devoteam

Creative tech for Better Change

devoteam

# Docker

Creative tech for Better Change

# Agenda

- Intro
- Recap
- Docker Architecture
- Docker Installation
- Docker Commands
- Lab

Vladimiro Luz
vladimiro.luz@devoteam.com

# Recap: What is Docker?

- An open-source platform that enables developers to automate the deployment, scaling, and management of applications using containers.
- Containers are lightweight, portable, and self-sufficient environments that include everything needed to run an application
    - Code
    - Runtime
    - system tools
    - Libraries
    - dependencies

# Recap: Why Docker?

**Efficiency:**

Containers share the host OS kernel, reducing overhead and improving performance.

**Isolation:**

Containers are isolated from each other, preventing conflicts and security vulnerabilities.

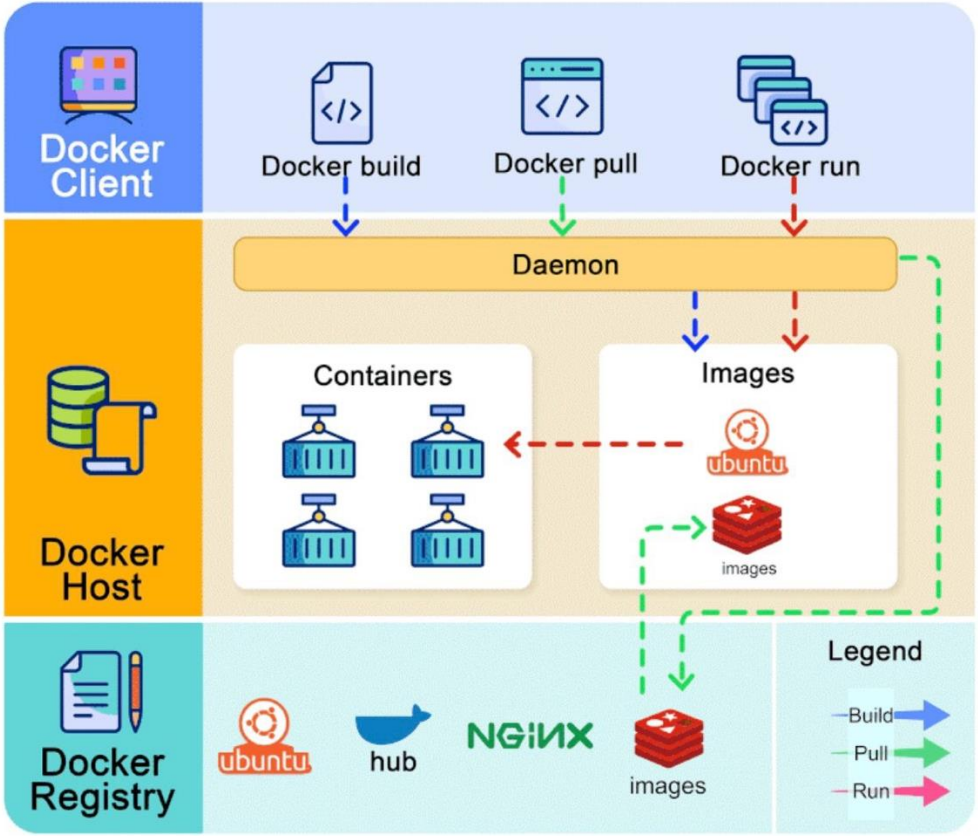
**Portability:**

Containers can run on any system that supports the containerization platform.

**Scalability:**

Containers can be easily scaled up or down to meet demand.

# Docker Architecture

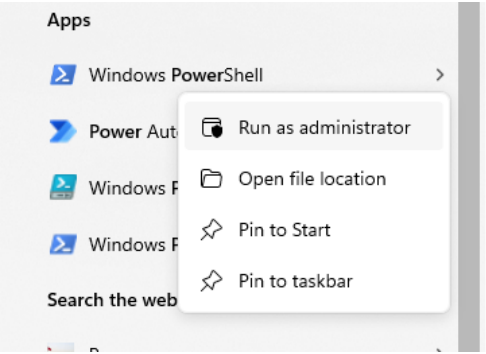



devoteam

# Docker Installation

Creative tech for Better Change

# Using WSL with Ubuntu

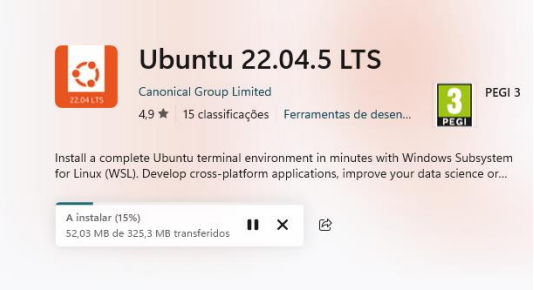



**Enable WSL and install Ubuntu:**
Open Windows PowerShell and run:
**wsl --install -d Ubuntu**

This will install WSL and the default Ubuntu distribution.

After the WSL installation, restart your computer if prompted.
If wsl fails, try to open Microsoft Store and search for "Ubuntu". Install the desired version (e.g., Ubuntu 22.04).

**Start Ubuntu:**
Open Ubuntu from the Start menu.
Complete the setup by creating a username and password.
You now have a fully functional Linux shell.



# Using WSL with Ubuntu

## Run on CLI:

sudo apt update && sudo apt install docker.io -y

# Docker Desktop for Mac

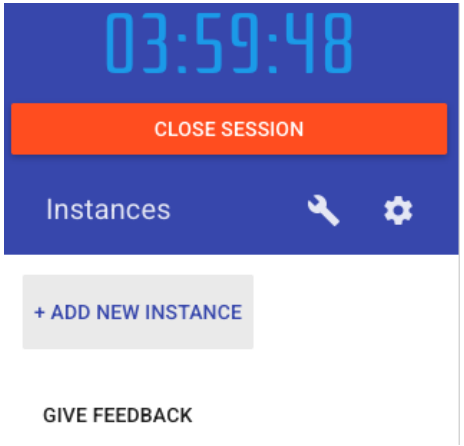
1. Download the Docker Desktop installer from:
   https://docs.docker.com/desktop/setup/install/mac-install/

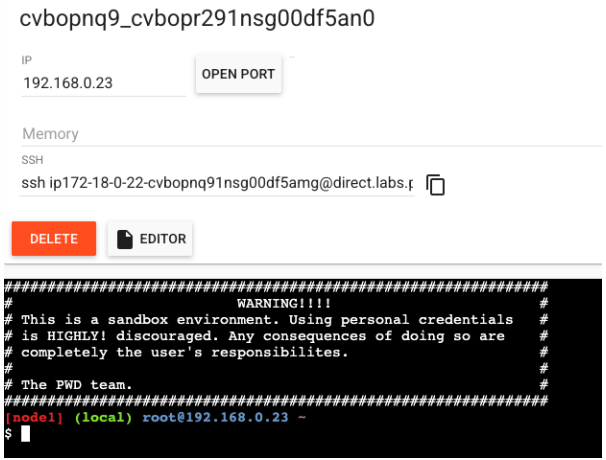
2. Double-click Docker.dmg to open the installer, then drag the Docker icon to the **Applications** folder. By default, Docker Desktop is installed at /Applications/Docker.app.
3. Double-click Docker.app in the **Applications** folder to start Docker.
4. The Docker menu displays the Docker Subscription Service Agreement.

# Using Docker Playground (PWD)

1. Access:

   **https://labs.play-with-docker.com/**
2. Loging using Docker (may need to create an account in **https://hub.docker.com/**)
3. Launch a new Instance



# Docker Commands (Useful)

| Command | Description |
|---|---|
| Docker --help | List available Docker commands and their descriptions |
| Docker --version | Display the installed Docker version |
| docker inspect <container/image> | Shows detailed information about a container or image |
| docker stats | Displays real-time resource usage of containers |
| docker system prune | Cleans up unused containers, images, and networks |

# Docker Commands (Image)

| Command | Description |
|---|---|
| docker pull <image> | Downloads an image from a registry (e.g., Docker Hub) |
| docker images | Lists all downloaded images on your system |
| docker rmi <image> | Removes an image from your system |
| docker build -t <image-name> . | Builds an image from a Dockerfile |
| docker commit | Create a new image from a container's changes |
| docker create | Create a new container without starting it |
| docker exec | Run a command inside an existing running container |
| docker pull | Download an image from Docker Hub |
| docker push | Upload an image to a registry (e.g., Docker Hub) |

# Docker Commands (Container)

| Command | Description |
|---|---|
| docker run <image> | Runs a container from an image |
| docker run -d <image> | Runs a container in detached (background) mode |
| docker run -it <image> bash | Runs a container interactively with a shell |
| docker ps | Lists all running containers |
| docker ps -a | Lists all containers (running + stopped) |
| docker stop <container> | Stops a running container |
| docker start <container> | Starts a stopped container |
| docker restart <container> | Restarts a container |
| docker rm <container> | Removes a container |
| docker logs <container> | Shows logs of a container |
| docker exec -it <container> bash | Enters a running container with an interactive shell |

# Docker Commands (Volume & Network)

| Command | Description |
|---|---|
| docker volume ls | Lists all volumes |
| docker volume create <name> | Creates a new volume |
| docker network ls | Lists all networks |
| docker network create <name> | Creates a new network |
| docker network connect <network> <container> | Connects a container to a network |

# Docker Commands (Volume & Network)

| Command | Description |
|---|---|
| docker-compose up | Starts containers defined in docker-compose.yml |
| docker-compose down | Stops and removes containers from docker-compose.yml |
| docker-compose ps | Lists containers managed by Docker Compose |

devoteam

# Lab 1

Creative tech for Better Change

# Exercise 1

Lets Check the Docker installation

```
$ docker -—version
```

Now lets run our first container

```
$ docker run hello-world
```

# Exercise 1

Can you find the size of the image we run?

```
$ docker images
```

What do we see when we list the running container?

```
$ docker ps
```

```
$ docker ps -a
```

To find more information about the image run:

```
$ docker inspect <imagename>
```

# Exercise 2

Create a new folder for out first container app

```
$ mkdir docker-app && cd docker-app
```

Create a simple app app.py (Python flask)?

```
from flask import Flask  # Import Flask framework

app = Flask(__name__)

@app.route('/')

def home():

    return "<h1>Welcome to the Fun Web App!</h1>"  # Define a simple web page

if __name__ == "__main__":

    app.run(host='0.0.0.0', port=5000)  # Start Flask web server
```

# Exercise 2

## Create a Dockerfile

```
# Use Python 3.9 as the base image

FROM python:3.9

WORKDIR /app

# Set the working directory inside the container

COPY . .

# Copy all application files into the container

RUN pip install flask

# Install Flask dependency

CMD ["python", "app.py"]

# Define command to run the app
```

# Exercise 2

Now lets build our image

```
$ docker build -t fun-app .
```

Run the container mapping the ports

```
$ docker run -d -p 5000:5000 fun-app
```

https://github.com/vladimiro-luz-bold/docker-lab1.git

# Exercise 3 - Docker Compose

Using Docker Compose. Start by creating a docker-compose.yml file

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
```

Start the services in detach mode

```
$ docker-compose up -d
```

# Exercise 4 - Pushing to a Container Registry

Start with login in Docker Hub

```
$ docker login
```

Lets tag our image

```
$ docker tag fun-app [username]/fun-app:v1
```

Push the image to the register

```
$ docker push [username]/fun-app:v1
```

# Exercise 5 - Persistent data storage in containers

Start by creating a new volume

```
$ docker volume create myvolume
```

Run a container with volume attached:

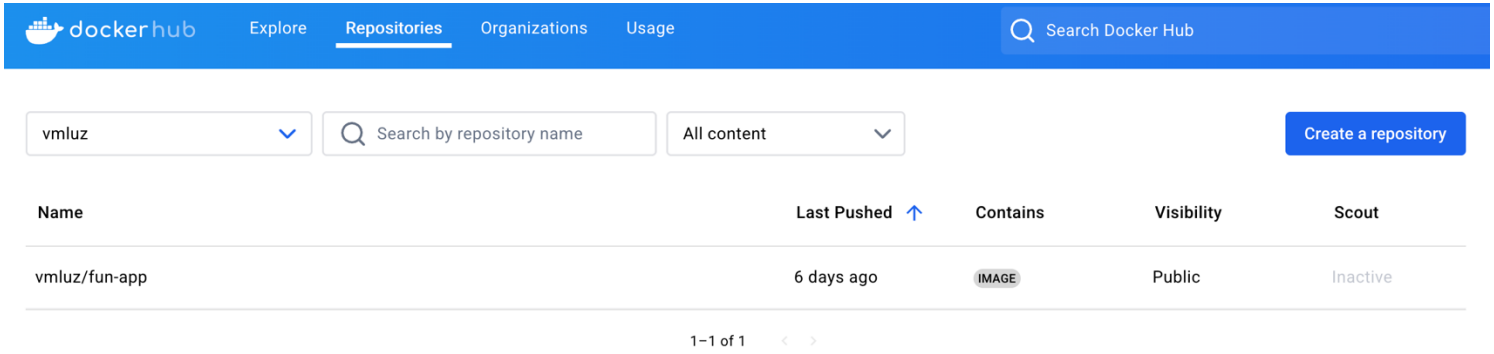
```
$ docker run -d --name mycontainer -v myvolume:/data busybox sleep 3600
```

Now lets inspect our container

```
$ docker inspect mycontainer
```

# Exercise 5 - Persistent data storage in containers

Now access Docker hub in your browser and check the images




# Exercise 6 - Creating and using Docker networks

Create two custom network

```
$ docker network create mynetwork1
$ docker network create mynetwork2
```

Run two containers in the same network:

```
$ docker run -d --name container1 --network mynetwork1 busybox sleep 3600
$ docker run -d --name container2 --network mynetwork1 busybox sleep 3600
```

Run an additional container on second network

```
$ docker run -d --name container3 --network mynetwork2 busybox sleep 3600
```

# Exercise 6 - Creating and using Docker networks

Testing the connection between container 1 and 2

```
$ docker exec -it container1 ping container2
```

What happens when we try to access container3 from container1

```
$ docker exec -it container1 ping container3
```

devoteam

# Thank **you.**