Docker

Technical Documentation

Prepared by
Sagar Lotiya

INDEX

Install Docker

- Install docker & docker compose

```
sudo apt-get update

Remove unwanted packages
for pkg in docker.io docker-doc docker-compose podman-docker
containerd runc; do sudo apt-get remove $pkg; done

rm -rf /var/lib/docker

sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

Add Docker’s official GPG key

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

Use the following command to set up the repository
echo \
  "deb [arch="$(dpkg --print-architecture)"
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null


sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin

sudo apt-get update

sudo groupadd docker

sudo usermod -aG docker $USER
```

```
newgrp docker
```

First Example using Composer

- **Docker network**
- **Dockerfile**
- **Compose file for building and running docker images**
- **Docker images**
- **How to search docker images?**
- **Docker context**
- **Docker volumes**
- **Docker single stage and multi stage**
- **Docker containers**

- Create docker network

```
docker network create --driver bridge --subnet 172.18.0.0/16 --gateway
172.18.0.1 practice-network
```

docker network create: This part of the command instructs Docker to create a new network.

--driver bridge: Specifies the network driver to be used. In this case, the bridge driver is used. **The bridge driver creates an isolated network that allows containers to communicate with each other, while also providing them with a bridge interface to communicate with the host system.**

--subnet 172.18.0.0/16: Defines the subnet for the network. This specifies the IP address range that containers within this network can use. In this case, the network's IP range is 172.18.0.0 to 172.18.255.255 with a subnet mask of /16, meaning the first two segments (octets) are fixed, and the last two octets are variable for each container.

--gateway 172.18.0.1: Sets the gateway IP address for the network. The gateway IP is the entry point for communication between containers in this network and external networks (such as the host system or other networks). In this case, the gateway IP is 172.18.0.1.

practice-network: This is the name of the network being created. You can choose any name you like. The network's name is used to reference it when connecting containers to this network.

- How to search for docker images?

```
Open:  https://hub.docker.com/search?q=&type=image
Select operating system & architecture:   Linux, x86_64
To check architecture in linux: uname -m
```

```
# Build stage
FROM golang:latest as builder

COPY ./GO /home/GO
WORKDIR /home/GO
RUN go build main.go

# Final stage
FROM busybox
COPY --from=builder /home/GO/main /home/GO/main
CMD ["/home/GO/main"]
```

main.go

```
package main

import "fmt"

func main() {

      fmt.Println("hello world")

}
```

```
docker compose -f deployment/GO.yml build
docker compose -f deployment/GO.yml up
```

```
docker images
```

```
docker ps
```

The output indicates that we did not see an image running — because our app is not like a web server that runs continuously. For apps that run continuously, we would see a container image running continuously with all the other details.

Command

- List docker packages

```
dpkg -l | grep -I docker
```

- Docker cheat sheet
    - **GENERAL COMMANDS**

```
Start the docker daemon
docker -d

Get help with Docker. Can also use -help on all subcommands
docker --help

Display system-wide information
docker info
```

    - **IMAGES**

```
Build an Image from a Dockerfile
docker build -t <image_name>

Build an Image from a Dockerfile without the cache
docker build -t <image_name> . -no-cache

List local images
docker images

Delete an Image
docker rmi <image_name>

Remove all unused images
docker image prune

Remove all unused docker things
docker system prune
```

- **CONTAINERS**

```
Create and run a container from an image, with a custom name:
docker run --name <container_name> <image_name>

Run a container with and publish a container's port(s) to the host.
docker run -p <host_port>:<container_port> <image_name>

Run a container in the background
docker run -d <image_name>

Start or stop an existing container:
docker start|stop <container_name> (or <container-id>)

Remove a stopped container:
docker rm <container_name>

Open a shell inside a running container:
docker exec -it <container_name> sh
```

```
Fetch and follow the logs of a container:
docker logs -f <container_name>

To inspect a running container:
docker inspect <container_name> (or <container_id>)

To list currently running containers:
docker ps

List all docker containers (running and stopped):
docker ps --all

View resource usage stats
docker container stats
```

- **VOLUME**

```
docker volume ls
docker volume rm  <NAME OF VOLUME>
```

- NETWORK

```
docker network ls
docker network rm <NAME OF NETWORK>
docker network prune
```

- **DOCKER HUB**

```
Login into Docker
docker login -u <username>

Publish an image to Docker Hub
docker push <username>/<image_name>
```

```
Search Hub for an image
docker search <image_name>

Pull an image from a Docker Hub
docker pull <image_name>
```

- Go inside docker container

```
docker exec -u root -it CONTAINER_ID /bin/sh -c "YOUR_COMMAND"
```

```
docker exec -u USER_ID:USER_GROUP -it CONTAINER_NAME /bin/sh
```

- Clear disk space

```
sudo su

cat <<EOF >> /etc/cron.daily/clear-tag-none
#!/bin/sh
docker rmi $(docker images -f "dangling=true" -q)
EOF

chmod +x /etc/cron.daily/clear-tag-none

cat <<EOF >> /etc/cron.daily/clear-container-logs
#!/bin/sh
cat /dev/null > /var/lib/docker/containers/*/*-json.log
EOF

chmod +x /etc/cron.daily/clear-container-logs
```

- Create local docker registry

  https://hub.docker.com/_/registry?_gl=1*1o8yo7z*_ga*MTQ3NDc4MjUxMC4xNjU

1NzkzODkw*_ga_XJWPQMJYHQ*MTY5MjQxNjY4NS4xMS4xLjE2OTI0MTY2ODYuNTkuMC4w

```
docker run -d -p 5000:5000 --restart=always --name registry
registry:2.8.2

docker image tag <YOUR_IMAGE> localhost:5000/<TAG_NAME>

docker push localhost:5000/<TAG_NAME>
```

The -d flag will run the container in detached mode. The -p flag publishes port 5000 on your local machine's network. We also give our container a name using the --name flag.

Uninstall Docker

- Uninstall docker

```
sudo apt-get purge docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin docker-ce-rootless-extras
```

- Remove Images, Volumes, Containers or Custom Configuration file

```
sudo rm -rf /var/lib/docker
sudo rm -rf /var/lib/containerd
```

A Complete Example using PHP, MySQL & Nginx

- Create local registry

```
docker run -d -p 5000:5000 --restart=always --name registry
registry:2.8.2
```

- Create, build & push corephp image [Refer Code]

```
docker build -t localhost:5000/corephp820:latest -f
deployment/docker/corephp/Dockerfile .
docker push localhost:5000/corephp820:latest
```

- Create mysql.yml [Refer Code]

```
docker compose -f deployment/mysql.yml build
docker compose -f deployment/mysql.yml up
```

Note: Docker provides restart policies to control whether your containers start automatically when they exit, or when Docker restarts. Restart policies ensure that linked containers are started in the correct order.

always: It will always restart the docker container unless it has been explicitly or manually stopped

- Create php-nginx.yml [Refer Code]

```
docker compose -f deployment/php-nginx.yml build
docker compose -f deployment/php-nginx.yml up
```

- Reload nginx without building it again

```
docker exec -it practice_nginx nginx -s reload
```

A Complete Example for Laravel as Backend

- Create laravel-nginx.yml [Refer Code]

```
docker compose -f deployment/laravel-nginx.yml build
docker compose -f deployment/laravel-nginx.yml up
```

```
docker exec -it 1000 laravel_php /bin/sh
composer create-project laravel/laravel ./
```

A Complete Example for Node.js

```
https://nodejs.org/dist/latest-v20.x/docs/api/synopsis.html
```

- Create node.yml [Refer Code]

```
docker compose -f deployment/node.yml build
docker compose -f deployment/node.yml up
```

A Complete Example for React as Frontend

- Create react.yml [Refer Code]

```
docker compose -f deployment/react.yml build
docker compose -f deployment/react.yml up
```

```
docker cp practice_react:/web/react/node_modules
/PATH_TO_YOUR_DIRCTORY/react/
```

References