#02 practical class

Docker virtualization

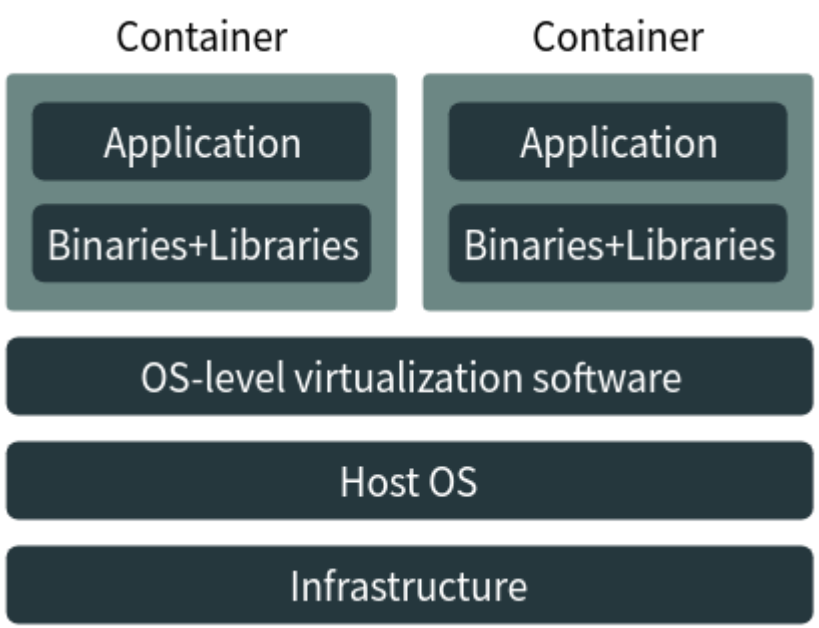
COMPUTING SYSTEMS AND INFRASTRUCTURES

(SISTEMAS E INFRAESTRUTURAS DE COMPUTAÇÃO)

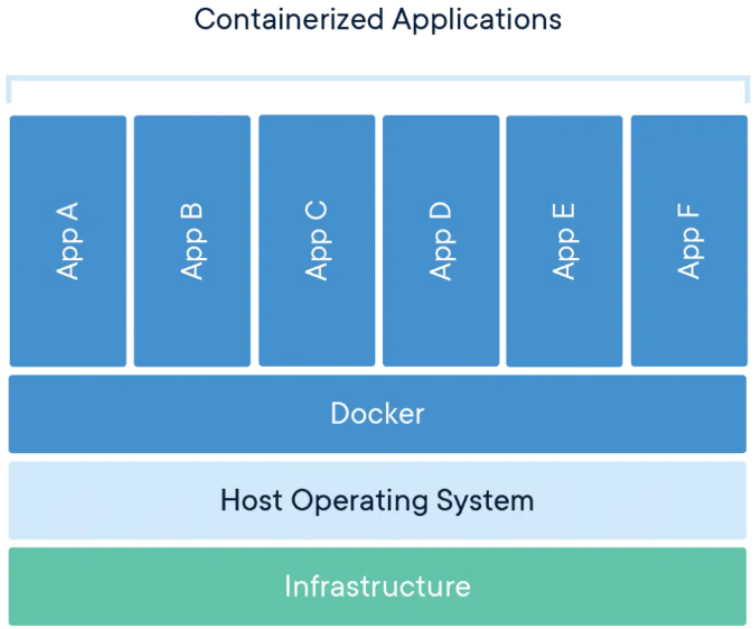
Overview

- Container-based virtualization
- Docker
- Using docker
- Building an image
- Deploying and testing docker

Container-based Virtualization



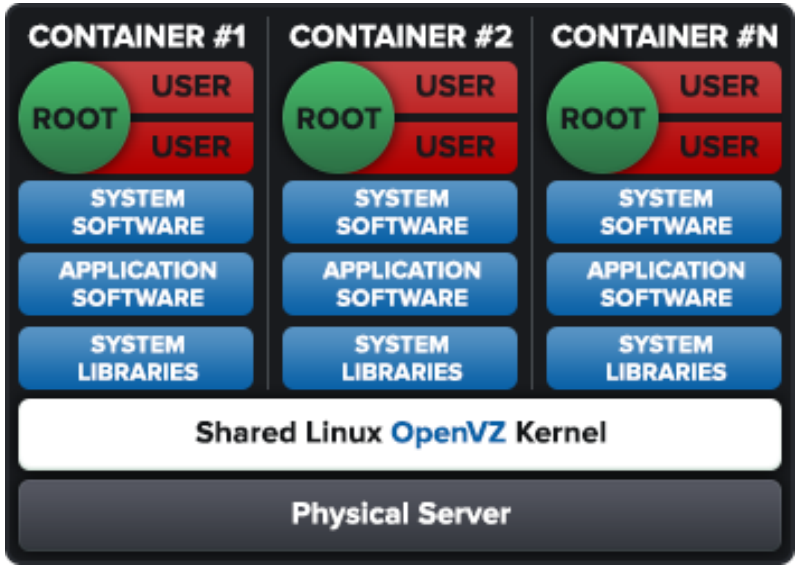
General view



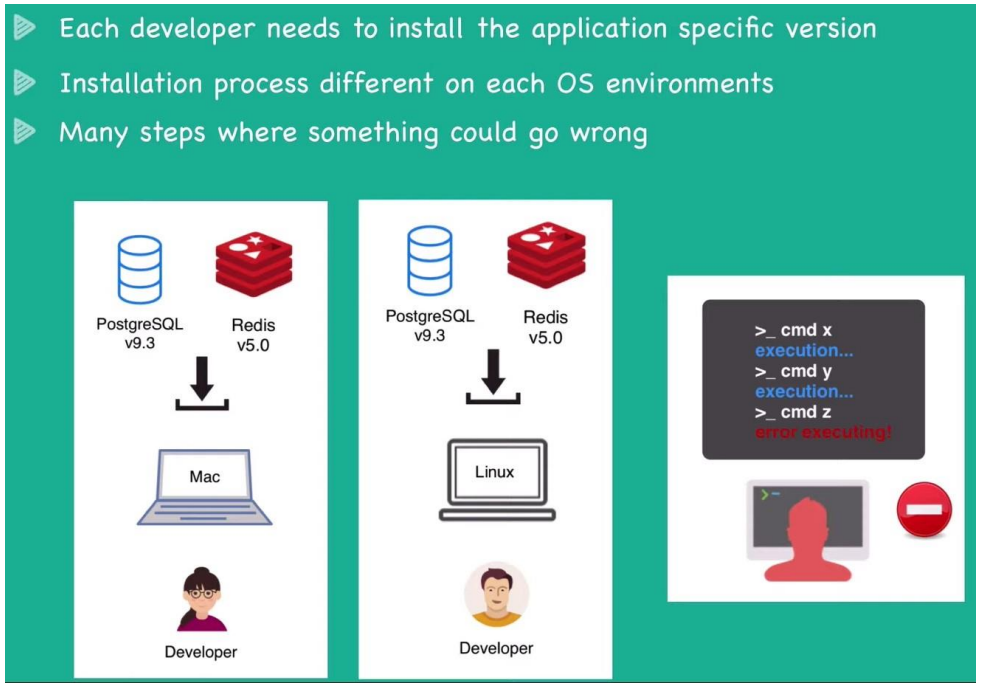
Docker view

Container-based Virtualization

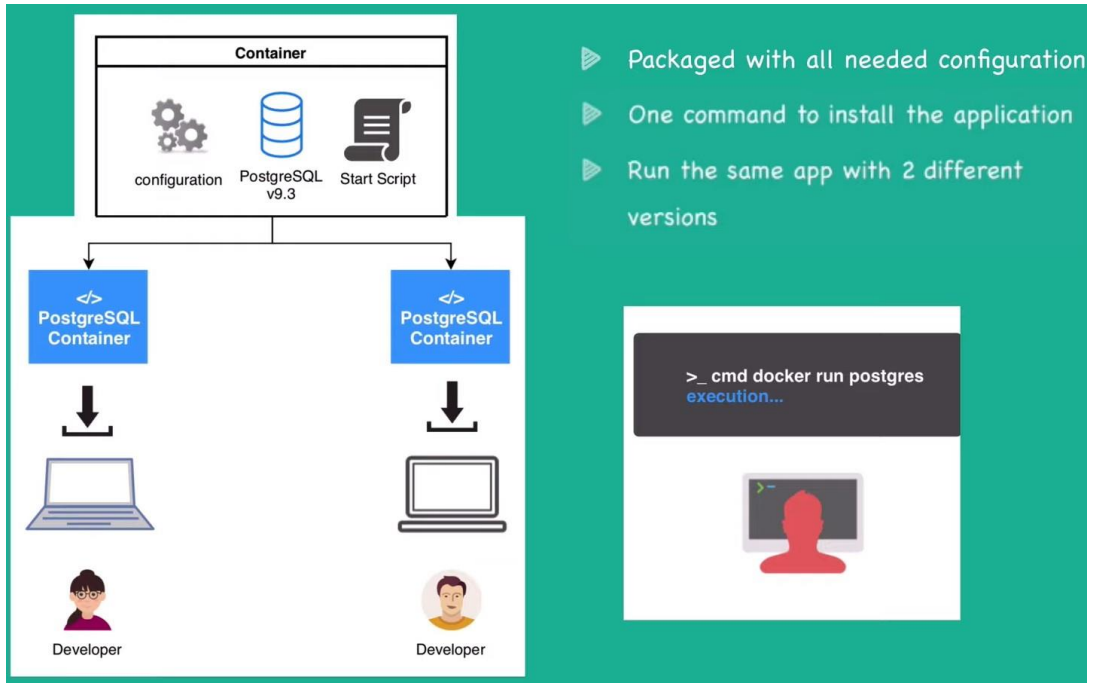
- Container-based virtualization for Linux
- Various isolated Linux containers run on a single physical server
- A container executes like a stand-alone server:
  - Started, stopped and rebooted independently
  - Uses a separate IP address
  - Communications between containers via a *bridging* interface
  - Resource usage *quotas* per container



Before & After containers



Before

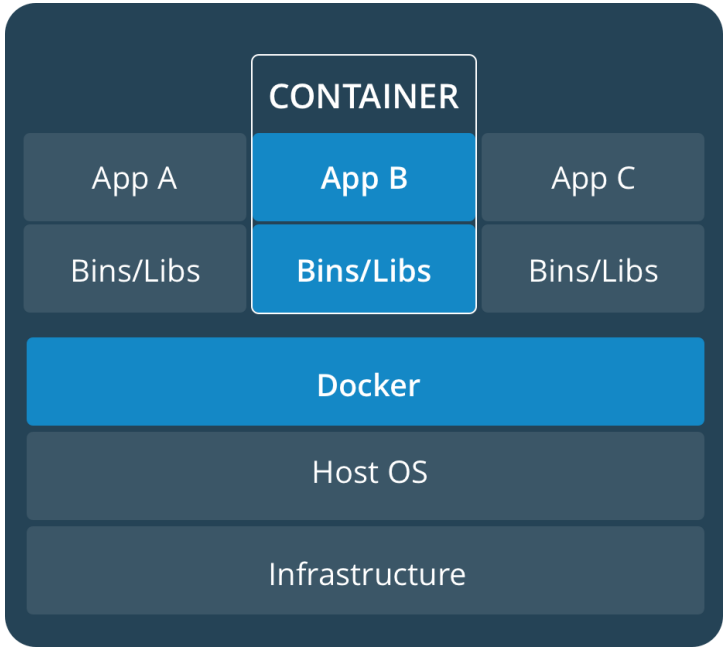


After

Docker

Docker is a platform for developers and sysadmins to build, run, and share applications with containers.

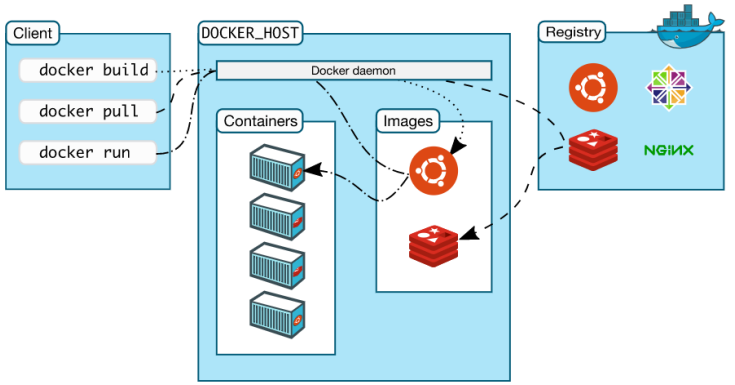
- **Flexible**: Even the most complex applications can be containerized.
- **Lightweight**: Containers leverage and share the host kernel, making them much more efficient in terms of system resources than virtual machines.
- **Portable**: You can build locally, deploy to the cloud, and run anywhere.
- **Loosely coupled**: Containers are highly self-sufficient and encapsulated, allowing you to replace or upgrade one without disrupting others.
- **Scalable**: You can increase and automatically distribute container replicas across a datacenter.
- **Secure**: Containers apply aggressive constraints and isolations to processes without any configuration required on the part of the user.

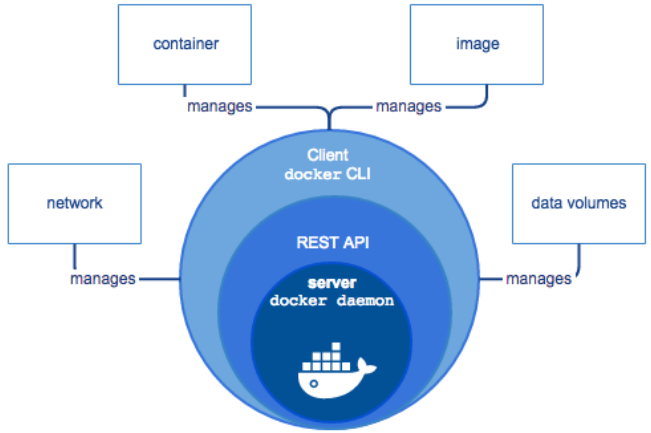


From https://docs.docker.com

Docker

- Docker daemon
  - The daemon (***dockerd***) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
- Docker client
  - The primary way that users interact with Docker.
  - The docker command uses the Docker API.
- Docker registries
  - Stores Docker images.
  - Docker Hub is a public registry that anyone can use.
  - You can run your own private registry.
  - When you use the docker pull or docker run commands, the required images are pulled from your configured registry.
  - When you use the docker push command, your image is pushed to your configured registry.





From https://docs.docker.com

Docker

- Images
  - An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.
  - You might create your own images or you might only use those created by others and published in a registry.
  - To build your own image, you must create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image.
- Containers
  - A container is a runnable instance of an image. You can create, start, stop, move, or delete a container. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
  - A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.
- Services
  - Services allow you to scale containers across multiple Docker daemons, which all work together as a swarm with multiple managers and workers. A service allows you to define the desired state, such as the number of replicas of the service that must be available at any given time. By default, the service is load-balanced across all worker nodes. To the consumer, the Docker service appears to be a single application.

From https://docs.docker.com

Using Docker

- docker pull
  - Download an image from a registry
- docker images
  - List images
- docker ps -a
  - List all the containers
- docker run
  - Create and run a new container from an image
- docker stop
  - Stop one or more running containers
- docker start
  - Start one or more stopped containers
- docker rm
  - Remove one or more containers
- docker inspect
  - Return low-level information on Docker objects
- docker logs
  - Fetch the logs of a container
- docker network
  - Manage networks

https://docs.docker.com/engine/reference/commandline/cli

Build an image: Dockerfile

Dockerfiles describe how to assemble a private filesystem for a container, and can also contain some metadata describing how to run a container based on this image.

Docker builds images automatically by reading the instructions from a Dockerfile (a text file that contains all commands, in order, needed to build a given image).

```
# Use the official image as a parent image
FROM node:current-slim

# Set the working directory
WORKDIR /usr/src/app

# Copy the file from your host to your current location
COPY package.json .

# Run the command inside your image filesystem
RUN npm install

# Inform Docker that the container is listening on the specified port at runtime.
EXPOSE 8080

# Run the specified command within the container.
CMD [ "npm", "start" ]

# Copy the rest of your app's source code from your host to your image filesystem.
COPY . .
```

https://docs.docker.com/develop/develop-images/dockerfile_best-practices

Build an image: Dockerfile

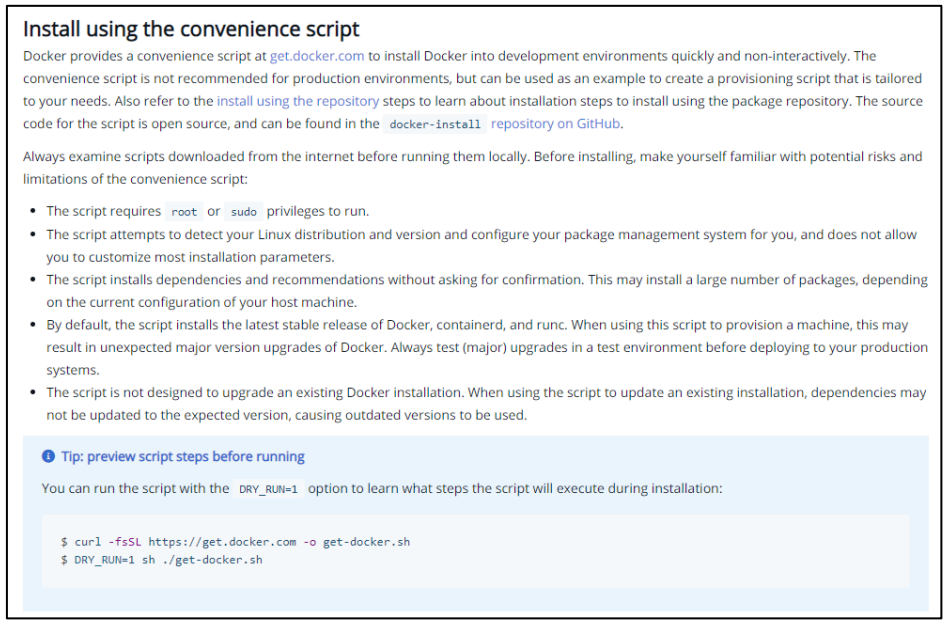
```
# docker image build -t test_docker:1.0
```

```
# docker container run --publish 8000:8080 --detach --name test_container test_docker:1.0
```

```
# docker container rm --force test_container
```

Deploying Docker

- Use the SIC VM provided
- Docker documentation
  - https://docs.docker.com/engine/install/debian
- Recommended script deployment



Install using the convenience script

Docker provides a convenience script at get.docker.com to install Docker into development environments quickly and non-interactively. The convenience script is not recommended for production environments, but can be used as an example to create a provisioning script that is tailored to your needs. Also refer to the install using the repository steps to learn about installation steps to install using the package repository. The source code for the script is open source, and can be found in the `docker-install` repository on GitHub.

Always examine scripts downloaded from the internet before running them locally. Before installing, make yourself familiar with potential risks and limitations of the convenience script:

- The script requires `root` or `sudo` privileges to run.
- The script attempts to detect your Linux distribution and version and configure your package management system for you, and does not allow you to customize most installation parameters.
- The script installs dependencies and recommendations without asking for confirmation. This may install a large number of packages, depending on the current configuration of your host machine.
- By default, the script installs the latest stable release of Docker, containerd, and runc. When using this script to provision a machine, this may result in unexpected major version upgrades of Docker. Always test (major) upgrades in a test environment before deploying to your production systems.
- The script is not designed to upgrade an existing Docker installation. When using the script to update an existing installation, dependencies may not be updated to the expected version, causing outdated versions to be used.

Tip: preview script steps before running

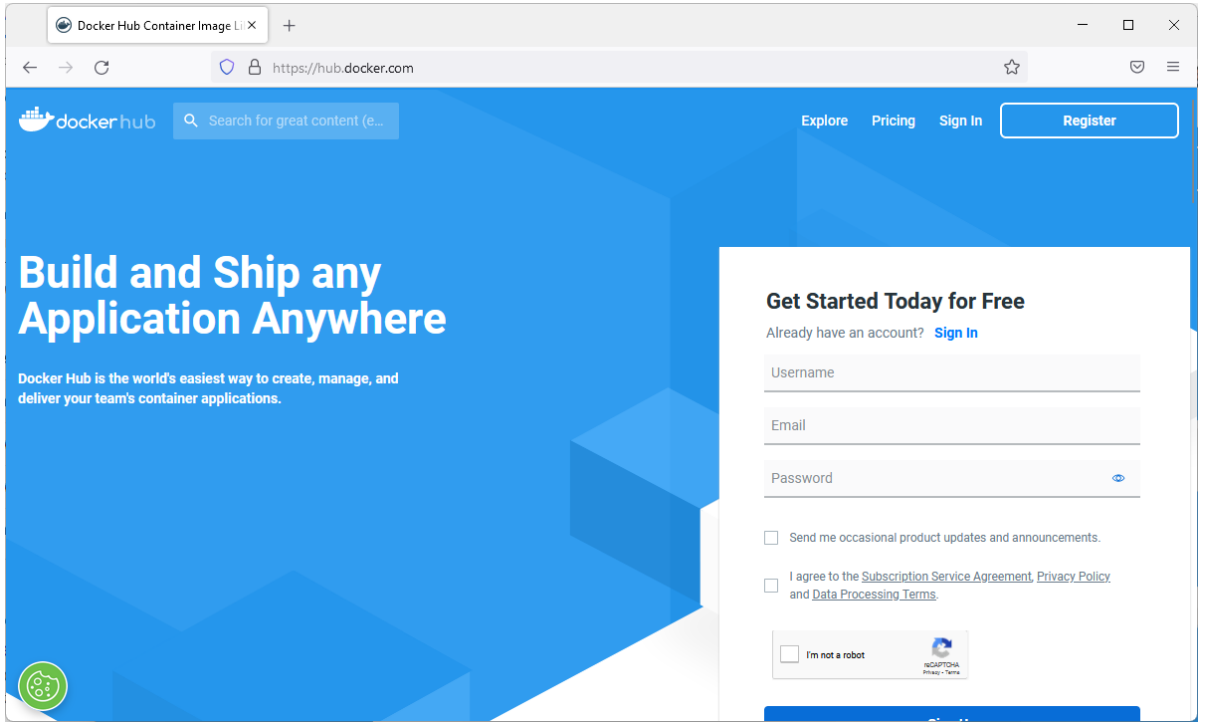
You can run the script with the `DRY_RUN=1` option to learn what steps the script will execute during installation:

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ DRY_RUN=1 sh ./get-docker.sh
```

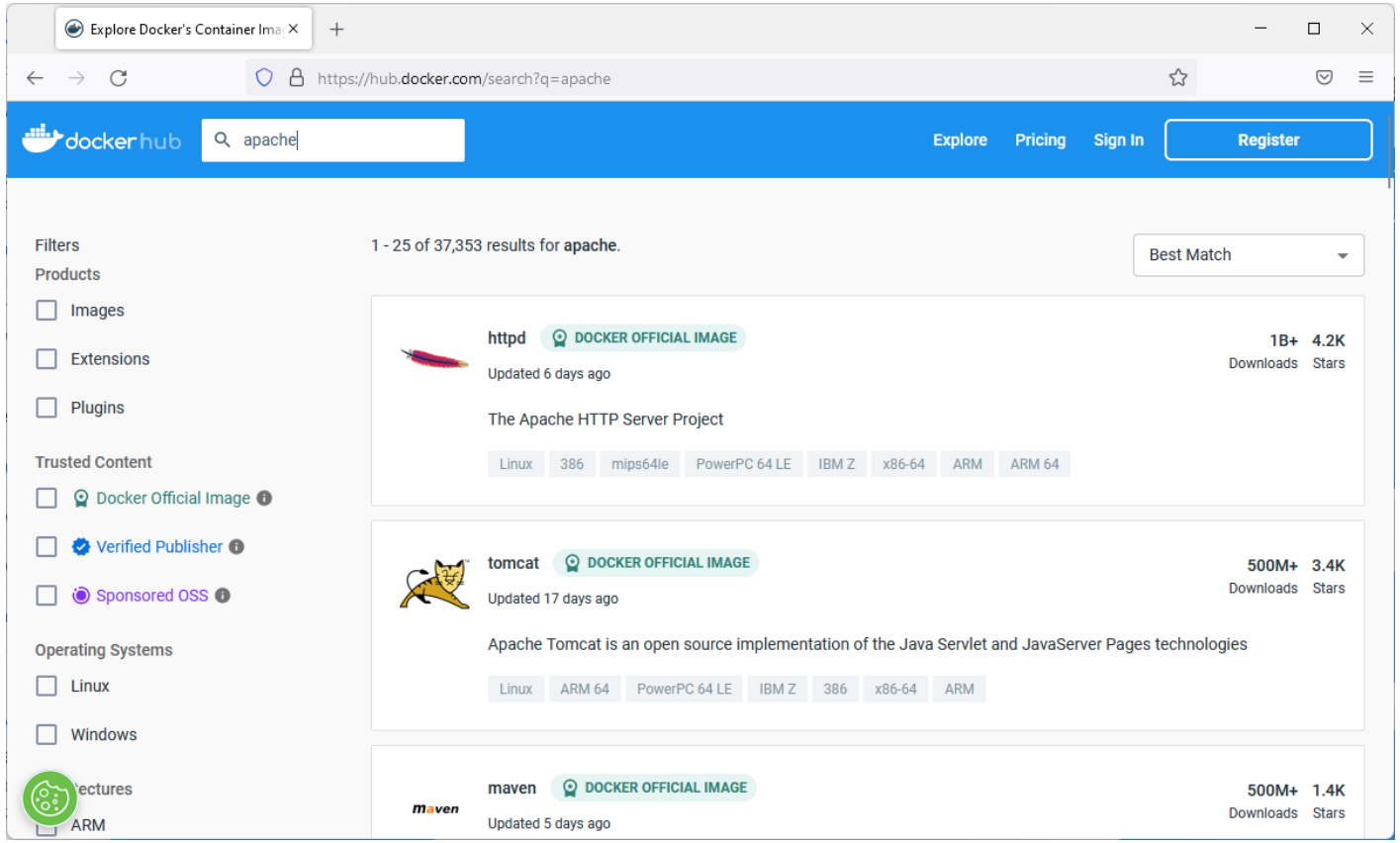
Docker Hub

- URL: https://hub.docker.com

- Public images repository
- Official and verified images
- Other images

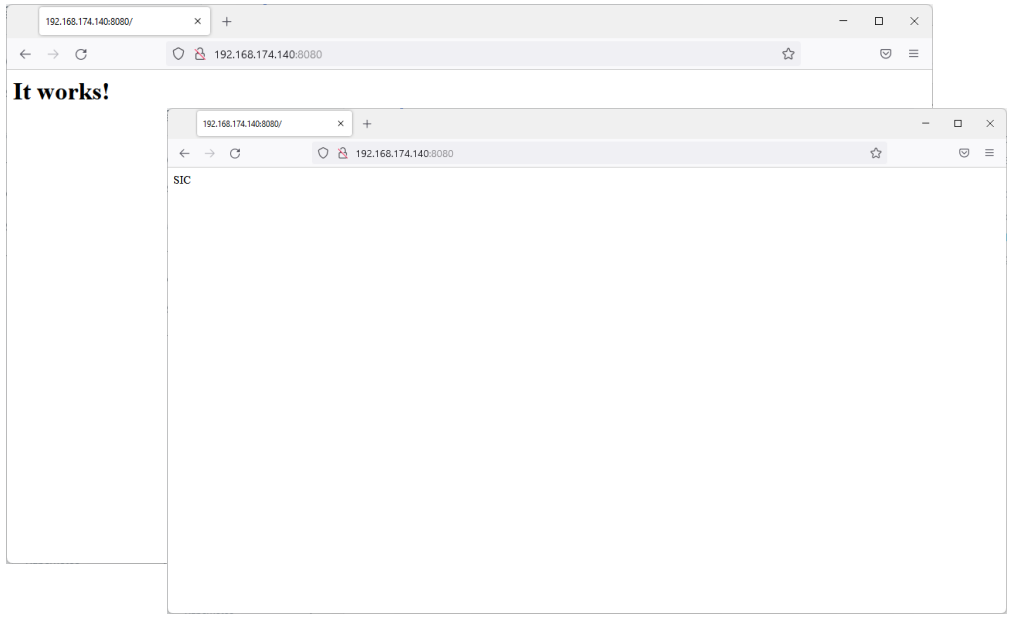


Docker Hub
Apache images



Web server

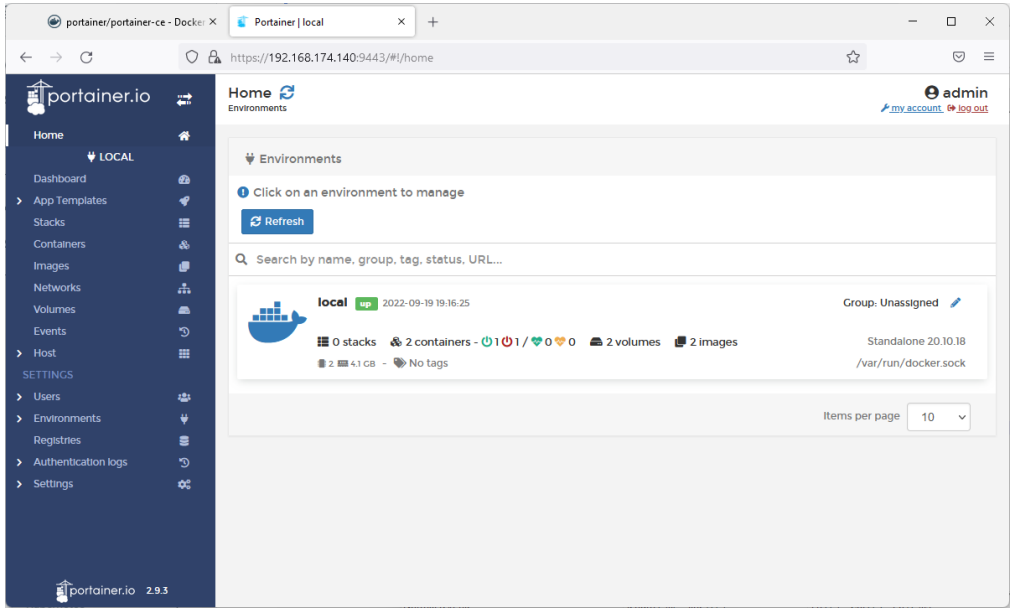
- URL: https://hub.docker.com/_/httpd
- Command: `docker run -d --name apache_server -p 8080:80 httpd`
- Website: `http://<VM-IP>:8080`
- Command: `docker exec -ti apache_server bash`
- Can you change the welcome message (**It works!**)?

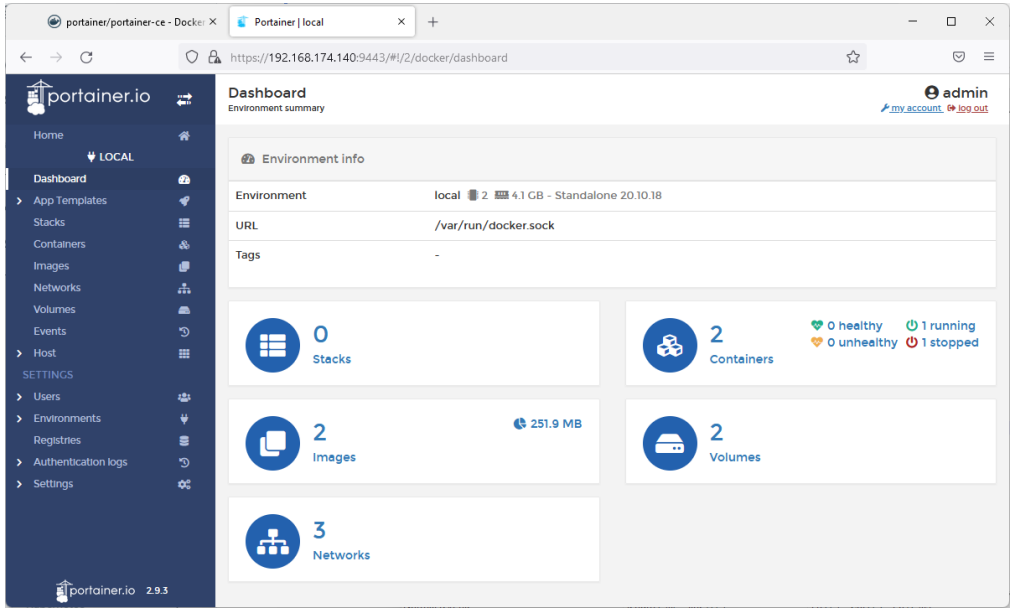


Portainer.io

- URL: https://docs.portainer.io/start/install-ce/server/docker/linux

```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
--restart=always -v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data portainer/portainer-ce:latest
```





Python script

- Create test script in /root/scripts/app.py

```
#!/usr/bin/env python3
import time

if __name__ == '__main__':
    print("Starting script...")
    for n in range(1,100):
        print("N: "+str(n))
        time.sleep(1)
    print("Ending script!")
```

- Run Python container:

```
docker run -dit --rm \
  --name python-script \
  -v /root/scripts:/usr/src/myapp \
  -w /usr/src/myapp \
  python:3 \
  python app.py
```

```
root@sic:~/script# docker run -dit --rm \
        --name python-script \
        -v /root/script:/usr/src/myapp \
        -w /usr/src/myapp \
        python:3 \
        python app.py
5b82f5534041dd33d7bfc21179266728b87df01cca941e961dea837abe70daf7
```

```
root@sic:~/script# docker logs python-script
Starting script...
N: 1
N: 2
N: 3
N: 4
N: 5
N: 6
N: 7
N: 8
```

Exercise

- Deploy Mosquitto MQTT docker container
- Use efrecon/mqtt-client to test the MQTT server (sub and pub)
- For more details, take a look at the **how-to-docker.txt** file in UCStudent