# #_ [ Docker Commands and Operations ] ( Extended-CheatSheet )

## 1. Basic Docker Commands

- Check Docker version: `docker --version`
- Display system-wide information: `docker info`
- List available Docker commands: `docker`
- Display Docker disk usage: `docker system df`
- Show Docker system information: `docker system info`

## 2. Image Management

- List local images: `docker images`
- Pull an image: `docker pull ubuntu:latest`
- Remove an image: `docker rmi ubuntu:latest`
- Remove all unused images: `docker image prune -a`
- Search Docker Hub for images: `docker search nginx`
- Build an image from Dockerfile: `docker build -t myapp:1.0 .`
- Tag an image: `docker tag myapp:1.0 myrepo/myapp:1.0`
- Push an image to registry: `docker push myrepo/myapp:1.0`
- Save image to tar archive: `docker save -o myapp.tar myapp:1.0`
- Load image from tar archive: `docker load -i myapp.tar`
- Show image history: `docker history myapp:1.0`
- Inspect an image: `docker inspect myapp:1.0`
- List image layers: `docker image inspect myapp:1.0 --format '{{.RootFS.Layers}}'`
- Remove all dangling images: `docker image prune`
- Remove all unused images: `docker image prune -a`

## 3. Container Management

- Run a container: `docker run -d --name mycontainer nginx`
- List running containers: `docker ps`
- List all containers (including stopped): `docker ps -a`
- Stop a container: `docker stop mycontainer`
- Start a stopped container: `docker start mycontainer`
- Restart a container: `docker restart mycontainer`
- Remove a container: `docker rm mycontainer`
- Remove all stopped containers: `docker container prune`

- Force remove a running container: `docker rm -f mycontainer`
- Run a command in a running container: `docker exec -it mycontainer /bin/bash`
- Attach to a running container: `docker attach mycontainer`
- Copy files from container to host: `docker cp mycontainer:/app/file.txt /host/path/`
- Copy files from host to container: `docker cp /host/path/file.txt mycontainer:/app/`
- Show container logs: `docker logs mycontainer`
- Follow container logs: `docker logs -f mycontainer`
- Show container resource usage: `docker stats mycontainer`
- Rename a container: `docker rename oldname newname`
- Create a new image from container changes: `docker commit mycontainer mynewimage:1.0`
- Export a container's filesystem: `docker export mycontainer > mycontainer.tar`
- Import a container filesystem: `docker import mycontainer.tar mynewimage:1.0`

## 4. Networking

- List networks: `docker network ls`
- Create a network: `docker network create mynetwork`
- Remove a network: `docker network rm mynetwork`
- Connect a container to a network: `docker network connect mynetwork mycontainer`
- Disconnect a container from a network: `docker network disconnect mynetwork mycontainer`
- Inspect a network: `docker network inspect mynetwork`
- Remove all unused networks: `docker network prune`
- Create a bridge network: `docker network create --driver bridge mybridgenetwork`
- Create an overlay network: `docker network create --driver overlay myoverlaynetwork`
- Create a macvlan network: `docker network create -d macvlan --subnet=172.16.86.0/24 --gateway=172.16.86.1 -o parent=eth0 mymacvlan`

## 5. Volumes

- List volumes: `docker volume ls`

- Create a volume: `docker volume create myvolume`
- Remove a volume: `docker volume rm myvolume`
- Inspect a volume: `docker volume inspect myvolume`
- Remove all unused volumes: `docker volume prune`
- Create a container with a volume: `docker run -v myvolume:/app nginx`
- Create a container with a bind mount: `docker run -v /host/path:/container/path nginx`
- Create a tmpfs mount: `docker run --tmpfs /app/temp nginx`

## 6. Dockerfile Commands

- Set base image: `FROM ubuntu:20.04`
- Run a command: `RUN apt-get update && apt-get install -y nginx`
- Copy files: `COPY app/ /app/`
- Set working directory: `WORKDIR /app`
- Set environment variable: `ENV NODE_ENV=production`
- Expose a port: `EXPOSE 80`
- Set default command: `CMD ["nginx", "-g", "daemon off;"]`
- Set entry point: `ENTRYPOINT ["nginx"]`
- Add metadata: `LABEL version="1.0" description="My nginx container"`
- Add health check: `HEALTHCHECK CMD curl --fail http://localhost || exit 1`
- Set user: `USER nginx`
- Add build argument: `ARG VERSION=1.0`
- Use build argument: `ENV VERSION=${VERSION}`
- Set volume: `VOLUME /app/data`
- Add file from URL: `ADD https://example.com/file.txt /app/`
- Set shell: `SHELL ["/bin/bash", "-c"]`
- Use multi-stage build: `FROM build-stage AS build ... FROM runtime-stage`

## 7. Docker Compose

- Run services: `docker-compose up`
- Run services in detached mode: `docker-compose up -d`
- Stop services: `docker-compose down`
- List services: `docker-compose ps`
- View service logs: `docker-compose logs`
- Execute command in service: `docker-compose exec web /bin/bash`
- Build services: `docker-compose build`
- Pull service images: `docker-compose pull`

- Scale a service: `docker-compose up -d --scale web=3`
- Show composition configuration: `docker-compose config`
- Validate composition file: `docker-compose config -q`
- Run a one-off command: `docker-compose run web npm test`
- Force recreate containers: `docker-compose up --force-recreate`
- Stop and remove containers: `docker-compose down --rmi all --volumes`
- View network settings: `docker-compose network ls`

## 8. Docker Swarm

- Initialize a swarm: `docker swarm init`
- Join a swarm as a worker: `docker swarm join --token <worker-token> <manager-ip>:<port>`
- Join a swarm as a manager: `docker swarm join --token <manager-token> <manager-ip>:<port>`
- List nodes in swarm: `docker node ls`
- Create a service: `docker service create --name myservice nginx`
- List services: `docker service ls`
- Scale a service: `docker service scale myservice=5`
- Update a service: `docker service update --image nginx:1.19 myservice`
- Remove a service: `docker service rm myservice`
- View service logs: `docker service logs myservice`

## 9. Docker Stack

- Deploy a stack: `docker stack deploy -c docker-compose.yml mystack`
- List stacks: `docker stack ls`
- List services in a stack: `docker stack services mystack`
- List tasks in a stack: `docker stack ps mystack`
- Remove a stack: `docker stack rm mystack`

## 10. Docker Registry

- Log in to a registry: `docker login myregistry.azurecr.io`
- Log out from a registry: `docker logout myregistry.azurecr.io`
- Tag image for registry: `docker tag myimage:1.0 myregistry.azurecr.io/myimage:1.0`
- Push image to registry: `docker push myregistry.azurecr.io/myimage:1.0`
- Pull image from registry: `docker pull myregistry.azurecr.io/myimage:1.0`

By: Waleed Mousa

## 11. Docker System

- Remove unused data: `docker system prune`
- Remove all unused data: `docker system prune -a`
- Show Docker disk usage: `docker system df`
- Show real-time events: `docker system events`
- Show Docker version and info: `docker system info`

## 12. Docker Context

- List contexts: `docker context ls`
- Create a new context: `docker context create mycontext`
- Use a context: `docker context use mycontext`
- Inspect a context: `docker context inspect mycontext`
- Remove a context: `docker context rm mycontext`

## 13. Docker Security

- View security options: `docker info --format '{{.SecurityOptions}}'`
- Run a container with security options: `docker run --security-opt="apparmor=unconfined" nginx`
- Enable user namespace remapping: `dockerd --userns-remap="default"`
- Run container with read-only root filesystem: `docker run --read-only nginx`
- Run container with dropped capabilities: `docker run --cap-drop ALL nginx`

## 14. Docker Plugins

- List plugins: `docker plugin ls`
- Install a plugin: `docker plugin install vieux/sshfs`
- Enable a plugin: `docker plugin enable vieux/sshfs`
- Disable a plugin: `docker plugin disable vieux/sshfs`
- Remove a plugin: `docker plugin rm vieux/sshfs`

## 15. Docker Buildx

- List buildx builders: `docker buildx ls`
- Create a new builder: `docker buildx create --name mybuilder`
- Use a builder: `docker buildx use mybuilder`

By: Waleed Mousa

- Build and push multi-platform image: `docker buildx build --platform linux/amd64,linux/arm64 -t myimage:1.0 --push .`
- Inspect builder: `docker buildx inspect`

## 16. Docker Performance

- View container stats: `docker stats`
- Limit container CPU: `docker run --cpus=".5" nginx`
- Limit container memory: `docker run --memory=512m nginx`
- Set container CPU priority: `docker run --cpu-shares=512 nginx`
- Limit container IO: `docker run --device-write-bps /dev/sda:1mb nginx`

## 17. Docker Debugging

- View container processes: `docker top mycontainer`
- Inspect container changes: `docker diff mycontainer`
- View image layers: `docker history myimage:1.0`
- Debug a container with strace: `docker run --cap-add=SYS_PTRACE --security-opt seccomp=unconfined myimage strace -f -p 1`
- Get a core dump from a container: `docker run --ulimit core=-1 --security-opt seccomp=unconfined myimage`

## 18. Docker Configuration

- Configure default address pools: `dockerd --default-address-pool base=172.80.0.0/16,size=24`
- Configure log driver: `dockerd --log-driver json-file --log-opt max-size=10m --log-opt max-file=3`
- Configure registry mirrors: `dockerd --registry-mirror https://mirror.gcr.io`
- Configure insecure registries: `dockerd --insecure-registry 10.0.0.0/24`
- Configure Docker daemon with config file: `echo '{"debug": true}' > /etc/docker/daemon.json`

## 19. Docker Storage Drivers

- Use overlay2 storage driver: `dockerd --storage-driver=overlay2`
- Use devicemapper storage driver: `dockerd --storage-driver=devicemapper`
- Configure devicemapper options: `dockerd --storage-opt dm.thinpooldev=/dev/mapper/thin-pool`

- Configure overlay2 options: `dockerd --storage-opt overlay2.override_kernel_check=true`

## 20. Docker Networking Advanced

- Create an ipvlan network: `docker network create -d ipvlan --subnet=192.168.1.0/24 -o ipvlan_mode=l2 ipvlannet`
- Create a user-defined bridge network with subnet: `docker network create --subnet 172.18.0.0/16 customnet`
- Configure DNS for a container: `docker run --dns 8.8.8.8 nginx`
- Add extra hosts to a container: `docker run --add-host host.docker.internal:host-gateway nginx`
- Use host networking: `docker run --network host nginx`
- Use container networking: `docker run --network container:mycontainer nginx`

## 21. Docker API

- Get Docker version via API: `curl --unix-socket /var/run/docker.sock http://localhost/version`
- List containers via API: `curl --unix-socket /var/run/docker.sock http://localhost/containers/json`
- Create a container via API: `curl -X POST --unix-socket /var/run/docker.sock -H "Content-Type: application/json" -d '{"Image":"nginx"}' http://localhost/containers/create`
- Start a container via API: `curl -X POST --unix-socket /var/run/docker.sock http://localhost/containers/{id}/start`
- Stop a container via API: `curl -X POST --unix-socket /var/run/docker.sock http://localhost/containers/{id}/stop`

## 22. Docker Content Trust

- Enable Docker Content Trust: `export DOCKER_CONTENT_TRUST=1`
- Sign an image: `docker trust sign myregistry.azurecr.io/myimage:1.0`
- Add a signer: `docker trust signer add --key cert.pem myname myregistry.azurecr.io/myimage`
- View signature information: `docker trust inspect myregistry.azurecr.io/myimage:1.0`
- Remove signature: `docker trust revoke myregistry.azurecr.io/myimage:1.0`

## 23. Docker Secrets (Swarm mode)

- Create a secret: `printf "mysecret" | docker secret create mysecret -`
- List secrets: `docker secret ls`
- Use secret in a service: `docker service create --name myservice --secret mysecret nginx`
- Remove a secret: `docker secret rm mysecret`

## 24. Docker Configs (Swarm mode)

- Create a config: `docker config create myconfig config.json`
- List configs: `docker config ls`
- Use config in a service: `docker service create --name myservice --config myconfig nginx`
- Remove a config: `docker config rm myconfig`

## 25. Docker Healthchecks

- Add healthcheck to Dockerfile: `HEALTHCHECK --interval=30s --timeout=10s CMD curl -f http://localhost/ || exit 1`
- Run container with custom healthcheck: `docker run --health-cmd="curl -f http://localhost/ || exit 1" --health-interval=30s nginx`
- View container health status: `docker inspect --format='{{.State.Health.Status}}' mycontainer`

## 26. Miscellaneous

- Get container IP address: `docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mycontainer`
- Get container environment variables: `docker exec mycontainer env`
- Get container mounted volumes: `docker inspect -f '{{range .Mounts}}{{.Source}} -> {{.Destination}}{{end}}' mycontainer`
- Run a container with a specific hostname: `docker run --hostname=myhost nginx`
- Set container DNS search domains: `docker run --dns-search=example.com nginx`
- Limit container logging: `docker run --log-opt max-size=10m --log-opt max-file=3 nginx`
- Override entrypoint: `docker run --entrypoint /bin/bash nginx`

## 27. Docker Experimental Features

- Enable experimental features: `export DOCKER_CLI_EXPERIMENTAL=enabled`

- Use squash option in build: docker build --squash -t myimage:1.0 .
- Use buildkit: DOCKER_BUILDKIT=1 docker build -t myimage:1.0 .
- Use checkpoint feature: docker checkpoint create mycontainer checkpoint1
- Restore from checkpoint: docker start --checkpoint checkpoint1 mycontainer

## 28. Docker Bench Security

- Run Docker Bench Security: docker run -it --net host --pid host --userns host --cap-add audit_control -v /var/lib:/var/lib -v /var/run/docker.sock:/var/run/docker.sock -v /usr/lib/systemd:/usr/lib/systemd -v /etc:/etc --label docker_bench_security docker/docker-bench-security

## 29. Docker Resource Constraints

195. Limit CPU usage: docker run --cpus=0.5 nginx
196. Set CPU shares: docker run --cpu-shares=512 nginx
197. Limit memory usage: docker run --memory=512m nginx
198. Set memory reservation: docker run --memory-reservation=256m nginx
199. Limit swap usage: docker run --memory-swap=1g nginx
200. Set kernel memory limit: docker run --kernel-memory=50m nginx

## 30. Docker Logging

- Use json-file logging driver: docker run --log-driver json-file nginx
- Use syslog logging driver: docker run --log-driver syslog nginx
- Set log rotation: docker run --log-opt max-size=10m --log-opt max-file=3 nginx
- Use gelf logging driver: docker run --log-driver gelf --log-opt gelf-address=udp://1.2.3.4:12201 nginx

## 31. Docker Manifest

- Create and push a manifest list: docker manifest create myrepo/myimage:latest myrepo/myimage:v1-linux-amd64 myrepo/myimage:v1-linux-arm64
- Push a manifest list: docker manifest push myrepo/myimage:latest
- Inspect a manifest: docker manifest inspect myrepo/myimage:latest

## 32. Docker Context

- Create a new context: `docker context create my-context --docker "host=ssh://user@host"`
- Use a specific context: `docker --context my-context ps`
- List available contexts: `docker context ls`

## 33. Docker Buildx (Advanced)

- Create a new builder instance: `docker buildx create --name mybuilder --driver docker-container`
- Use the new builder: `docker buildx use mybuilder`
- Build for multiple platforms: `docker buildx build --platform linux/amd64,linux/arm64,linux/arm/v7 -t myrepo/myimage:latest .`

## 34. Docker Compose with Swarm

- Deploy a stack with compose file: `docker stack deploy -c docker-compose.yml mystack`
- List stacks: `docker stack ls`
- List services in a stack: `docker stack services mystack`
- Remove a stack: `docker stack rm mystack`

## 35. Docker Stats and Monitoring

- View real-time container stats: `docker stats`
- View stats for specific containers: `docker stats container1 container2`
- Format stats output: `docker stats --format "table {{.Name}}\t{{.CPUPerc}}\t{{.MemUsage}}"`

## 36. Advanced Network Operations

- Create an overlay network for swarm: `docker network create --driver overlay myoverlaynet`
- Create a macvlan network: `docker network create -d macvlan --subnet=192.168.0.0/24 --gateway=192.168.0.1 -o parent=eth0 mymacvlan`
- Disconnect a container from a network: `docker network disconnect mynetwork mycontainer`
- Prune unused networks: `docker network prune`

## 37. Docker System Commands

- View system-wide information: `docker system info`
- Show docker disk usage: `docker system df`
- Remove unused data: `docker system prune --volumes`

## 38. Miscellaneous Advanced Operations

- Create a docker hub repository: `docker run --rm -it xd20110642/dockerhub-cli create myrepo`
- Set up docker content trust: `export DOCKER_CONTENT_TRUST=1`
- Use multi-stage builds to optimize image size: `FROM build-image AS build ... FROM runtime-image`
- Use BuildKit's new frontend: `#syntax=docker/dockerfile:1.2`
- Use heredoc syntax in Dockerfile (requires BuildKit): `RUN <<EOF ... EOF`
- Use SSH forwarding in builds: `RUN --mount=type=ssh ssh-add /path/to/key`