Docker Cheat Sheet

Table of Contents

---

1. Installation

Install Docker:

Update package list and install prerequisites:

```
sudo apt-get update

sudo apt-get install \

ca-certificates \

curl \

gnupg \

Lsb-release
```

Add Docker's official GPG key:

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL
https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Set up the stable repository:

```
echo \ "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
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 update
```

Install Docker Engine:

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

Verify Docker Installation:

```
docker --version
```

Install Docker Compose:

Download the current stable release of Docker Compose:

```
sudo curl -L
"https://github.com/docker/compose/releases/download/
$(curl -s
https://api.github.com/repos/docker/compose/releases/
latest | grep -Po '(?<="tag_name":
```

```
")[^"]*')/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

Apply executable permissions to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Verify Docker Compose Installation:

```
docker-compose --version
```

---

2. Basic Commands

Version Info:

```
docker --version
```

```
docker info
```

---

3. Container Lifecycle

Run a Container:

```
docker run -d -p <host_port>:<container_port> --name
<container_name> <image>
```

List Containers:

```
docker ps
```

```
docker ps -a
```

Stop, Start, Restart, Remove a Container:

```
docker stop <container_id>
```

```
docker start <container_id>
```

```
docker restart <container_id>
```

```
docker rm <container_id>
```

View Logs:
```
docker logs <container_id>
```

Execute Command in Running Container:
```
docker exec -it <container_id> <command>
```

```
docker exec -it <container_id> bash
```

---

4. Image Management

Build an Image:
```
docker build -t <image_name>:<tag> .
```

List Images:
```
docker images
```

Remove an Image:
```
docker rmi <image_id>
```

Pull an Image:
```
docker pull <image>
```

Push an Image:
```
docker push <image>
```

Tag an Image:
```
docker tag <existing_image>:<tag> <new_image>:<tag>
```

---

5. Networking

List Networks:
```
docker network ls
```

Create a Network:
```
docker network create <network_name>
```

Connect a Container to a Network:
```
docker network connect <network_name> <container_id>
```

Disconnect a Container from a Network:
```
docker network disconnect <network_name>
<container_id>
```

---

6. Volumes

List Volumes:
```
docker volume ls
```

Create a Volume:
```
docker volume create <volume_name>
```

Attach a Volume to a Container:
```
docker run -v <volume_name>:/path/in/container
<image>
```

Remove a Volume:
```
docker volume rm <volume_name>
```

---

7. Docker Compose

Start Services:
```
docker-compose up
```

Start Services in Detached Mode:
```
docker-compose up -d
```

Stop Services:
```
docker-compose down
```

View Logs:
```
docker-compose logs
```

Execute Command in a Service:
```
docker-compose exec <service_name> <command>
```

---

8. Docker Scout

Install Docker Scout:
```
curl -fsSL
https://raw.githubusercontent.com/docker/scout-cli/ma
in/install.sh -o install-scout.sh

sh install-scout.sh
```

Check Version:
```
docker scout version
```

Analyze Image for CVEs:
```
docker scout cves <image>
```

---

9. Security

Scan an Image for Vulnerabilities:
```
docker scan <image>
```

Login to Docker Hub:
```
docker login
```

Logout from Docker Hub:
```
docker logout
```

---

10. Dockerfile Instructions

Basic Dockerfile Example:
```
# Use an official OpenJDK runtime as a parent image

FROM openjdk:8-jre-alpine
```

```
# Set the working directory

WORKDIR /app

# Copy the local file into the container

COPY . .

# Make port 8080 available to the world outside this container

EXPOSE 8080

# Run the application

ENTRYPOINT ["java", "-jar", "app.jar"]
```

Common Instructions:

- `FROM`: Specifies the base image.
- `WORKDIR`: Sets the working directory.
- `COPY`: Copies files from the host to the container.
- `RUN`: Executes commands in the container.
- `CMD`: Specifies the command to run within the container.
- `ENTRYPOINT`: Configures a container that will run as an executable.
