Docker Cheatsheet

Images

Blueprints of applications that form the basis of containers.

PULL IMAGE

```
> docker pull hello-world
> docker pull <username>/<img_tag>
```

LIST IMAGES

```
> docker images
```

BUILD IMAGE

```
> docker build .
# with tag
> docker build -t <img_tag> .
```

INSPECT IMAGE

```
> docker inspect <username>/<img_tag>
> docker history <username>/<img_tag>
```

TAG IMAGE

```
> docker tag <img_tag_or_id> <new_tag>
```

PUSH IMAGE

```
> docker build -t <username>/<img_tag> .
> docker push <username>/<img_tag>
```

REMOVE IMAGE(S)

```
> docker rmi <img_tag>
# remove all images
> docker rmi $(docker images -a -q)
```

Containers

Created from images to run applications

RUN CONTAINER

```
> docker run hello-world
> docker run <username>/<img_tag>
# detached with port mapping <host>:<container>
> docker run -d --name <name> -p 80:80 <img_tag>
```

RUN SHELL ON CONTAINER

```
> docker exec -it <name_or_id> /bin/bash
# run shell on last run container
> docker exec -it $(docker ps -aq) /bin/bash
# TIP: Detach with Ctrl + P, Ctrl + Q
```

GET LOGS FROM CONTAINER

```
> docker logs -f <name_or_id>
```

INSPECT CONTAINER

```
> docker inspect <name_or_id>
```

LIST CONTAINERS

```
> docker ps -a
# list only running containers
> docker ps
```

START/STOP CONTAINER

```
> docker start <name_or_id>
> docker stop <name_or_id>
# a less graceful shutdown with SIGKILL
> docker kill <name_or_id>
```

REMOVE CONTAINER(S)

```
> docker rm <name_or_id>
# remove all exited containers
> docker rm $(docker ps -aq -f status=exited)
# remove all containers
> docker rm $(docker ps -aq)
```

Sample Dockerfile

```
# our base image
FROM python:3.6-alpine

# install Python modules needed by the app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r \
    /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates /usr/src/app/templates/

# expose the container's port 9000
EXPOSE 9000

# run the application
CMD ["python", "/usr/src/app/app.py"]
```

Networks

CREATE A NETWORK

```
> docker network create --driver=bridge <name>
```

CONNECT TO A NETWORK

```
# run an image and connect to the network
> docker run -d --net <ntwrk_name> <img_name>
# connect a running container to a network
> docker network connect [OPTIONS] \
    <ntwrk_name> <container_name_or_id>
```

LIST NETWORKS

```
> docker network ls
```

INSPECT NETWORK

```
> docker network inspect <ntwrk_name>
```

REMOVE NETWORK

```
> docker network rm <ntwrk_name>
```

Smooth DevOps

With ❤ from mLab