Docker

Docker is a platform that enables developers to create, deploy, and run applications in containers. Containers are lightweight, portable, and isolated environments that include everything needed to run the application, such as code, runtime, libraries, and dependencies.

From a practical standpoint it's just a way to package software so it can run on any hardware

How to Install Docker:

On Linux:

- nobody need it(it's just a joke, take it easy, I use Arch btw)

On Windows or macOS:

1. Download Docker Desktop from the official website.
2. Install the application and follow the on-screen instructions.
3. After installation, ensure Docker Desktop is running.

3 Main Things

- DockerFile - is a blueprint for building a docker image
- Docker Image - is a template for running docker containers
- Container - is just a running process

Useful Commands:

```
# Verify Docker is installed and working
docker --version

# Run a test container to check the installation
docker run hello-world

# List running containers
docker ps

# List all containers (running + stopped)
docker ps -a

# Pull an image from Docker Hub (example: nginx)
```

```
docker pull nginx

# Run a container from an image (nginx on port 80)
docker run -d -p 80:80 nginx

# Stop a container
docker stop <container_id>

# Remove a container
docker rm <container_id>

# Remove an image
docker rmi <image_name>

# Build an image from a Dockerfile
docker build -t <image_name> .

# View logs of a running container
docker logs <container_id>

# Start Docker service (Linux only)
sudo systemctl start docker

# Enable Docker to start on boot (Linux only)
sudo systemctl enable docker

# Add user to the Docker group (optional for Linux)
sudo usermod -aG docker $USER
```

> **How to use Docker**

Dockerfile

example:

```
# Base image
FROM python:3.9-slim

# Set the working directory
WORKDIR /app

# Copy the application code into the container
COPY . .
```

```
# Install required dependencies
RUN pip install -r requirements.txt

# define environment variables
ENV PORT=8080

# Expose the application port
EXPOSE 8080

# Command to run the application
CMD ["python", "app.py"]
```

Explanation of Dockerfile Instructions:

1. `FROM`: Specifies the base image. Example: `python:3.9-slim`.
2. `WORKDIR`: Sets the working directory inside the container.
3. `COPY`: Copies files from your local machine to the container.
4. `RUN`: Executes commands during the image build process (e.g., install dependencies).
5. `EXPOSE`: Exposes a port for external access to the container.
6. `CMD`: Specifies the command to run when the container starts.

Docker Image

We build docker image by running `docker build` command

```
# here you can specify name of your image and version
# <username>/<image-name>:<version>
docker build -t kazuhirosan/my-python-app:1.0 .
```

Also u can push thy image into cloud service

```
# You need to authenticate with the registry you're pushing the image to.
docker login

# Tag the image
# docker tag <local-image-name> <username>/<repository>:<tag>
docker tag my-python-app kazuhirosan/my-python-app:latest

# Once the image is tagged, push it to the registry:
# docker push <username>/<repository>:<tag>
docker push kazuhirosan/my-python-app:latest
```

Container

You use the `docker run` command to start a container based on an image.

```
# docker run [OPTIONS] <image_name>
docker run my-python-app

# here whe add -p flag and asign ports <Local>:<Container> to be able to see app running in browser
docker -p 5000:8080 my-python-app
```

To close Container you can run docker stop command, which:

- Gracefully stops a container by sending a `SIGTERM` signal to the main process inside the container, giving it time to clean up.
- If the process doesn't stop in a default timeout (10 seconds), it sends a `SIGKILL` signal to force termination.

```
# Stop a Specific Container
docker stop <container_id_or_name>

# Stop all containers
docker stop $(docker ps -q)
```

Or you can run `docker kill` command, which immediately stops the container by sending a `SIGKILL` signal

```
# kiiiil them AAAAALL!!!
docker kill <container_id_or_name>
```

Volume for Containers

When you want to your containers to share info, you must create volume, this can be done by typing those commands:

```
# Create a volume
docker volume create my-shared-volume
```

```
# Run the first container
docker run -d -v my-shared-volume:/app/data --name container1 my-container

# Run the second container
docker run -d -v my-shared-volume:/app/data --name container2 another-
container

# Verify the volume
docker volume ls
docker volume inspect my-shared-volume
```

Or you can create `docker-compose` file, and then run it by `docker-compose up` command, which will automatically create and share the volume between the two containers ex:

docker-compose.yml

```
version: '3'
services:
  app1:
    image: my-container
    volumes:
      - my-shared-volume:/app/data
  app2:
    image: another-container
    volumes:
      - my-shared-volume:/app/data

volumes:
  my-shared-volume:
```

console

```
docker-compose up
```

WSL2 Issues

Steps to Enable WSL2 Support

1. Enable Virtual Machine Platform and WSL
   Open a **PowerShell** window as Administrator and run the following commands:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all
/norestart
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-
Linux /all /norestart
```

2. **Enable Hyper-V (if required)**:
   If the error persists, you might also need to enable **Hyper-V**. Run:

```
dism.exe /online /enable-feature /featurename:Microsoft-Hyper-V-All /all
/norestart
```

3. **Ensure Virtualization is Enabled in BIOS**
   - **Reboot your computer** and enter the **BIOS/UEFI settings** (usually by pressing `Del`, `F2`, or another key during startup).
   - Look for **Intel VT-x** (or **AMD-V** if you're using an AMD processor) and **Intel VT-d** or **SVM Mode** (for AMD).
   - **Enable** these settings if they are disabled.
   - Save and exit the BIOS/UEFI settings.
4. **Install WSL2**:
   If you haven't done so already, install WSL2 by running in console:

```
wsl --set-default-version 2
```

5. **Restart Your PC**

No distro in WSL2

To check if you have any distro, you can type this command in console:

```
wsl -l -v
```

if you get such response `Windows Subsystem for Linux has no installed distributions.`, Congratulations! You don't have any distro, yahoo....

Install a Linux Distribution for WSL:

1. **List Available Distributions**: Open PowerShell and run the following command to see a list of available Linux distributions for WSL:

```
wsl --list --online
```

2. **Install a Distribution**: Choose a distribution from the list (e.g., **Ubuntu**, **Debian**, **Kali Linux**, etc.), and install it by running:

```
wsl --install -d <DistroName>
```

If you haven’t already set WSL 2 as the default version, run the following command:

```
wsl --set-default-version 2
```

You can verify installation by first command in topic.