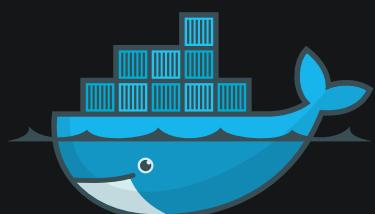




# Atelier11 : Conteneurisation et Orchestration d'une Application Web

Module : Virtualisation

Master M1 : Sécurité IT et Big Data



Préparé par :

Bouassab Chaimae

Encadré par :

Pr Mohamed BEN AHMED

### 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 مارس 26 - atelier11 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ATELIER11 node\_modules app.js Dockerfile package-lock.json package.json

app.js > ...

```
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!');
7 });
8
9 app.listen(port, () => {
10   console.log(`App running on port ${port}`);
11});
```

Debug Console (Ctrl+Shift+Y)

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

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\$

Ln 12, Col 1 Spaces: 4 UTF-8 LF {} JavaScript

This screenshot shows the Visual Studio Code interface with the 'ATELIER11' workspace open. The 'app.js' file is selected in the Explorer sidebar. The code defines a simple Express application that listens on port 3000 and responds with 'Hello World!'. Below the code editor, the terminal window shows the results of an npm install command, indicating 69 packages added and 70 packages audited in 5 seconds. It also shows 14 packages looking for funding and 0 vulnerabilities found.

Activities Visual Studio Code 19:24 مارس 26 - atelier11 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ATELIER11 node\_modules app.js Dockerfile package-lock.json package.json

package.json > ...

```
1 {
2   "name": "docker-demo",
3   "version": "1.0.0",
4   "main": "app.js",
5   "dependencies": {
6     "express": "4.17.1"
7   },
8   "scripts": {
9     "start": "node app.js"
10   }
11}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

This screenshot shows the Visual Studio Code interface with the 'ATELIER11' workspace open. The 'package.json' file is selected in the Explorer sidebar. The JSON object defines a project named 'docker-demo' with version '1.0.0', main file 'app.js', and dependency 'express' at version '4.17.1'. It also includes a script named 'start' that runs 'node app.js'.

Activities Visual Studio Code 19:24 مارس 26 - atelier11 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

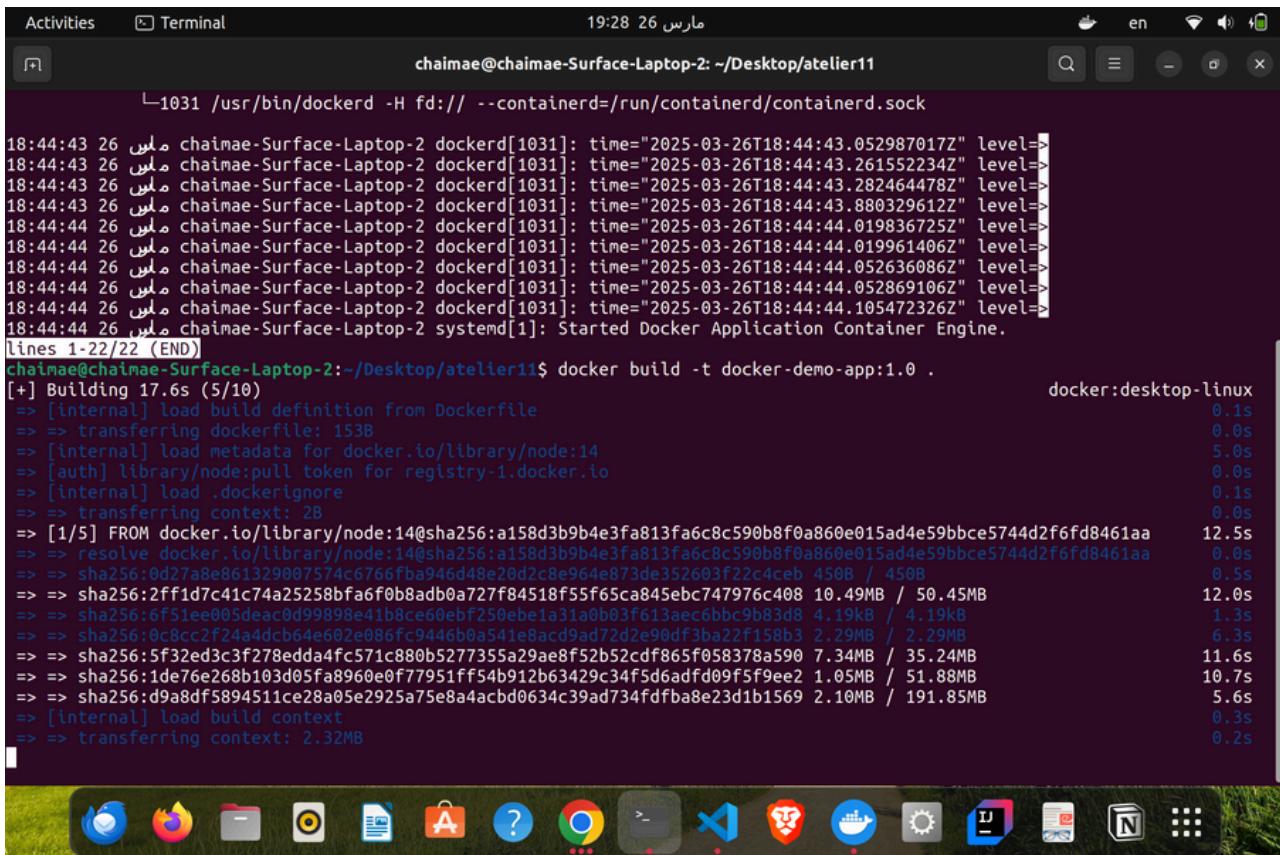
EXPLORER ATELIER11 node\_modules app.js Dockerfile package-lock.json package.json

Dockerfile > ...

```
1 FROM node:14
2 WORKDIR /usr/src/app
3 COPY package*.json .
4 RUN npm install
5 COPY .
6 EXPOSE 3000
7 CMD [ "npm", "start" ]
```

This screenshot shows the Visual Studio Code interface with the 'ATELIER11' workspace open. The 'Dockerfile' is selected in the Explorer sidebar. The Dockerfile uses the 'node:14' base image, sets the working directory to '/usr/src/app', copies the package files, installs dependencies, copies the local files, exposes port 3000, and starts the application with npm start.

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

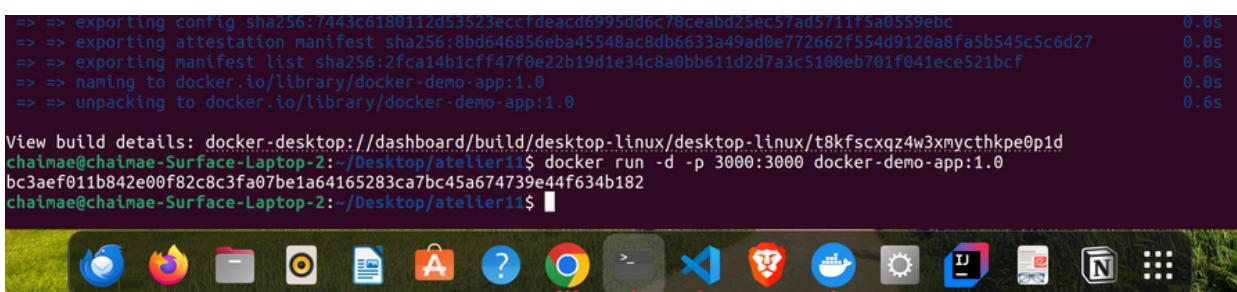


```
Activities Terminal 19:28 26 مارس chaimae@chaimae-Surface-Laptop-2: ~/Desktop/atelier11
└─1031 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

18:44:43 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:43.052987017Z" level=>
18:44:43 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:43.261552234Z" level=>
18:44:43 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:43.282464478Z" level=>
18:44:43 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:43.880329612Z" level=>
18:44:44 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:44.019836725Z" level=>
18:44:44 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:44.019961406Z" level=>
18:44:44 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:44.052636086Z" level=>
18:44:44 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:44.052869106Z" level=>
18:44:44 26 ملیم chaimae-Surface-Laptop-2 dockerd[1031]: time="2025-03-26T18:44:44.105472326Z" level=>
18:44:44 26 ملیم chaimae-Surface-Laptop-2 systemd[1]: Started Docker Application Container Engine.
lines 1-22/22 (END)

chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ docker build -t docker-demo-app:1.0 .
[+] Building 17.6s (5/10)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 153B
=> [internal] load metadata for docker.io/library/node:14
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerrignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa 12.5s
=> => resolve docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa 0.0s
=> => sha256:0d27a8e8613290907574c6766fba946d48e20d2c8e964e873de352603f22c4ceb 450B / 450B 0.5s
=> => sha256:2ff1d7c41c74a25258bfaf6f0b8adb0a727f84518f55f65ca845ebc747976c408 10.49MB / 10.45MB 12.0s
=> => sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8 4.19KB / 4.19KB 1.3s
=> => sha256:0c8cc2f24a4dc64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3 2.29MB / 2.29MB 6.3s
=> => sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590 7.34MB / 35.24MB 11.6s
=> => sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 1.05MB / 51.88MB 10.7s
=> => sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 2.10MB / 191.85MB 5.6s
=> [internal] load build context
=> => transferring context: 2.32MB 0.3s
=> => 0.2s
```

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.



```
=> => exporting config sha256:7443c6180112d53523eccfdeacd6995dd6c70ceabd25ec57ad5711f5a0559ebc 0.0s
=> => exporting attestation manifest sha256:8bd646856eba45548ac8db6633a49ad0e772662f554d9120a8fa5b545c5c6d27 0.0s
=> => exporting manifest list sha256:2fc14b1cff47f0e22b19d1e34c8a0bb611d2d7a3c5100eb701f041ece521bcf 0.0s
=> => naming to docker.io/library/docker-demo-app:1.0 0.0s
=> => unpacking to docker.io/library/docker-demo-app:1.0 0.6s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/t8kfstcxqz4w3xmycthkpe0pid
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ docker run -d -p 3000:3000 docker-demo-app:1.0
bc3ae011b842e00f82c8c3fa07be1a64165283ca7bc45a674739e44f634b182
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```



# Étape 3 : Gestion des conteneurs Docker

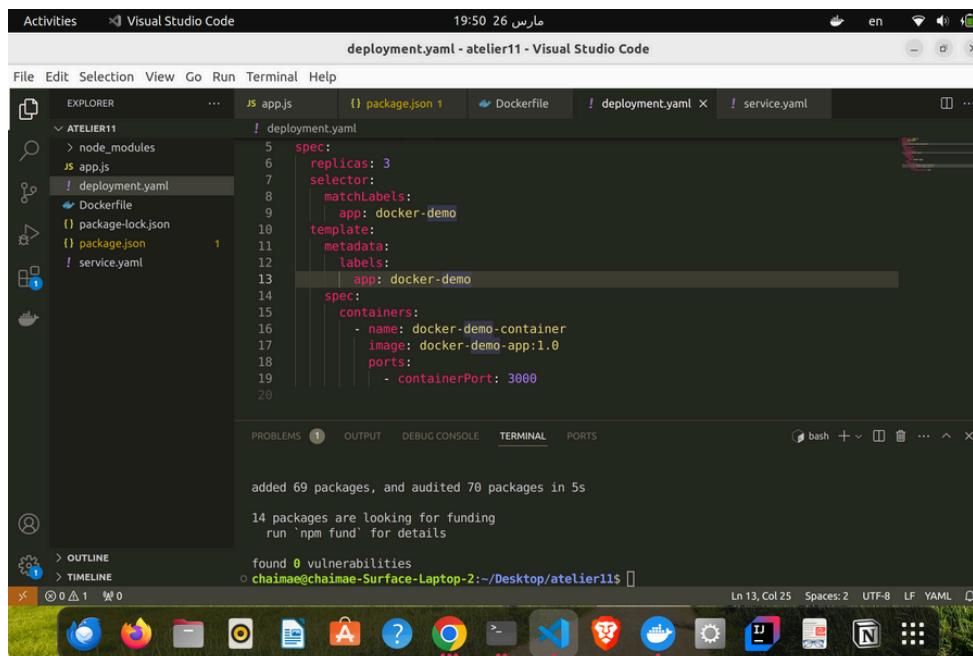
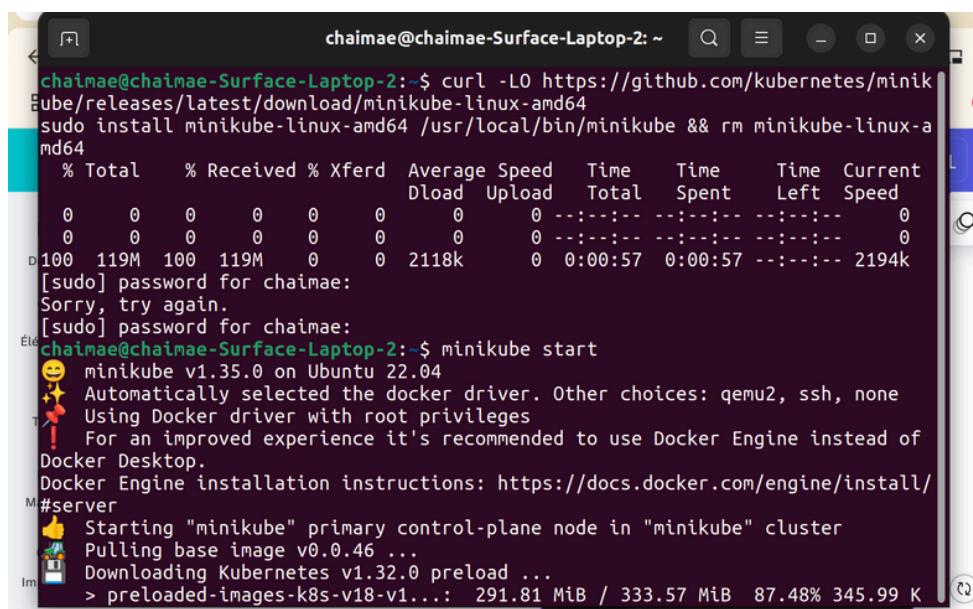
## Liste des conteneurs en cours d'exécution :

```
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ docker run -d -p 3000:3000 docker-demo-app:1.0
bc3aef011b842e00f82c8c3fa07be1a64165283ca7bc45a674739e44f634b182
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS
 NAMES
bc3aef011b84   docker-demo-app:1.0   "docker-entrypoint.s..."   2 minutes ago   Up 2 minutes   0.0.0.0:3000->3000/tcp
admiring_shaw  mysql:latest    "docker-entrypoint.s..."   7 days ago    Up 12 minutes  33060/tcp, 0.0.0.0:3307->3306
/tcp           node-app_mysql-container_1
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```



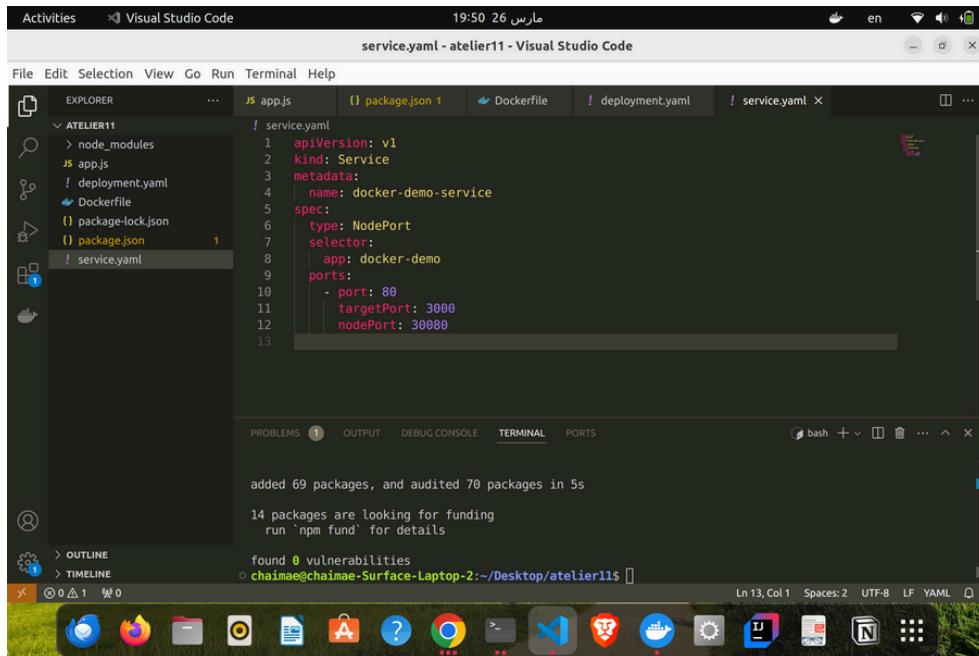
## Partie 2 : Orchestration avec Kubernetes

```
chaimae@chaimae-Surface-Laptop-2: ~
chaimae@chaimae-Surface-Laptop-2: $ curl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
% Total    % Received % Xferd  Average Speed   Time     Time     Current
          Dload Upload Total   Spent   Left Speed
0          0     0     0     0     0     0 --:--:-- --:--:-- --:--:-- 0
0          0     0     0     0     0     0 --:--:-- --:--:-- --:--:-- 0
D 100  119M  100  119M  0     0  2118k  0  0:00:57  0:00:57 --:--:-- 2194k
[sudo] password for chaimae:
Sorry, try again.
[sudo] password for chaimae:
chaimae@chaimae-Surface-Laptop-2: $ minikube start
minikube v1.35.0 on Ubuntu 22.04
Automatically selected the docker driver. Other choices: qemu2, ssh, none
Using Docker driver with root privileges
For an improved experience it's recommended to use Docker Engine instead of Docker Desktop.
Docker Engine installation instructions: https://docs.docker.com/engine/install/
#server
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Downloading Kubernetes v1.32.0 preload ...
> preloaded-images-k8s-v18-v1...: 291.81 MiB / 333.57 MiB 87.48% 345.99 K
```



```
deployment.yaml - atelier11 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
File Explorer  ... JS app.js ! package.json  Dockerfile ! deployment.yaml  ! service.yaml  ...
ATELIER11
> node_modules
JS app.js
! deployment.yaml
Dockerfile
! package-lock.json
! package.json
! service.yaml
5 spec:
6   replicas: 3
7   selector:
8     matchLabels:
9       app: docker-demo
10  template:
11    metadata:
12      labels:
13        app: docker-demo
14  spec:
15    containers:
16      - name: docker-demo-container
17        image: docker-demo-app:1.0
18        ports:
19          - containerPort: 3000
20
PROBLEMS  OUTPUT DEBUG CONSOLE TERMINAL PORTS
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$
```





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

```
sudo snap install kubectl
chaimae@chaimae-Surface-Laptop-2:~$ kubectl apply -f deployment.yaml
Command 'kubectl' not found, but can be installed with:
sudo snap install kubectl
chaimae@chaimae-Surface-Laptop-2:~$ sudo snap install kubectl
error: This revision of snap "kubectl" was published using classic confinement
and thus may perform arbitrary system changes outside of the security
sandbox that snaps are usually confined to, which may put your system at
risk.

If you understand and want to proceed repeat the command including
--classic.
chaimae@chaimae-Surface-Laptop-2:~$ sudo snap install kubectl --classic
kubectl 1.32.3 from Canonical✓ installed
chaimae@chaimae-Surface-Laptop-2:~$
```

```
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl apply -f deployment.yaml
deployment.apps/docker-demo-deployment created
service/docker-demo-service created
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```

### Étape 3 : Vérifier le déploiement

Vérifier que les pods sont créés :

```
service/docker-demo-service created
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
docker-demo-deployment-c87497cc6-54rl2  0/1     ImagePullBackOff  0          66s
docker-demo-deployment-c87497cc6-f22dz  0/1     ErrImagePull    0          66s
docker-demo-deployment-c87497cc6-gb7ln  0/1     ErrImagePull    0          66s
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```

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

```
docker demo deployment c87497cc6 7/2202 0/1 ErrImagePull 0 66s
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 :**

NAMESPACE	NAME	TARGET PORT	URL
default	docker-demo-service	80	http://192.168.49.2:30080

✖ 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

hai Mae@hai Mae-Surface-Laptop-2: ~/Desktop/atelier11c

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

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

```
chainae@chainae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl scale deployment docker-demo-deployment --replicas=5
deployment.apps/docker-demo-deployment scaled
chainae@chainae-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
chainae@chainae-Surface-Laptop-2:~/Desktop/atelier11$
```

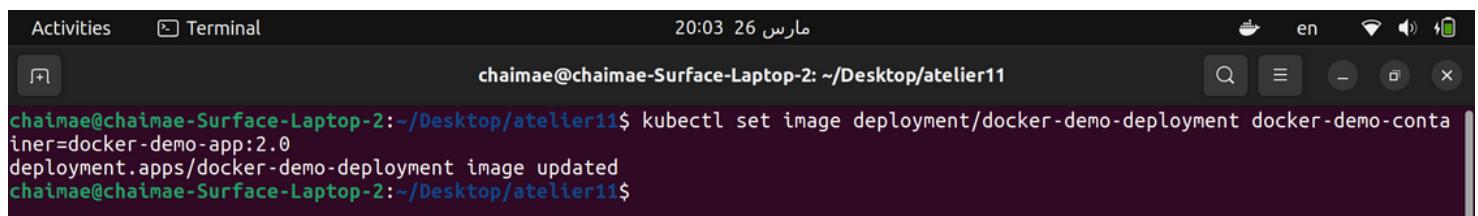
# Vérifier que 5 pods sont maintenant en cours d'exécution :

```
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 :

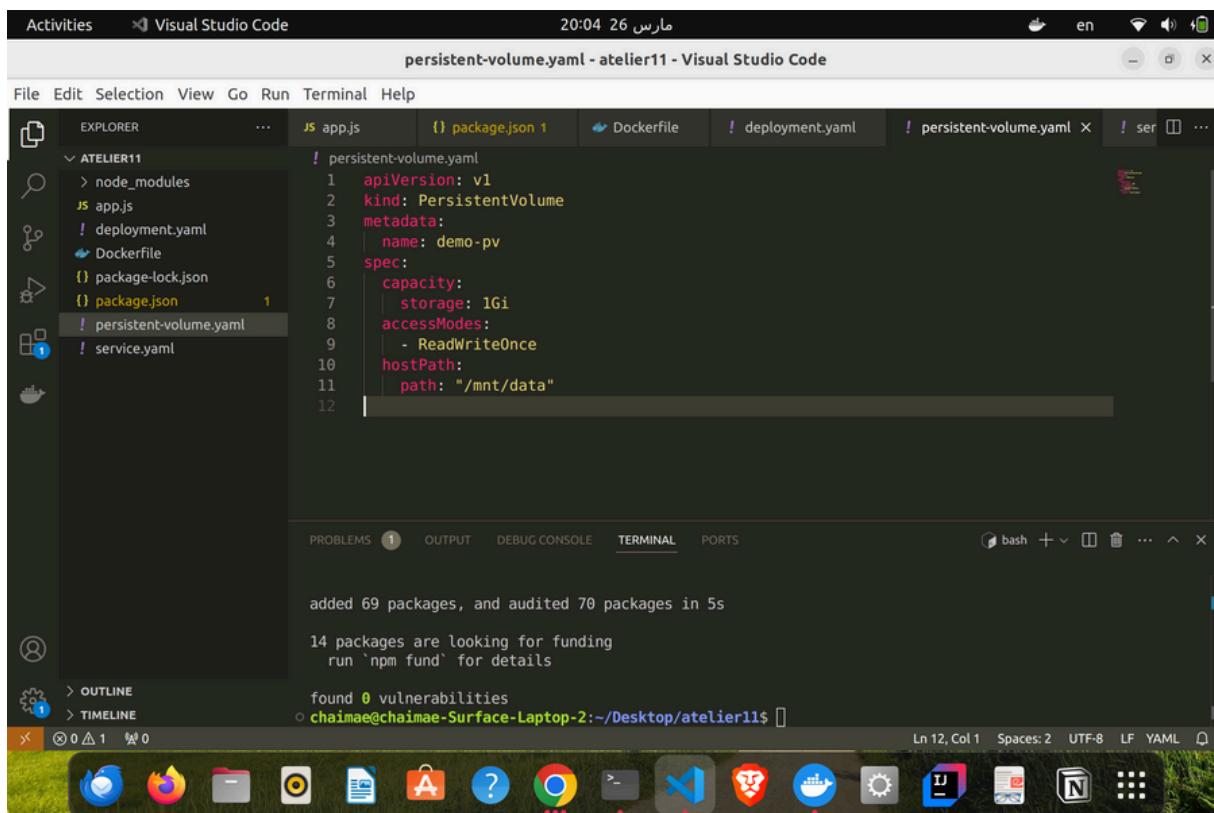


```
Activities Terminal 20:03 26 مارس
chaimae@chaimae-Surface-Laptop-2: ~/Desktop/atelier11$ kubectl set image deployment/docker-demo-deployment docker-demo-conta iner=docker-demo-app:2.0
deployment.apps/docker-demo-deployment image updated
chaimae@chaimae-Surface-Laptop-2: ~/Desktop/atelier11$
```

### 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) :



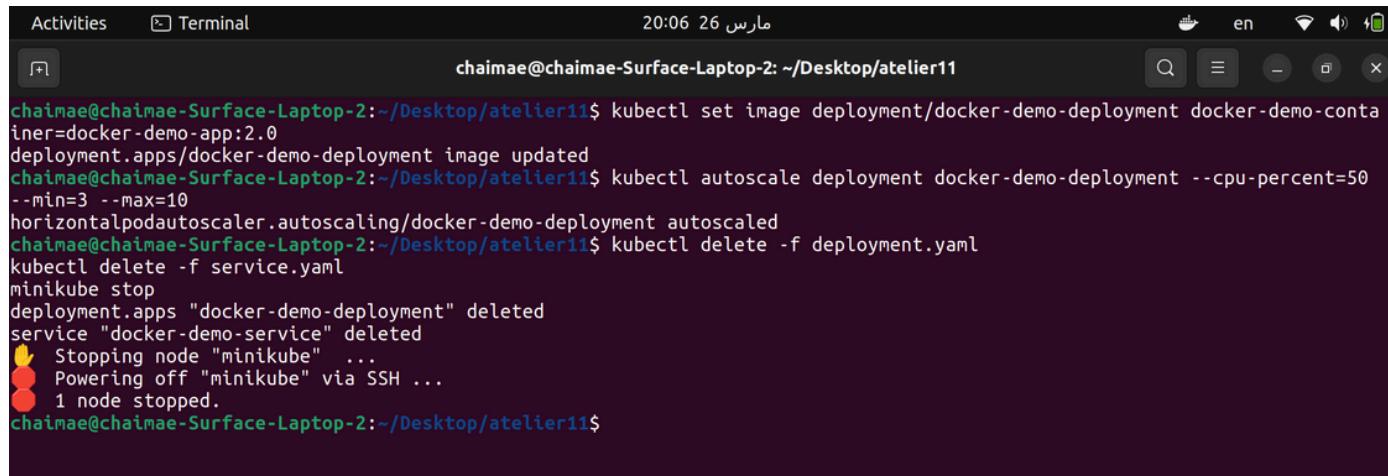
```
! persistent-volume.yaml
1 apiVersion: v1
2 kind: PersistentVolume
3 metadata:
4   name: demo-pv
5 spec:
6   capacity:
7     storage: 1Gi
8   accessModes:
9     - ReadWriteOnce
10  hostPath:
11    path: "/mnt/data"
```

## Étape 3 : Auto-scaling Horizontal

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

## Étape 4 : Nettoyage

Supprimer les déploiements et services :



The screenshot shows a terminal window on a Mac OS X desktop. The title bar says "Activities Terminal". The status bar at the top right shows "مارس 26 20:06 en". The terminal window has a dark background with white text. It displays the following command-line session:

```
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl set image deployment/docker-demo-deployment docker-demo-conta
iner=docker-demo-app:2.0
deployment.apps/docker-demo-deployment image updated
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl autoscale deployment docker-demo-deployment --cpu-percent=50
--min=3 --max=10
horizontalpodautoscaler.autoscaling/docker-demo-deployment autoscaled
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$ kubectl delete -f deployment.yaml
kubectl delete -f service.yaml
minikube stop
deployment.apps "docker-demo-deployment" deleted
service "docker-demo-service" deleted
Stopping node "minikube" ...
Powering off "minikube" via SSH ...
1 node stopped.
chaimae@chaimae-Surface-Laptop-2:~/Desktop/atelier11$
```

## 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.

