'application dans un conteneur Docker : avec la commande suivante pour démarrer le conteneur à partir de l'image que nous avons construite.

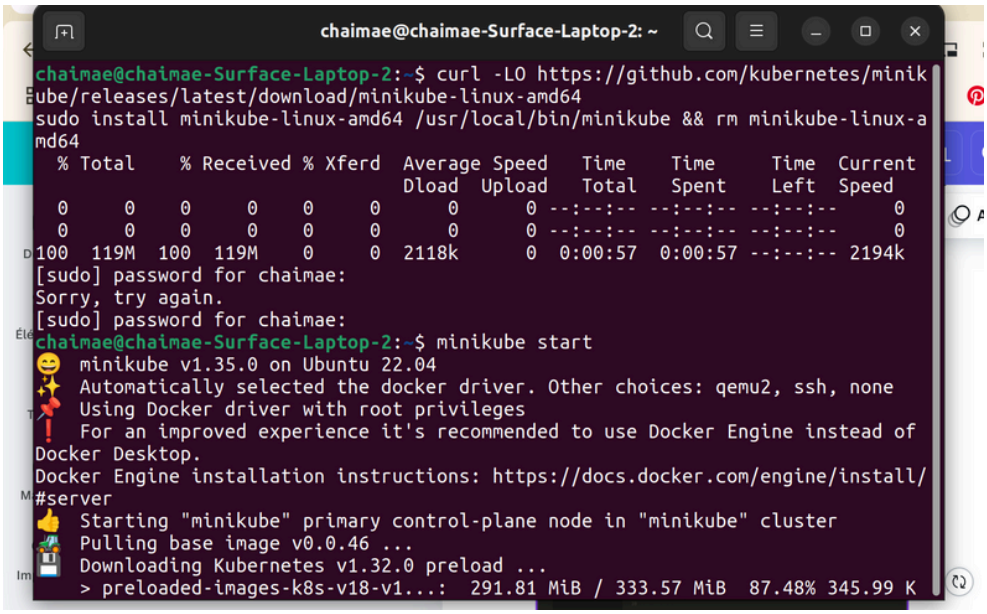


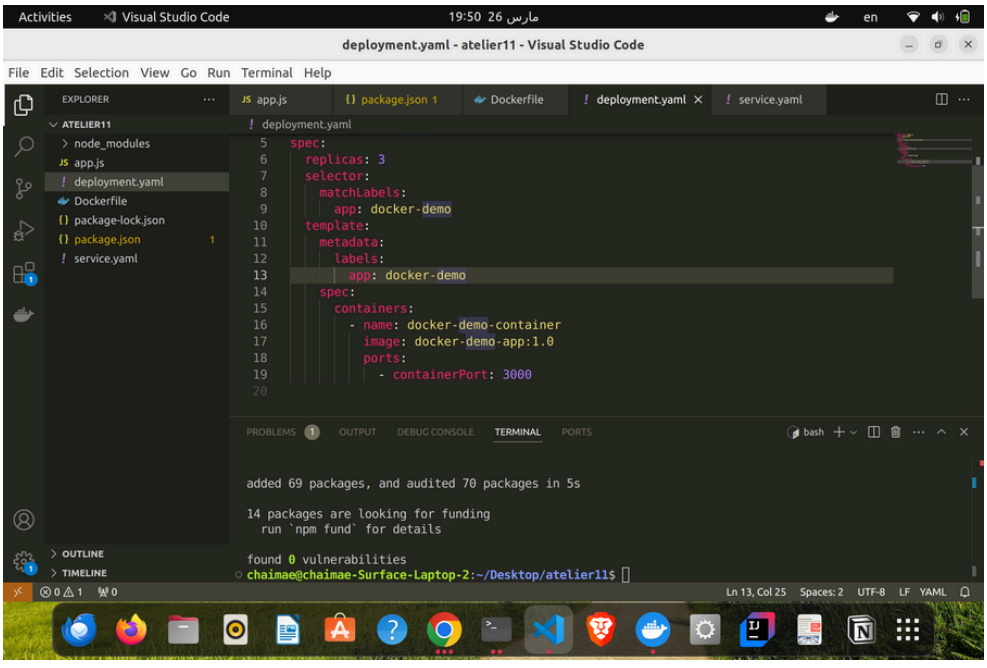
Étape 3 : Gestion des conteneurs Docker

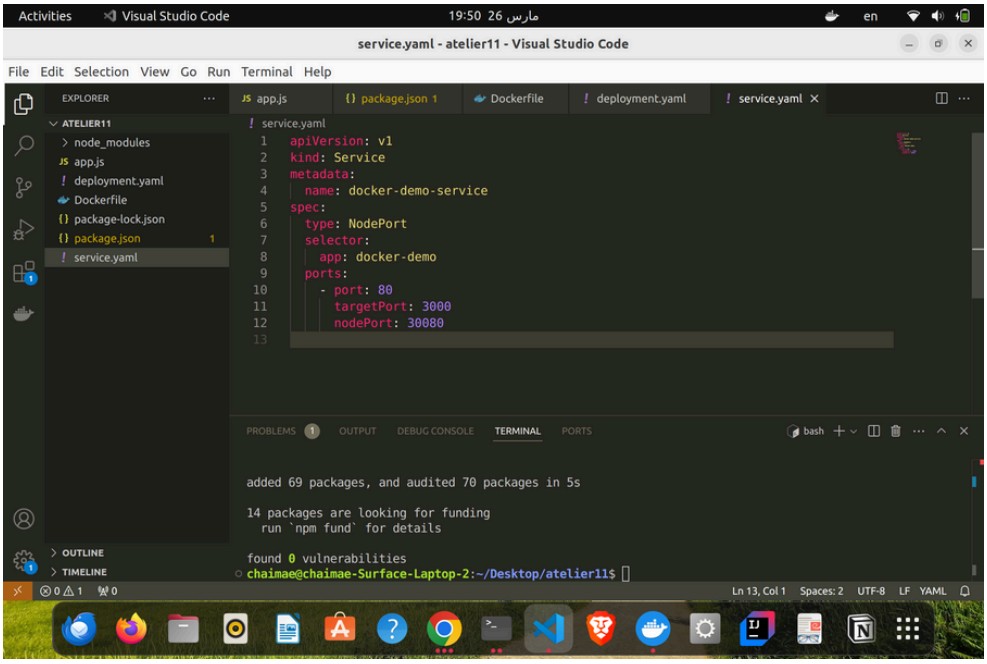
Liste des conteneurs en cours d'exécution :



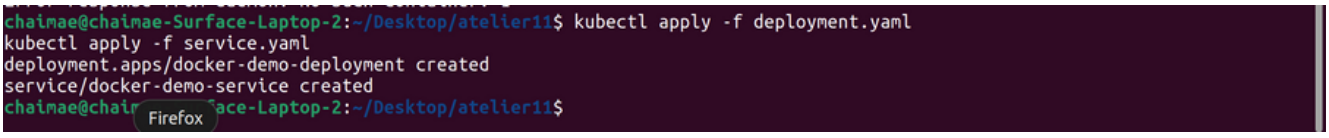
Partie 2 : Orchestration avec Kubernetes





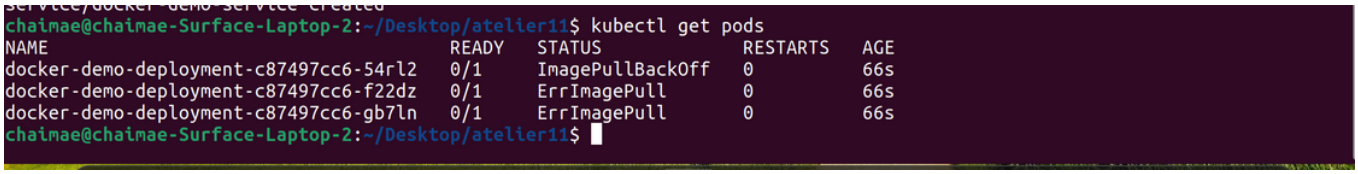


on applique les fichiers YAML pour déployer l'application dans Kubernetes :



Étape 3 : Vérifier le déploiement

Vérifier que les pods sont créés :



Vérifier le service et l'accès à l'application :

```
docker-demo-deployment-c87497cc6-gb7ln   0/1     ErrImagePull       0          66s
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl get svc
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
docker-demo-service    NodePort    10.110.218.154   <none>        80:30080/TCP   119s
kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP        6m10s
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```

Accéder à l'application sur Kubernetes : avec la commande suivante pour obtenir l'URL exposée par Minikube :

```
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ minikube service docker-demo-service
|-----------|---------------------|-------------|---------------------------|
| NAMESPACE |        NAME         | TARGET PORT |            URL            |
|-----------|---------------------|-------------|---------------------------|
| default   | docker-demo-service |          80 | http://192.168.49.2:30080 |
|-----------|---------------------|-------------|---------------------------|

X Exiting due to SVC_UNREACHABLE: service not available: no running pod for service docker-demo-service found


    If the above advice does not help, please let us know:
    https://github.com/kubernetes/minikube/issues/new/choose

    Please run `minikube logs --file=logs.txt` and attach logs.txt to the GitHub issue.
    Please also attach the following file to the GitHub issue:
    - /tmp/minikube_service_aa092421ac01c082671bcb1520a7a49347ac7c79_0.log
```

Étape 4 : Mise à l'échelle de l'application

Modifier le nombre de réplicas pour tester la mise à l'échelle :

```
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl scale deployment docker-demo-deployment --replicas=5
deployment.apps/docker-demo-deployment scaled
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl get pods
NAME                                     READY   STATUS             RESTARTS   AGE
docker-demo-deployment-c87497cc6-54rl2   0/1     ImagePullBackOff   0          5m19s
docker-demo-deployment-c87497cc6-dss9w   0/1     ImagePullBackOff   0          52s
docker-demo-deployment-c87497cc6-f22dz   0/1     ImagePullBackOff   0          5m19s
docker-demo-deployment-c87497cc6-gb7ln   0/1     ImagePullBackOff   0          5m19s
docker-demo-deployment-c87497cc6-xc49l   0/1     ErrImagePull       0          52s
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```

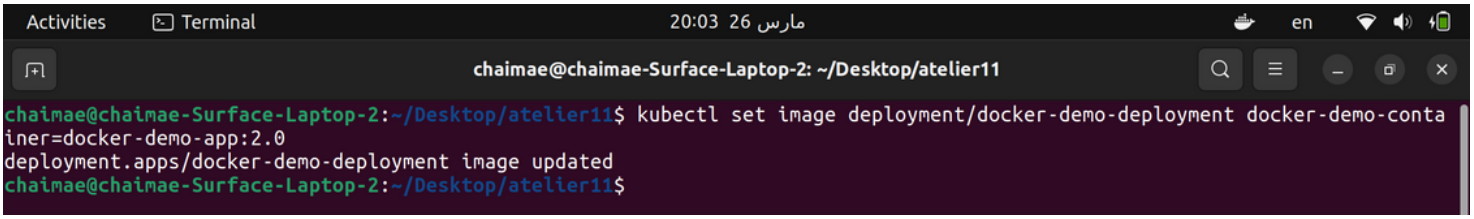
Vérifier que 5 pods sont maintenant en cours d'exécution :

```
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl scale deployment docker-demo-deployment --replicas=5
deployment.apps/docker-demo-deployment scaled
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl get pods
NAME                                     READY   STATUS             RESTARTS   AGE
docker-demo-deployment-c87497cc6-54rl2   0/1     ImagePullBackOff   0          5m19s
docker-demo-deployment-c87497cc6-dss9w   0/1     ImagePullBackOff   0          52s
docker-demo-deployment-c87497cc6-f22dz   0/1     ImagePullBackOff   0          5m19s
docker-demo-deployment-c87497cc6-gb7ln   0/1     ImagePullBackOff   0          5m19s
docker-demo-deployment-c87497cc6-xc49l   0/1     ErrImagePull       0          52s
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```

Partie 3 : Concepts avancés

Étape 1 : Mise à jour sans interruption (Rolling Update)

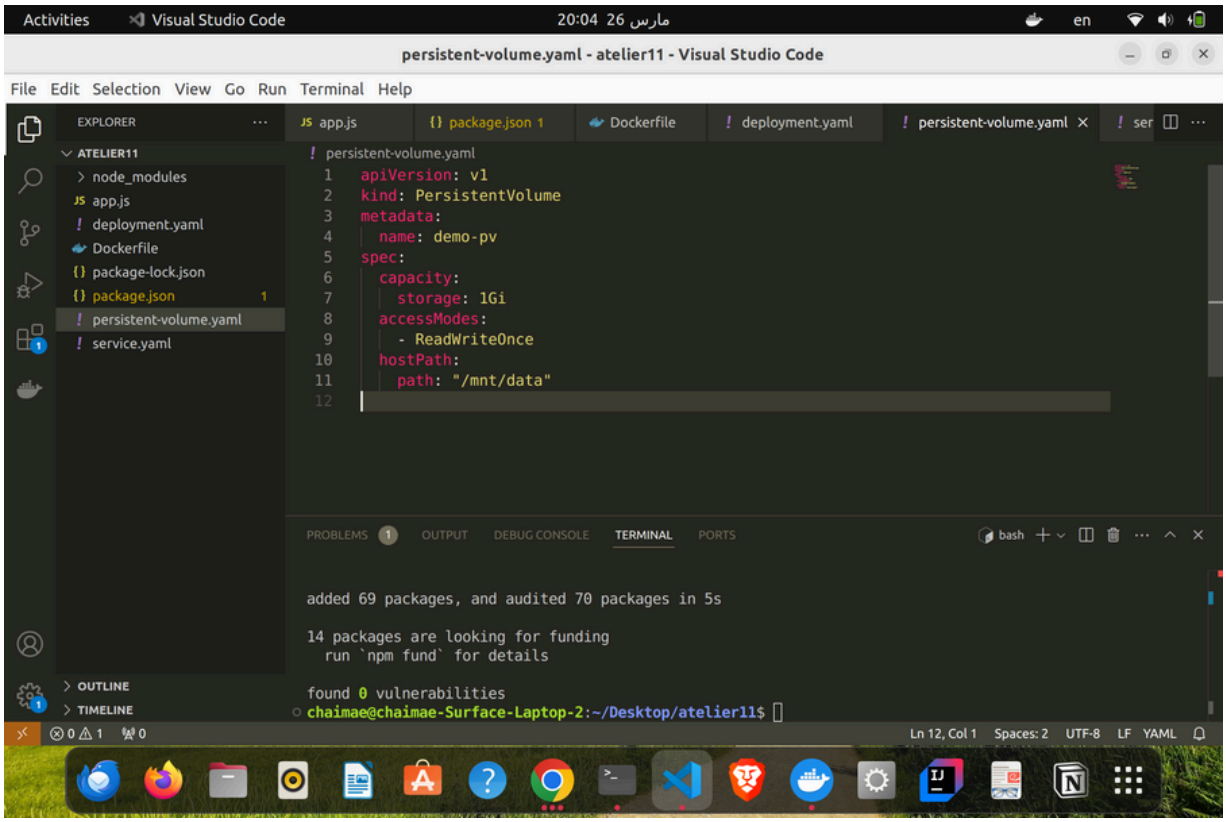
1. **Mettre à jour l'application sans interruption :**



Étape 2 : Volumes persistants

Ajouter un volume persistant pour stocker les données :

Créez un fichier YAML pour un volume persistant (par exemple, persistent-volume.yaml) :

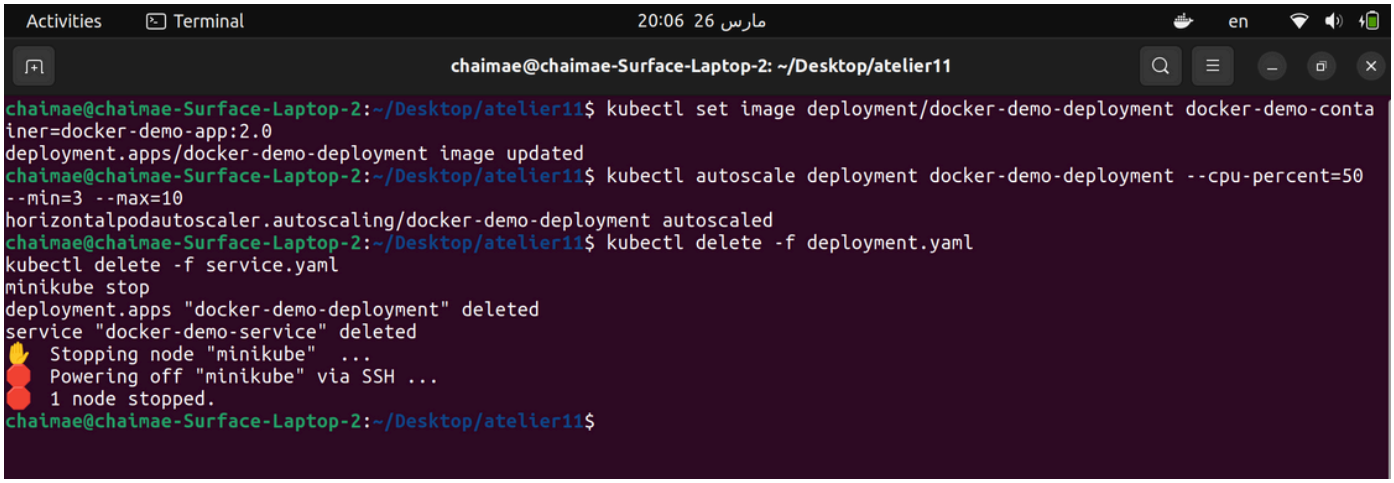


Étape 3 : Auto-scaling Horizontal

Configurer l'auto-scaling basé sur l'utilisation des ressources :

Étape 4 : Nettoyage

Supprimer les déploiements et services :



Conclusion

Cet atelier a permis d'acquérir des compétences essentielles dans la conteneurisation et l'orchestration des applications avec Docker et Kubernetes. En créant une application web simple en Node.js et en la déployant dans un cluster Kubernetes local via Minikube, les concepts clés de Docker, tels que la création d'images et la gestion des conteneurs, ont été explorés. De plus, l'utilisation de Kubernetes a permis d'implémenter des fonctionnalités avancées comme la mise à l'échelle et l'auto-scaling, essentielles pour le déploiement d'applications robustes. Cet atelier met en évidence l'importance de ces outils dans le développement moderne et la gestion efficace des applications, en offrant une meilleure flexibilité, scalabilité et résilience dans les environnements de production.