Noroff
School of technology
and digital media

Monday - Week 4

Working with Docker

What is Docker?

- Docker is a container platform in which your software runs
- It abstracts your software from the system on which it runs

- Docker is not a virtual machine*

Why use Docker? What does it let us do?

- Docker isolates our applications from the host
- It provides the application with all the necessities
- Unlike a VM it has low overhead
    - The individual apps can share certain resources

- It scales easily (Kubernetes)

Docker on windows

- Docker will ‘hook’ into the unix kernel on Linux (and Mac) but on Windows it needs more
- Docker will install and enable Hyper-V when installed on Windows
- This means that virtualisation software may have issues

- Alternatively: Create a Linux VM and install Docker in it

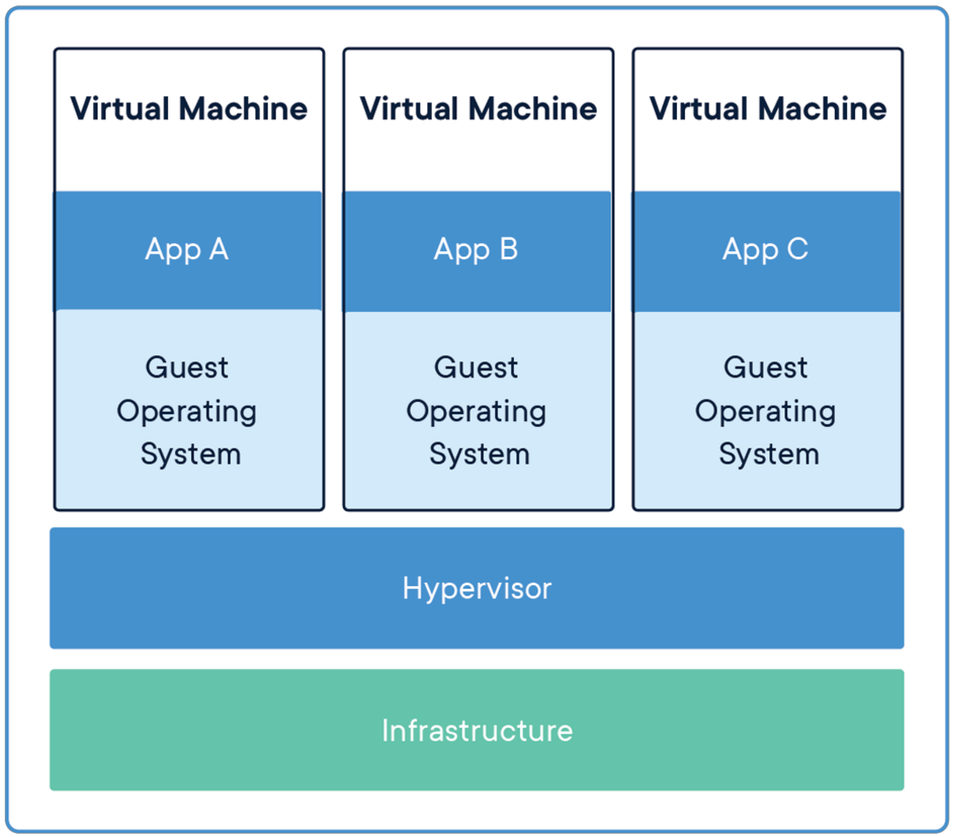
Installation (Windows)

- https://docs.docker.com/docker-for-windows/install/

  - Download
  - Install
  - Login

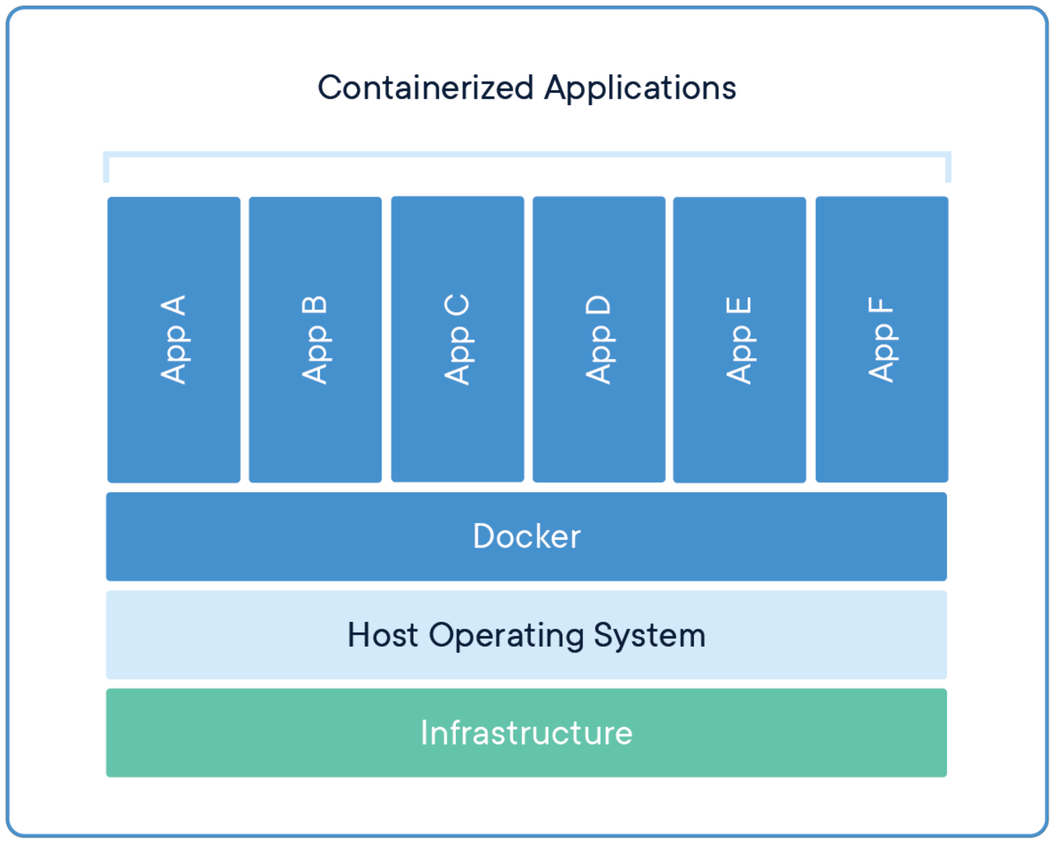
Virtual Machines

- Hypervisor allocates resources to VM's
- Each VM runs an OS
- Apps run in isolation



Containers

- Host OS runs Docker
- Docker allocates resources to Applications
- Apps share OS
- Apps have own user space



The Docker Process (Lifecycle?)

- Build Application
- Create Dockerfile
- Create Container (Dockerfile + Image)
- Run
- Pause/Stop/Kill

Dockerfile (sample)

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
ENTRYPOINT [“java”,"-Djava.security.egd=file:/dev/./urandom",
      "-jar","/app.jar"]
```

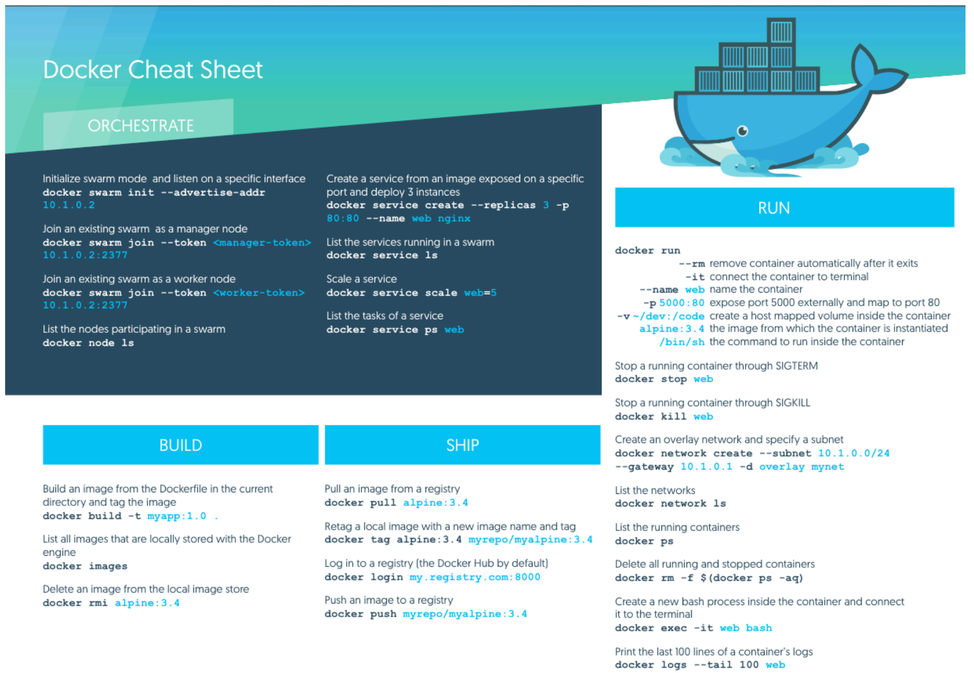
Layers and Best Practices

- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

- The Dockerfile is ‘layered’
- Each command should represent a change from the line before

Cheat Sheet

- https://www.docker.com/sites/default/files/Docker_CheatSheet_08.09.2016_0.pdf



Docker Cheat Sheet

ORCHESTRATE

Initialize swarm mode and listen on a specific interface
`docker swarm init --advertise-addr 10.1.0.2`

Join an existing swarm as a manager node
`docker swarm join --token <manager-token> 10.1.0.2:2377`

Join an existing swarm as a worker node
`docker swarm join --token <worker-token> 10.1.0.2:2377`

List the nodes participating in a swarm
`docker node ls`

Create a service from an image exposed on a specific port and deploy 3 instances
`docker service create --replicas 3 -p 80:80 --name web nginx`

List the services running in a swarm
`docker service ls`

Scale a service
`docker service scale web=5`

List the tasks of a service
`docker service ps web`

BUILD

Build an image from the Dockerfile in the current directory and tag the image
`docker build -t myapp:1.0 .`

List all images that are locally stored with the Docker engine
`docker images`

Delete an image from the local image store
`docker rmi alpine:3.4`

SHIP

Pull an image from a registry
`docker pull alpine:3.4`

Retag a local image with a new image name and tag
`docker tag alpine:3.4 myrepo/myalpine:3.4`

Log in to a registry (the Docker Hub by default)
`docker login my.registry.com:8000`

Push an image to a registry
`docker push myrepo/myalpine:3.4`

RUN

`docker run`
`--rm` remove container automatically after it exits
`-it` connect the container to terminal
`--name web` name the container
`-p 5000:80` expose port 5000 externally and map to port 80
`-v ~/dev:/code` create a host mapped volume inside the container
`alpine:3.4` the image from which the container is instantiated
`/bin/sh` the command to run inside the container

Stop a running container through SIGTERM
`docker stop web`

Stop a running container through SIGKILL
`docker kill web`

Create an overlay network and specify a subnet
`docker network create --subnet 10.1.0.0/24 --gateway 10.1.0.1 -d overlay mynet`

List the networks
`docker network ls`

List the running containers
`docker ps`

Delete all running and stopped containers
`docker rm -f $(docker ps -aq)`

Create a new bash process inside the container and connect it to the terminal
`docker exec -it web bash`

Print the last 100 lines of a container's logs
`docker logs --tail 100 web`

Another (more detailed) Cheat Sheet

- https://dmitryfrank.com/projects/docker-quick-ref

Noroff
School of technology
and digital media

Monday - Week 4

Services, Swarms, and Kubernetes

Running more than one copy of our App

- We need to scale the application up
  - More simultaneous users
  - Dedicated session-user pairs

- We create a service

Services

- docker-compose.yml
  - Deploy multiple instances at once
  - Load-balance / Scale

- Get Started, Part 3: Services
  https://docs.docker.com/get-started/part3/

Docker Swarms

- What if we need more instances of our service?
- We deploy it to multiple machines at once
- This is a swarm

- Get Started, Part 4: Swarms
  https://docs.docker.com/get-started/part4/

Kubernetes

- Kubernetes was originally a Google Cloud Solution
- It’s open source
- It competes with the likes of Spring Cloud

“container-orchestration system”

- It handles all the grunt work of managing multiple services across multiple hosts
- Resources can be shared within the clusters easily

- It works with more than just Docker

Noroff
School of technology
and digital media

Monday - Week 4

Tasks

Task - Install Docker

- https://docs.docker.com/docker-for-windows/install/
- If you use VM's already:
  - Install Ubuntu into a new VM
  - Then install Docker inside the Ubuntu VM
- Otherwise:
  - Install Docker in Windows

Task - Docker and Spring Boot

- https://spring.io/guides/gs/spring-boot-docker/

- This steps you through a “Hello World” Spring App

- NOTE: “docker push” will fail (near the end), because you are not part of the ‘springio’ organisation

This is not obvious enough in the tutorial

After the Push

A "docker push" will fail for you (unless you are part of the "springio" organization at Dockerhub), but if you change the configuration to match your own docker ID then it should succeed, and you will have a new tagged, deployed image.

Task - Push a docker container to Heroku

- https://devcenter.heroku.com/articles/container-registry-and-runtime

- This step will let you ‘test’ your application remotely.