

Cloud Computing

DOCKER CONTAINERS

So Far

Introduction

Cloud Service Models

Amazon Web Services

Next

Introduction

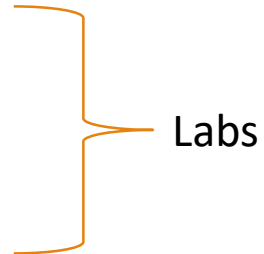
Cloud Service Models

Amazon Web Services

~~Google Cloud Platform~~

~~Microsoft Azure~~

~~Virtualization~~



Labs

Containers

Big Data

Machine Learning

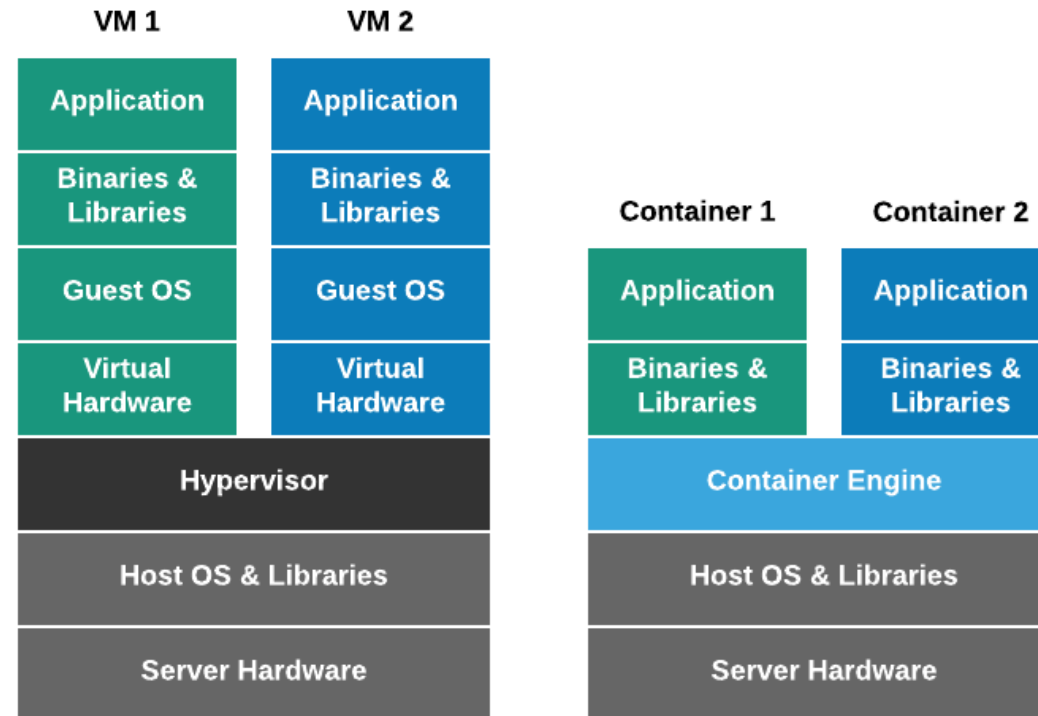
Containers

Lightweight VM alternative

Microservice approach

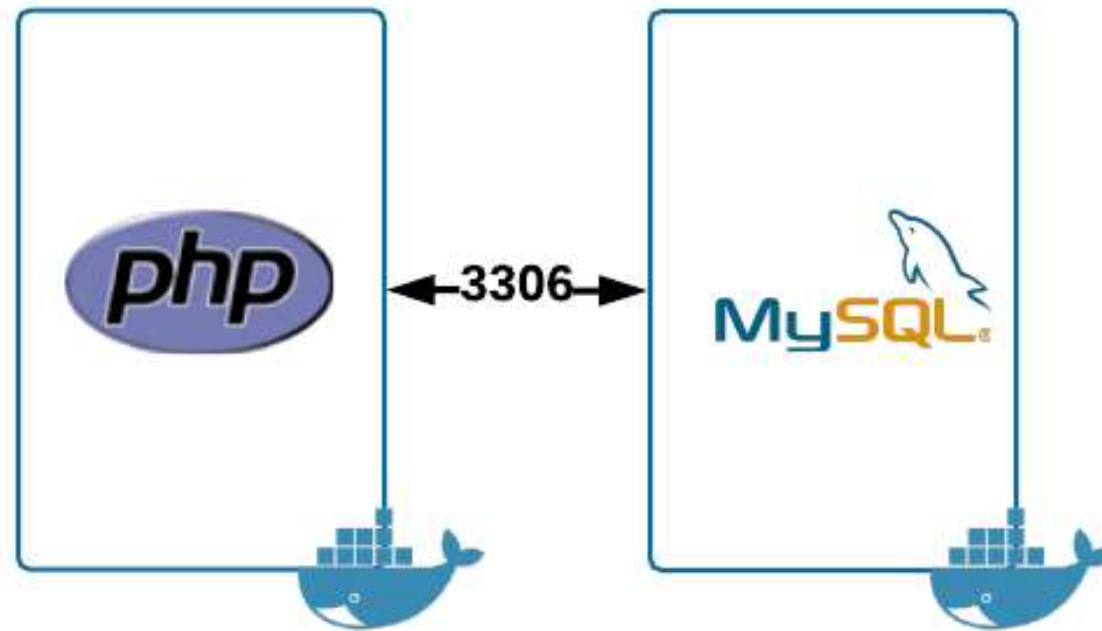
Portable & Scalable

Lightweight



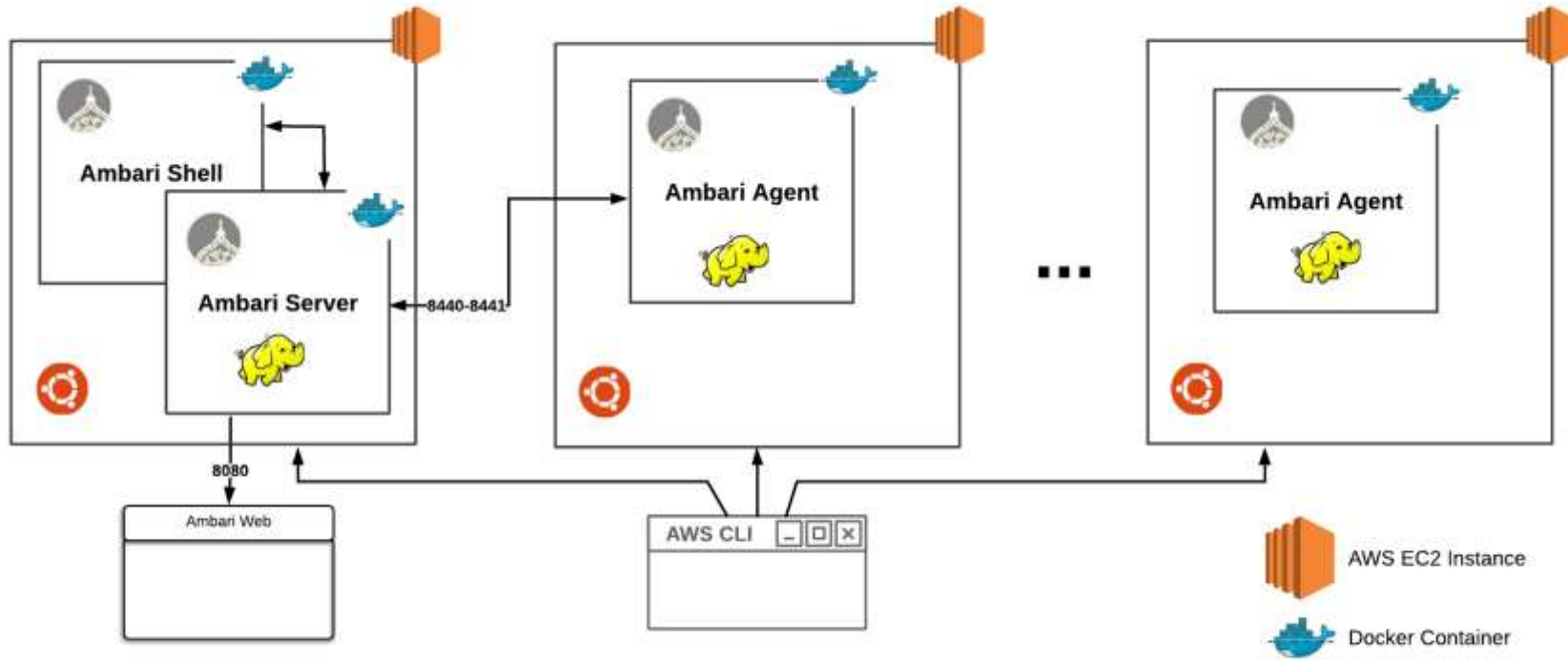
Licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Microservice Approach



Licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Portable & Scalable



Licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

Docker

Introduced in 2012

Dominates the container market

Integrated into all Cloud Platforms

Images and Containers

An **image** is an executable package that includes code, libraries, environment variables, and configuration files needed to run an application, e.g. a MySQL image

A **container** is an instance of the image, i.e. you create a container by running an image

Example: You create a PHP server by starting a container from a PHP image

Docker Store

Official/Certified Docker images store

<https://store.docker.com>

Docker Hub

Stores Docker images created by Docker community users

<https://hub.docker.com>

Docker Commands - Containers

docker run: run a command on a new container

docker exec: run a command on n existing (running) container

docker ps: show running containers

docker stop: stop a running container

docker rm: remove a stopped container

docker commit: commit changes of container to new image

Docker Commands - Images

docker pull: pull an image from hub/store

docker login: login to the docker hub (for push)

docker push: push an image to hub/store

docker images: show locally available images

docker rmi: remove an image

docker build: build an image from a *Dockerfile*

Start a Container

`docker run [<options>] <image> [<command>]`

`docker run -i -t --rm -p 3306:3306 ubuntu bash`

`-i -t` : interactive terminal

`--rm` : remove container when finished

`docker run -d --name db -p 3306:3306 -v $PWD:/data mysql`

`-d`: daemon (run in service mode)

`--name`: container name

`-p` : map container port to host port

`-v`: map container path to host path

Docker services

Different components that define a single application

Each service is a container that runs from an image

Services of the same app are typically connected via a private network

Services can be started using the `docker run` command

Alternatively, we use a docker-compose file to define app services

Example: A webserver may use a *WordPress* service connected to a *MySQL* service via port 3306

docker-compose file

Defines all services of the app, e.g. Wordpress & MySQL

Defines all parameters needed to start each service/container

Defines volumes and networks used by the application

Containers are created/destroyed using `docker-compose up/down`

docker-compose file - Example

version: "3"

services:

db:

image: "mysql:latest"

container_name: database

restart: always

environment:

- MYSQL_USER=username
- MYSQL_PASSWORD=password

volumes:

- '\$PWD/data:/data'

docker-compose file - Example

web:

image: wordpress:latest

container_name: web

ports:

- "80:80"
- "443:443"

restart: always

volumes:

- \$PWD/html:/var/www/html

depends_on:

- db

Dockerfile

Defines the application and environment of a custom image

Built into image using the `docker build` command

More persistent alternative of using the `docker commit` command

Dockerfile commands

FROM: Start from an existing image

WORKDIR: Set working directory within image

ADD: Copy local file into the image

RUN: Run command inside image, e.g. apt install

EXPOSE: Expose certain ports to the world, e.g. ports 80 and 443 for HTTP/HTTPS

ENV: Define an environment variable

CMD: Default command to be run when the container is started

Further Reading

<https://docs.docker.com>

Non-academic Reading

