Running Containers

docker run -it ubuntu bash	Run container and specify command
docker run -it ubuntu	Run container
docker run -tid ubuntu	Run container detatched
docker create -ti ubuntu	Create a container without starting it
docker run -tid --name smelly-hippo ubuntu	named container
docker ps	show running containers
docker ps -a	show all containers
docker ps --filter name=web1	show matching containers
docker ps --filter name=web1 -q	show matching container ID
docker inspect smelly-hippo	inspect container

Container Lifecycle Stuff

docker start smelly-hippo	start
docker stop smelly-hippo	stop
docker stop smelly-hippo funny-frog	stop mutliple
docker restart smelly-hippo	restart container
docker pause smelly-hippo	pauses a running container, freeze in place
docker unpause smelly-hippo	unpause a container
docker wait smelly-hippo	blocks until running container stops
docker kill smelly-hippo	sends SIGKILL, faster than stop
docker rm smelly-hippo	remove
docker rm smelly-hippo funny-frog	remove multiple
docker rm -f smelly-hippo	force remove
docker container rm -f $(docker ps -aq)	Remove all containers, running or stopped

Docker Images

docker images	show images
docker history ubuntu	show history of image
docker image rm user1/funny-frog	remove image
docker image remove 113a43faa138	remove by id
docker image remove user1/funny-frog	remove image
docker rmi user1/funny-frog	remove image
docker rmi $(docker images -q)	remove all images
Commit container to an image:	
docker commit smelly-hippo	no repo name
docker commit smelly-hippo test1	repo name
docker commit smelly-hippo loworbitflux/test1	repo name
docker commit smelly-hippo loworbitflux/test1:my-update	tagged
docker commit smelly-hippo loworbitflux/test1:v1.2.3	tagged

Export / Import / Save / Load

docker export	export container to tarball archive stream
docker import	create image from tarball, excludes history (smaller image)
docker load	load an image from tarball, includes history (larger image)
docker save	save image to tar archive stream (includes parent layers)
Examples:	
docker load < my-image.tar.gz	
docker save my_image:my_tag | gzip > my-image.tar.gz	
cat my-container.tar.gz | docker import - my-image:my_tag	
docker export my-container | gzip > my-container.tar.gz	

Volumes / Storage

docker info | grep -i storage	check storage driver
docker inspect web	look for “Mounts”
docker volume ls	show voluems
docker volume create testvol1	create a volume
docker volume inspect testvol1	inspect a volume
docker volume ls -f dangling=true	find dangling (unused) volumes
docker volume rm volume1	remove volume
Running containers with volumes:	
docker run -d --name test1 -v /data ubuntu	unamed volume mounted on /data
docker run -d --name test2 -v vol1:/data ubuntu	named volume
docker run -d --name test3 -v /src/data:/data ubuntu	bind mount
docker run -d --name test4 -v /src/data:/data:ro ubuntu	RO
docker run -d --volumes-from test2 --name test5 ubuntu	storage can be shared
docker rm -v test1	remove container and unnamed volume
Access and sharing parameters:	
:ro	for read only
:z	shared all containers can read/write
:Z	private, unshared
-	
/var/lib/docker/overlay2	Defalt volume storage location on Ubuntu Linux

Resource Limits and Controls

docker run -tid -c 512 ubuntu	50% cpu
docker run -tid --cpuset-cpus=0,4,6 ubuntu	use these cpus
docker run -tid -m 300M ubuntu	limit memory
docker create -ti --storage-opt size=120G ubuntu	limit storage, not on aufs

Stats, Logs, and Events

docker stats	resourse stats for all containers
docker stats smelly-hippo	resource stats for one container
docker top smelly-hippo	shows processes in a container
docker logs web	container logs
docker events	watch events in real time
docker port nostalgic_colden	shows public facing port of container
docker diff practical_sinoussi	show changes to a container's file system

Docker Hub / Registry

docker login	Login to Registry
docker logout	Logout of Registry
docker tag 7d9495d03763 loworbitflux/smelly-hippo:latest	Tag an image
docker push loworbitflux/smelly-hippo	Push to registry
docker search mysql	Search for an image
docker pull mysql	Pull it down
docker run user1/funny-frog	Will be downloaded if it isn’t here

Building Docker Images From A Dockerfile

mkdir mydockerbuild	Create build dir
cd mydockerbuild	cd into build dir
vi Dockerfile	Edit build instructions
docker build -t mydockerimage .	Build the image (note the dot ".")
docker images	Show images
docker run mydockerimage	Run the new image

Simple Dockerfile Example

FROM ubuntu
RUN apt update
RUN apt install nginx -y
CMD ["/usr/sbin/nginx"]

Big Dockerfile Example

FROM ubuntu	base image
RUN apt update	run commands while building
RUN apt install nginx -y	run commands while building
WORKDIR ~/	working dir that CMD is run from
ENTRYPOINT echo	default application
CMD "echo" "Hello docker!"	main command / default application
CMD ["--port 27017"]	params for ENTRYPOINT
CMD "Hello docker!"	params for ENTRYPOINT
ENV SERVER_WORKS 4	set env variable
EXPOSE 8080	expose a port, not published to the host
MAINTAINER authors_name	deprecated
LABEL version="1.0"	add metadata
LABEL author="User One"	add metadata
USER 751	UID (or username) to run as
VOLUME ["/my_files"]	sets up a volume
COPY test relativeDir/	copies "test" to \`WORKDIR\`/relativeDir/
COPY test /absoluteDir/	copies "test" to /absoluteDir/
COPY ssh_config /etc/ssh/ssh_config	copy over a vile
COPY --chown=user1:group1 files* /data/	also changes ownership
ADD /dir1 /dir2	like copy but does more ...

Expose Ports

docker run -tid -p 1234:80 nginx	expose container port 80 on host port 1234
docker run -tid -p 80:5000 ubuntu	bind port
docker run -tid -p 8000-9000:5000 ubuntu	bind port to range
docker run -tid -p 80:5000/udp ubuntu	udp ports
docker run -tid -p 127.0.0.1:80:5000 ubuntu	bind port on an interface
docker run -tid -p 127.0.0.1::5000 ubuntu	bind any port, specific interface
docker run -tid -P ubuntu	exposed ports to random ports

Networks

docker network ls	show networks, bridge is default
docker network inspect bridge	show network details and connected containers
Create Bridge Network, Specify Subnet and Gateway:	
docker network create -d bridge my-network	
docker network create -d bridge --subnet 172.25.0.0/16 my-network	
docker network create --subnet 203.0.113.0/24 --gateway 203.0.113.254 my-network	
docker network rm my-network	remove network
Run container and specify network:	
docker run -tid --net=my-network --name test1 ubuntu	
Run container, specify network and IP:	
docker run -tid --net=my-network --ip=172.25.3.3 --name=test1 ubuntu	
Connect container to network:	
docker network connect net1 test1	
docker network connect net1 test2 --ip 172.25.0.102	
Disconnect container from network:	
docker network disconnect net1 test1	Disconnect container from this network
docker network disconnect -f test1 test2	Force disconnect
Find container's IP address:	
docker inspect -f '{{json .NetworkSettings.Networks}}' container1	
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' container1	