Introduction to Docker

Liberty Global | Merijntje Tak | 2020-07-07

About me

Merijntje Tak

IPA: [mərɛintʃə] Cyrillic: Мерайнче

- DevOps Engineer in the Operational Data Hub (ODH) project

  All ODH services run inside Docker
  - Stateful services
  - Dynamically scalable (elastic) services
  - User tooling
- 6 years professional Docker experience
- UNIX/Linux and infrastructure background
- https://github.com/mtak/

Topics

- What is Docker?
- Docker concepts
- Operations
- Networking
- Creating Docker images
- Docker registries

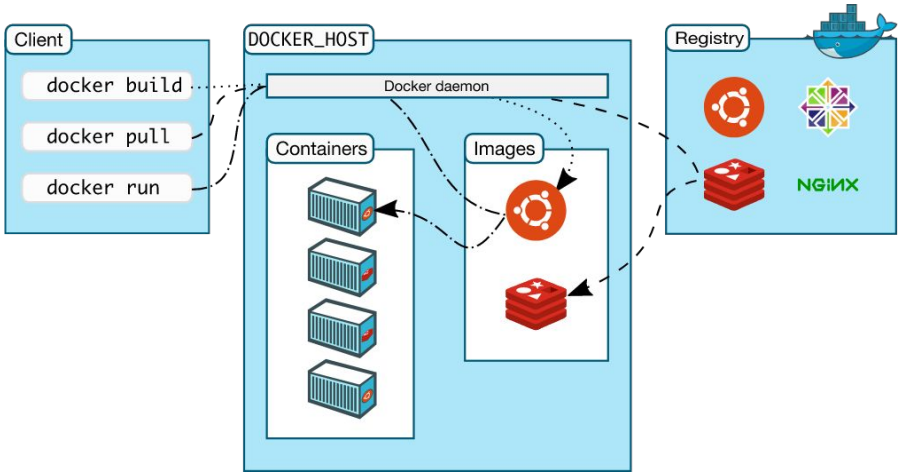
Concepts

What is Docker?

Concepts

Ecosystem for developing, shipping and running applications, which includes:

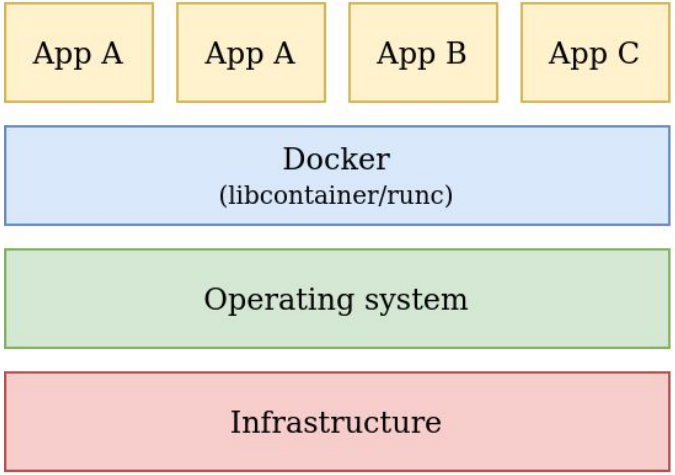
- Management tooling for running containerized or sandboxed apps on a variety of operating systems
- File format for containerized apps
- Development tooling for creating containerized applications
- Repository for containerized applications



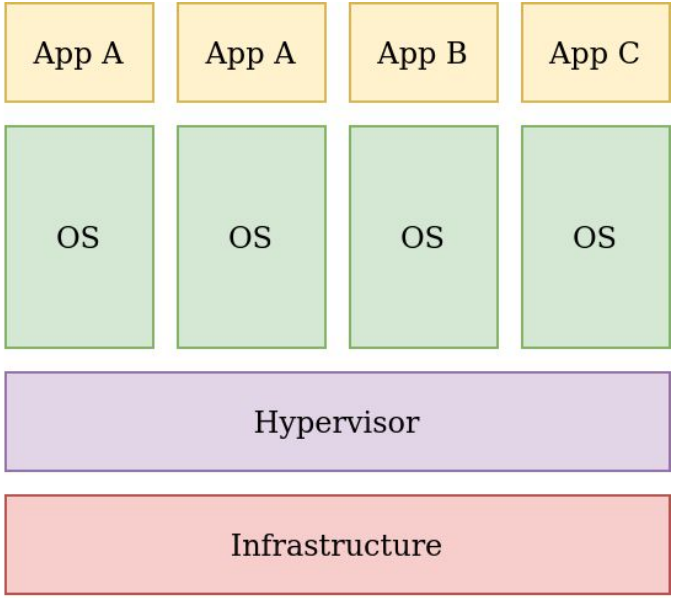
https://docs.docker.com/get-started/overview/

Docker containers vs virtual machines

Concepts



Docker



Virtual machine

Docker images vs containers

Concepts

Image

- Read-only package containing a runtime environment, libraries, executable and config files
- Copy-on-write paradigm via layers
- UnionFS layered format, allowing multiple images to share common layers
- Stored in a repository, which can be accessed by Docker
- Built using Dockerfile

“Class”

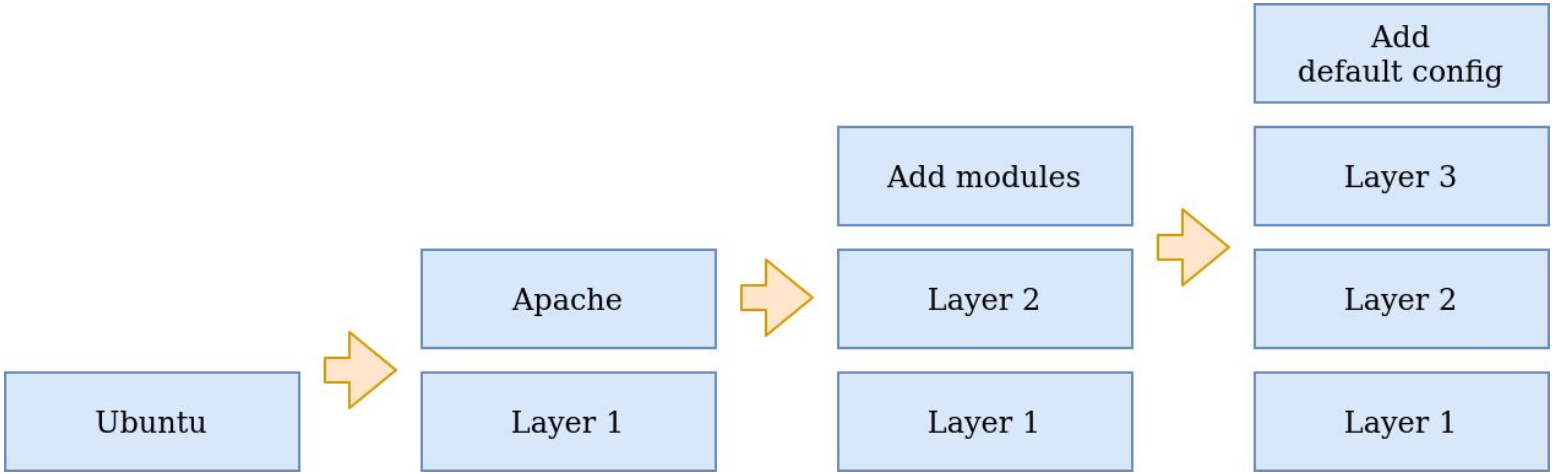
Container

- Runnable instance of an image
- Creates additional R/W layer on top of the image
- Can carry runtime-specific information via:
  - Environment variables
  - Configuration files via mount
  - Command-line parameters

“Object”

UnionFS and docker layers

Concepts



Using containers

Running a Docker container

Operations

```
$ docker run --rm merijntjetak/hello-world:latest
```

- `docker run` – Run a docker container
- `--rm` - Remove container after exit
- `merijntjetak/hello-world:latest` - Docker image reference
  - `merijntjetak` - Docker Hub namespace
  - `hello-world` - Container name
  - `latest` - Version tag

(If an image is not locally available, Docker will search for the image in it's default repository: Docker Hub)

Running a Docker container

Operations

```
mtak@rubiks:~$ docker run --rm merijntjetak/hello-world:0.1

#     #
#     #  ######  #       #        ####
#     #  #       #       #       #    #
#######  #####   #       #       #    #
#     #  #       #       #       #    #
#     #  #       #       #       #    #
#     #  ######  ######  ######   ####

#     #                                  ###
#  #  #   ####   #####   #       #####    ###
#  #  #  #    #  #    #  #       #    #   ###
#  #  #  #    #  #    #  #       #    #    #
#  #  #  #    #  #####   #       #    #
#  #  #  #    #  #   #   #       #    #   ###
 ## ##    ####   #    #  ######  #####    ###



   Congratulations Developer!

You have launched your first Docker container. What just happened:
 - dockerd downloaded this container from Docker Hub
 - dockerd created a new container
 - dockerd started shell script /hello_world.sh

mtak@rubiks:~$
```

Running a Docker container

Operations

Docker container runtime can be influenced with parameters to the docker run command:

```
$ docker run --rm -e NAME=Merijntje merijntjetak/hello-world:latest
```

This will pass the environment variable NAME with value “Merijntje” to the Docker container:

```
mtak@rubiks:~$ docker run --rm -e NAME=Merijntje merijntjetak/hello-world:0.1

#     #
#     #  ######  #       #       ####
#     #  #       #       #      #    #
#######  #####   #       #      #    #

 ## ##    ####   #    #  ######  #####    ###


   Congratulations Merijntje!

You have launched your first Docker container. What just happened:
```

Running a Docker container

Operations

Common run-time options:

- `-e` - Pass environment variable
- `-v` - Mount loopback filesystem (volume)
- `-d` - Detached from terminal, for running daemons
- `--name` - Give a container an easy to remember name

Full reference available at:

- https://docs.docker.com/engine/reference/run/
- `man docker-run`

Running a Docker container

Operations

- Loopback volumes are the primary means to store persistent data in Docker containers
- Overlays an existing directory in a container
- Multiple `-v` options can be used at the same time
- Syntax: `-v <host location>:<container location>[:<options>]`
  - Common options: `rw` / `ro`
- Works for individual files, as well as directories

Running a Docker container

Assignment 1/2

- Run Docker container `merijntjetak/hello-world:latest`
- Run the container again, but add the environment variable NAME with your name as value to this container
- The docker container will behave differently if a file named `/motd.txt` exists. Create this file, and add it to the Docker container using the `-v` option. Observe the container's behaviour.

- Always make sure to clean up, use the `--rm` flag!

Running a Docker container

Key 1/2

```
mtak@rubiks:~$ cat motd.txt

Mounting the file /motd.txt inside the container will only print the contents of /motd.txt on screen.

mtak@rubiks:~$ docker run --rm -v `pwd`/motd.txt:/motd.txt merijntjetak/hello-world:0.1

Mounting the file /motd.txt inside the container will only print the contents of /motd.txt on screen.

mtak@rubiks:~$
```

Running a Docker container

Running daemons

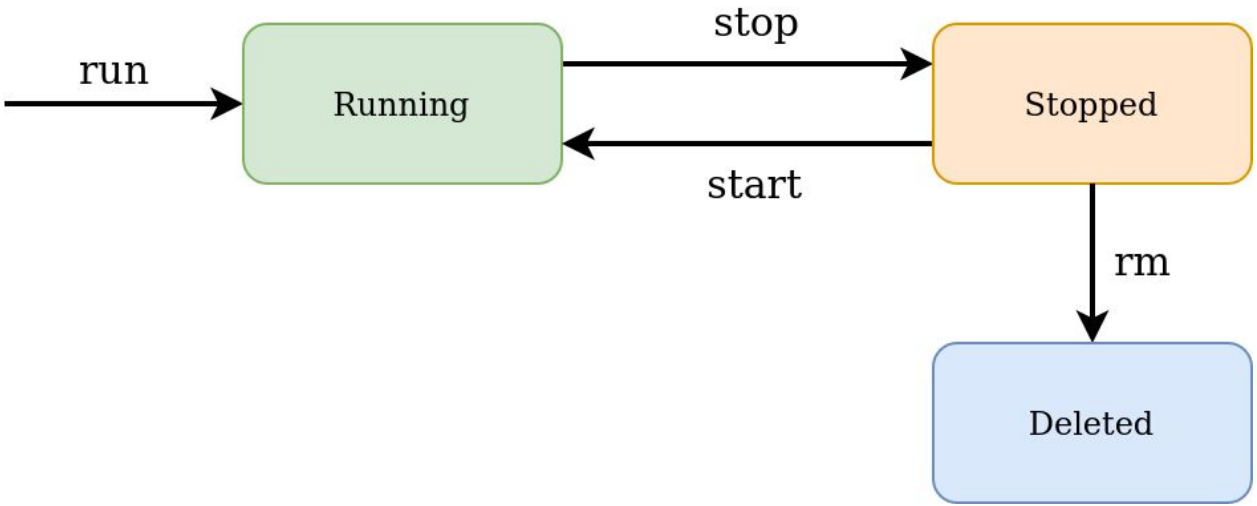
- Use the `-d` option to daemonize a container
- Use `--name` to add a memorable name to a container
- By default, Docker uses port-forwarding to expose specific ports on containers

```
mtak@rubiks:~$ docker run -d -p 5678:5678 --name echo-service hashicorp/http-echo -text="hello world"
edc50a3993ad3898cb8b6630924b7cdb43ff8cf9b8e7304c23cdfa565346d2f6
mtak@rubiks:~$
mtak@rubiks:~$
mtak@rubiks:~$
mtak@rubiks:~$ curl -i http://localhost:5678/
HTTP/1.1 200 OK
X-App-Name: http-echo
X-App-Version: 0.2.3
Date: Mon, 03 Aug 2020 14:51:33 GMT
Content-Length: 12
Content-Type: text/plain; charset=utf-8

hello world
mtak@rubiks:~$
```

Running a Docker container

Docker container lifecycle



Running a Docker container

View running Docker containers

- Running containers can be viewed with the `$ docker ps` command

```
mtak@training:~$ docker ps
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS              PORTS                                   NAMES
7753f9393bd8        guacamole/guacamole   "/opt/guacamole/bin/…"   3 days ago          Up 3 days           0.0.0.0:8080->8080/tcp                  system-guacamole
25d6b86384a4        guacamole/guacd       "/bin/sh -c '/usr/lo…"   3 days ago          Up 3 days           4822/tcp                                system-guacd
2b5390a74af8        mysql:latest          "docker-entrypoint.s…"   3 days ago          Up 3 days           0.0.0.0:3306->3306/tcp, 33060/tcp       system-mysql
```

- View stopped/exited containers with the `-a` flag

```
mtak@training:~$ docker ps -a
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS                      PORTS                                   NAMES
56cb880f514b        tak.io/my-container:0.2  "/bin/sh -c 'python3…"   6 seconds ago       Exited (0) 3 seconds ago                                            tender_meninsky
7753f9393bd8        guacamole/guacamole      "/opt/guacamole/bin/…"   3 days ago          Up 3 days                   0.0.0.0:8080->8080/tcp                  system-guacamole
25d6b86384a4        guacamole/guacd          "/bin/sh -c '/usr/lo…"   3 days ago          Up 3 days                   4822/tcp                                system-guacd
2b5390a74af8        mysql:latest             "docker-entrypoint.s…"   3 days ago          Up 3 days                   0.0.0.0:3306->3306/tcp, 33060/tcp       system-mysql
mtak@training:~$
```

Running a Docker container

Stop and kill containers

- Docker containers can be stopped with the `$ docker stop <containername>` command
  - Docker will send a SIGTERM to the main process inside the container
  - After a grace period (10s default), it will send a SIGKILL to the main process

- If a container is particularly stubborn; there is also `$docker kill <containername>`
  - This will send a SIGKILL to the main process in the container

```
mtak@rubiks:~/dev/docker-introduction/images(master)$ docker ps | grep webserver
94d29f19e731        webserver           "apache2ctl -D FOREG…"   21 hours ago        Up 21 hours
mtak@rubiks:~/dev/docker-introduction/images(master)$ docker stop 94d29f19e731
94d29f19e731
```

Running a Docker container

View container logs

- If a container is running without the -ti options, all STDOUT and STDERR will be captured by Docker and stored in a log. This log can be viewed with:

```
$ docker logs <containername>
```

```
mtak@training:~$ docker logs system-guacd
guacd[6]: INFO: Guacamole proxy daemon (guacd) version 1.2.0 started
guacd[6]: INFO: Listening on host 0.0.0.0, port 4822
guacd[6]: ERROR:        Guacamole protocol violation. Perhaps the version of guacamole-client is incompatible with this version of guacd?
guacd[6]: INFO: Creating new client for protocol "ssh"
guacd[6]: INFO: Connection ID is "$ccdeebf1-b76f-4f18-95ef-9a81a3f5cc08"
guacd[9]: INFO: User "@143ce311-70e9-44f5-86df-a7e88b8dba5c" joined connection "$ccdeebf1-b76f-4f18-95ef-9a81a3f5cc08" (1 users now present)
guacd[9]: ERROR:        Unable to connect to any addresses.
guacd[9]: INFO: User "@143ce311-70e9-44f5-86df-a7e88b8dba5c" disconnected (0 users remain)
guacd[9]: INFO: Last user of connection "$ccdeebf1-b76f-4f18-95ef-9a81a3f5cc08" disconnected
guacd[6]: INFO: Connection "$ccdeebf1-b76f-4f18-95ef-9a81a3f5cc08" removed.
guacd[6]: INFO: Creating new client for protocol "ssh"
guacd[6]: INFO: Connection ID is "$2c4007f9-5434-4219-99f8-fa90562d3fd3"
guacd[19]: INFO:        User "@b530f761-7a96-4bf6-8d49-a03c5d69e4a1" joined connection "$2c4007f9-5434-4219-99f8-fa90562d3fd3" (1 users now present)
guacd[19]: ERROR:       Unable to connect to any addresses.
guacd[19]: INFO:        User "@b530f761-7a96-4bf6-8d49-a03c5d69e4a1" disconnected (0 users remain)
guacd[19]: INFO:        Last user of connection "$2c4007f9-5434-4219-99f8-fa90562d3fd3" disconnected
guacd[6]: INFO: Connection "$2c4007f9-5434-4219-99f8-fa90562d3fd3" removed.
```

- NOTE: `docker logs` will also accept parameters like -f and --tail

Running a Docker container

View container details

- All details related to a container can be found with the `$ docker inspect <containername>` command. This massive JSON will include:
  - Container and image name and version
  - Mount points (volumes)
  - Network settings
  - Environment variables
  - Command and arguments
  - Overlay settings
  - Etc.

```
mtak@training:~$ docker inspect system-guacamole
[
    {
        "Id": "7753f9393bd8c61979d2e85b33ddc0f55ecc892d08e92e7c78ba2e8e1bb319f2",
        "Created": "2020-08-03T09:29:17.244069875Z",
        "Path": "/opt/guacamole/bin/start.sh",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7292,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2020-08-03T09:32:21.967264263Z",
            "FinishedAt": "2020-08-03T09:32:21.472031121Z"
        },
        "Image": "sha256:0c9ec5e99dce3c6b75128d5a6e2a731446c71c0f88bb1c6d398e7ccd589e5075
        "ResolvConfPath": "/var/lib/docker/containers/7753f9393bd8c61979d2e85b33ddc0f55ec
        "HostnamePath": "/var/lib/docker/containers/7753f9393bd8c61979d2e85b33ddc0f55ecc8
        "HostsPath": "/var/lib/docker/containers/7753f9393bd8c61979d2e85b33ddc0f55ecc892d
        "LogPath": "/var/lib/docker/containers/7753f9393bd8c61979d2e85b33ddc0f55ecc892d08
        "Name": "/system-guacamole",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "docker-default",
        "ExecIDs": null,
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
```

Running a Docker container

Assignment 2/2

1. Start the hashicorp/http-echo Docker container:
   - As a daemon
   - Using switch `-p 5678:5678`
   - With name `echo-service`
   - Use the final parameter `-text="Test service"`
2. View the running container
3. Stop the container
4. Start the container again, using the same command. An error is displayed. What does this error indicate?
5. Verify that the error is correct
6. Remove the container

Running a Docker container

Key 2/2

```
mtak@rubiks:~$ docker run -d -p 5678:5678 --name echo-service hashicorp/http-echo -text="Test service"
10e0a3b24eb6d0102c5241f56b672508921940e923876f7491891dfffe07d492
mtak@rubiks:~$ docker ps
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS              PORTS                    NAMES
10e0a3b24eb6        hashicorp/http-echo   "/http-echo '-text=T…"   7 seconds ago       Up 5 seconds        0.0.0.0:5678->5678/tcp   echo-service
mtak@rubiks:~$ docker stop echo-service
echo-service
mtak@rubiks:~$ docker run -d -p 5678:5678 --name echo-service hashicorp/http-echo -text="Test service"
docker: Error response from daemon: Conflict. The container name "/echo-service" is already in use by container "10e0a3b24eb6d0102c5241f56b672508921940e923876f7491891dfffe0
7d492". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
mtak@rubiks:~$ docker ps -a
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS                      PORTS               NAMES
10e0a3b24eb6        hashicorp/http-echo   "/http-echo '-text=T…"   40 seconds ago      Exited (137) 12 seconds ago                     echo-service
mtak@rubiks:~$ docker rm echo-service
echo-service
```

Docker networking

Docker networking

Concepts

- **Bridge networking**
  Host-only networking which allows Docker containers to communicate in an internal network. External access can be provided for via port forwarding. A Docker-managed combination of iptables and Linux bridges

- **Host networking**
  Directly connect the container to the host system's interfaces, running like any other service

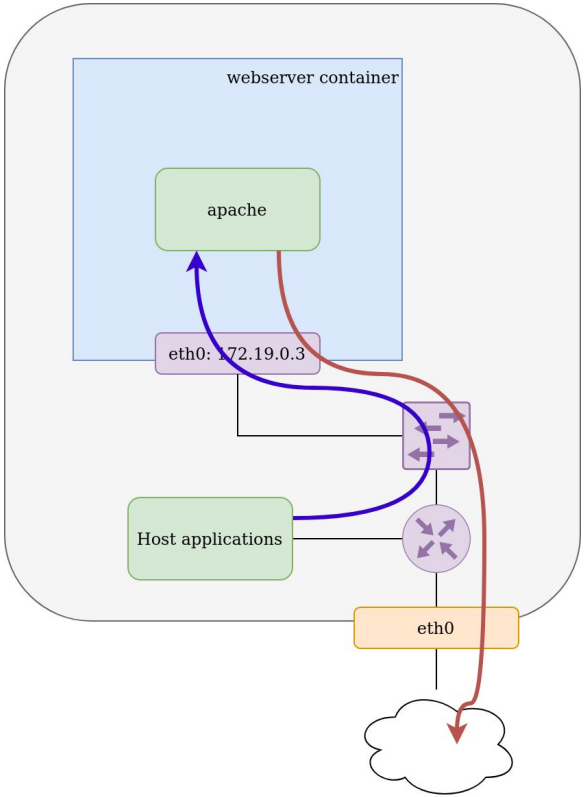
- **Overlay networks**
  Docker managed network that connects Docker daemons on several hosts together and allow cluster communications between containers on separate hosts (out of scope)

Bridge networking

Concepts

- Docker's default bridging
- Creates an internal bridge, with a separate subnet

- Allows host<->container communications
- No container<->container comms



Bridge networking

Concepts

```
mtak@rubiks:~$ docker run --rm -ti --name="webserver" hashicorp/http-echo -text="hello world"
2020/08/04 09:27:47 Server is listening on :5678
2020/08/04 09:27:47 [ERR] Unknown signal window changed
2020/08/04 09:28:14 192.168.32.2:5678 192.168.32.1:52896 "GET / HTTP/1.1" 200 12 "curl/7.58.0" 12.764µs
```

```
mtak@rubiks: ~
mtak@rubiks:~$ docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' webserver
192.168.32.2
mtak@rubiks:~$ curl http://192.168.32.2:5678/
hello world
mtak@rubiks:~$
```

```
mtak@rubiks:~$ docker run --rm -ti alpine sh
/ # wget -SO- https://mtak.nl/ >/dev/null
Connecting to mtak.nl (136.144.137.41:443)
  HTTP/1.1 200 OK
  Date: Tue, 04 Aug 2020 09:31:46 GMT
  Server: Apache/2.4.10 (Debian)
  Last-Modified: Wed, 02 May 2018 09:19:37 GMT
  ETag: "1ea9-56b359600ddb7"
  Accept-Ranges: bytes
  Content-Length: 7849
  Connection: close
  Content-Type: text/html

-                    100% |***************************
/ #
```

Bridge networking

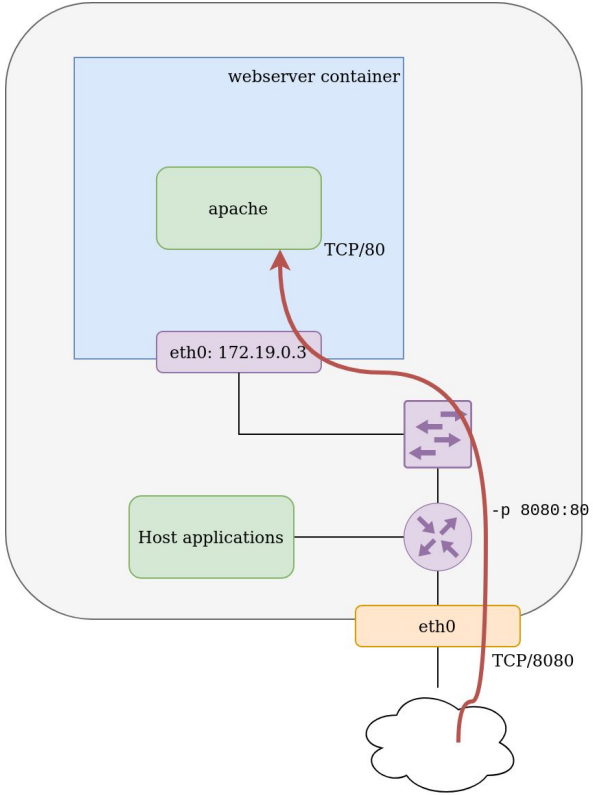
Concepts

Docker port forwarding

- Allows individual ports to be forwarded from the outside world to the Docker container

- Set up a port forward

```
$ docker run -p 8080:80 containername
```



Bridge networking

Concepts

```
mtak@training:~$ docker run --rm -ti --name="webserver" -p 8000:5678 hashicorp/http-echo -text="hello world"
2020/08/04 10:24:05 Server is listening on :5678
2020/08/04 10:24:05 [ERR] Unknown signal window changed
2020/08/04 10:24:15 training.int.mtak.nl:8000 10.100.1.2:33478 "GET / HTTP/1.1" 200 12 "curl/7.58.0" 25.301µs
```

```
mtak@rubiks: ~
mtak@rubiks:~$ curl http://training.int.mtak.nl:8000/
hello world
mtak@rubiks:~$
```

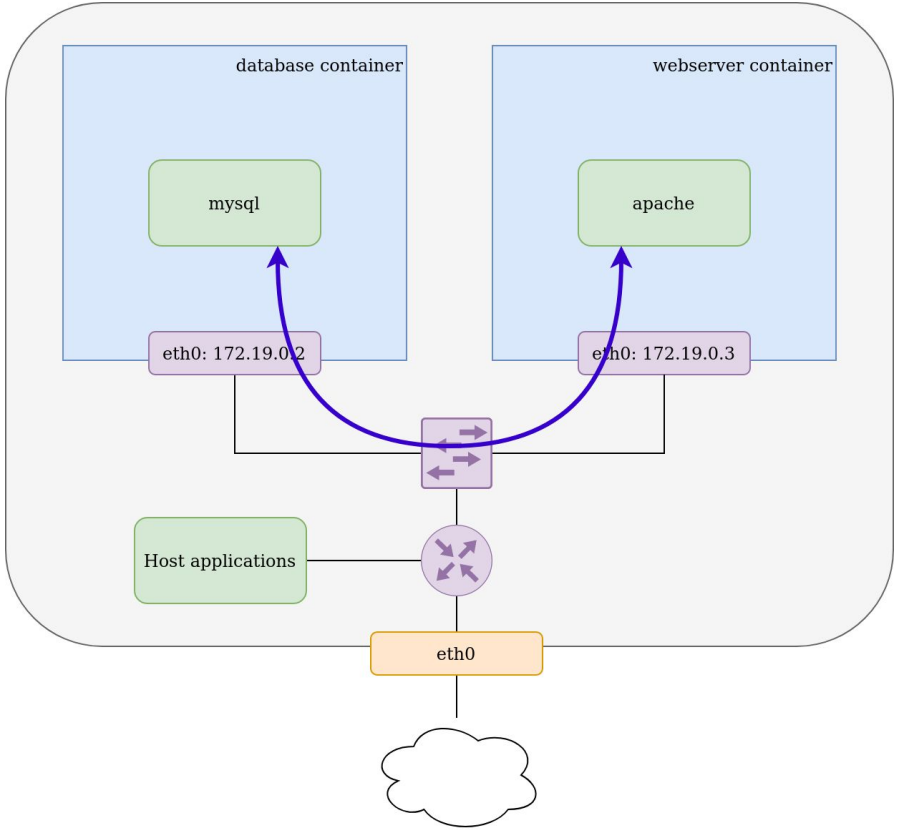
Bridge networking

Concepts

User-defined bridges allow:

- Container<->container communication
- Multiple user bridges can be created (eg. one per stack)
- /etc/hosts in a container is updated with all other containers, so you can refer to other containers by name

Note: option --link is deprecated



Bridge networking

Concepts

1. Set up a bridge

```
$ docker network create my-bridge
```

2. Attach containers to the bridge

```
$ docker run --name="webserver" --network="my-bridge" containername
```

1. Stop and remove the containers

```
$ docker stop webserver && docker rm webserver
```

2. Destroy the bridge

```
$ docker network rm my-bridge
```

Bridge networking

Concepts

```
mtak@rubiks:~$ docker run --rm -ti --name="webserver" --network="my-bridge" hashicorp/http-echo -text="hello world"
2020/08/04 10:31:06 Server is listening on :5678
2020/08/04 10:31:06 [ERR] Unknown signal window changed
2020/08/04 10:31:12 webserver:5678 192.168.128.3:47984 "GET / HTTP/1.1" 200 12 "Wget" 69.831µs
```

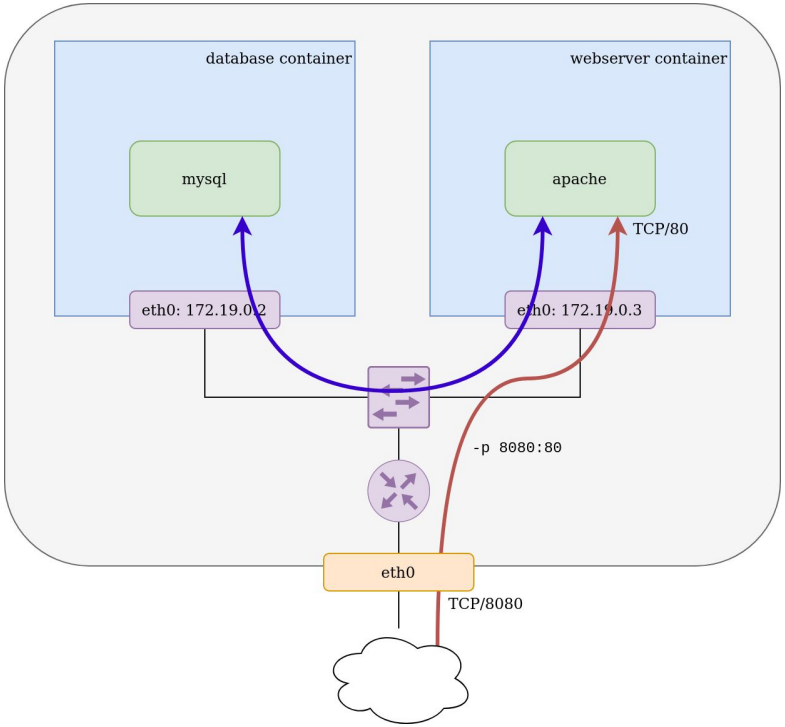
mtak@rubiks: ~

```
mtak@rubiks:~$ docker run --rm -ti --network="my-bridge" alpine sh
/ # wget -O- http://webserver:5678/
Connecting to webserver:5678 (192.168.128.2:5678)
hello world
-                    100% |*******************************************************************************|
/ #
```

Bridge networking

Concepts

- User-defined bridges can be used in conjunction with port forwarding to create application stacks



Bridge networking

Assignment

1. Create a bridge network called “my-webapp”

2. Run the container merijntjetak/my-webapp-db:latest with:
   - Name: “my-webapp-db”
   - Connected to the bridge created in step 1

3. Run the container merijntjetak/my-webapp-webapp:latest with:
   - Port forwarding port 8080 to port 80
   - Connected to the bridge created in step 1
   - Set the environment variable MYSQL_HOST to the name of the database container

4. Browse to the webpage (CloudShell->Web Preview->Preview on port 8080)

Bridge networking

Assignment

1. Create a bridge network called “my-webapp”

```
$ docker network create my-webapp
```

2. Run the container merijntjetak/my-webapp-db:0.1 with:

```
$ docker run --rm -d --network="my-webapp" --name="my-webapp-db" merijntjetak/my-webapp-db:latest
```

3. Run the container merijntjetak/my-webapp-webapp:0.1 with:

```
$ docker run --rm -d --network="my-webapp" --name="my-webapp-webapp"
-e MYSQL_HOST=my-webapp-db -p 8080:80 merijntjetak/my-webapp-webapp:latest
```

4. Browse to the webpage

Host networking

Concepts

- In Docker host networking, applications are directly tied to the host's TCP stack. They will behave like any other daemon running on the system. The option is `--net=host`

```
mtak@rubiks:~$ docker run -d --net=host --name echo-service hashicorp/http-echo -text="Test service"
9aefc9ef275e47bb6a948f881563f2c66ba5041afe7687131468279914576b57
mtak@rubiks:~$
```

```
mtak@rubiks: ~
mtak@rubiks:~$ curl -i http://localhost:5678/
HTTP/1.1 200 OK
X-App-Name: http-echo
X-App-Version: 0.2.3
Date: Tue, 04 Aug 2020 13:35:53 GMT
Content-Length: 13
Content-Type: text/plain; charset=utf-8
```

- You *will* have to manage port configurations though the application and avoid collisions

```
mtak@rubiks:~$ docker run -ti --net=host --name echo-service2 hashicorp/http-echo -text="Test service"
2020/08/04 13:37:38 [ERR] Error starting server: listen tcp :5678: bind: address already in use
```

Creating a Docker image

Building a Docker image

Concepts

- Docker images are defined by using a Dockerfile
- Dockerfile is a scripting/definition language, usually in a file called Dockerfile
- A Dockerfile contains
  - What your image is based on
  - What steps are taken to install packages, add files, run commands etc
  - Runtime options
- Docker provides a toolchain to build the images based on information in the Dockerfile

Building a Docker image

Concepts

```
FROM ubuntu:20.04

RUN apt-get update
RUN apt-get -y install python3-pip
RUN rm -rf /var/lib/apt/lists/*

RUN pip install pprint

RUN mkdir /app

COPY ./app.py /app/app.py

CMD python /app/app.py
```

- Use Ubuntu 20.04 from Docker hub as base image
- Run apt-get update
  Install pip
  Clean up the mess afterwards
- Run this command inside the Docker image, installing pprint
- Make a directory
- Copy a file into the docker container
- Set the default command to run when starting the container

Building a Docker image

Concepts

- After creating the Dockerfile, run the following command to build the image:

```
$ docker build -t my-container .
```

- Docker will build a container named my-container, with version tag: `latest`

- Version tags can be added, separated from the container name by a colon:

```
$ docker build -t my-container:0.1 .
```

```
mtak@rubiks:~/dev/docker-introduction/images/example-docker-build(master)$ docker run --rm my-container:latest
[   ['spam', 'eggs', 'lumberjack', 'knights', 'ni'],
    'spam',
    'eggs',
    'lumberjack',
    'knights',
    'ni']
```

Building a Docker image

Concepts

```
mtak@rubiks:~/dev/docker-introduction/images/example-docker-build(master)$ docker build  -t my-container .
Sending build context to Docker daemon  3.072kB
Step 1/8 : FROM ubuntu:20.04
 ---> 1e4467b07108
Step 2/8 : RUN apt-get update
 ---> Using cache
 ---> abce57461fe1
Step 3/8 : RUN apt-get -y install python3-pip
 ---> Using cache
 ---> 2fae6aebfb35
Step 4/8 : RUN rm -rf /var/lib/apt/list/*
 ---> Using cache
 ---> cc640aa07ce0
Step 5/8 : RUN pip3 install pprint
 ---> Using cache
 ---> 57792d1b6beb
Step 6/8 : RUN mkdir /app
 ---> Using cache
 ---> 1f46a0dc3f12
Step 7/8 : COPY ./app.py /app/app.py
 ---> Using cache
 ---> 03166269f250
Step 8/8 : CMD python3 /app/app.py
 ---> Using cache
 ---> ae1e4fafdbc7
Successfully built ae1e4fafdbc7
Successfully tagged my-container:latest
```

- All these are UnionFS layers
- Every RUN, COPY, ADD and CMD command adds a layer to the Docker container
- All layers combined, make up this container
- Inefficient, why?

Building a Docker image

Concepts

```
mtak@rubiks:~/dev/docker-introduction/images/example-docker-build(master)$ docker history my-container
IMAGE          CREATED        CREATED BY                                      SIZE      COMMENT
ae1e4fafdbc7   5 minutes ago  /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "pyth…   0B
03166269f250   5 minutes ago  /bin/sh -c #(nop) COPY file:038b842ac1fa77ac…   151B
1f46a0dc3f12   5 minutes ago  /bin/sh -c mkdir /app                           0B
57792d1b6beb   5 minutes ago  /bin/sh -c pip3 install pprint                   5.22kB
cc640aa07ce0   5 minutes ago  /bin/sh -c rm -rf /var/lib/apt/list/*           0B
2fae6aebfb35   5 minutes ago  /bin/sh -c apt-get -y install python3-pip       293MB
abce57461fe1   6 minutes ago  /bin/sh -c apt-get update                       22.6MB
1e4467b07108   12 days ago    /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      12 days ago    /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>      12 days ago    /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   811B
<missing>      12 days ago    /bin/sh -c [ -z "$(apt-get indextargets)" ]     1.01MB
<missing>      12 days ago    /bin/sh -c #(nop) ADD file:65a1cc50a9867c153…   72.9MB
```

Building a Docker image

Space optimizations

```
FROM ubuntu:20.04

RUN apt-get update && \
    apt-get -y install python3-pip && \
    rm -rf /var/lib/apt/lists/*

RUN pip install pprint

RUN mkdir /app

COPY ./app.py /app/app.py

CMD python /app/app.py
```

A single RUN command will only produce one layer, not storing the apt cache

Building a Docker image

Space optimization

```
mtak@rubiks:~/dev/docker-introduction/images/example-docker-build-optimized(master)$ docker history my-container:0.2
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
2ecc7b82042a        7 seconds ago       /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "pyth…   0B
0951591dc87b        7 seconds ago       /bin/sh -c #(nop) COPY file:038b842ac1fa77ac…   151B
caeb24b5fb06        7 seconds ago       /bin/sh -c mkdir /app                           0B
d114fd513324        8 seconds ago       /bin/sh -c pip3 install pprint                  5.22kB
5d08af61d764        10 seconds ago      /bin/sh -c apt-get update &&     apt-get -y …   293MB
1e4467b07108        12 days ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           12 days ago         /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>           12 days ago         /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   811B
<missing>           12 days ago         /bin/sh -c [ -z "$(apt-get indextargets)" ]     1.01MB
<missing>           12 days ago         /bin/sh -c #(nop) ADD file:65a1cc50a9867c153…   72.9MB
```

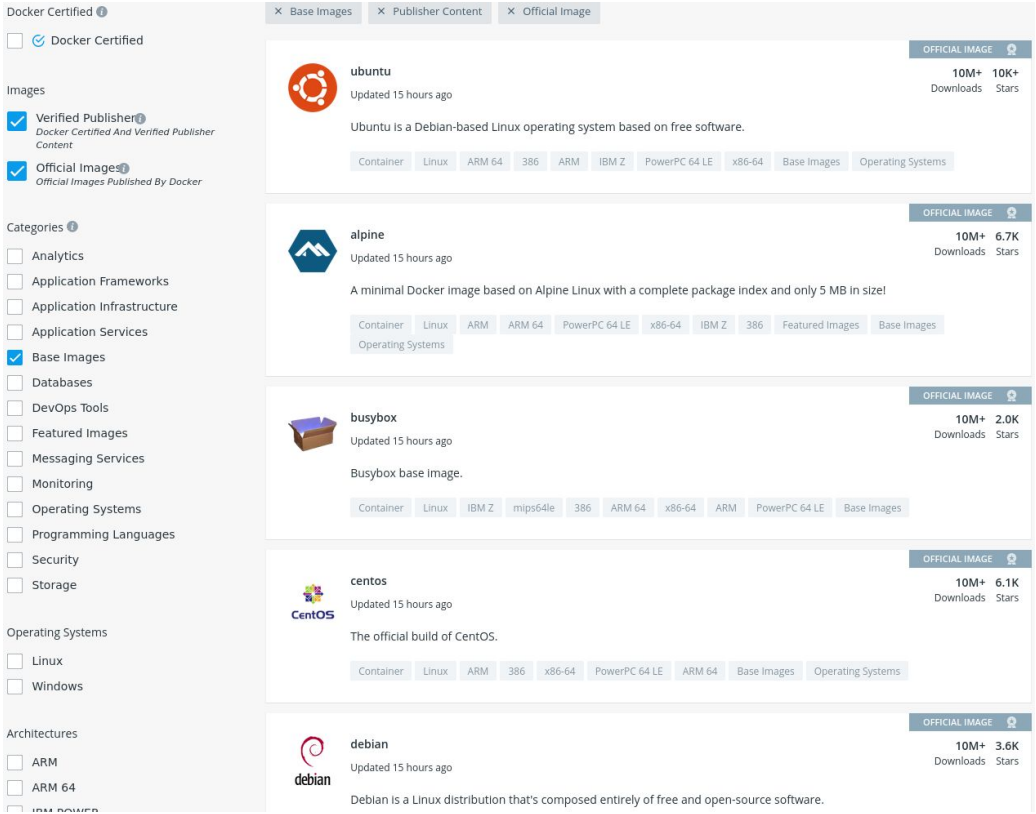
```
mtak@rubiks:~/dev/docker-introduction/images/example-docker-build-optimized(master)$ docker images | grep my-container
my-container                                0.2                 2ecc7b82042a        55 seconds ago      367MB
my-container                                latest              ae1e4fafdbc7        19 minutes ago      389MB
```

Saving 22MB

Building a Docker image

Base images

- Base images can be found on Docker Hub (https://hub.docker.com)
- Alpine is a very small and popular base image, which comes with a package manager (apk)



Building a Docker image

CMD vs ENTRYPOINT

- CMD and ENTRYPOINT statements do the same thing, but have different purposes
- CMD sets a default command to run when a container starts. When a command is provided in a docker run command, this will override the CMD value in the Dockerfile

```
mtak@rubiks:~/dev/docker-introduction/images/cmd-vs-entrypoint/cmd(master)$ cat Dockerfile
FROM alpine:latest

CMD ["uname"]
mtak@rubiks:~/dev/docker-introduction/images/cmd-vs-entrypoint/cmd(master)$ docker run --rm cmd-test
Linux
mtak@rubiks:~/dev/docker-introduction/images/cmd-vs-entrypoint/cmd(master)$ docker run --rm cmd-test ls
bin
dev
etc
```

Building a Docker image

CMD vs ENTRYPOINT

- ENTRYPOINT will set a fixed command to be run when a container starts
- Any arguments passed via docker run will be added as parameters to the ENTRYPOINT command

```
mtak@rubiks:~/dev/docker-introduction/images/cmd-vs-entrypoint/entrypoint(master)$ cat Dockerfile
FROM alpine:latest

ENTRYPOINT ["uname"]
mtak@rubiks:~/dev/docker-introduction/images/cmd-vs-entrypoint/entrypoint(master)$ docker run --rm entrypoint-test
Linux
mtak@rubiks:~/dev/docker-introduction/images/cmd-vs-entrypoint/entrypoint(master)$ docker run --rm entrypoint-test -a
Linux 232e7d1de017 5.3.0-7648-generic #41~1586790036~18.04~600aeb5-Ubuntu SMP Mon Apr 13 17:47:15 UTC  x86_64 Linux
```

Building a Docker image

Assignment

- Create a Docker image named <name>-webserver, with version tag 0.1
- Choose a suitable base image (your favourite distro, Alpine because of it’s small size)
- Install a webserver
- Customize the index.html file
- Make sure the webserver runs when the Docker container starts without any arguments
- Start the container, listening on port 8080 and view the result (Top right -> Web preview)

Building a Docker image

Key

```
mtak@rubiks:~/dev/docker-introduction/images/create-image-assignment-key(master)$ cat Dockerfile
FROM ubuntu:20.04

# Avoid tzdata package configuration interactive menu
ENV TZ=Europe/Amsterdam
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

RUN apt-get update && apt-get install -y \
        apache2 \
        && rm -rf /var/lib/apt/lists/

COPY src/index.html /var/www/html/index.html

EXPOSE 80

CMD ["apache2ctl", "-D", "FOREGROUND"]
mtak@rubiks:~/dev/docker-introduction/images/create-image-assignment-key(master)$
```

```
mtak@rubiks:~/dev/docker-introduction/images/create-image-assignment-key(master)$ docker run --rm -p8080:80 webserver
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 192.168.32.2. Set the 'S
```

Building a Docker image

Multi-stage builds

```
FROM golang:1.7.3 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go    .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app .
CMD ["./app"]
```

Docker registries

Docker registries

Concepts

- Remote Docker image storage
- Registries leverage Docker layers and deduplicate them
- Ideal for publishing containers for usage or deployment
- Docker tags allow for versioning

- You've already used it! Docker Hub is a fancy Docker registry
- By default, Docker will use Docker Hub
- Other registries can be used

Docker registries

Concepts

- Namespacing for containers stored on registries is a bit odd:

```
tak.io/merijntjetak/hello-world:0.1
```

- tak.io is the registry hostname
  - The default insecure registry port is 5000
  - The default secure registry port is 443
  - Registry authentication can *only* be done with SSL, but the Docker client can be reconfigured to allow insecure registries
- merijntjetak/hello-world is the image name
- 0.1 is the version number

Docker registries

Authentication

- Registries might be authenticated
- Authentication might only be required to push images, like Docker Hub
- Authentication might be required to push and pull images, like tak.io

- You can log on to a registry, by issuing the docker login command, followed by the registry

```
$ docker login tak.io
```

- This will request your username and password and store it locally for further use
- The next time you interact with the registry, it will send these credentials

- Log out of a registry by using the docker logout command

```
$ docker logout tak.io
```

Docker registries

Upload images

- To upload a Docker image, first retag it to include the registry name:

```
$ docker tag my-container:0.2 tak.io/my-container:0.2
```

- Then push

```
$ docker push tak.io/my-container:0.2
```

```
mtak@rubiks:~$ docker tag my-container:0.2 tak.io/my-container:0.2
mtak@rubiks:~$ docker push !$
docker push tak.io/my-container:0.2
The push refers to repository [tak.io/my-container]
f44f996e655e: Pushed
63412a605a6f: Pushed
ccdabd9e6ad8: Pushed
486024670813: Pushing [=======>                                    ]  41.28MB/293MB
095624243293: Pushed
a37e74863e72: Pushed
8eeb4a14bcb4: Pushed
ce3011290956: Pushing [=======>                                    ]  11.36MB/72.85MB
```

Docker registries

Download images

- The easiest way to download an image from the registry, is simply by running it:
- Docker pull will achieve the same thing, without running the image
- Images are stored on the local machine

```
mtak@training:~$ docker run tak.io/my-container:0.2
Unable to find image 'tak.io/my-container:0.2' locally
0.2: Pulling from my-container
3ff22d22a855: Pull complete
e7cb79d19722: Pull complete
323d0d660b6a: Pull complete
b7f616834fd0: Pull complete
1205141072ca: Pull complete
2adac37124de: Pull complete
7d73eb3ad9f9: Pull complete
93efac9b27f6: Pull complete
Digest: sha256:8c23825e7c74ae2f6b79301ae11379437ce2d7aa518e7cd0ec2eeac8bbfa722c
Status: Downloaded newer image for tak.io/my-container:0.2
[   ['spam', 'eggs', 'lumberjack', 'knights', 'ni'],
    'spam',
    'eggs',
    'lumberjack',
    'knights',
    'ni']
mtak@training:~$
```

Docker registries

Local image operations

- View images stored locally with `$ docker image ls`
- Remove an image from the local machine with `$ docker image rm containername`

```
mtak@training:~$ docker image ls
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
tak.io/my-container           0.2                 2ecc7b82042a        56 minutes ago      367MB
mysql                         latest              e3fcc9e1cc04        13 days ago         544MB
guacamole/guacd               latest              9713c6008034        5 weeks ago         398MB
guacamole/guacamole           latest              0c9ec5e99dce        5 weeks ago         507MB
hello-world                   latest              bf756fb1ae65        7 months ago        13.3kB
tak.io/elasticsearch-curator  5.5.4               db45d4dd7feb        2 years ago         94.4MB
tak.io:5000/http-echo         latest              a6838e9a6ff6        3 years ago         3.97MB
mtak@training:~$ docker rmi tak.io/my-container:0.2
Untagged: tak.io/my-container:0.2
Untagged: tak.io/my-container@sha256:8c23825e7c74ae2f6b79301ae11379437ce2d7aa518e7cd0ec2eeac8bbfa722c
Deleted: sha256:2ecc7b82042a778aedd3348e622195c6588011add4c0be1ef2ab5347a312ce3e
Deleted: sha256:60b468cd4ddd6036f2151d09cdf5e22dcb5cdb9f4753283bb5414ce4dad25a6b
Deleted: sha256:1768b76c34e0974562a12d49f1bb3895575b95ac7cafe6f7434af19bc116fe62
Deleted: sha256:f2ce5f61bd45ca9a3c839756a2b984d6a039ce3e449b6e7c088244cf6425a15d
Deleted: sha256:a16e996c1867b697238ab8cbd6fff800b1693e48c6c609bd21d434ef594b8c87
Deleted: sha256:7515ee845913c8df9826c988341a09e0240e291c66bdc436a067e070d7910a1f
Deleted: sha256:50ebe6a0675f1ed7ca499a2ec7d8cc993d495dd66ca1035c218ec5efcb6fbb8c
Deleted: sha256:2515e0ecfb82d58c004c4b53fcf9230d9eca8d0f5f823c20172be01eec587ccb
Deleted: sha256:ce30112909569cead47eac188789d0cf95924b166405aa4b71fb500d6e4ae08d
```

Docker registries

Assignment

1. Upload your webserver image to the registry at tak.io

   The credentials for the registry are:

   Username: training

   Password: LibertyGlobal1

   NOTE: make sure to include your name in the image tag to avoid collisions!

2. Remove the image from your local instance, using the old and new tags

3. Pull the image back to your machine

Docker registries

Key

```
mtak@rubiks:~$ docker image ls | grep webserver
webserver                                   latest              f18d7f3eac55        40 minutes ago      187MB
mtak@rubiks:~$ docker tag webserver:latest tak.io/webserver:0.1
mtak@rubiks:~$ docker push tak.io/webserver:0.1
The push refers to repository [tak.io/webserver]
1d7f81079a98: Pushed
dbf122e2ae97: Pushed
8ecdb6227cb1: Pushed
095624243293: Mounted from my-container
a37e74863e72: Mounted from my-container
8eeb4a14bcb4: Mounted from my-container
ce3011290956: Mounted from my-container
0.1: digest: sha256:305c5678ec1b7db80411c29a28ed2fd68af2f93d0e3966096ac67f6792f2c04e size: 1778
mtak@rubiks:~$
```

Layer reused from another container


GROW
WITH US

Miscellaneous

- Look into Docker Compose and Docker Service, it will make running services from containers a lot easier
- Steal stuff from Docker Hub and GitHub. Look at the source repositories and the Dockerfiles in them and learn the tricks from people before you
- Always tag your images appropriately. Your colleagues will hate you if you don't!

Course repo: https://github.com/mtak/docker-introduction

Dockerfile reference: https://docs.docker.com/engine/reference/builder/

Docker run reference: https://docs.docker.com/engine/reference/run/