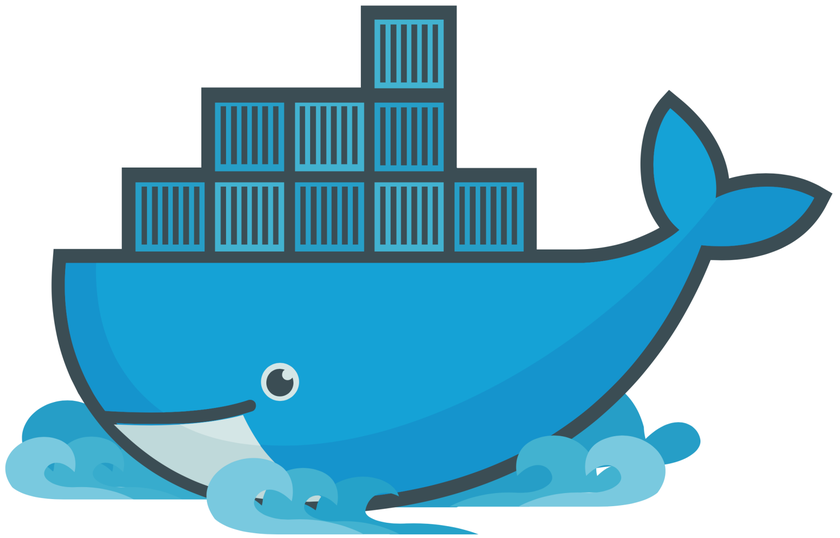
Introduction to Docker

Peter Story



https://www.docker.com/

Find Teammates for Course Project

- If you already have three or four teammates, you are all set!
- If you have a project idea, **but don’t have enough teammates,** write your name(s) and project idea on left-hand side of the board
- **If you don’t have a project idea,** write your name on the right-hand side of the board
- For the first 5 minutes, discuss with others to form your team

Today you will learn…

- How to install and run PostgreSQL using Docker

What Problems Does Docker Solve?

- If you develop software on your laptop, how do you run it elsewhere?
  - A web application, which you need to run on a server
  - A program, which a colleague wants to run on their PC so they can help with development
- Your laptop's hard drive died, and after restoring from your backups your software won't run! What changed?!
- Research replicability: different software versions may give different results

What Problems Does Docker Solve?

- Challenges:
  - What dependencies does your software require?
  - Does your software support the host's OS?
- Without Docker, you might spend a half hour, a few hours, or even days setting up your software on a different computer
- Using Docker, you can get your software running in minutes!

What Problems Does Docker Solve?

- Larger principle: configuration as code

Install and Run a Web Server

- Download and run the Nginx web server from Docker Hub:

```
docker run \
  --rm \
  --volume ./public_html:/usr/share/nginx/html \
  --publish 9999:80 \
  nginx
```

Install and Run PostgreSQL

- Download and run PostgreSQL from Docker Hub:

```
docker run \
  --rm \
  --volume ./postgres_data:/var/lib/postgresql/data \
  --env POSTGRES_PASSWORD=mysecretpassword \
  --env POSTGRES_USER=myusername \
  --name postgresdemo \
  postgres:16.1
```

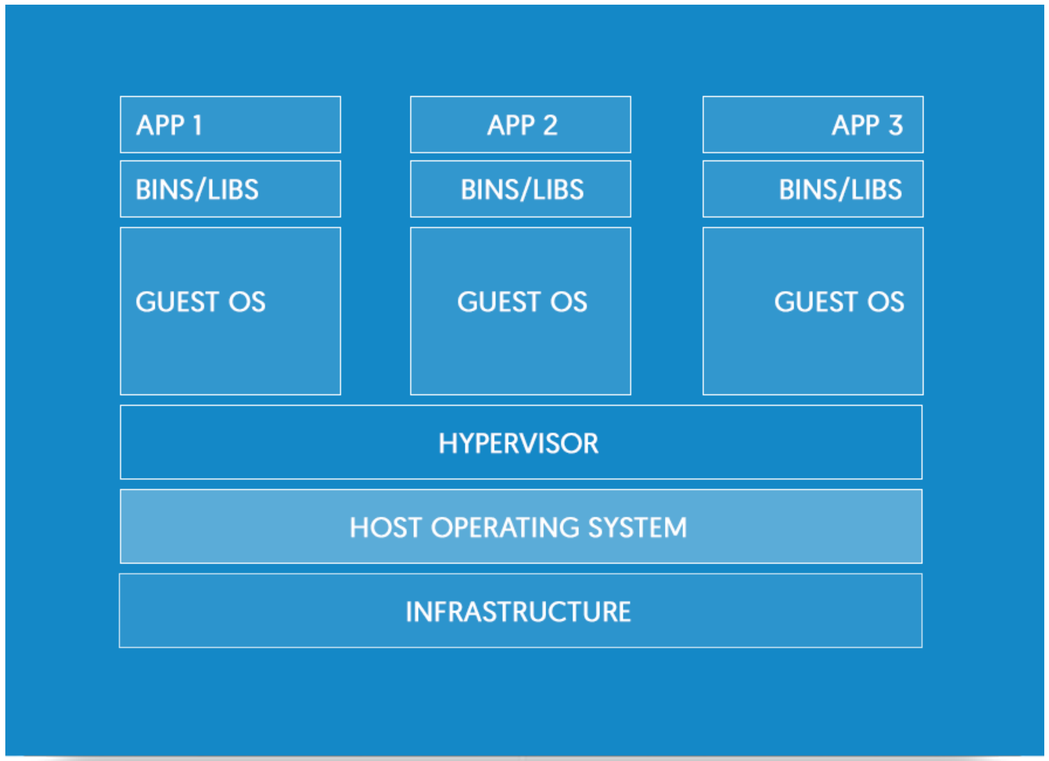
- Open a SQL shell:

```
docker exec \
  --interactive \
  --tty \
  postgresdemo psql --username=myusername
```

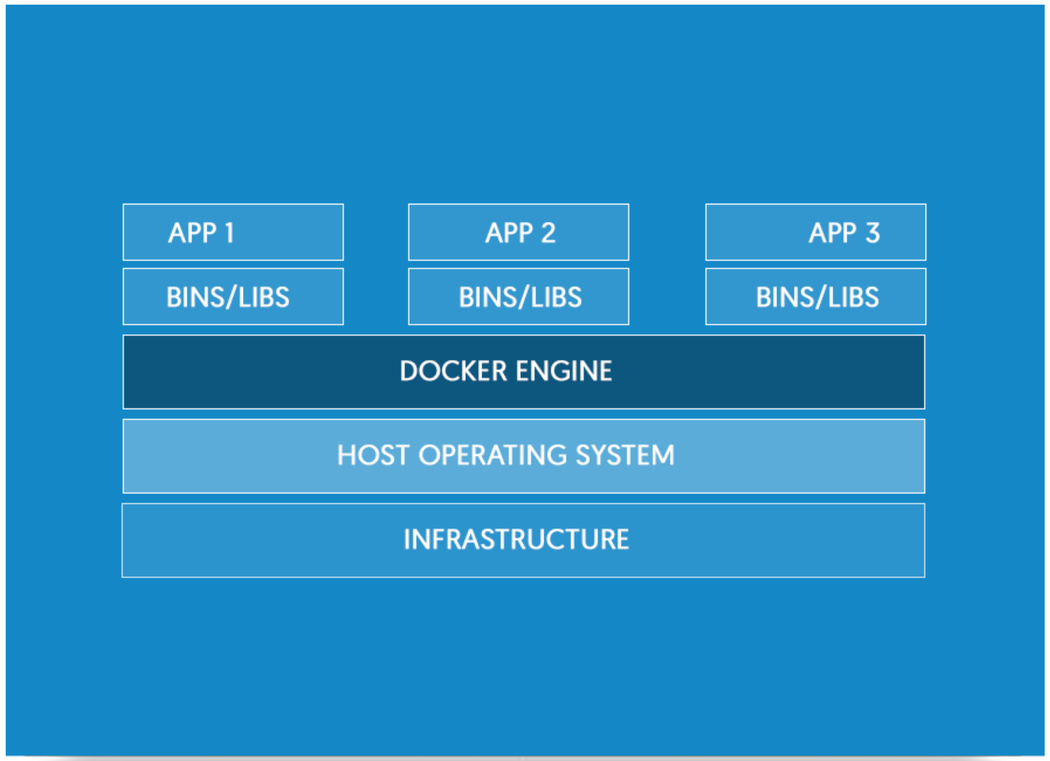
What is Docker?

- Tools for encapsulating software and its dependencies in “Docker Images”
- Tools for distributing and running “Docker Images”
  - Docker Hub
  - Docker Swarm
  - Docker Cloud

What is Docker?



Virtual Machines:
OS, Binaries and Libraries, App



Docker Containers:
Binaries and Libraries, App

https://www.docker.com/what-docker

Docker Concepts

- **Images**
  - Layered: building on top of a base image
  - Immutable
- **Containers**
  - Instantiated images
  - Mutable
  - Ephemeral
- **Volumes**
  - Connect container filesystem to the host, or multiple containers together
  - Used to persist data
- **Networks**
  - Docker containers can communicate using user-defined networks

Docker Images

- A Docker image is a binary artifact encapsulating a filesystem and metadata
  - For example, the Nginx image from Docker Hub includes all the resources needed to run the Nginx server (program binary, default config files, etc.)
  - It also includes instructions for how to run the server (the “entrypoint” does initial setup, and the “command” points to the Nginx binary)
- Represented as a series of immutable layers

Docker Containers

- A Docker Container *instantiates* an image

Action	Docker CLI	Container State
Container is created from an image	`docker run IMAGE`	Running
Main process in container exits	N/A	Running → Stopped
Stop signal is sent to container	`docker stop CONTAINER`	Running → Stopped
Container is started by Docker	`docker start CONTAINER`	Stopped → Running
Container is removed	`docker rm CONTAINER`	Stopped → Deleted (container is gone)

Creating Docker Images

- Often, you will use pre-built images from Docker Hub:
  - PostgreSQL, NGINX, Apache, Rails, Python, etc.
  - Keep security in mind: trust official repos, maybe trust automated builds (if you read their Dockerfiles), be wary of others
- To package your own software, create your own images using Dockerfiles

Dockerfile

```
FROM ubuntu:22.04

RUN mkdir /root/hello_world

COPY hello.sh /root/hello_world

CMD ["/root/hello_world/hello.sh"]
```

Dockerfile Explanation

Dockerfile	Docker Image	Explanation
`FROM ubuntu:22.04`	Stripped down Ubuntu distribution	Downloaded from Docker Hub
`RUN mkdir /root/hello_world`	Plus a folder created at /root/hello_world	Adds a layer
`COPY hello.sh /root/hello_world`	Plus a file at /root/hello_world/hello.sh	Adds a layer
`CMD ["/root/hello_world/hello.sh"]`	When the image is run, this command will be run	Adds a layer

Docker Compose

- An important Docker design principle: one process per container
  - **DON'T** install your program, MySQL, Nginx, etc. in the same image/container
  - It is common to use multiple worker processes (e.g., for web requests)
- If you need multiple processes, use Docker Compose to manage multiple containers

Demos

Run Hello World in a Container

Command run in the container

```
docker run ubuntu echo 'Hello world'
```

Image name, available on Docker Hub (latest is used by default)

https://docs.docker.com/engine/tutorials/dockerizing/

Run an Interactive Container

```
docker run -it ubuntu bash
```

```
docker run --help

…

-t, --tty              Allocate a pseudo-TTY
-i, --interactive      Keep STDIN open even if not attached
```

Run a Daemon Container

```
$ docker run -d ubuntu sh -c "while true; do echo hello world; sleep 1; done"

$ docker ps [-a]

$ docker logs CONTAINER_NAME
```

```
docker run --help

…

-d, --detach  Run container in background and print container ID
```

Enter a Running Container

```
docker exec -it CONTAINER_NAME bash
```

```
docker exec --help

…

-t, --tty              Allocate a pseudo-TTY
-i, --interactive      Keep STDIN open even if not attached
```

Stop and Remove a Daemon Container

```
$ docker stop CONTAINER_NAME

$ docker rm CONTAINER_NAME
```

Build an Image

- View:
  - Dockerfile
  - hello.sh (must be executable!)
- Run these commands:
  - `docker build --tag hello_world .`
  - `docker run hello_world`

Volume Map Content to a Web Server

- Review the documentation: https://hub.docker.com/_/nginx/
- View: html/index.html
- Run this command:
  - `docker run --rm --volume ./html:/usr/share/nginx/html:ro --publish 127.0.0.1:80:80 nginx:1.25`
- Load localhost
- Edit the HTML file, then refresh

Compose Files

- Instead of remembering Docker's CLI syntax, describe the setup in a docker-compose.yml file
- View: docker-compose.yml
- Run:
  - `docker compose up`
  - `docker ps`

```
version: '3.4'
services:
  nginx:
    image: nginx:1.25
    ports:
      - "127.0.0.1:80:80"
    volumes:
      - "./html:/usr/share/nginx/html:ro"
```

Compose Commands

- Ensures all containers are started. If necessary, they will also be built and created. -d will start them in the background. You can also specify a service name, to just start one container.
  `docker compose up [–d] [service_name]`

- Restart all containers.
  `docker compose restart [service_name]`

- Stop all containers.
  `docker compose stop [service_name]`

Compose Commands

- Stop and remove all containers.
  `docker compose down [service_name]`

- **Dangerous:** Stop and remove all containers and volumes.
  `docker compose down -v`

- View a container's logs. -f follows the logs, so they are continually updated.
  `docker compose logs [-f] [service_name]`

List…

- List containers:
  `docker ps [-a]`
- List images:
  `docker image ls`
- List networks:
  `docker network ls`
- List volumes:
  `docker volume ls`

Documentation

- Installing Docker
- Dockerfile reference
- Docker Compose reference
- Docker CLI