

# DOCKER

Author: *VAMOS Team*  
Date: January 19, 2024

## 1. What is Docker? Why do we need Docker?

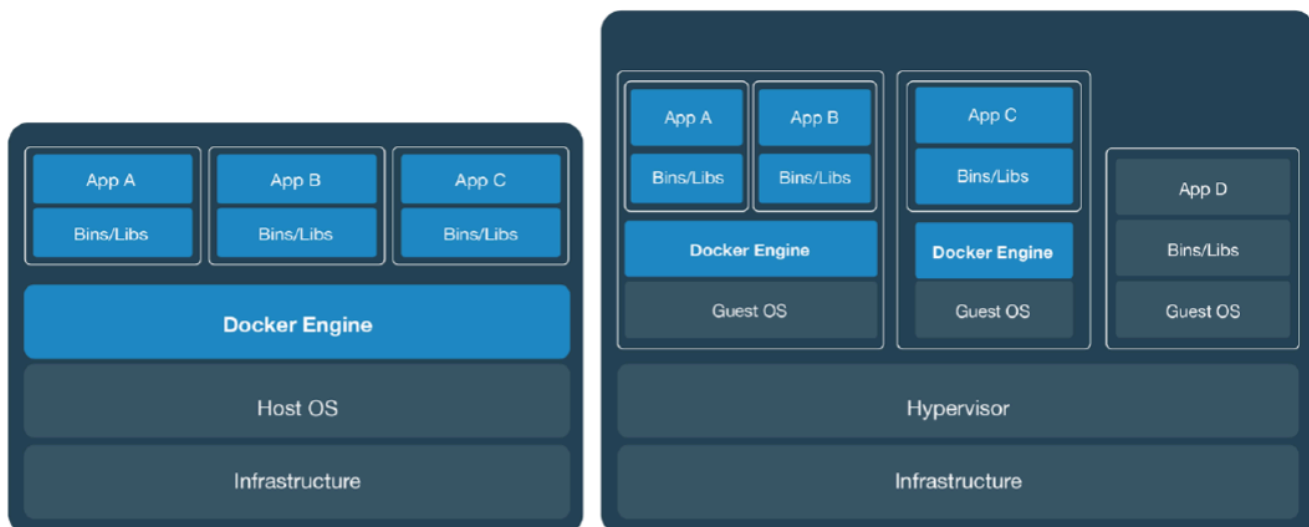
Docker is a technology that allows us to wrap the application and its dependencies into one package, which are **portable** (run anywhere) and **executable** (run anytime).

- Those packages are called **images**.
- When executing an image, we get a **container**.

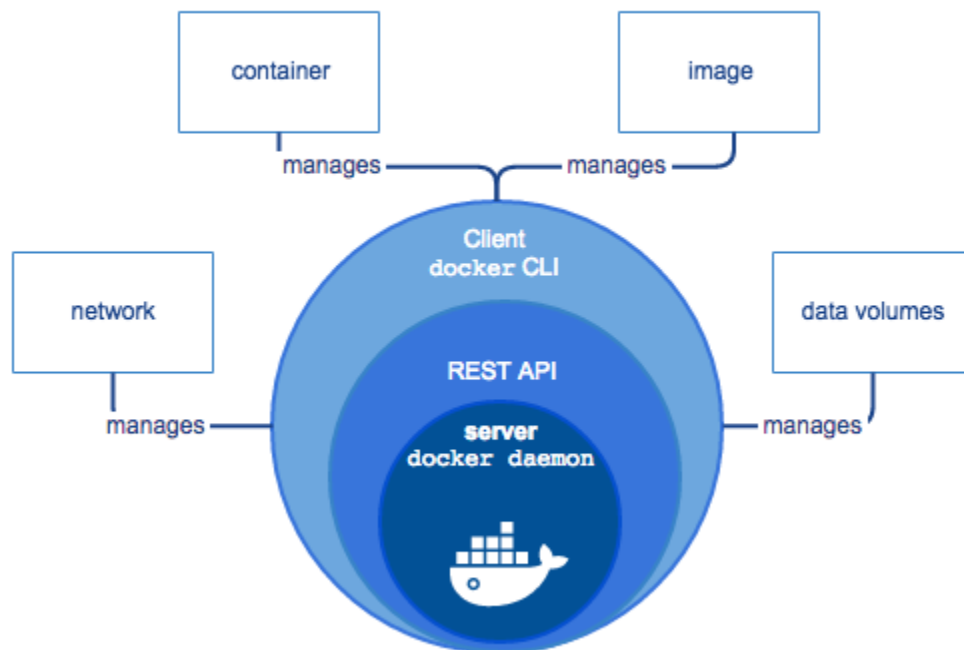
A container is almost the same as a virtual machine, but it has one big difference: **the container does not have its own OS kernel**, but it must use the same kernel as the physical machine.

After all, a container is just a process on a docker-based operating system managed by Docker.

## Containers Vs VMs



## 2. Docker Engine:



Docker is the company name, and the thing this company provide is Docker Engine  
**Docker Engine** includes 2 main parts:

- **Docker daemon:** core module that acts as a server

- **Docker client (cli):** communicates via HTTPS (Rest api in the image above)

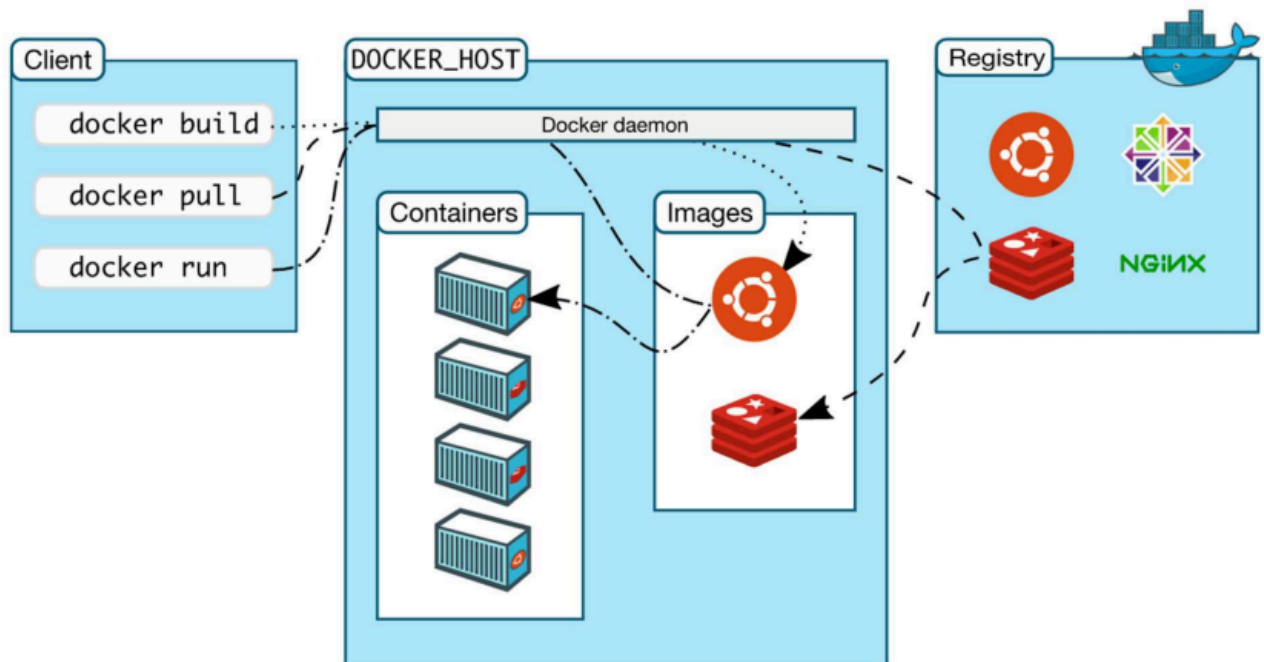
Because the docker daemon shares the same kernel with the operating system:

If you install docker on windows, it can only run window containers .Due to such limitations so when installing docker on windows, windows will ask you to install docker daemon on linux via hyper V or Virtual box, while docker cli will be installed on windows.

=> This is the reason why we need to download WSL-Windows Subsystem for Linux before we download docker on it.

### 3. Registry:

- Docker packages (containerised) application and its dependencies into image.
- Images are stored in the Docker repository call container registry.
- Some Docker registry providers:
  - Docker Hub
  - Amazon Elastic Container Registry (ECR)
  - Google Container Registry (GCR)
  - Azure Container Registry (ACR)



## 4. Basic commands:

Syntax docker

docker <component> <command>

### 4.1. Image:

- `docker image pull <image>`
- `docker image pull <image>: <tag>`
- `docker image push <image>: <tag>`
- `docker image ls | docker images`

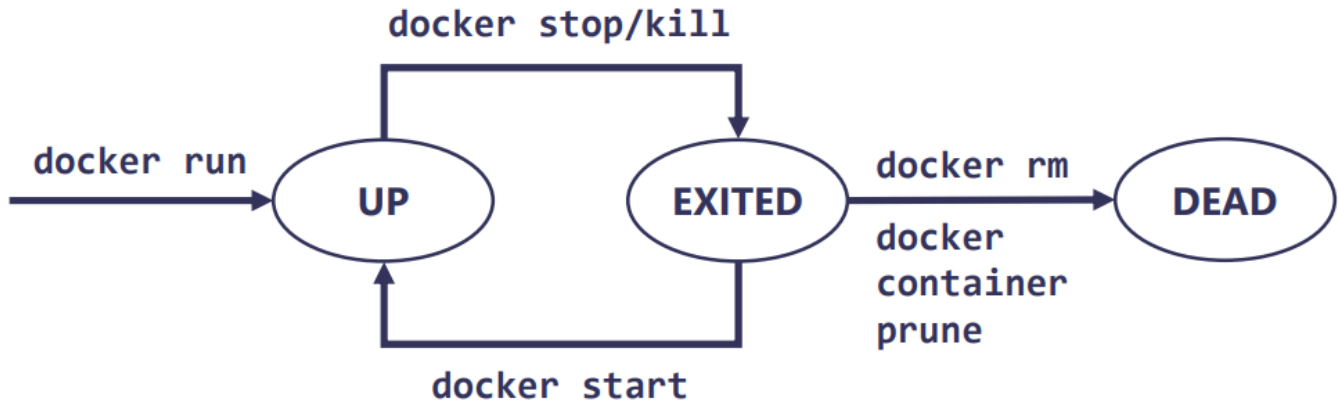
- docker image prune
- Short-hand:
  - docker pull
  - docker push

## 4.2. Container:

- docker container run <image>
- docker container ls | docker container ls -a
- **docker ps | docker ps -a**
- docker container stop <container\_id>
- docker container start
- docker container prune
- docker container exec <container\_id> <command>
- Short-hand:
  - docker run
  - docker stop
  - docker exec

## 5. Core Concept of container:

### 5.1. Docker container life cycle:



## 5.2. Execute commands insides container:

### Syntax

- `docker exec <container_id> <command>`

### Example

- `docker exec <container_id> echo Hello World`
- `docker exec <container_id> echo $PATH`
- `docker exec <container_id> sh -c "echo $PATH"`
- `docker exec <container_id> cat /etc/os-release`

Good To Remember To SSH into any container, use `exec shell` or `bash` with `-it`

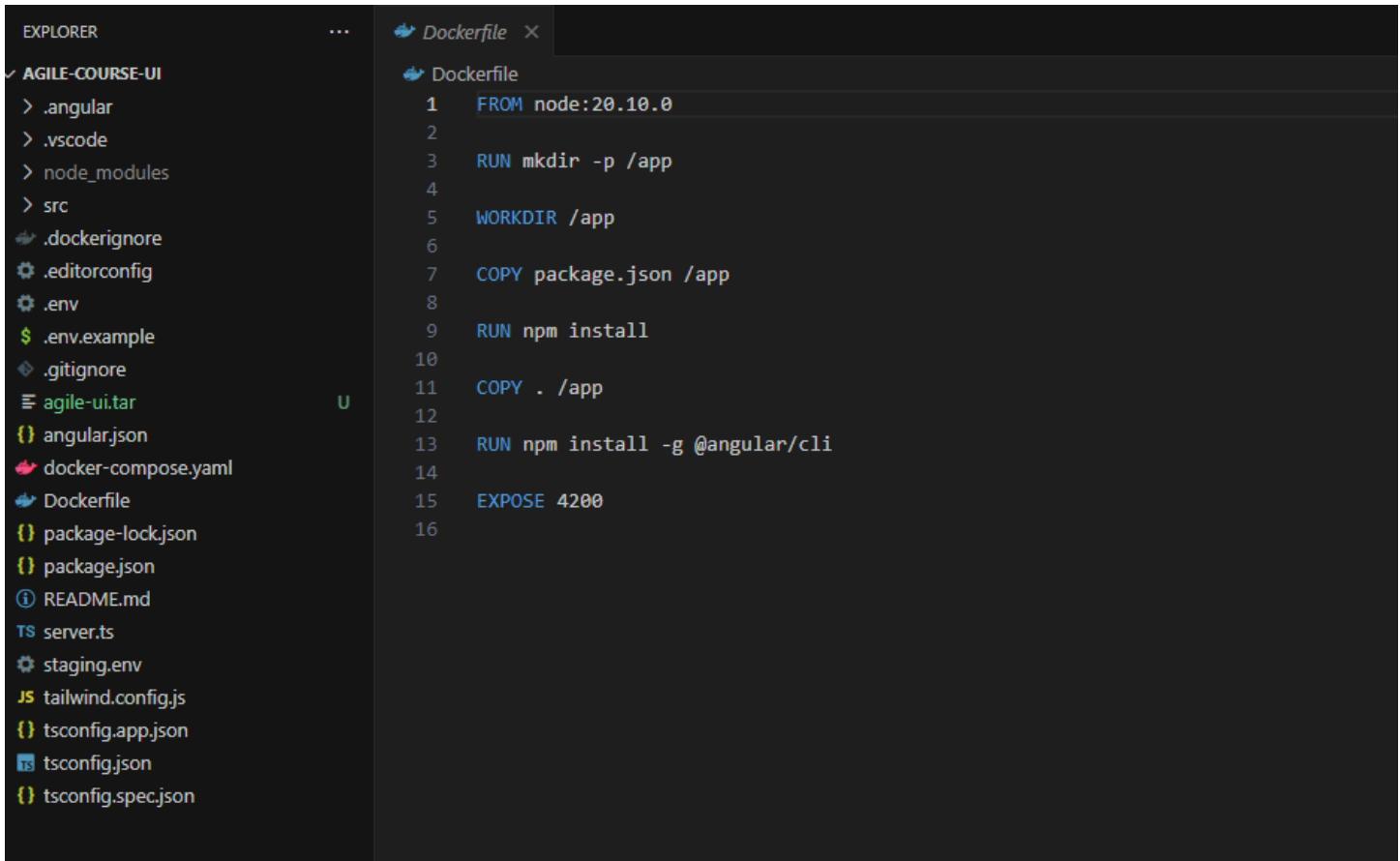
- `docker exec -it <container_id> sh`
- `docker exec -it <container_id> bash`

```
1 version: "3"
2
3 services:
4   angular-fe:
5     container_name: ${UI_CONTAINER_NAME}
6     build:
7       context: .
8       dockerfile: Dockerfile
9     restart: always
10    command: sh -c "ng serve ${CONFIGURATION} --host 0.0.0.0"
11    ports:
12      - "${UI_PORT}:4200"
13
```

**execute command inside container**

## 6. Docker File:

Dockerfile is a template that allows us to instruct Docker to build our own image step by step.



The screenshot shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a file tree for a project named 'AGILE-COURSE-UI'. The files listed include .angular, .vscode, node\_modules, src, .dockerignore, .editorconfig, .env, .env.example, .gitignore, agile-ui.tar, angular.json, docker-compose.yaml, Dockerfile, package-lock.json, package.json, README.md, server.ts, staging.env, tailwind.config.js, tsconfig.app.json, tsconfig.json, and tsconfig.spec.json. The 'Dockerfile' file is selected and open in the main editor area. The Dockerfile content is as follows:

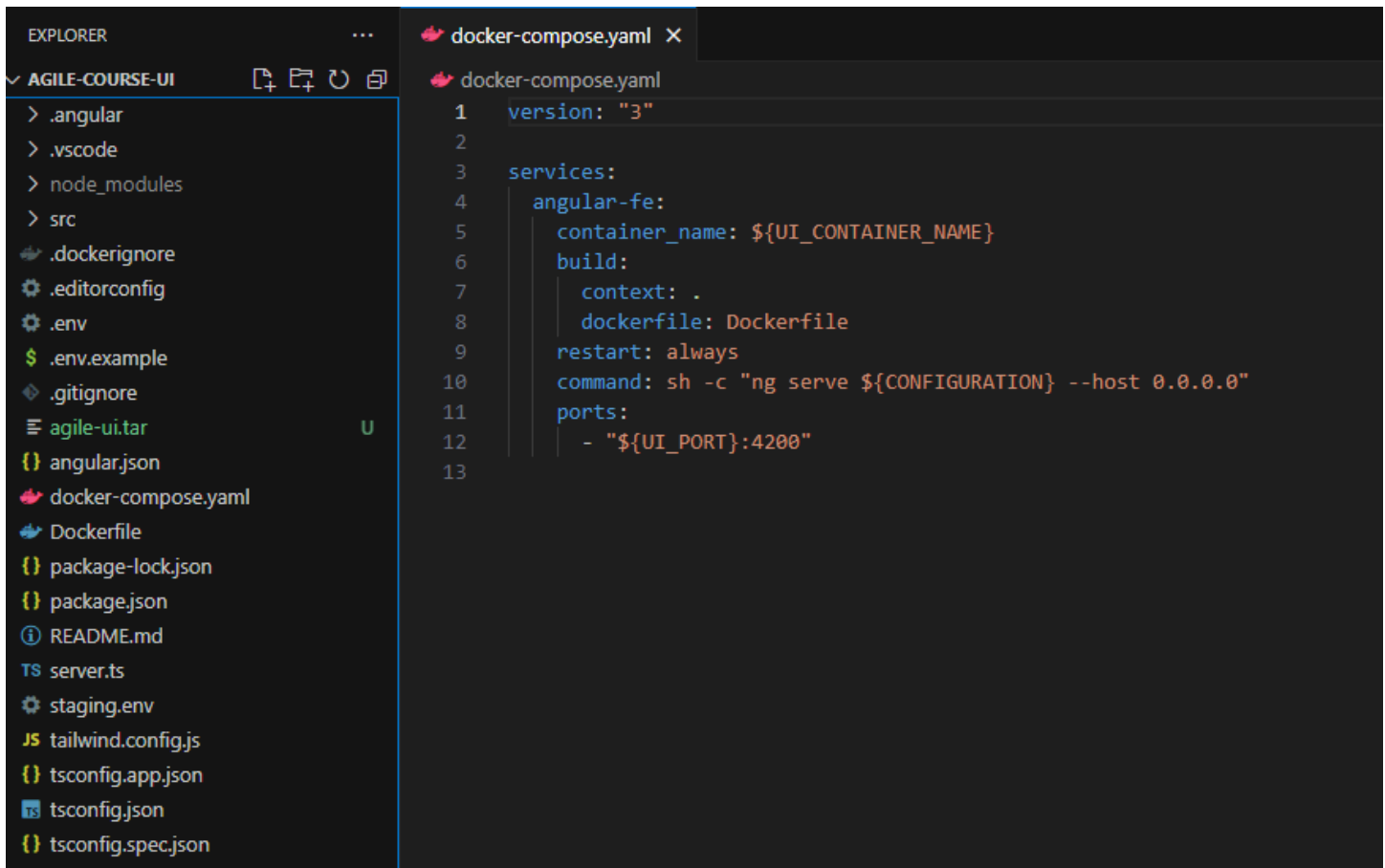
```
1 FROM node:20.10.0
2
3 RUN mkdir -p /app
4
5 WORKDIR /app
6
7 COPY package.json /app
8
9 RUN npm install
10
11 COPY . /app
12
13 RUN npm install -g @angular/cli
14
15 EXPOSE 4200
16
```



## 7. Docker-compose:

Docker Compose is a tool for defining and running multi-container applications.

docker-compose.yaml



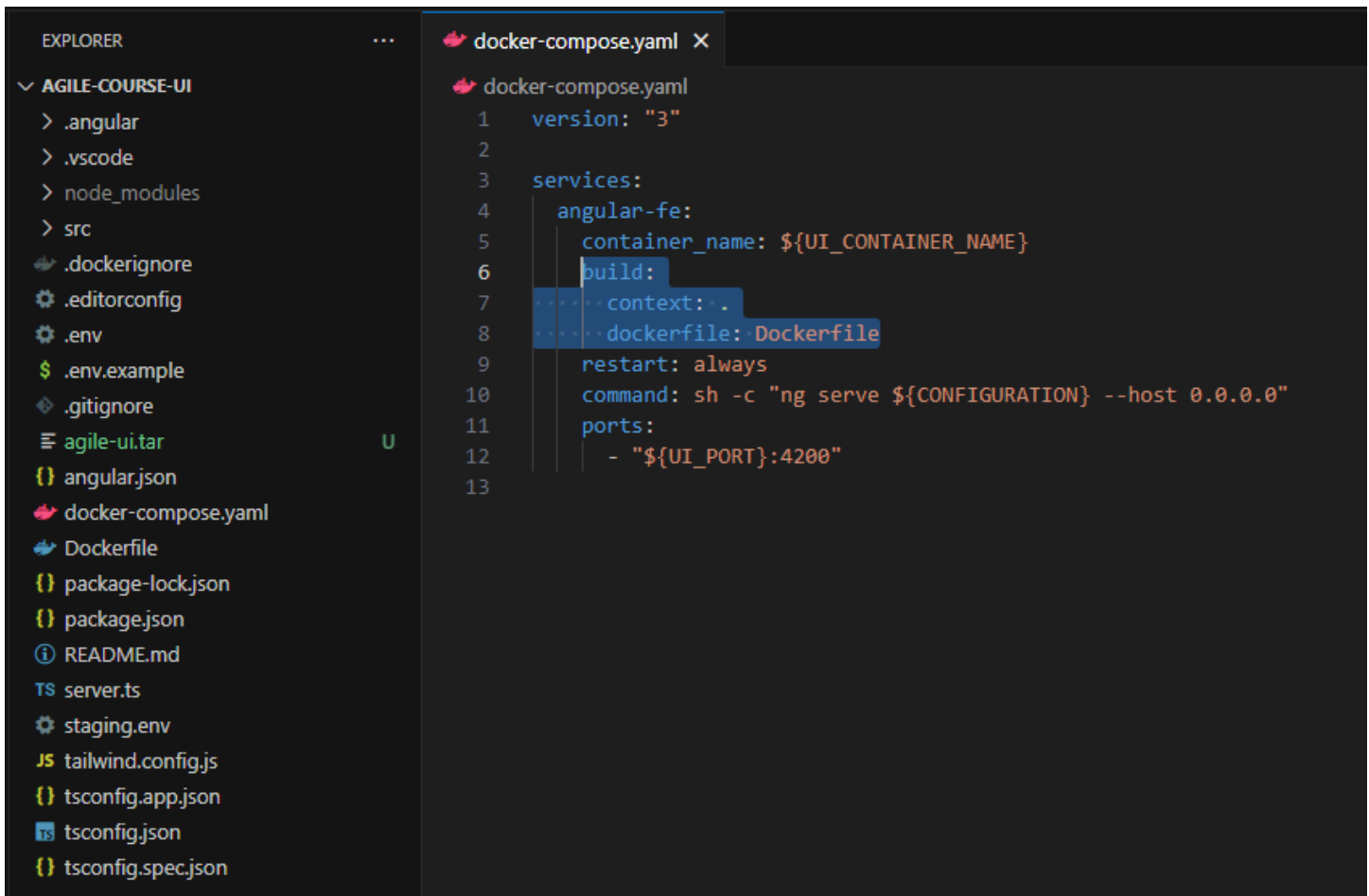
```
1 version: "3"
2
3 services:
4   angular-fe:
5     container_name: ${UI_CONTAINER_NAME}
6     build:
7       context: .
8       dockerfile: Dockerfile
9     restart: always
10    command: sh -c "ng serve ${CONFIGURATION} --host 0.0.0.0"
11    ports:
12      - "${UI_PORT}:4200"
13
```

Build image easily with compose

Place docker-compose.yaml in the same folder of Dockerfile and build.

### Syntax

- docker-compose build



The screenshot shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a file tree for a project named 'AGILE-COURSE-UI'. The tree includes folders like '.angular', '.vscode', 'node\_modules', and 'src', as well as files like '.dockerignore', '.editorconfig', '.env', '.env.example', '.gitignore', 'agile-ui.tar', 'angular.json', 'docker-compose.yaml', 'Dockerfile', 'package-lock.json', 'package.json', 'README.md', 'server.ts', 'staging.env', 'tailwind.config.js', 'tsconfig.app.json', 'tsconfig.json', and 'tsconfig.spec.json'. The 'docker-compose.yaml' file is selected and open in the main editor area. The file content is as follows:

```
1 version: "3"
2
3 services:
4   angular-fe:
5     container_name: ${UI_CONTAINER_NAME}
6     build:
7       context: .
8       dockerfile: Dockerfile
9     restart: always
10    command: sh -c "ng serve ${CONFIGURATION} --host 0.0.0.0"
11    ports:
12      - "${UI_PORT}:4200"
13
```

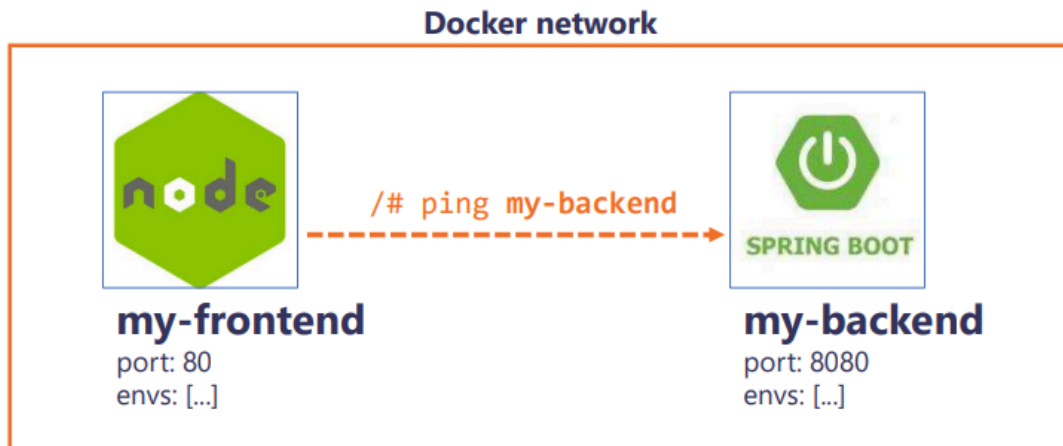
Run container easily with compose

## Syntax

- docker-compose up
- docker-compose up <service\_name>
- docker-compose up -d <service\_name>
- docker-compose logs -f <service\_name>
- docker-compose stop <service\_name>
- docker-compose down

## 8. Network:

When attaching containers into the same network, thanks to the built-in network resolution, Docker can resolve service's name into its actual IP. So that containers can now communicate with each other through service names instead of IPs.



## 9. Rancher Desktop:

### WSL2 (Windows Subsystem Linux 2):

- Docker engine only runs on Linux environment
- Demonstrates operational parity with native Linux environments.
- Key components, including the Rancher server and associated services, are executed within Docker containers meticulously managed by Docker within the WSL2 environment.

### Rancher Desktop:

- Rancher Desktop is a container management platform that provides a user interface and tools for managing, orchestrating containers in a clustered environment.
- When you deploy Rancher Desktop within WSL2, it operates similarly to how it would on a native Linux environment.
- Rancher Desktop components, including the Rancher server and any other services, run within Docker containers managed by Docker within the WSL2 environment.

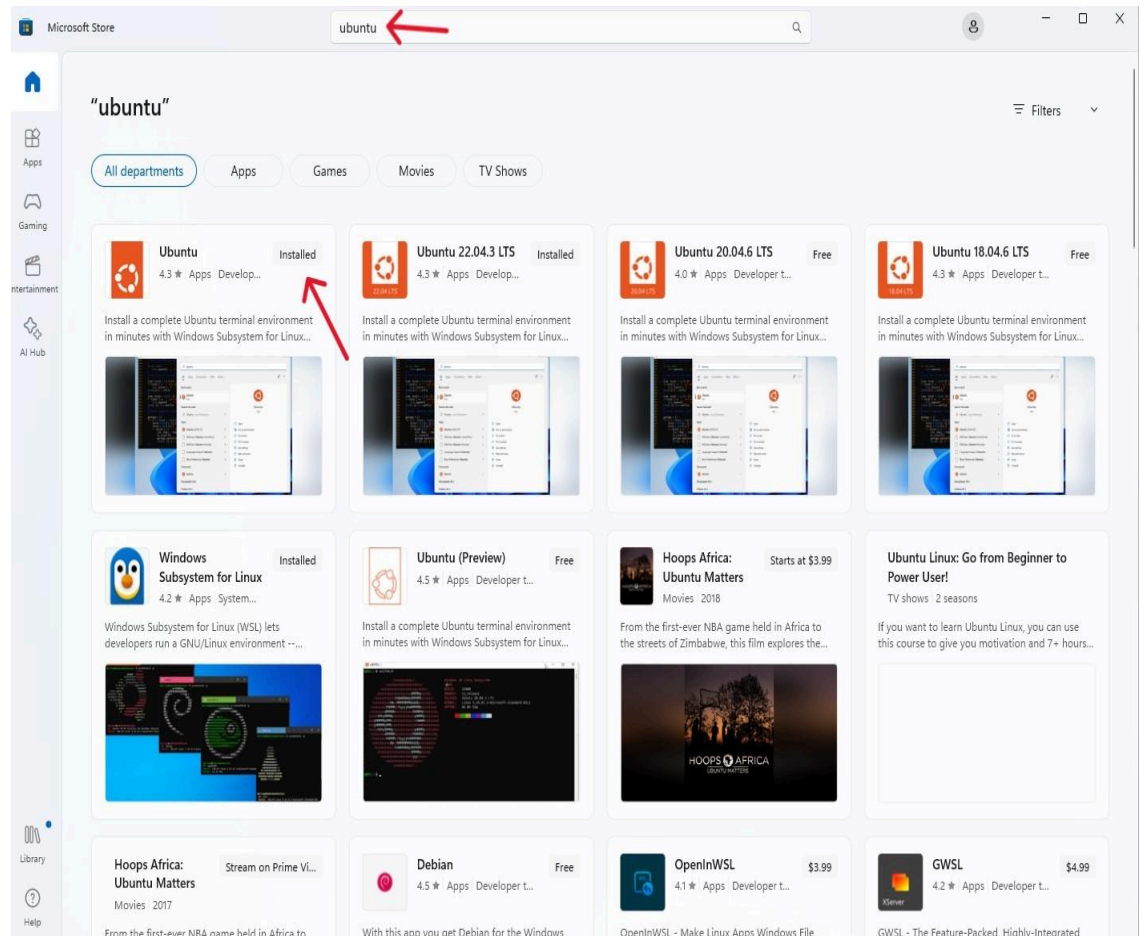
## Installation:

- **Prerequisite:**

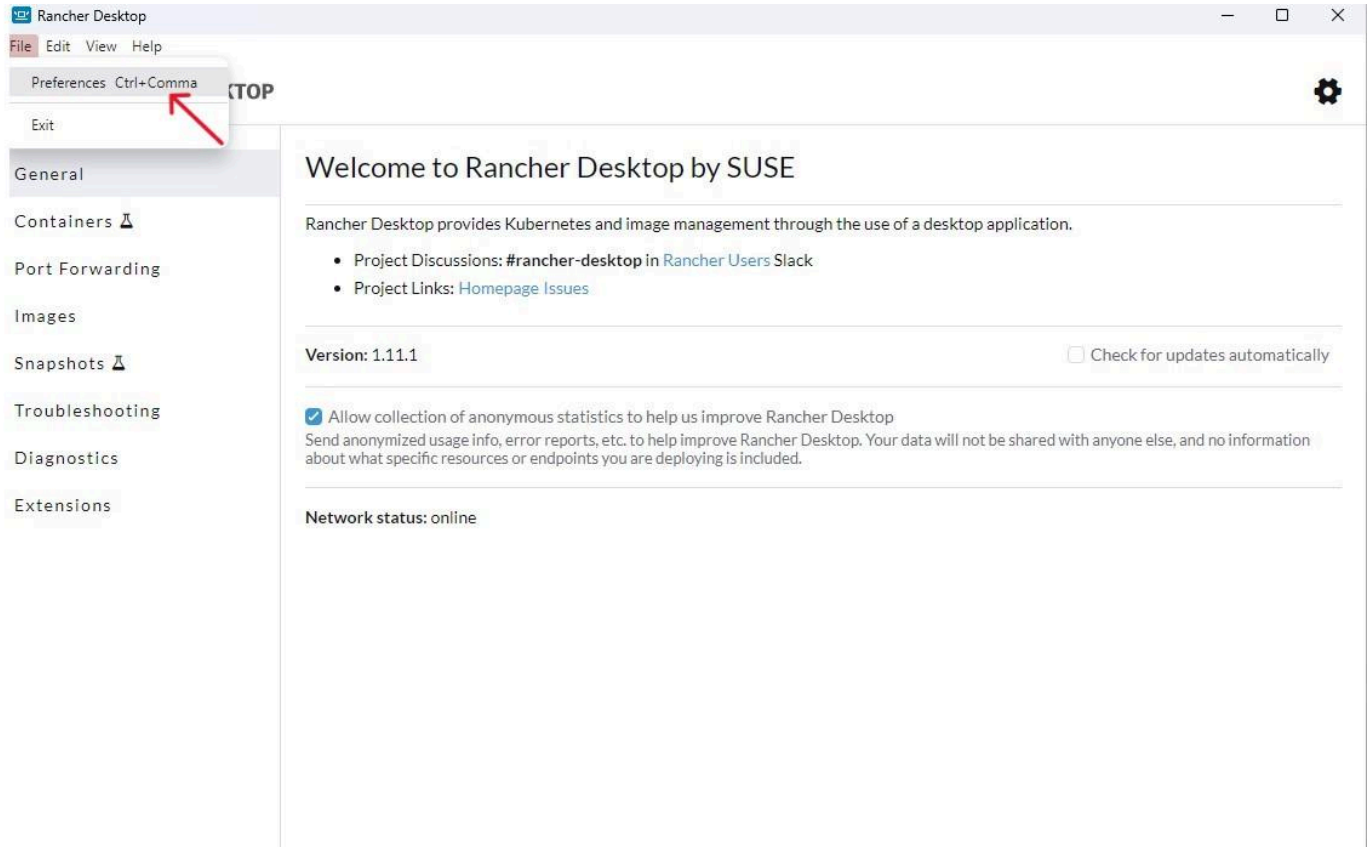
- Install WSL2 first, to install WSL2 please visit <https://learn.microsoft.com/en-us/windows/wsl/install>
- To install Rancher Desktop, please visit the website for fully instruction: <https://docs.rancherdesktop.io/getting-started/installation/>

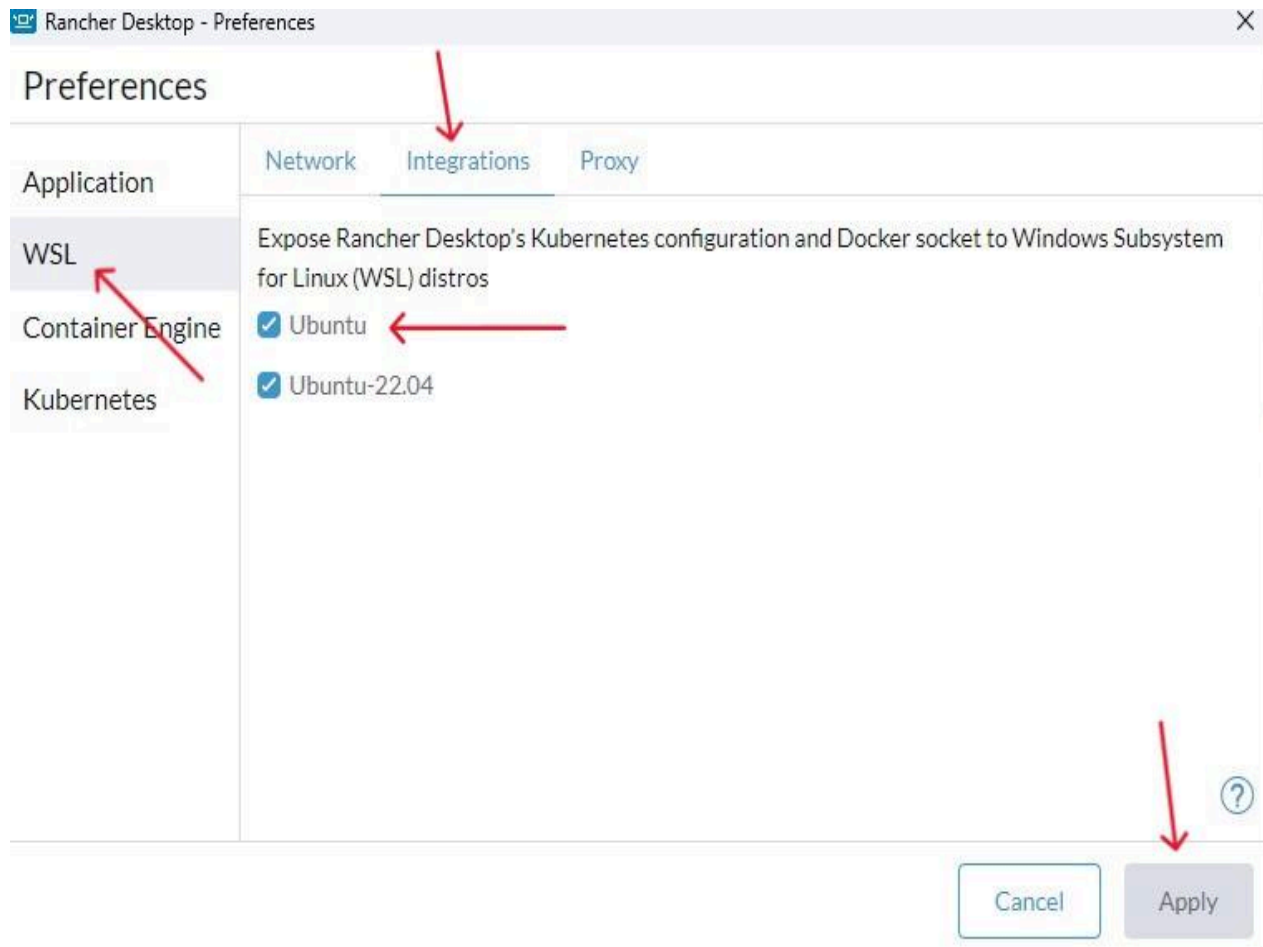
- **Setting:**

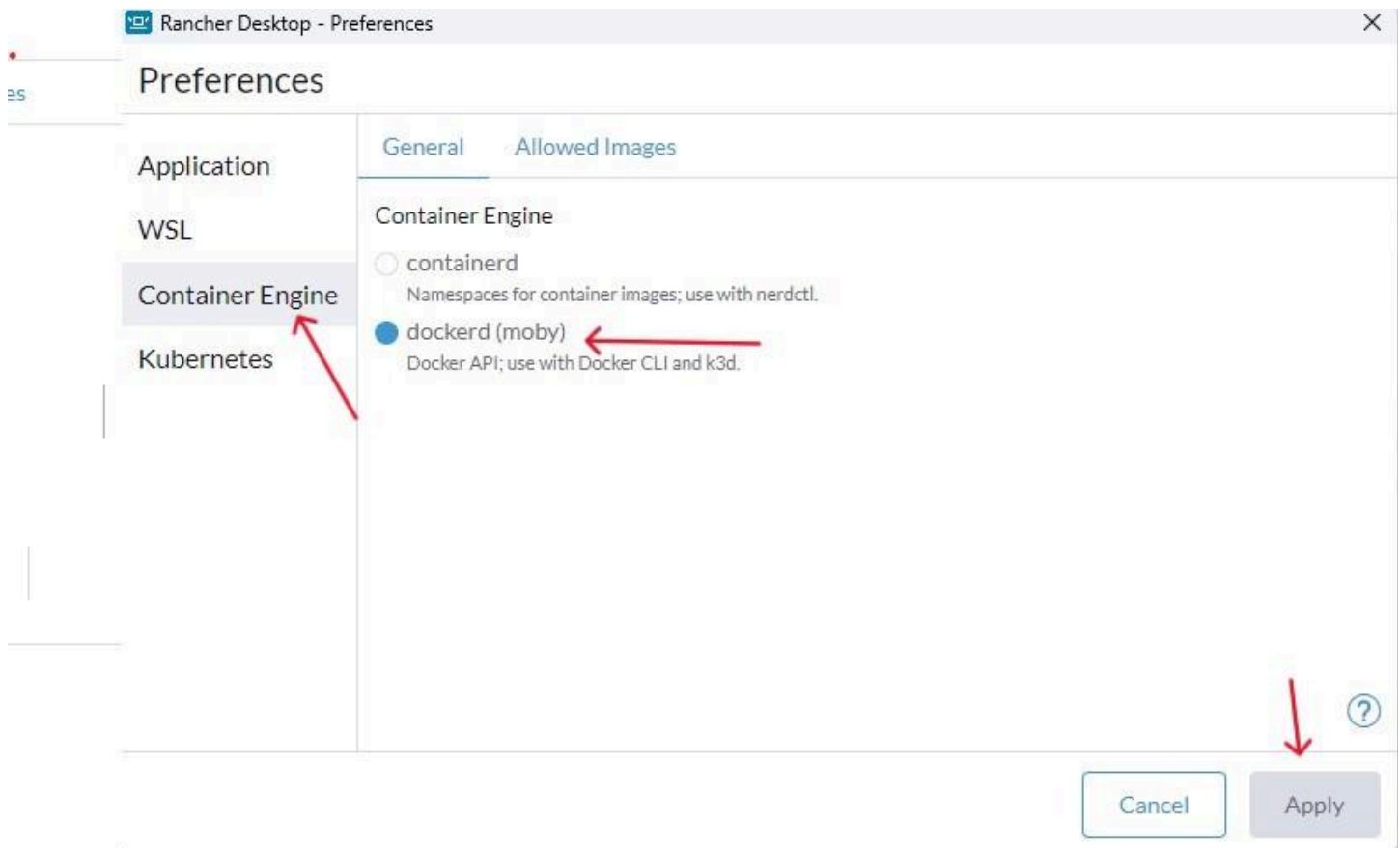
- After install Rancher Desktop, go to Microsoft store and install Ubuntu



- Open Rancher Desktop, configure these setting:







**\*Note:**

- For Running Docker standalone without Rancher, simply running the command in:

```
sudo service docker start
```

- This command is effective only when WSL2 is installed before Rancher Desktop. It initiates services in a Linux-based environment, as Docker engine operates as services. In the event of Rancher Desktop being installed before WSL2, Docker Engine is established within an alternative Linux environment on the Windows platform.

## 10. References:

Install Windows Subsystem for Linux 2:

---

<https://learn.microsoft.com/en-us/windows/wsl/install>

Docs Docker engine:

<https://docs.docker.com/engine/>

Docker and Compose Tutorial:

<https://www.youtube.com/watch?v=yWCse8S2qsM>

Docker and Compose Tutorial PDF:

<https://drive.google.com/file/d/1ETOW4RhvYdBEiUfu9N9ZNBFNdwbhKGOI/view>