www.devopsshack.com
office@devopsshack.com

Search www.devopsshack.com

DOCKER
CHEAT SHEET

215K+ SUBSCRIBERS
46K+ FOLLOWERS

@adityajaiswal7
@devopsshack

CREATED BY

DevOps Shack

DevOps Shack

Docker Cheat Sheet

1. Docker Installation & Setup

Install Docker (Linux)

```
curl -fsSL https://get.docker.com
sudo systemctl start docker
sudo systemctl enable docker
```

Check Docker Version

```
docker --version
docker info
```

Run Docker Without `sudo`

```
sudo usermod -aG docker $USER
newgrp docker
```

Verify Installation

```
docker run hello-world
```

2. Docker Basics

Pull an Image

```
docker pull ubuntu
```

List Docker Images

```
docker images
```

Run a Container (Interactive Mode)

```
docker run -it ubuntu ba
```

Run a Container in Detached Mode

```
docker run -d ubuntu
```

Name a Container

```
docker run --name my_container ubuntu
```

Run a Container with a Specific Port

```
docker run -p 8080:80 nginx
```

List Running Containers

```
docker ps
```

List All Containers (Including Stopped)

```
docker ps -a
```

Stop a Running Container

```
docker stop container_id
```

Remove a Container

```
docker rm container_id
```

Remove All Stopped Containers

```
docker container prune
```

Remove an Image

```
docker rmi image_id
```

Remove All Unused Images

```
docker image prune -a
```

3. Docker Images

Create a Custom Docker Image

Create a Dockerfile:
dockerfile

```
FROM ubuntu

RUN apt-get update && apt-get install -y curl

CMD ["ba"]
```

Build the Image:

```
docker build -t my_custom_image .
```

Run the Image:

```
docker run -it my_custom_image
```

Tag an Image

```
docker tag my_custom_image myrepo/myimage:v1
```

Pu an Image to Docker Hub

```
docker login
```

```
docker pu myrepo/myimage:v1
```

4. Working with Containers

Execute Commands in Running Container

```
docker exec -it container_id ba
```

View Container Logs

```
docker logs container_id
```

Follow Live Logs

```
docker logs -f container_id
```

Copy Files Between Host & Container

```
docker cp file.txt container_id:/path/to/destination
```

```
docker cp container_id:/path/to/file.txt .
```

Restart a Container

```
docker restart container_id
```

5. Networking in Docker

List Networks

```
docker network ls
```

Create a Network

```
docker network create my_network
```

Connect a Container to a Network

```
docker network connect my_network container_id
```

Run a Container in a Custom Network

```
docker run --network=my_network alpine ping google.com
```

Disconnect a Container from a Network

```
docker network disconnect my_network container_id
```

Remove a Network

```
docker network rm my_network
```

6. Docker Volumes (Persistent Storage)

List Volumes

```
docker volume ls
```

Create a Volume

docker volume create my_volume

Run a Container with a Volume

docker run -v my_volume:/data ubuntu

Inspect a Volume

docker volume inspect my_volume

Remove a Volume

docker volume rm my_volume

Remove All Unused Volumes

docker volume prune

7. Docker Compose

Install Docker Compose

```
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

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

Example docker-compose.yml

yaml

```
version: "3.8"

services:

  web:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - my_data:/usr/are/nginx/html
volumes:
  my_data:
```

Start Services

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

Stop Services

```
docker-compose down
```

View Logs

```
docker-compose logs -f
```

8. Dockerfile Best Practices

Minimize Layers

dockerfile

```
FROM ubuntu
RUN apt-get update && apt-get install -y curl
```

Use Multi-Stage Builds

dockerfile

```
FROM golang:1.16 AS builder
WORKDIR /app
COPY . .
RUN go build -o myapp
FROM alpine
COPY --from=builder /app/myapp /myapp
CMD ["/myapp"]
```

Set a Working Directory

dockerfile

```
WORKDIR /app
```

Use Environment Variables

Dockerfile

```
ENV APP_ENV=production
```

Expose Ports

dockerfile

```
EXPOSE 8080
```

9. Security Best Practices

Use Non-Root User

dockerfile

```
RUN adduser -D myuser
USER myuser
```

Scan Images for Vulnerabilities

```
docker scan my_image
```

Limit Container Capabilities

```
docker run --cap-drop=ALL --cap-add=NET_BIND_SERVICE nginx
```

Read-Only Filesystem

```
docker run --read-only nginx
```

10. Debugging & Monitoring

View Running Processes in a Container

```
docker top container_id
```

View Stats

```
docker stats
```

Inspect a Container

```
docker inspect container_id
```

View Disk Usage

```
docker system df
```

Clean Up Unused Resources

```
docker system prune -a
```

11. Kubernetes with Docker

Enable Kubernetes (Docker Desktop)

```
kubectl get nodes
```

Deploy an App

yaml

```
apiVersion: apps/v1
kind: Deployment
Metadata:
  name: myapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
```

```
      app: myapp
    spec:
      containers:
      - name: myapp
        image: nginx
```

```
kubectl apply -f deployment.yaml
```

Expose a Service

```
kubectl expose deployment myapp --type=NodePort --port=80
```

Check Running Pods

```
kubectl get pods
```

12. Miscellaneous

Run a Temporary Container

```
docker run --rm -it alpine
```

Limit CPU & Memory Usage

```
docker run --memory=512m --cpus=1 nginx
```

Export & Import Containers

```
docker export container_id > container.tar
docker import container.tar new_image
```

Save & Load Images

```
docker save -o myimage.tar my_image
docker load -i myimage.tar
```

13. Docker Swarm (Native Orchestration)

Docker Swarm allows you to manage a cluster of Docker nodes and deploy services efficiently.

Initialize Docker Swarm

```
docker swarm init --advertise-addr <manager-ip>
```

Check Swarm Status

```
docker info | grep Swarm
```

Get Join Token for Worker Nodes

```
docker swarm join-token worker
```

Get Join Token for Manager Nodes

```
docker swarm join-token manager
```

Join a Worker Node

```
docker swarm join --token <token> <manager-ip>:2377
```

List All Nodes in Swarm

```
docker node ls
```

Promote a Worker to Manager

```
docker node promote worker-node
```

Demote a Manager to Worker

```
docker node demote manager-node
```

Leave Swarm (From Worker)

```
docker swarm leave
```

Remove a Node from Swarm

```
docker node rm worker-node
```

14. Deploying Services in Docker Swarm

Deploy a Service

```
docker service create --name web --replicas 3 -p 8080:80 nginx
```

List Running Services

```
docker service ls
```

List Tasks for a Service

```
docker service ps web
```

Scale a Service

```
docker service scale web=5
```

Update a Running Service

```
docker service update --image nginx:latest web
```

Remove a Service

```
docker service rm web
```

15. Docker Secrets (Secure Storage of Credentials)

Docker Secrets allow sensitive data (passwords, API keys) to be securely managed in Swarm.

Create a Secret

```
echo "my-secret-password" | docker secret create db_password -
```

List All Secrets

```
docker secret ls
```

Inspect a Secret

```
docker secret inspect db_password
```

Use Secret in a Service

```
docker service create --name mysql --secret db_password mysql:latest
```

Remove a Secret

```
docker secret rm db_password
```

16. Docker Configs (Manage Config Files)

Docker Configs allow configuration files to be stored securely for use by containers.

Create a Config

```
echo "server { listen 80; }" | docker config create nginx_config -
```

List Configs

```
docker config ls
```

Inspect a Config

```
docker config inspect nginx_config
```

Use Config in a Service

```
docker service create --name nginx --config nginx_config nginx
```

Remove a Config

```
docker config rm nginx_config
```

17. Docker BuildKit (Efficient Image Builds)

Enable BuildKit

```
export DOCKER_BUILDKIT=1
```

Use BuildKit in Dockerfile

dockerfile

```
# syntax=docker/dockerfile:1.2
```

Build an Image with BuildKit

```
docker build -t myimage --progress=plain .
```

Multi-Stage Build Example

dockerfile

```
# Build Stage
FROM golang:1.18 AS builder
WORKDIR /app
COPY . .
RUN go build -o myapp
# Runtime Stage
FROM alpine
COPY --from=builder /app/myapp /usr/local/bin/myapp
CMD ["myapp"]
```

18. Docker Security Best Practices

Scan Images for Vulnerabilities

```
docker scan myimage
```

Run Containers as Non-Root User

dockerfile

```
RUN adduser -D myuser
USER myuser
```

Use Read-Only Filesystem

```
docker run --read-only nginx
```

Limit Container Capabilities

```
docker run --cap-drop=ALL --cap-add=NET_BIND_SERVICE nginx
```

Block New Privileges

```
docker run --security-opt=no-new-privileges nginx
```

19. Docker Logging & Monitoring

View Container Logs

```
docker logs -f container_id
```

Use JSON Log Driver

```
docker run --log-driver=json-file nginx
```

Use Syslog Driver

```
docker run --log-driver=syslog nginx
```

Monitor Resource Usage

```
docker stats
```

Check Disk Usage

```
docker system df
```

20. Docker System Cleanup & Maintenance

Remove Unused Containers

```
docker container prune
```

Remove Unused Images

```
docker image prune -a
```

Remove Unused Networks

```
docker network prune
```

Remove Unused Volumes

```
docker volume prune
```

Clean Up Everything (Dangling Images, Containers, Volumes)

```
docker system prune -a --volumes
```

21. Docker Bench Security (Security Audit Tool)

Install Docker Bench

```
git clone https://github.com/docker/docker-bench-security.git
cd docker-bench-security
sudo  docker-bench-security.
```

22. Docker Multi-Host Networking with Overlay Networks

Create an Overlay Network

```
docker network create --driver overlay my_overlay
```

Run a Service in Overlay Network

```
docker service create --network my_overlay --name web nginx
```

Inspect Network

```
docker network inspect my_overlay
```

23. Docker Desktop Features

Enable Kubernetes in Docker Desktop

- Go to Settings > Kubernetes and enable it.

Verify:

```
kubectl get nodes
```

Run GUI Applications inside Docker (Linux)

```
xhost +local:
```

```
docker run -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix
firefox
```

24. Docker vs Podman (Rootless Containers)

Podman is a daemonless container engine.

Install Podman (Linux)

```
sudo apt install podman -y
```

Run a Container with Podman

```
podman run -it ubuntu ba
```

List Containers

```
podman ps
```

Alias Docker to Podman

```
alias docker=podman
```

25. Running Systemd Inside a Docker Container

Use Systemd in a Container

```
docker run -d --privileged --name systemd-container ubuntu:latest /sbin/init
```

Attach to Systemd

```
docker exec -it systemd-container ba
```

26. Running Docker-in-Docker (DIND)

Run Docker Inside a Container

```
docker run --privileged -d docker:dind
```

Use Docker CLI inside Container

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

27. Deploying Kubernetes with k3s in Docker

Install k3s

```
curl -sfL https://get.k3s.io |  -
```

Check Cluster Status

```
kubectl get nodes
```

Deploy an Application

```
kubectl create deployment myapp --image=nginx
```