Communication Anglaise

Table des matières

Course 1

Mission 1

Step 1 :First, I install Docker Engine on Debian

I update the apt package index and install packages.

I have download the lastest and I drop the file in my virtual debian.

Step 2 : We will need git to pull following repository from github. Now, we can run the Docker desktop.

Step 3: Clone the repository github.

Step 4: Modify the file Dockerfile to install A2SR

```
  GNU nano 5.4                    Dockerfile
# Complete the file :)

# Description : From used to creat a base image
FROM node:latest

# Explain : It is a command to used and execute the new image
RUN apk add --no-cache python2 g++ make

# Explain : It is the working directory
WORKDIR /app

# Description :copy files and the directories to the container
COPY . .

# Explain : He install dependencies for the project
RUN yarn install --production

# Explain : He execute the file index.js to install the option
CMD ["node", "src/index.js"]

                          [ 19 lignes écrites ]
^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich.^\ Remplacer  ^U Coller    ^J Justifier ^_ Aller ligne
```

He build the project and run the project :

- Docker build -t nodesaxelgauvrit .

-docker run -p 127.0.0.:3000:3000 nodesaxelgauvrit

Docker is very interest to create just the application. I create a contenair and isole all contenair. We don't have need more performance to launch the application. He is very practice to less the storage because we just install the service.

Mission 2

Make a modification in : src/static/js/app.js

```
return (
    <Form onSubmit={submitNewItem}>
        <InputGroup className="mb-3">
            <Form.Control
                value={newItem}
                onChange={e => setNewItem(e.target.value)}
                type="text"
                placeholder="New car"
                aria-describedby="basic-addon1"
            />
            <InputGroup.Append>
                <Button
                    type="submit"
                    variant="success"
                    disabled={!newItem.length}
                    className={submitting ? 'disabled' : ''}
                >
                    {submitting ? 'Adding...' : 'Add car'}
                </Button>
            </InputGroup.Append>
        </InputGroup>
    </Form>
);
```

Build and update the container

Docker build -t nodesaxelgauvritv2 .

docker run -p 127.0.0.1 :3000:3000 nodesaxelgauvritv2

Create an account and a public registry

gnp –generate-key

pass init CBC69578943CCCD8F8265C972015F3A137BE43C6

choose a pass phrase : 1234567890

Push your image

docker login

username

password

docker tag nodesaxelgauvritv2 axelgauvrit/nodesaxelgauvritv2

docker push axelgauvrit/nodesaxelgauvritv2

```
root@master:/home/administrateur# docker image push axelgauvrit/nodesaxelgauvritv2
Using default tag: latest
The push refers to repository [docker.io/axelgauvrit/nodesaxelgauvritv2]
0e7b459e0204: Pushed
310b37a47944: Pushed
328c700e9b13: Pushed
434f1fa5863b: Pushed
3cd484903201: Pushed
f98e42232d41: Pushed
3658e8b65f49: Pushed
52fb12b3f764: Pushed
0dafd2b156dd: Pushed
152e2eadff76: Pushed
738226f36892: Pushed
3fcfc59d80ac: Pushed
latest: digest: sha256:8b0d72f47ac051cc1b92e95f4ce62cabd218a1362ece47ada5dfd4c09b7908ed size: 2845
```

docker pull wolfanto/todolist_a2sr_v2

The download finish i launch the new docker image by Antonin.

Mission 3

The mission 3 is to save all the task whe you close the container and share all the projet to the gitHub.

I have a problem with the connection for the git, to resolve the problem i create a token in my github.

```
root@master:/home/administrateur/A2SR/Course_1# git push -u origin main
Username for 'https://github.com': axelgauvrit
Password for 'https://axelgauvrit@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Échec d'authentification pour 'https://github.com/axelgauvrit/A2SR.git/'
```

```
root@master:/home/administrateur/A2SR/Course_1# git push -u origin main
Username for 'https://github.com': axelgauvrit
Password for 'https://axelgauvrit@github.com':
Énumération des objets: 55, fait.
Décompte des objets: 100% (55/55), fait.
Compression par delta en utilisant jusqu'à 3 fils d'exécution
Compression des objets: 100% (54/54), fait.
Écriture des objets: 100% (55/55), 4.41 Mio | 5.49 Mio/s, fait.
Total 55 (delta 3), réutilisés 0 (delta 0), réutilisés du pack 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/axelgauvrit/A2SR.git
 * [new branch]      main -> main
La branche 'main' est paramétrée pour suivre la branche distante 'main' depuis 'origin'.
```

Conclusion

So for the first cours, I devellop my skills for the Git and Docker. Git is a web iste to share all the code for all people and generate a new version will push a new code. Docker is plateform to use just application, no the all environnement of the operating system. I practice to install Docker and run a little application. I modified the code to personalize for me. And for the finish i share my code.

Course 2

Mission 1

We have installed Docker Desktop for Windows to use kubernetes.

We download the programm Docker in official web site of Docker. To active kubernetes, we go to the settings and check the activation of kubernetes

After we installed Kubernetes dashboard with powershell. It is to have a graphical interface.

Winget install -e –id kubernetes.kubectl

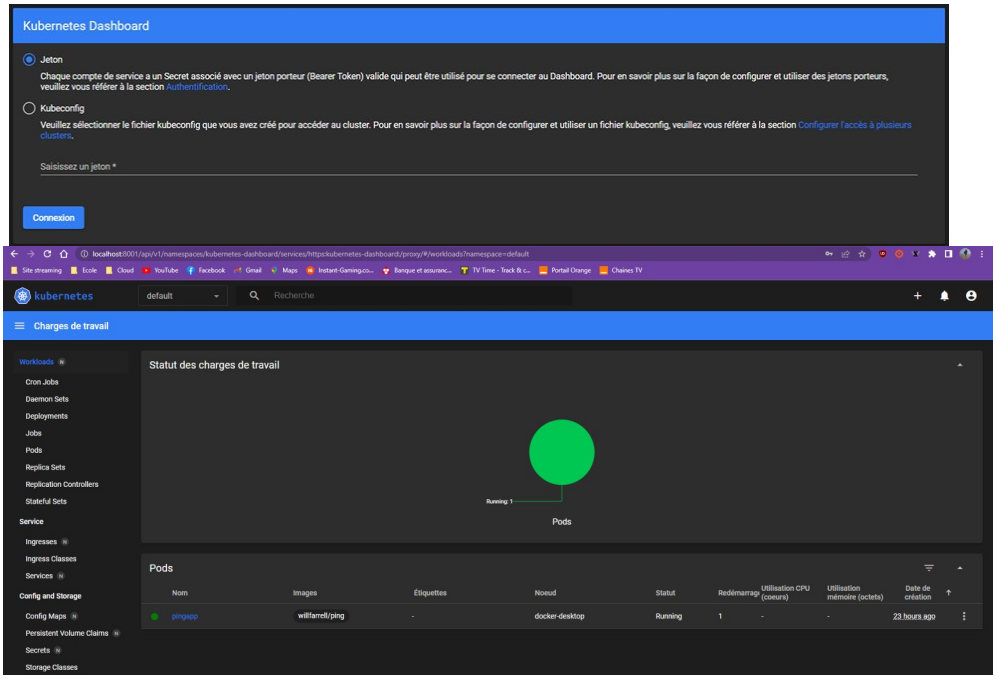
I using kubectl, we deployed kubernetes dashboard via a YAML file on the official kubernetes github

After the finish of the installation. We launch the kubectl. To run this we create a « Service Account » and a « ClusterRoleBindining ». After these steps, we generated the token authenticate. We copy the jeton launch with the command generate the token. Launch with kubectl proxy

kubectl -n kubernetes-dashboard create token admin-user

He give the token to connect of the interface.

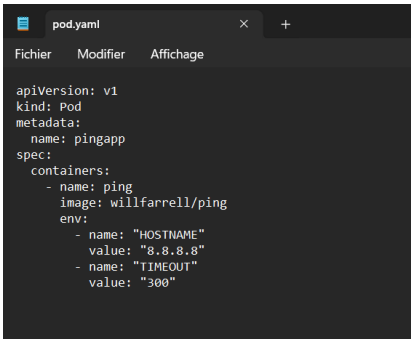
http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

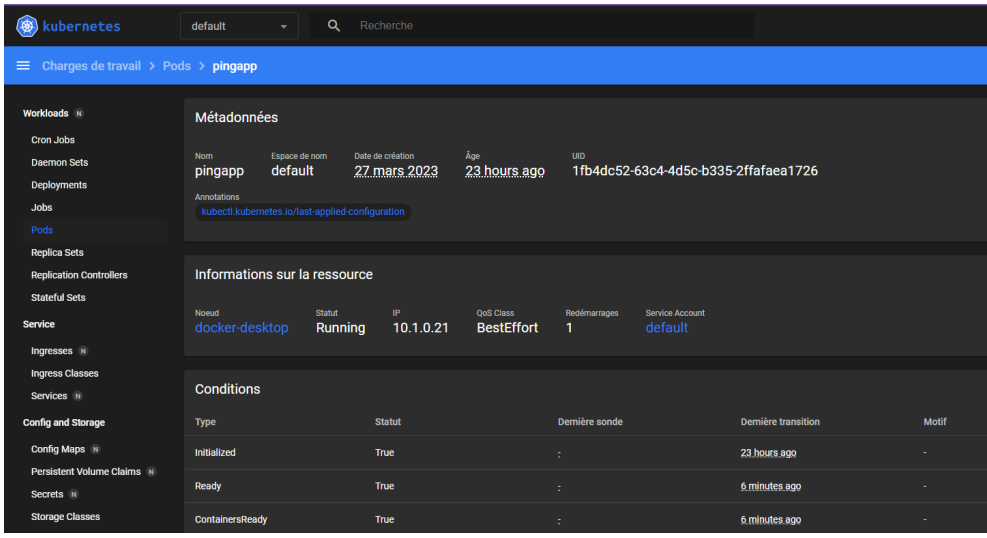


Mission 2

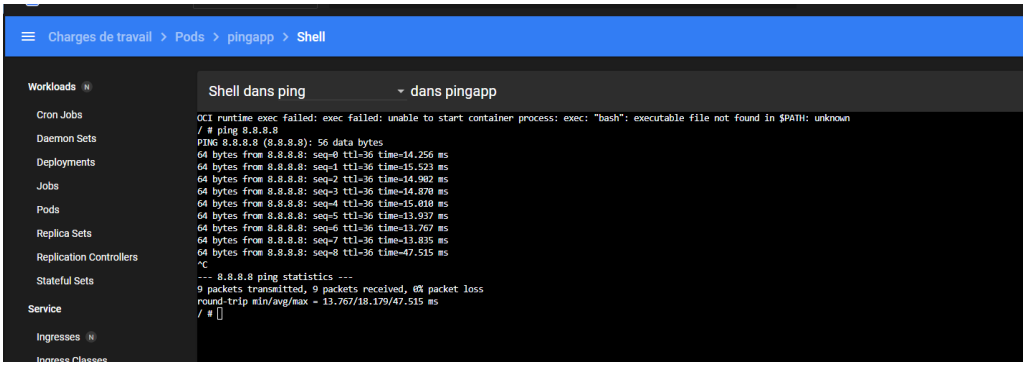
We updated the A2SR github projet with the « git pull » to add all the file for the mission 1.

We have a problem of the YAML so we change line to correct

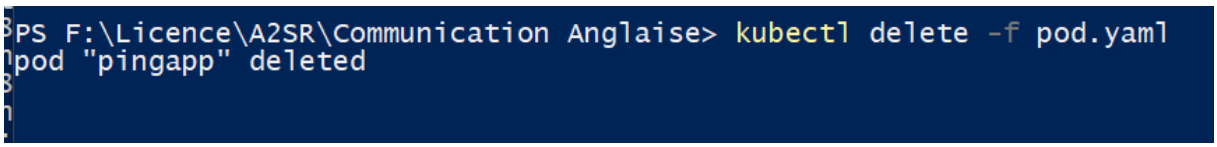




Now we test the pod on kubernetes.



After verifying the pod we go delete the pod .



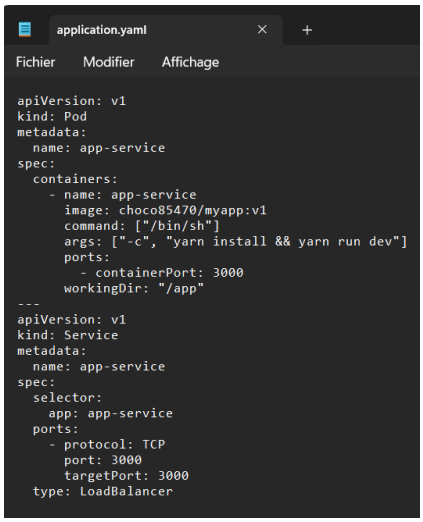
Conclusion

For the conclusion of this course N°2, we learned how to use docker and kubernetes on windows. We also installed a dashboard for Kubernetes. We leaned how to use Kubernetes and the dashboard to create, modify, and delete a Pod. We have push all code on the github.

Course 3

Mission 1

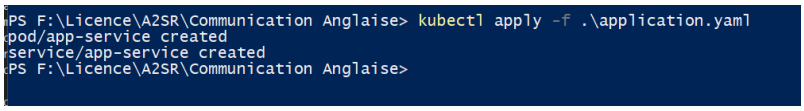
We created a Pod and its service in YAML file, after we apply in the kubectl utility.



application.yaml

Fichier Modifier Affichage

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app-service
spec:
  containers:
    - name: app-service
      image: choco85470/myapp:v1
      command: ["/bin/sh"]
      args: ["-c", "yarn install && yarn run dev"]
      ports:
        - containerPort: 3000
      workingDir: "/app"
---
apiVersion: v1
kind: Service
metadata:
  name: app-service
spec:
  selector:
    app: app-service
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: LoadBalancer
```



```
PS F:\Licence\A2SR\Communication Anglaise> kubectl apply -f .\application.yaml
pod/app-service created
service/app-service created
PS F:\Licence\A2SR\Communication Anglaise>
```

We merge two file the pod and the service.

We have a lot of parameters to launch the project. The teacher give this project in last course. The goal si to launch a web site. To connect of the web site we open ports with the kubectl utility.

apiVersion: The version of the Kubernetes API to use.

kind: The type of object being defined.

metadata: Information about the Pod, such as its name.

spec: The specifications for the Pod, including which container image to use, any environment variables, and which ports to expose.

The Service object describes how traffic will be routed to the Pod. The parameters of this object are:

apiVersion: The version of the Kubernetes API to use.

kind: The type of object being defined.

metadata: Information about the Service, such as its name.

spec: The specifications for the Service, including which Pod(s) to target, which ports to listen on, and which type of Service to use (in this case, a LoadBalancer).

The goal of the command

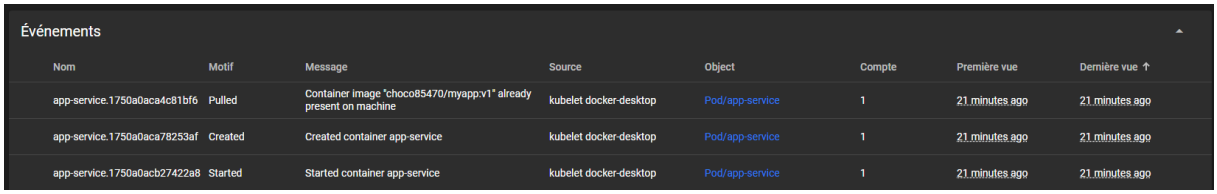
kubectl: The Kubernetes command-line tool used to interact with the Kubernetes API.

port-forward: The sub-command used to forward traffic from a local port to a port on a Kubernetes Service or Pod.

service/app-service: The name of the Kubernetes Service to forward traffic to.

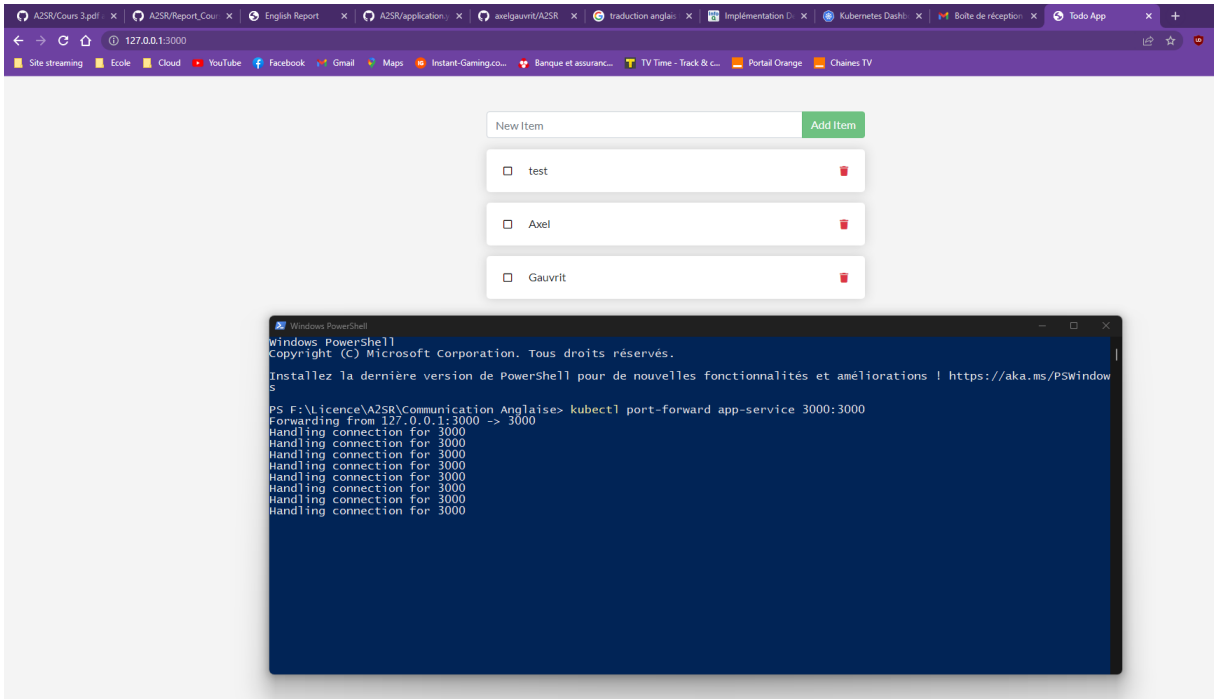
3000:3000: The local port (3000) to forward traffic from and the target port (also 3000) on the Kubernetes Service to forward traffic to. This specifies that traffic received on the local machine at port 3000 should be forwarded to port 3000 of the Kubernetes Service named "app-service".

Just the log of the pod and services



Événements

Nom	Motif	Message	Source	Object	Compte	Première vue	Dernière vue ↑
app-service.1750a0aca4c81bf6	Pulled	Container image "choco85470/myapp:v1" already present on machine	kubelet docker-desktop	Pod/app-service	1	21 minutes ago	21 minutes ago
app-service.1750a0aca78253af	Created	Created container app-service	kubelet docker-desktop	Pod/app-service	1	21 minutes ago	21 minutes ago
app-service.1750a0acb27422a8	Started	Started container app-service	kubelet docker-desktop	Pod/app-service	1	21 minutes ago	21 minutes ago



Conclusion

I learned how to set up a service and a pod. We added a port on my local machine to connect to the website. we have touched a lot of knowledge like Github, Docker, Kubernetes.
Finally we publish the code on github.