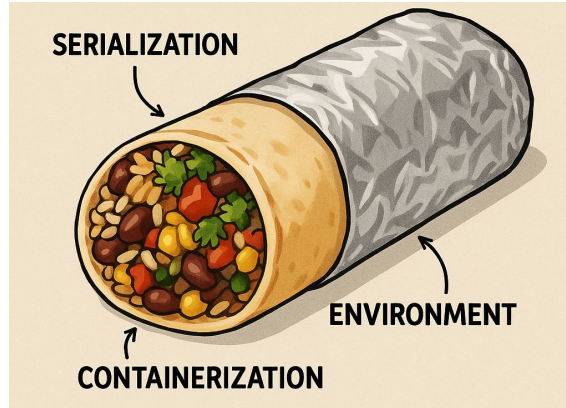


Starting @ 1:10 PM

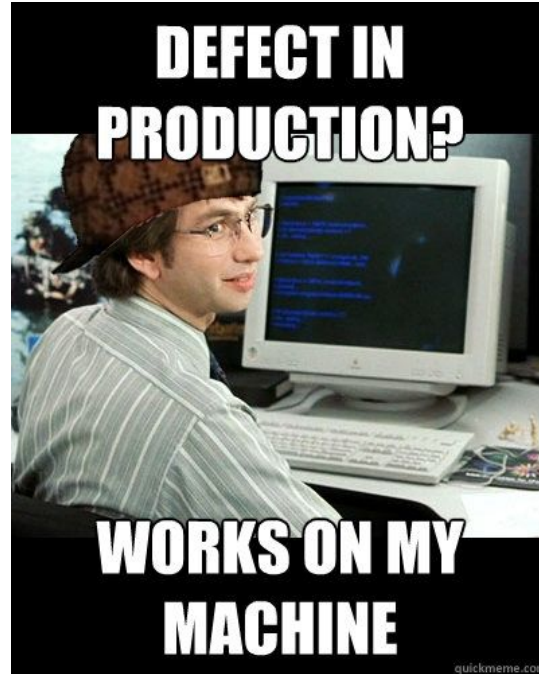
Containerization

The Three-Layer Packaging Burrito



Serialization is the filling, environment is the tortilla, containerization is the foil wrap.

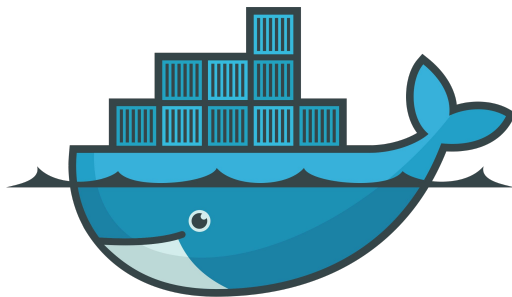
Why Containerization is important?



Tools used in Containerization

1. Docker
2. Kubernetes
3. AWS ECS (Elastic Container Service)
4. Azure Container Service

Docker



What is Docker?

Docker is a platform that enables developers to automate the deployment of applications within lightweight, portable containers.

These containers package an application and its dependencies, ensuring consistent behavior across different environments.



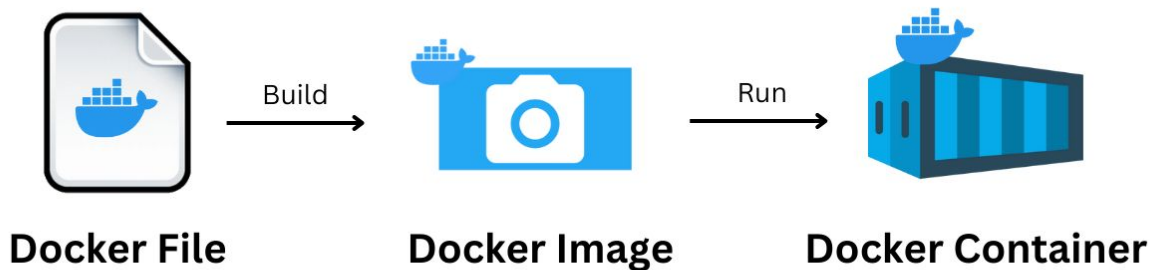
Images vs Containers

Image: a template that includes all of the files, binaries, libraries, and configurations to run a container.

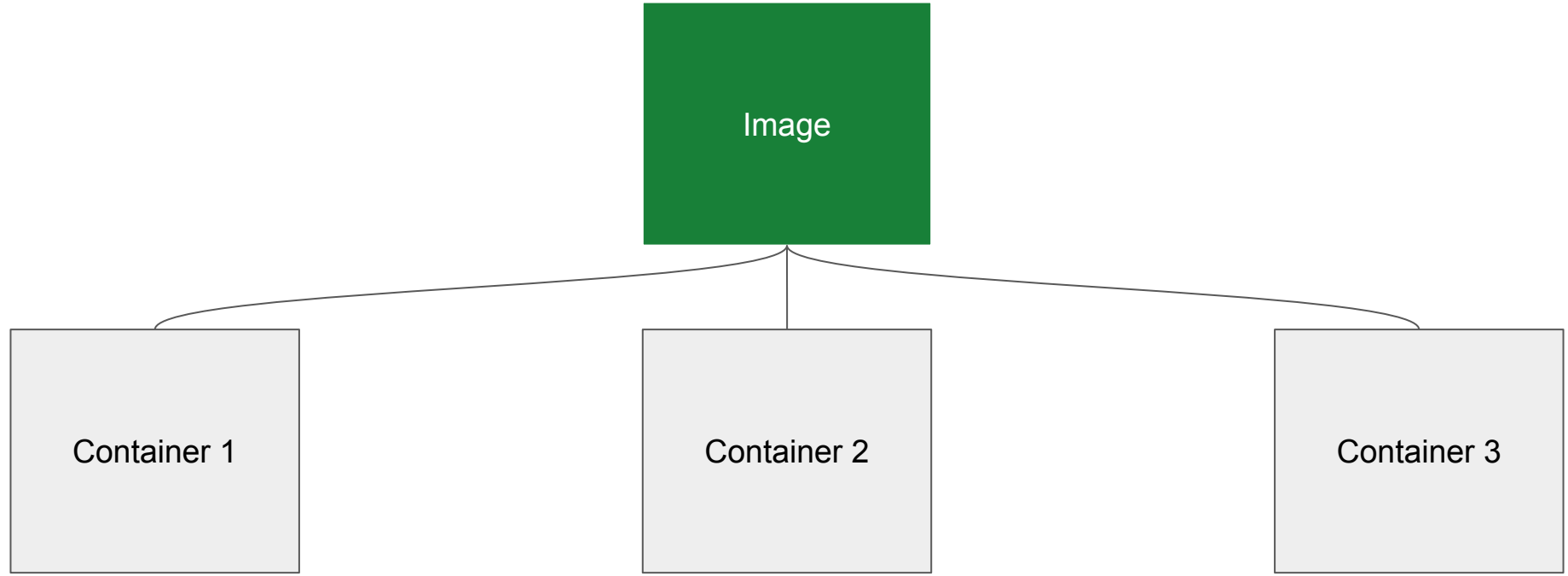
- Image: recipe
- Containers: dish made from the recipe

We can use the same image in multiple containers.

To run a container I should determine what I will do with it.



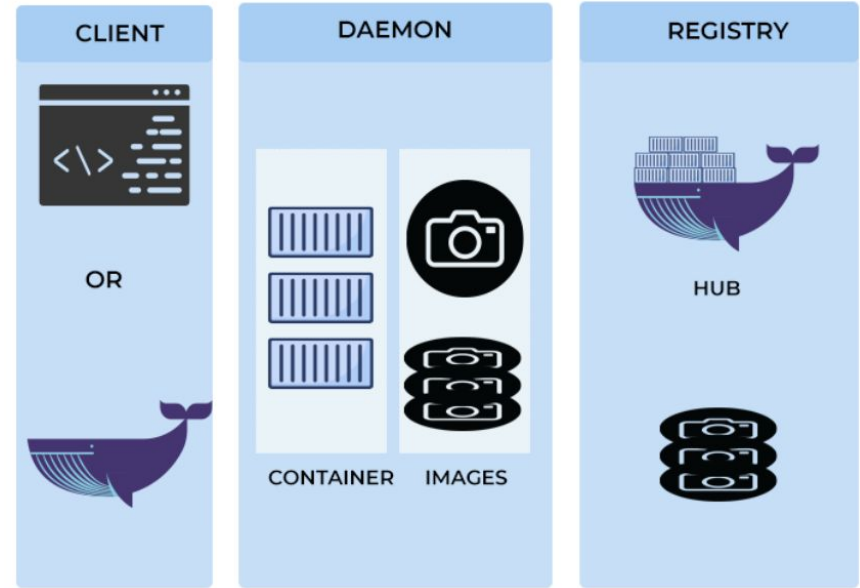
Images vs Containers



Docker Architecture Overview

Docker uses a client-server architecture:

1. **Docker Client:** the primary way users interact with Docker via type Docker commands (send requests)
2. **Docker Daemon:** manages Docker objects like images, containers, networks, and volumes. (server that handle requests)
3. **Docker Registry:** stores Docker images (e.g., Docker Hub).



Containers vs VMs



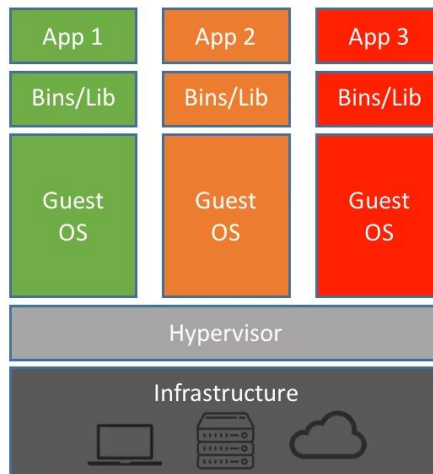
Containers vs VMs

- **Virtual Machines:** have their own complete operating systems (Guest OS), managed by a **Hypervisor** (application that allows you to create VMs). This makes them heavier and slower.

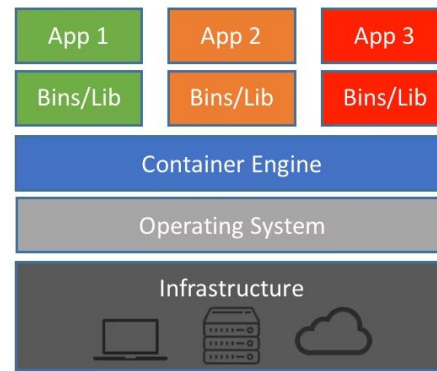
بيخلي ال OS يقدر يتكلم مع ال hardware ، بيقدم زي

virtual hardware لكل VM

- **Containers:** share a single operating system kernel, managed by a **Container Engine** (like Docker). This makes them lighter and faster.



Machine Virtualization



Containers

Basic Docker Commands

Running your first container

Let's start by running a simple container:

```
docker run hello-world
```

This command does the following:

1. Checks for the hello-world image locally
2. If not found, pulls the image from Docker Hub
3. Creates a container from the image
4. Runs the container, which prints a hello message
5. Exits the container

Basic Docker Commands

Here are some essential Docker commands for working with containers:

Listing Containers

To see all running containers:

```
docker ps
```

To see all containers (including stopped ones):

```
docker ps -a
```

Basic Docker Commands

Starting and Stopping Containers:

To stop a running container:

```
docker stop <container_id_or_name>
```

To start a stopped container:

```
docker start <container_id_or_name>
```

To restart a container:

```
docker restart <container_id_or_name>
```


Basic Docker Commands

Removing Containers:

To remove a stopped container:

```
docker rm <container_id_or_name>
```

To force remove a running container:

```
docker rm -f <container_id_or_name>
```

Docker in VSCode

Let's go to vsCode

Containerizing an Application

DockerFile

Dockerfile — a text document that contains a series of instructions that used to create a Docker image automatically.

`FROM python:latest`

- Use the latest official Python image from Docker Hub.

`WORKDIR /app`

- Set the working directory inside the container to `/app`.

`COPY requirements.txt .`

- Copy the `requirements.txt` file from your local machine to the `/app` directory inside the container.

`RUN pip install -r requirements.txt`

- Install Python dependencies listed in `requirements.txt`.

`COPY hello.py .`

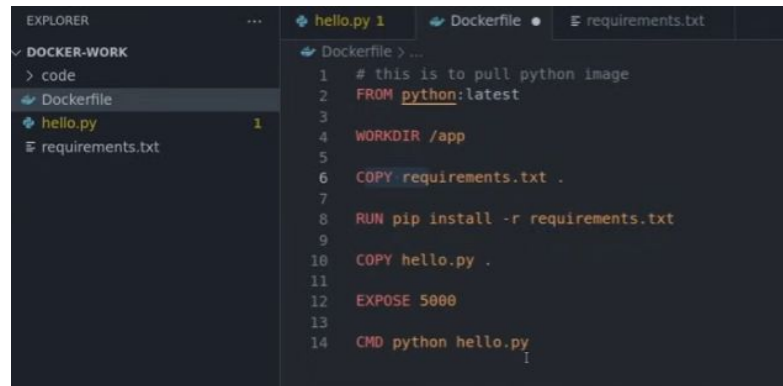
- Copy the `hello.py` script into the container.

`EXPOSE 5000`

- Inform Docker that the application will use port 5000 (useful for web apps, like Flask).

`CMD python hello.py`

- This is the command that will run when the container starts: it runs your `hello.py` script



The screenshot shows a code editor interface with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'DOCKER-WORK'. It contains a 'code' directory with files 'Dockerfile', 'hello.py', and 'requirements.txt'. The 'Dockerfile' is selected. On the right, the editor shows the content of the 'Dockerfile' with line numbers 1 through 14. The instructions are: 1. Comment: '# this is to pull python image', 2. 'FROM python:latest', 3. Blank line, 4. 'WORKDIR /app', 5. Blank line, 6. 'COPY requirements.txt .', 7. Blank line, 8. 'RUN pip install -r requirements.txt', 9. Blank line, 10. 'COPY hello.py .', 11. Blank line, 12. 'EXPOSE 5000', 13. Blank line, 14. 'CMD python hello.py'.

```
1 # this is to pull python image
2 FROM python:latest
3
4 WORKDIR /app
5
6 COPY requirements.txt .
7
8 RUN pip install -r requirements.txt
9
10 COPY hello.py .
11
12 EXPOSE 5000
13
14 CMD python hello.py
```

Docker Compose

Docker Compose: is a powerful tool for defining and running multi container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

The `docker-compose.yml` file is the core of Docker Compose. It defines all the components and configurations of your application.

Lab

Your task is as follows:

- Create a `Dockerfile` at the root of the project
- Using `Python:latest` image
- Set the working directory inside the container to `/app`.
- Copy all project files into the container.
- Install dependencies from `requirements.txt`.
- Expose port `8000`.
- Add a `CMD` instruction to run the FastAPI app with `uvicorn`.
- Build the image and name it “your-name”
- Run the container and name it “docker-your-name”

BONUS:

- View logs from the running container

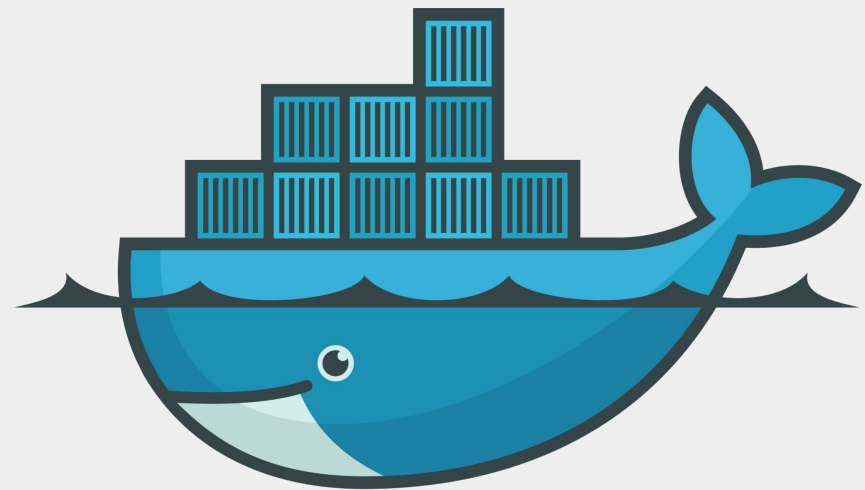
Basic Docker Commands

Here are some essential Docker commands for working with containers:

- | | |
|------------------|--|
| >> docker build | # Build an image from a Dockerfile |
| >> docker images | # List all images on a Docker host |
| >> docker run | # Run an image |
| >> docker ps | # List all running and stopped instances |
| >> docker stop | # Stop a running instance |
| >> docker rm | # Remove an instance |
| >> docker rmi | # Remove an image |

Thank You!

Any Questions?



docker