
docker

Objectives

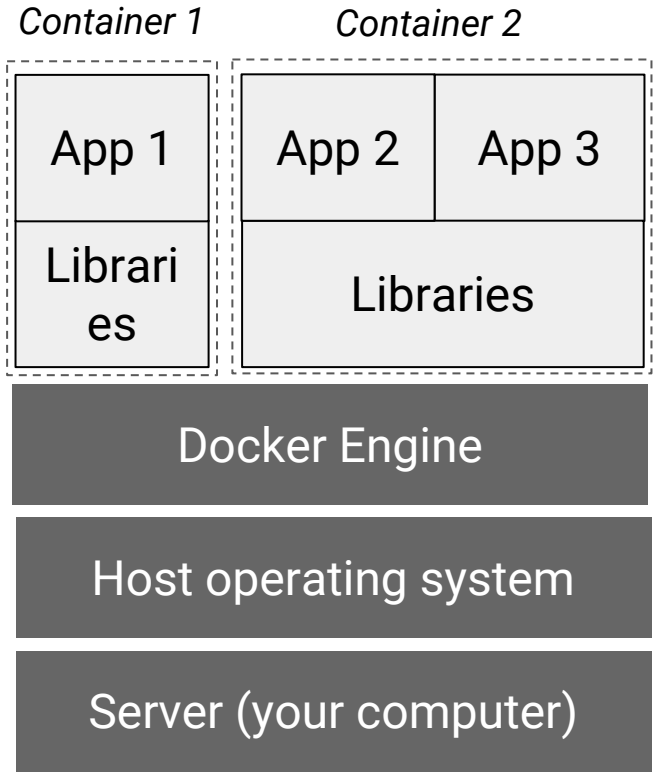
- Docker motivation
- Docker components
- Install Docker
- Describe how to make a Docker container from scratch
- Introduce Docker Hub
- Docker volumes
- Docker command reference
- References

Docker motivation

From founder and CTO of Docker Soloman Hykes, as motivation for Docker:

“Shipping code to the server should not be hard.”

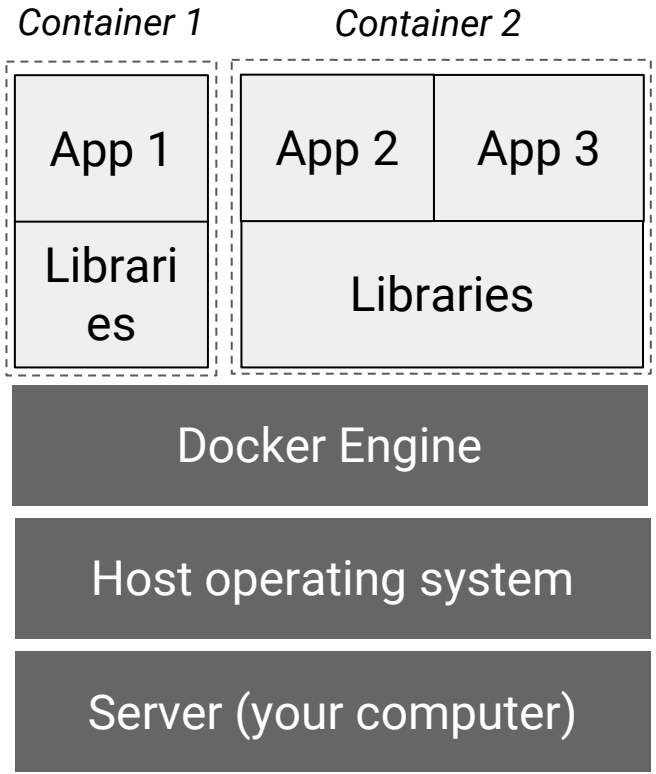
The developer can specify what’s inside the container, and IT just needs to handle the container.



No matter where the container goes, what’s inside will work the same way.

What is Docker

- Docker is an open-source project based on Linux containers.
- Docker is a software *containerization* platform.
- A docker *container* is a stand-alone piece of software that includes everything needed to run it.
- Containers are isolated from each other, but can share libraries where possible.
- Containers are created with Linux, but share a *kernel* with almost any type of OS.



A kernel is the central part of the OS.

Docker Timeline and Popularity

Docker was made open source in 2013 by Soloman Hykes at dotCloud.

Main contributors in 2016: Docker team, Cisco, Google, Huawei, IBM, Microsoft.

Has been downloaded more than 13 billion times as of 2017.

Main reasons for popularity:

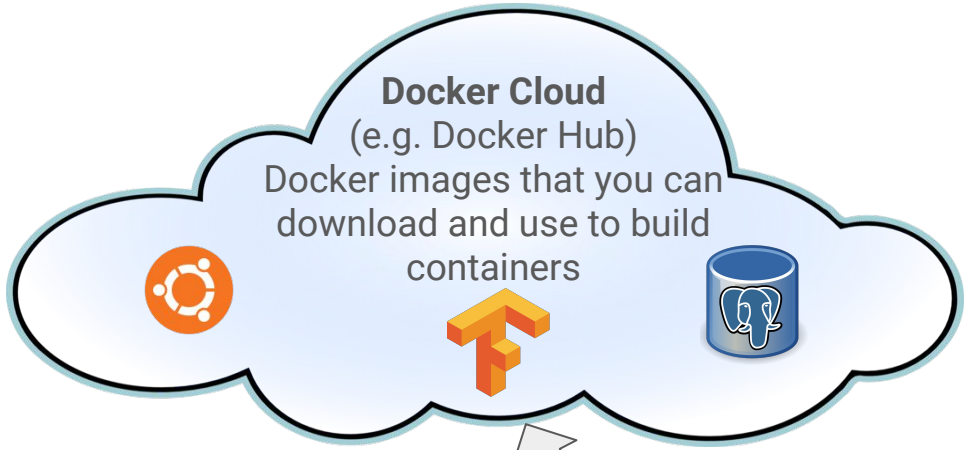
Ease of use: Build and test it on your laptop and it will run anywhere.

Speed: Containers are sandboxed environments running on the kernel that take up fewer resources and are faster to start-up than virtual machines.

Docker Hub: the free app-store for Docker images

Modularity and Scalability: An application can be divided into multiple containers but those containers can communicate.

Docker Components



Docker Client (laptop)

$ docker build
$ docker run
$ docker start
$ docker stop
$ docker pull
$ docker push

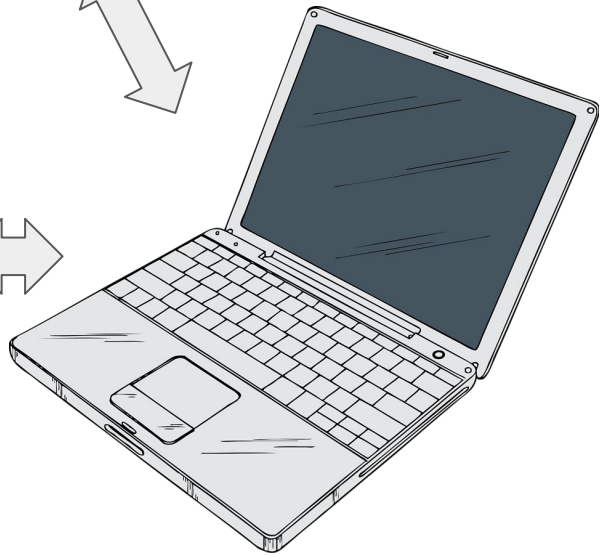
Docker Host (laptop)

Images

Containers

Volumes (where data persists)

Daemon (executes the commands)



Breakout - Install Docker

- For us, Docker is a relatively simple way to make an application that you know will work the same for someone else (as long as they have Docker, too.)
- The CE (Community Edition) is free.
- Install Docker Ubuntu Linux
- Install Docker Mac
- Install Docker Toolbox for Windows and older versions for Mac
- After installation, can you get information on your installation?
  - `$ docker info`
    - For Mac you may want to allocate more memory to Docker (4 GB instead of 2)
- After installation, can you run Docker's hello-world container?
  - `$ docker run hello-world`

Making a Docker container from scratch

- Docker containers contain running applications.
  - All dependencies are taken care of inside the container.
- You are going to want to make containers for your applications.
  - A Flask app, for example.
- A Docker *container* is made by running a Docker *image*.
  - `$ docker run <image>`
- You can get existing Docker images from Docker Hub.
- You can make your own Docker *image* by building from a *Dockerfile*.
  - `$ docker build -t <name_of_image> <path_Dockerfile_and_sup_files>`
- A *Dockerfile* is a textfile that defines the environment inside the container.
  - Can pull from parent images, specify programs to install, define environment variables, expose ports, run commands.

Making a Docker container from scratch (1/3)

A text *Dockerfile* defines what's in your container:

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Source: Docker Docs, Getting Started

Making a Docker container from scratch (2/3)

Put the *Dockerfile* and other files/directories required by your application in the same directory and then build them into a *Docker Image* using the *Docker Client*:

```
$ ls
Dockerfile              app.py                  requirements.txt
```

```
$ docker build --tag=friendlyhello .
```

View the *Docker Image* “friendlyhello” that you’ve created:

```
$ docker image ls

REPOSITORY            TAG                 IMAGE ID
friendlyhello         latest              326387cea398
```

Source: Docker Docs, Getting Started

Making a Docker container from scratch (3/3)

Use the *Docker Client* and your *Docker Image* to build your container. The -p flag maps the laptop's port 4000 to the container's port 80.

```
$ docker run -p 4000:80 friendlyhello
```

You should see a message that Python is serving your app at http://0.0.0.0:80.

But that message is coming from inside the container, which doesn't know you mapped port 80 of that container to 4000, making the correct URL http://localhost:4000

Source: Docker Docs, Getting Started

Docker command reference

Basic Docker commands

Command	Description
`docker`	Show available docker commands
`docker COMMAND --help`	Gets help on COMMAND
`docker info`	View details about your Docker installation
`docker login`	Logs in to Docker Hub so you can pull your own images
`docker build`	Use to make an image from a Dockerfile
`docker run <image>`	Pull (if needed) and start a container from an image
`docker image ls`	Show local images
`docker volume ls`	Show local volumes
`docker start <name | id>`	Start a pre-existing container
`docker logs <id>`	Return log of container (get token for notebook after starting)
`docker stop <name | id>`	Stop a container
`docker container ls`	Show running containers
`docker container ls -a`	Show running and stopped containers
`docker container rm <name | id>`	Removes a container

Breakout: Dockerfile -> Docker Image -> Docker Container

`Dockerfile, app.py` and `requirements.txt` have been given to you in an `app` directory. Go through the three steps described previously to create a Docker Image, and then build (and run) a container!

Questions:

- What do you type to get syntax help on a particular Docker command?
- What would you type at the command line to list your Docker Images?
- What would you type at the command line to list your Docker Containers?
- How do you show currently running containers?
- How do you show stopped containers?
- How do you stop a container?
- How do you delete a container?
- How do you build a new container and specify its name?

Docker Hub

Docker Hub is a service provided by Docker for finding and sharing container images with your team. It provides the following major features:

- Repositories: Push and pull container images.
- Teams & Organizations: Manage access to private repositories of container images.
- Official Images: Pull and use high-quality container images provided by Docker.
- And look at all the other images that are available.

Breakout:

1) Sign up for Docker Hub
2) Push your `friendlyhello` image to Docker Hub.

Docker Volumes

- Docker volumes are the preferred way of persisting data needed and created by containers.
- Containers by themselves have a writable layer (you can store data in a container without using a volume), but a volume is better because:
  - it doesn't increase the size of the container using it
  - the volume exists outside of the lifecycle of the container
  - the volume can be shared with other containers
- You can create a volume when a container is made.
- You can move data back and forth between the host file system and the docker container (my_data in my_container).

```
docker cp /tmp/my_data/. my_container:/my_data
```

More on Docker volumes

Detail syntax mounting a volume to a container

Make a tensorflow container named *tf_container* and a volume called *tf_volume*.

```
$ docker run -it \
          --name tf_container \
          --mount source=tf_volume,target=/notebooks \
          -p 8888:8888 \
          gcr.io/tensorflow/tensorflow:latest-py3
```

Detail syntax mounting a volume to a container

Make a tensorflow container named *tf_container* and a volume called *tf_volume*.

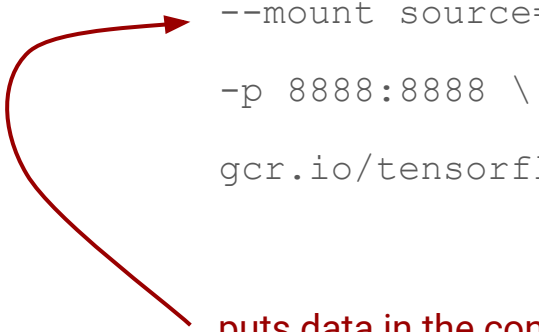
```
$ docker run -it \
             --name tf_container \
             --mount source=tf_volume,target=/notebooks \
             -p 8888:8888 \
             gcr.io/tensorflow/tensorflow:latest-py3
```

in an interactive terminal session

name the container

client port:container port

Docker image:tag



puts data in the container /notebooks directory into Docker volume tf_container

Docker command reference

Basic Docker commands

Command	Description
`docker`	Show available docker commands
`docker COMMAND --help`	Gets help on COMMAND
`docker info`	View details about your Docker installation
`docker login`	Logs in to Docker Hub so you can pull your own images
`docker build`	Use to make an image from a Dockerfile
`docker run <image>`	Pull (if needed) and start a container from an image
`docker image ls`	Show local images
`docker volume ls`	Show local volumes
`docker start <name | id>`	Start a pre-existing container
`docker logs <id>`	Return log of container (get token for notebook after starting)
`docker stop <name | id>`	Stop a container
`docker container ls`	Show running containers
`docker container ls -a`	Show running and stopped containers
`docker container rm <name | id>`	Removes a container

References

- Docker Documentation
- Docker on Github
- Docker Hub
- Solomon Hykes talk at Twitter, 2 Oct. 2013
- Preethi Kasireddy - Beginner friendly Docker concepts

Objectives

- Docker motivation
- Docker components
- Install Docker
- Describe how to make a Docker container from scratch
- Introduce Docker Hub
- Docker volumes
- Docker command reference
- References