DOCKER

Image

* A file which is essentially a snapshot of a container
* Images are created with the **build** command and they will produce a container when started with **run**
* images can be found online and installed via **docker pull**; e.g. **docker pull ubuntu**
* new images can be created from containers

|  |  |
| --- | --- |
| **Command** | **Description** |
| Docker images | View list of images |
| docker run -it repo:tag *bash* | -it interactive terminal  Repo:tag dockerImage:version |
| Docker ps | List of running containers |
| Docker rmi image-name | Remove image |

Workflow

*Example:*

1. run a container from an image
2. create a file
3. exit out of container
4. $ docker ps -l
   1. This gets the last container – make note of hash
5. Docker commit {hash}
6. New image has been created with the file present in step 2
7. Give image a name
   1. $ docker tag {hash} docker-image-name

Running processes in container

|  |  |
| --- | --- |
| **Command** | **Description** |
| Docker run **--rm** -it image-name *sleep 5* | --rm remove container when process finishes |
| Docker run -it **-d** image-name | -d detached (leave container running) |
| Docker **attach** {container-name} | attach |
| Docker **exec** -it {container-name} *bash* | Exec execute *bash* command |

*Ctrl + p & ctrl + q – exit out of container but leave running*

Manage containers

|  |  |
| --- | --- |
| **Command** | **Description** |
| Docker ps **-a** | View all processes including stopped containers |
| Docker **logs** {container-name} | View logs |
| Docker **kill** {container-name} | Kill process – this will stop container |
| Docker **system prune** | Purge all unused images/containers/networks… |

Container networking

* Programs in containers are isolated from the internet by default
* You can group your containers into *private* networks
* You explicitly set who can connect to whom

Expose ports to let connections in:

1. Explicitly specify the port inside the container and outside
2. Expose as many ports are you want

*Example:*

Docker run -it **-p** **45678:45678 -p 45679:45679** **–name** echo-server ubuntu:14:04 bash

-p specify the port – hostPort:dockerPort

--n name name the container

*Docker:*

$ nc -lp 45678 | nc -lp 45679

*Host terminal 1:*

$ nc localhost 45678

$ hello world

*Host terminal 2:*

$ nc localhost 45679

$ hello world

*nc – netcat:*

*a networking utility for reading from and writing to network connections using TCP or UDP*

Create a docker network

$ docker network create {name}

$ docker run --rm -it --net {networkName} --name cat-server ubuntu:14.04 bash

$ ping dog-server

$ ping cat-server

$ docker run --rm -it --net {networkName} --name dog-server ubuntu:14.04 bash

$ ping dog-server

$ ping cat-server

Volumes

* Virtual ‘disks’ to store and share data between containers and hosts
* Not part of images!

$ docker run -it **-v** /path/to/hostFolder:/path/in/container ubuntu bash

Even after the container is destroyed, the changes will persist on host. Try adding a file in the container shared folder and exiting from the container and checking the host location to see the file still has the changes made.

Docker files

*Dockerfile (simple example)*

FROM busybox

RUN echo “building simple docker image”

CMD echo “hello container”

$ docker build -t {image-name} . 🡨 notice ‘.’ Path

*Dockerfile (installing a program with docker build)*

FROM Debian:sid

RUN apt-get -y update

RUN apt-get install nano

CMD [“/bin/nano”, “/tmp/notes”]

|  |  |
| --- | --- |
| **Syntax** | **Description** |
| FROM | What image to start running from |
| MAINTAINER | Documentation of author/contact etc… |
| RUN | Run command line |
| ADD | Adds local files |
| ENV | Sets environment variables |
| ENTRYPOINT | Specifies the start of the command to run |
| CMD | Specifies the whole command to run |
| EXPOSE | Maps a port into the container |
| VOLUME | Defines volumes |
| WORKDIR | Set the working directory the container starts in |
| USER | Sets which use the container will run as |

*Dockerfile (node app – add this to the app root repo)*

FROM node:latest

WORKDIR /app

COPY package.json /app

RUN npm install

COPY . /app

CMD [“npm”, “start”]

$ docker build -t node-docker .

*Node-docker = name of image*

*. = location of dockerfile*

$ docker run -it -p 8000:3000 node-docker

Kubernetes introduction

* An open source platform designed to automate deploying, scaling and operations of containers
* Schedules and runs containers on *clusters* of virtual machines
* A *cluster* is a set of connected computers working together as a unified computing resource that can create the illusion of being one machine
* You can use docker containers to develop and build applications, and then use Kubernetes to run the applications on your infrastructure

QUIZ

1. Pull down the ubuntu image from the internet, remove and re-add
2. List all images
3. Create a container from an image and execute interactive shell at the same time
4. List all containers
5. Execute a command on a running container
6. Destroy all unused containers
7. Kill a running container
8. Expose a port on a container and send messages through it using 2 terminal processes on the host using netcat
9. Create 2 docker containers and ping the opposite container from each running container
10. Add a shared folder between the container and host using *volumes*
11. Create a dockerfile to run a basic node app
12. What is Kubernetes?