

# OUEST SOLUTIONS INFORMATIQUES

IUT La Roche-Sur-Yon

GREPILLOUX Antonin  
LP A2SR

2022-2023



OUEST SOLUTIONS INFORMATIQUES



IUT La Roche-sur-Yon  
Pôle Sciences et technologie

Nantes Université





# **SOMMAIRE**

## Table des matières

SOMMAIRE .....	2
Course 1 .....	3
Course 2 .....	7
Course 3 .....	13
LEXIQUE .....	14
BIBLIOGRAPHIE .....	15



# **Course 1**

## **Mission 1**

### **Découverte du module, Communication en langue anglaise :**

The discovery of the "*Communication en langue anglaise*" module began with a personal introduction from the teacher/trainer, sharing their profile and background.

After this brief introduction, all students introduced themselves, one by one, including a description of themselves and their company.

### **Découverte et utilisation de Docker et Git :**

During this first session, we practiced Docker and Git through english communication between students and the teacher/trainer.

It started with a presentation of Docker, including its workings and features.

We then practiced Docker, with the installation of Docker Desktop on the Linux Debian environment.

Secondly, using the Git tool, we retrieved a Github project to build a Docker container.

We followed the official Docker documentation, using the commands provided.

We edited a file named Dockerfile, containing various configuration elements for the container.

We defined several elements, such as "node" and the IUT proxies.

Finally, After defining our configuration file, we created the container.

We then launched the container to use it.

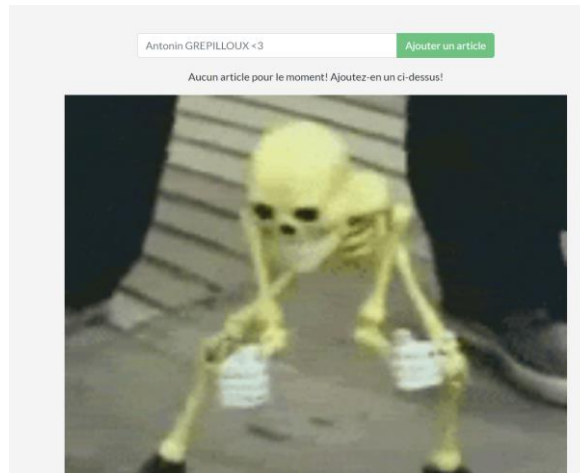
The container indicated it was listening on port 3000, so we opened a web browser and entered our IP and listening port.

## Mission 2

### Modification de l'application :

We had to modify our application to see the change with the old container.

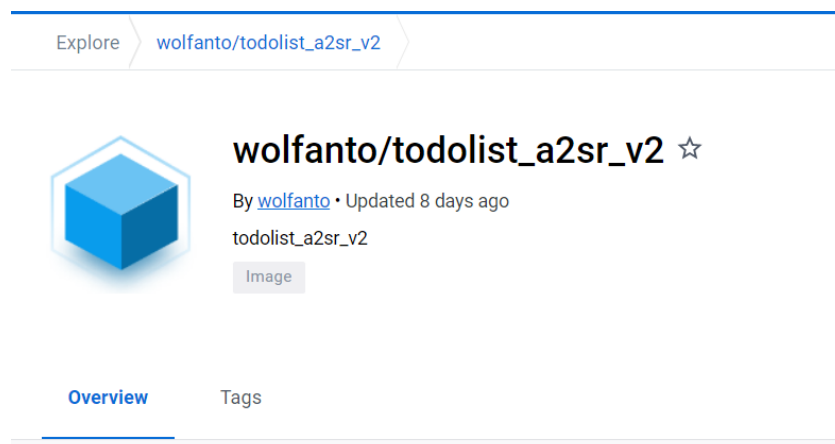
For that, I modify the src/static/app.js file to translate the sentences in French. Then I also modify the HTML page to add a GIF image. Then we rebuild our application.



### Partage et installation de l'image Docker :

After rebuilding our application, we try it out to see our changes.

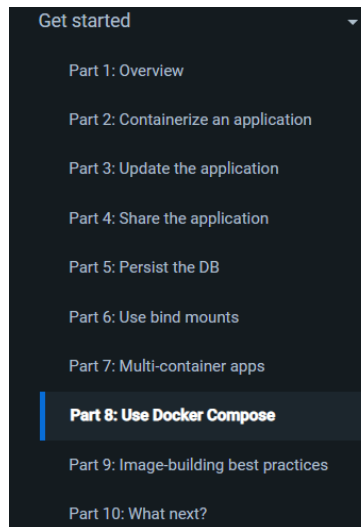
We create an account on the hub docker site that will allow us to share images with our classmates so they can see our changes. We use various commands to send and retrieve the images from the Docker site.



## Mission 3

### **Suivi du tutoriel Docker :**

We follow the tutorial proposed by Docker which will allow us to understand how to use Docker. We follow several steps explaining us step by step how it works. We could understand how to create a container, update it, share it, set several options and configure the configuration files.



### **Mise en place DB Persist :**

After all the steps of the tutorial, we had to set up a persistent database to save the todolist items, even if the container is turned off and then on again.

For this we create a persistent volume for the container.

### **Découverte et utilisation de Docker Compose :**

Docker Compose is a tool that was developed to help define and share multi-container applications. With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

We had to create a file named docker-compose.yml with a set of information allowing the application to work properly. For this we define the service (Node), and an installation command. We also define the port on which to connect.

We indicate the directory folder and the volume. We also specify the MySQL environment. Then, via the Docker Compose file, we run it, and check if the container has been created.



## **Conclusion :**

To summarize, in the first session, Docker and Git were practiced through English communication between students and teacher. There was a presentation of Docker and its workings, followed by the installation of Docker Desktop on Linux Debian.

Using Git, a Github project was retrieved to build a Docker container, following the official documentation. A Dockerfile was edited with configuration elements for the container, created and launched.

The container was listening on port 3000 and was used by entering the IP and port in a web browser.

We modified our application and shared it with our classmates. We also followed the Docker tutorial with the implementation of a persistent database.

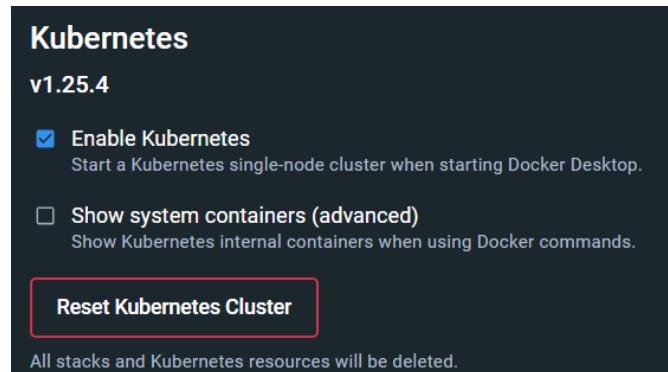
Then we saw how to use Docker Compose. We also created a GitHub repository and committed our repository.

# Course 2

## Mission 1

We have installed Docker Desktop for Windows to use Docker and Kubernetes.

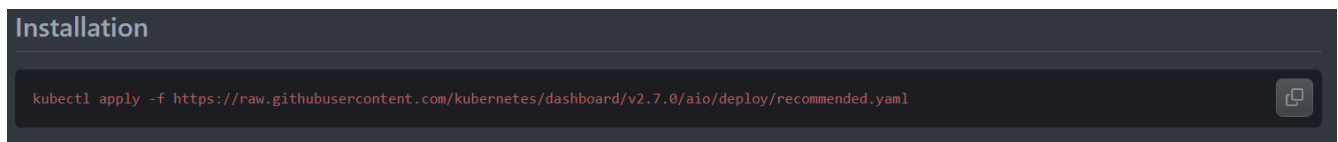
For this we took the Docker Desktop installer and installed it. Then, in the settings of Docker Desktop we went to the Kubernetes tab and checked the box to enable Kubernetes.



Then we installed Kubernetes dashboard. For this we installed via Powershell the Kubectl utility.

*winget install -e --id Kubernetes.kubectl*

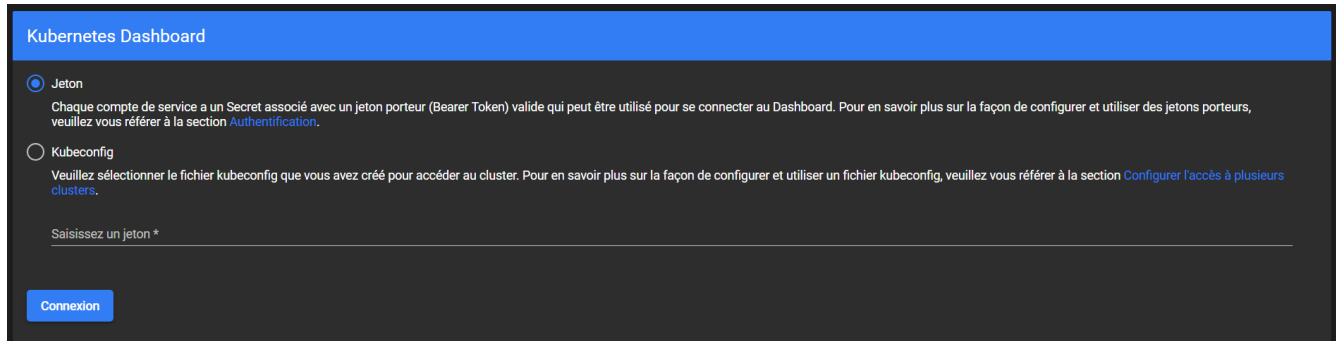
Using kubectl, we have deployed Kubernetes Dashboard via a YAML file on the official Kubernetes github.



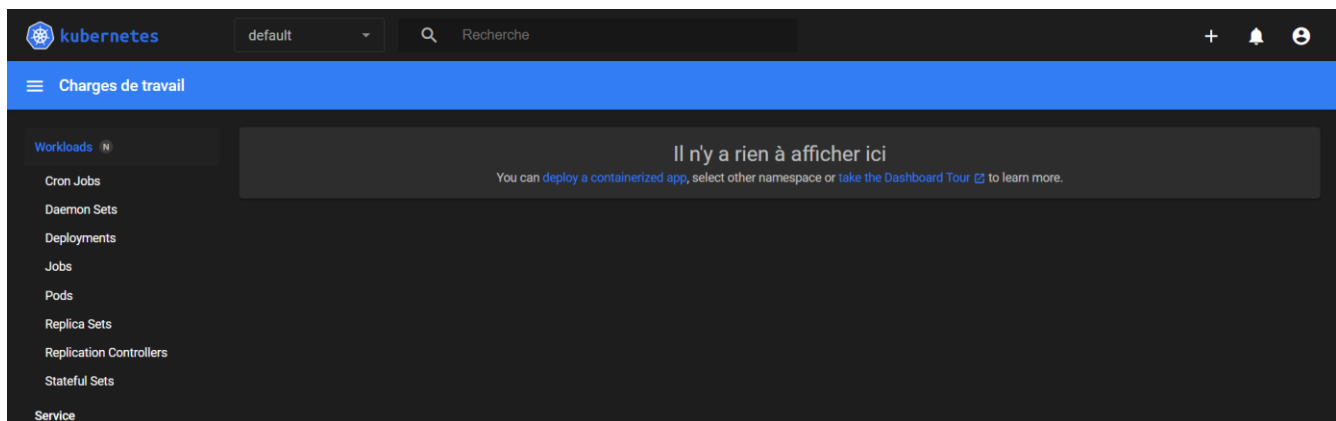
After installing Kubernetes, we started it. After that, Kubernetes Dashboard asked us for a token to authenticate. For this we created a "Service Account" and a "ClusterRoleBinding". After these steps, we generated the Token to authenticate. After authentication on Kubernetes Dashboard we now have access to the Dashboard.

*kubectl proxy*

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



The image shows the login page of the Kubernetes Dashboard. It has a blue header with the text "Kubernetes Dashboard". Below the header, there are two radio buttons for authentication: "Jeton" (selected) and "Kubeconfig". The "Jeton" option has a description: "Chaque compte de service a un Secret associé avec un jeton porteur (Bearer Token) valide qui peut être utilisé pour se connecter au Dashboard. Pour en savoir plus sur la façon de configurer et utiliser des jetons porteurs, veuillez vous référer à la section [Authentification](#)." The "Kubeconfig" option has a description: "Veuillez sélectionner le fichier kubeconfig que vous avez créé pour accéder au cluster. Pour en savoir plus sur la façon de configurer et utiliser un fichier kubeconfig, veuillez vous référer à la section [Configurer l'accès à plusieurs clusters](#)." Below these options is a text input field labeled "Saisissez un jeton \*" and a blue button labeled "Connexion".





## Mission 2

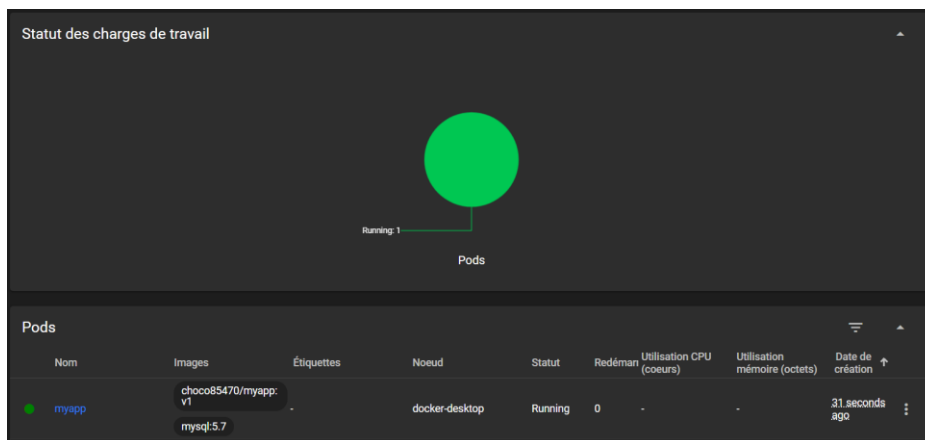
We updated the A2SR github project with the "git pull" command and deployed a YAML file on our Kubernetes cluster.

After installing Kubernetes Dashboard, we created a Pod.

The default YAML file needs to be modified to work. When we install it an error appears, it comes from a syntax error.

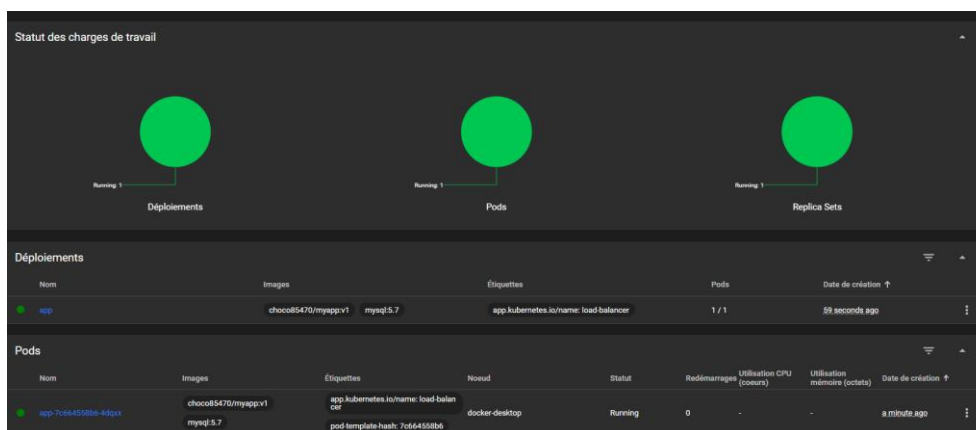
Here is the line to correct.

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: myapp
5  spec:
6    containers:
7    - name: app
8      image: choco85470/myapp:v1
9      command: ["/bin/sh"]
10     args: ["-c", "yarn install && yarn run dev"]
```



If we wanted to access our pod (our application), we had to create a Deployment with our Pod in it.

Then we had to create a Service to do Port Forwarding to be able to use our application.



We also needed to check our pod logs to notice a ping.

However, while browsing the Kubernetes Dashboard looking for logs of our cluster and application, we notice no trace of a ping.

It is possible that the instruction is not clear, and that we need to open a console in our application and perform a ping.

Événements							
Nom	Motif	Message	Source	Sub-object	Compte	Première vue	Dernière vue ↑
app-7c664558b6-4dqxx.1743bb47412b1af8	Started	Started container mysql	kubelet docker-desktop	spec.containers(mysql)	1	13 minutes ago	13 minutes ago
app-7c664558b6-4dqxx.1743bb4727ba41e8	Pulled	Container image "choco85470/myapp:v1" already present on machine	kubelet docker-desktop	spec.containers(app)	1	13 minutes ago	13 minutes ago
app-7c664558b6-4dqxx.1743bb472d286dc	Created	Created container app	kubelet docker-desktop	spec.containers(app)	1	13 minutes ago	13 minutes ago
app-7c664558b6-4dqxx.1743bb473422f8f8	Started	Started container app	kubelet docker-desktop	spec.containers(app)	1	13 minutes ago	13 minutes ago
app-7c664558b6-4dqxx.1743bb473467d9c8	Pulled	Container image "mysql:5.7" already present on machine	kubelet docker-desktop	spec.containers(mysql)	1	13 minutes ago	13 minutes ago
app-7c664558b6-4dqxx.1743bb4739a585c0	Created	Created container mysql	kubelet docker-desktop	spec.containers(mysql)	1	13 minutes ago	13 minutes ago
app-7c664558b6-4dqxx.1743bb46c2664970	Scheduled	Successfully assigned default/app-7c664558b6-4dqxx to docker-desktop			0	13 minutes ago	13 minutes ago

```
Shell dans app      dans app-7c664558b6-4dqxx
OCI runtime exec failed: exec failed: unable to start container process: exec: "bash": executable file not found in $PATH: unknown
/app # ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=36 time=70.563 ms
64 bytes from 8.8.8.8: seq=1 ttl=36 time=31.377 ms
64 bytes from 8.8.8.8: seq=2 ttl=36 time=17.231 ms
64 bytes from 8.8.8.8: seq=3 ttl=36 time=18.039 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 17.231/34.302/70.563 ms
/app #
```

After installing and verifying our Pod, we need to delete it.

To do this, we can delete it via command line or directly from the Kubernetes dashboard.

```
Windows PowerShell
PS E:\Anglais\Cours2 Mission 2> kubectl delete pod myapp
pod "myapp" deleted
PS E:\Anglais\Cours2 Mission 2>
```

Déploiements					
Nom	Images	Étiquettes	Pods	Date de création ↑	
app	choco85470/myapp:v1	mysql:5.7	app.kubernetes.io/name: load-balancer	1 / 1	13 minutes ago

Pods							
Nom	Images	Étiquettes	Noeud	Statut	Redéman	Utilisation CPU (coeurs)	Utilisation mémoire (octets)
app-7c664558b6-4dqxx	choco85470/myapp:v1	app.kubernetes.io/name: load-balancer	docker-desktop	Running	0	-	-

### Mission 3

In course 1, we created a docker-compose.yaml file to deploy our application on Docker. Now we need to do the same thing via Kubernetes.

To do so, we converted our docker-compose.yaml to be compatible with the syntax required by Kubernetes.


(Image non representative du résultat escompté)

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      app.kubernetes.io/name: load-balancer
6    name: app
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app.kubernetes.io/name: load-balancer
12   template:
13     metadata:
14       labels:
15         app.kubernetes.io/name: load-balancer
16     spec:
17       containers:
18         - image: choco85470/myapp:v1
19           name: app
20           command: ["/bin/sh"]
21           args: ["-c", "yarn install && yarn run dev"]
22           ports:
23             - containerPort: 3000
24           workingDir: "/app"
25           volumes:
26             - .:/app
27           environment:
28             MYSQL_HOST: mysql
29             MYSQL_USER: root
30             MYSQL_PASSWORD: secret
31             MYSQL_DB: todos
32         - image: mysql:8.0
33           name: mysql
34           env:
35             - name: MYSQL_ROOT_PASSWORD
36               value: "secret"
37             - name: MYSQL_DATABASE
38               value: "todos"
39       volumes:
40         todo-mysql-data:
```

Our application is not accessible from our browser. To solve this problem we have created a service to do Port Forwarding to be able to use our application.

```
Windows PowerShell
PS C:\Users\anton\convert> kubectl expose deployment app --type=LoadBalancer --name=app
service/app exposed
PS C:\Users\anton\convert>
```

Services						
Nom	Étiquettes	Type	IP cluster	Terminaisons internes	Terminaisons externes	Date de création ↑
app	app.kubernetes.io/name: load-balancer	LoadBalancer	10.105.184.84	app:3000 TCP app:32626 TCP	localhost:3000	15 minutes ago
kubernetes	component: apiserver provider: kubernetes	ClusterIP	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	23 hours ago



After all the assignments were done, we uploaded our work to our Github repository.

```
git config --global user.name "username"
```

```
git config --global user.email username@mail.com
```

```
git init
```

```
git add . (to add all the files in the current directory in the .git)
```

```
git commit -m "Commit English Courses" (to commit modification)
```

```
git remote add origin https://github.com/wolfanto/A2SR.git (to specify where is my github repository)
```

```
git push -u origin main (to push the main branch)
```

To conclude course n°2, we installed several services such as Docker Desktop and Kubernetes, and understand how they work. We also installed a dashboard for Kubernetes. We learned how to use Kubernetes and its dashboard to create and delete a Pod.

Then we converted our docker-compose.yaml to Kubernetes to install our application correctly.

Finally, we published our code on our Github repository.

# **Course 3**

[Texte Session]



# **LEXIQUE**

Debian 11 : Linux distribution

Linux : Linux is an open-source operating system. An operating system is software that directly manages physical system components and resources, such as the processor, memory, and storage. It acts as the interface between applications and hardware.

Docker : Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Git : Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Docker Desktop : Docker Desktop is an application for MacOS, Linux, and Windows machines for the building and sharing of containerized applications and microservices.

Kubernetes : Kubernetes is an open source orchestration system for automating the management, placement, scaling and routing of containers that has become popular with developers and IT operations teams in recent years.

kubectl : kubectl is the Kubernetes-specific command line tool that lets you communicate and control Kubernetes clusters.

YAML : YAML is a data serialization language that is often used for writing configuration files.

Github : GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.

Kubernetes Dashboard : The Kubernetes Dashboard is a web-based management interface that enables you to: deploy and edit containerized applications. assess the status of containerized applications. troubleshoot containerized applications.

# **BIBLIOGRAPHIE**

*Docker*. (n.d.). <https://hub.docker.com/>

*Install on Linux*. (2023, January 30). Docker Documentation.

<https://docs.docker.com/desktop/install/linux-install/>

Perochon, M. (n.d.). *GitHub - mperochon/A2SR*. GitHub.

<https://github.com/mperochon/A2SR>

Wikipedia contributors. (2023, January 16). *Linux*.

<https://fr.wikipedia.org/wiki/Linux>