Docker >

Docker Cheat Sheet

IMAGES

Info

show images on host
```
docker images
```

get image ID by name
```
docker images --format="{{.ID}}"
myImageName
```

inspect image
```
docker inspect image:version | jq .
```

Remove

remove image by name
```
docker rmi reponame:tag
```

remove dangling images (w no names or tags)
```
docker image prune
```

remove all images that dont have a child container running
```
docker image prune -a
```

remove ALL images
```
docker rmi $(docker images -qa)
```

Build

build an image from Dockerfile
```
docker build -t user/image-name .
```

Dockerfile config:

```
FROM debian:wheezy  #provide a base image
RUN apt-get update && apt-get install -y cowsay  #run additional commands to build container

ENTRYPOINT /usr/bin/telnet

ADD /host_dir /target_dir #copy contents to container from the host

CMD "echo" "all done!"  #cmd to execute once container is built

ENV PROXY_SERVER http://www.proxy.com #set env vars

EXPOSE 8250  #expose port from service inside container to outside

USER 125  #UID of user running the container
```

CONTAINERS

Info

show all containers and their status
```
docker ps -a
```

get just container IDs
```
docker ps -qa
```

show diffs made to container
```
docker diff containerName
```

see all operations done inside the container
```
docker logs containerName (or container ID)
```

get JSON info about a container
```
docker inspect <containerName or containerID>
```

Bash into a container
```
docker exec -ti <container Name> bash
```

Bash into a STOPPED container,
```
docker run -it --entrypoint /bin/bash <image_id>
```

get JSON output of all containers
```
curl -s --unix-socket /var/run/docker.sock
http:/containers/json
```

SSH into existing running container
```
docker attach $containerID
docker exec -ti $containerID bash
```

bind container network to the underlying host
```
docker run --network host -d --rm -e
VAULT_DEV_ROOT_TOKEN_ID=abcd -p "8200:8200"
artifactory.corp.local/docker.io/vault:0.9.5

will listen on 0.0.0.0:8200
```

Run/Stop

run a container, and login via terminal
```
docker run -it imageName
```

run a container w specific hostname and name
```
docker run -it --name myName --hostname myHostname
imageName
```

run container as daemon, bind to a port, give container a name
```
docker run -d -p 5200:5200 --name myName imageName
```

Stop a running container

```
VOLUME ["/tmp"] #enable access from
container to dir on the host machine

WORKDIR ~/dev  #where CMD will execute
from

COPY file.conf /etc/app/app.conf #copy
from host to container
```

COMPOSE

run docker compose up,
```
docker-compose -f my-compose-file.yaml up
-d
```

stop container,
```
docker-compose stop
```

remove stopped containers
```
docker-compose rm -f
```

SWARM

start swarm on leader node
```
docker swarm init
```

leader node generates token

add node to swarm
```
docker swarm join \
>     --token <TOKEN> \
>     <IP>:2377
This node joined a swarm as a worker.
```

start a service from a docker-compose file
```
docker stack deploy -c docker-
compose.yaml jira-maestro
```

see all running services
```
docker service ls
```

inspect service
```
docker service inspect <service name>
```

see which Swarm nodes are running the service
```
docker service ps <service ID>
```

scale out a service to more instances
```
docker service scale <service name>=5 (or
# of instances)
```

remove a service
```
docker service rm <service name>
```

```
docker ps (get container ID)
docker stop $containerID (or by Name)
```

Stop all Running containers
```
docker ps -a --format="{{.ID}}" | xargs docker stop
```

Start container
```
docker start $containerID
```

exit the container if logged in via terminal
exit

run a container, make a container Volume be available to host, ie, can read Container's application logs directly from Docker host
```
docker run -p 5100:5100 --volume <path to volume on
host>:<path to volume on container>
docker run -p 5100:5100 --
volume /host/data/app1:/opt/container/app1
```

Remove

remove container by name
```
docker rm <name>
```

remove all stopped containers
```
docker container prune
```

remove all containers
```
docker rm $(docker ps -aq)
```

prune Everything (remove dangling images, containers, networks)
```
docker system prune -f
```

Container Quick run
Splunk
```
docker image pull splunk/splunk
docker run -d -e "SPLUNK_START_ARGS=--accept-license" -
e "SPLUNK_USER=root" -p "8000:8000" -p "8088:8088" -p
"8089:8089" --name splunk splunk/splunk
```