Docker Lecture: https://www.youtube.com/watch?v=e_YEMorNAQc&list=PLzOJMOiC_x7e2gt9Fn1S0aqMfLXr1n3Hw

What is Docker?

Docker is a tool that makes development **efficient** and **predictable**

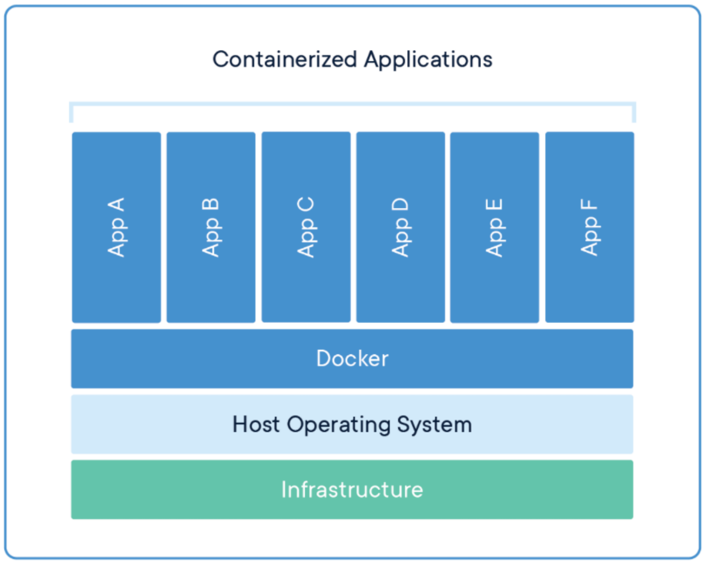
What is a Container?

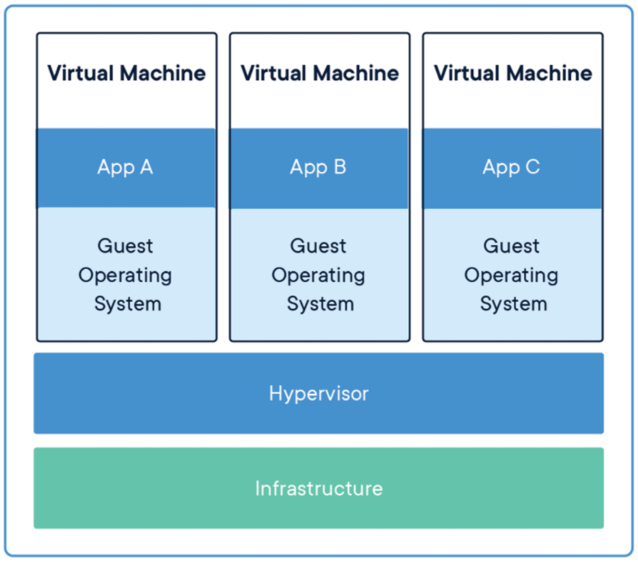
A standard unit of software that packages up code an all its dependencies so the application runs quickly and reliably from one computing environment to another.

Features:

- Lightweight
- Standalone
- Includes everything you need to run an application: code, runtime, system tools, system libraries, and settings

Containers vs. Virtual Machines





Containers compared to VMs:

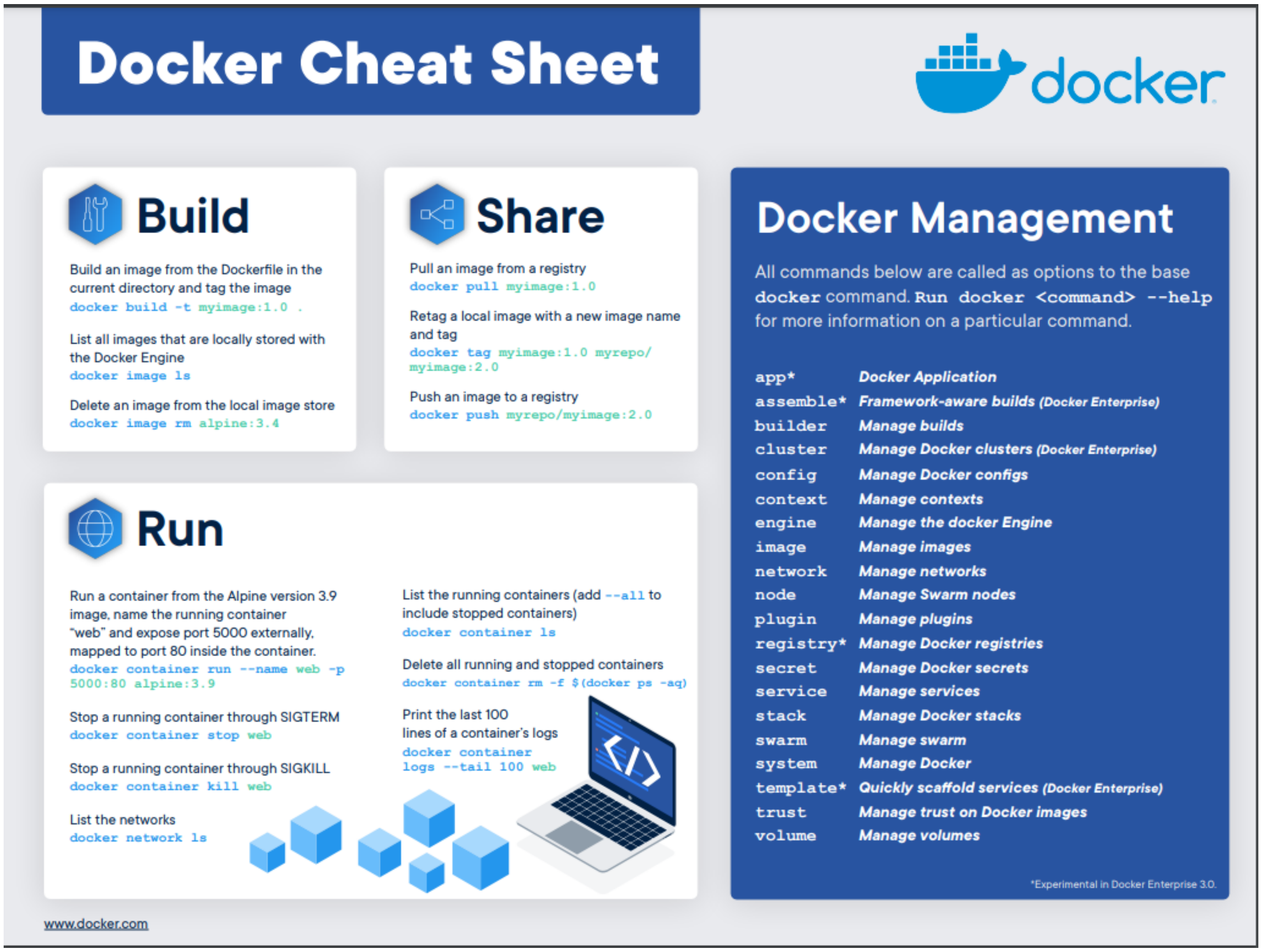
- Take up less space (10 MB vs 10 GB)
- Are more versatile (Can handle more applications and require fewer VMs and Operating Systems)
- Faster to boot

Anatomy of a Dockerfile

Dockerfile: A file used to specify how to build a docker image

```
FROM parentimage:latest
#this is a comment
ENV “environmentVariableKey”=“environmentVariableValue”
RUN commandToRunInShell
RUN ["executable", "param1", "param2"]
COPY ./localsourceFolder /DestFolderWithinImage
WORKDIR /DestFolderWithinImage
EXPOSE 80/tcp
#above line exposes port 80 to the docker virtual network
CMD ["executable","param1","param2"]
#above line is the one command to run in the container
#alternative method specifies executable in ENTRYPOINT and calls CMD
with only params
```

More information: https://www.freecodecamp.org/news/the-docker-handbook/#how-to-write-the-development-dockerfile



Docker Cheat Sheet

Build

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

List all images that are locally stored with the Docker Engine
```
docker image ls
```

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

Share

Pull an image from a registry
```
docker pull myimage:1.0
```

Retag a local image with a new image name and tag
```
docker tag myimage:1.0 myrepo/
myimage:2.0
```

Push an image to a registry
```
docker push myrepo/myimage:2.0
```

Run

Run a container from the Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container.
```
docker container run --name web -p
5000:80 alpine:3.9
```

Stop a running container through SIGTERM
```
docker container stop web
```

Stop a running container through SIGKILL
```
docker container kill web
```

List the networks
```
docker network ls
```

List the running containers (add `--all` to include stopped containers)
```
docker container ls
```

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

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

Docker Management

All commands below are called as options to the base `docker` command. Run `docker <command> --help` for more information on a particular command.

`app*`	***Docker Application***
`assemble*`	***Framework-aware builds (Docker Enterprise)***
`builder`	***Manage builds***
`cluster`	***Manage Docker clusters (Docker Enterprise)***
`config`	***Manage Docker configs***
`context`	***Manage contexts***
`engine`	***Manage the docker Engine***
`image`	***Manage images***
`network`	***Manage networks***
`node`	***Manage Swarm nodes***
`plugin`	***Manage plugins***
`registry*`	***Manage Docker registries***
`secret`	***Manage Docker secrets***
`service`	***Manage services***
`stack`	***Manage Docker stacks***
`swarm`	***Manage swarm***
`system`	***Manage Docker***
`template*`	***Quickly scaffold services (Docker Enterprise)***
`trust`	***Manage trust on Docker images***
`volume`	***Manage volumes***

*Experimental in Docker Enterprise 3.0.

www.docker.com

Docker Cheat Sheet: https://www.docker.com/sites/default/files/d8/2019-09/docker-cheat-sheet.pdf