Wednesday, September 30, 2020

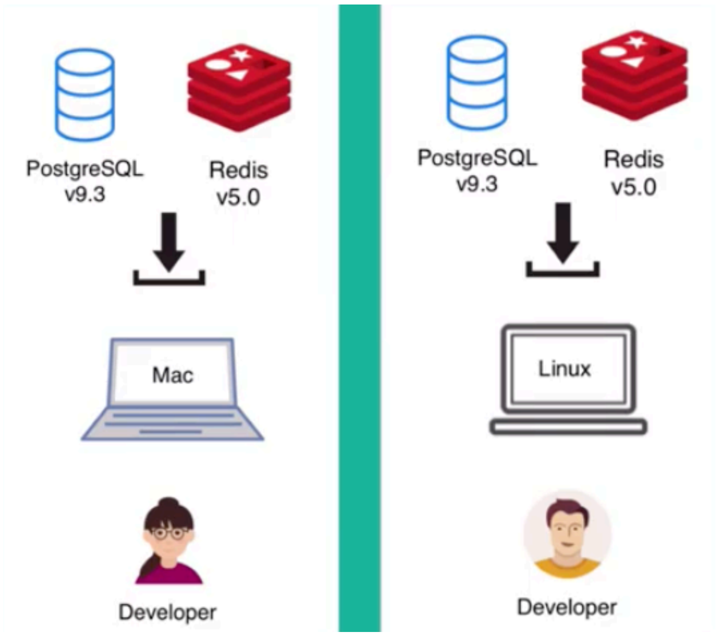
Docker and Kubernetes

Kubernetes - Orchestration tool

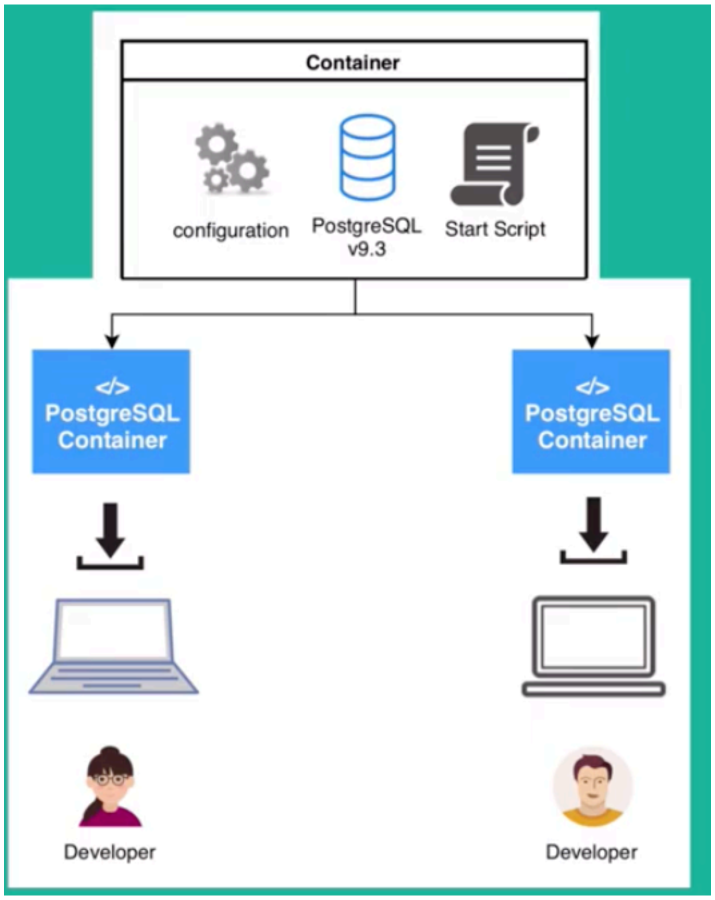
- Container: a package of applications with all the necessary dependencies and configuration
  - Easily shared and moved around
- Container repository
  - Private vs public repos
  - Docker and docker hub (public repo)

Application Development

Before Containers: Inefficiency and unreliable, too varied



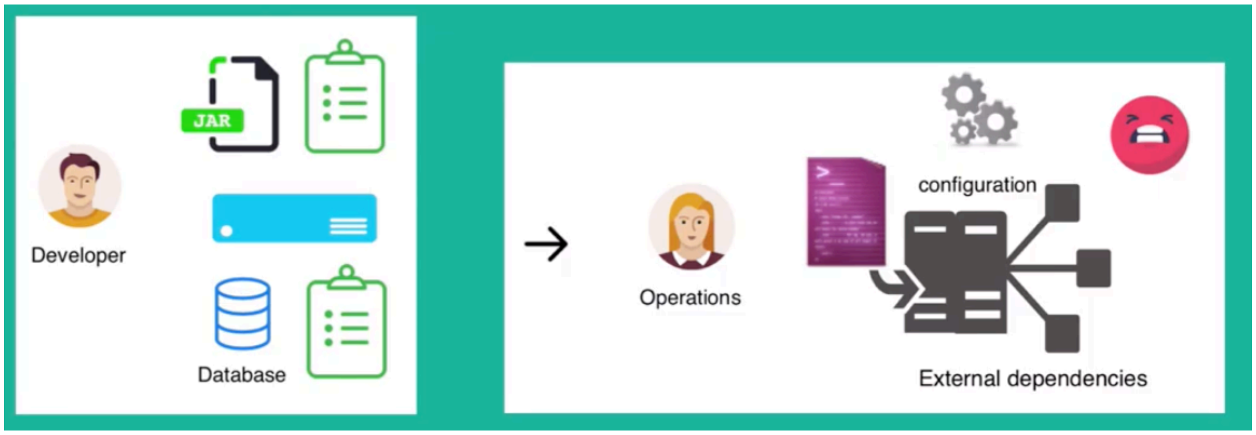
After Containers: same process for all os, one command to run, less variables, reliable and consistent



Wednesday, September 30, 2020

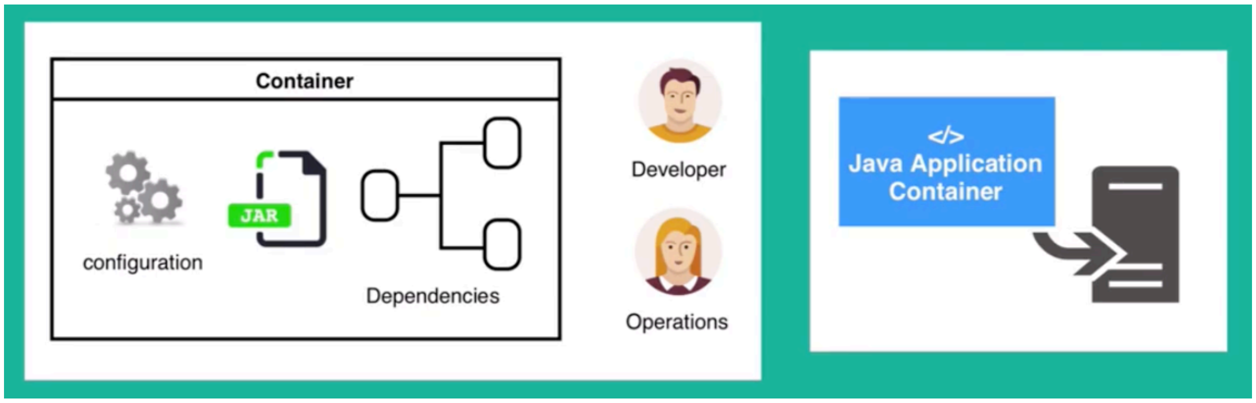
Application Deployment

Before



- Configuration on the server needed
- External dependencies on the server OS
  - Dependency versions

After



Wednesday, September 30, 2020

What is a Container?

Layers of linux base image, application image with configuration

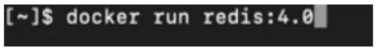
Image : image is the actual package, the applications, the configurations

- Tags are basically versions, like latest

Container : running the image and allocating memory for that application

- has its own virtual filesystem

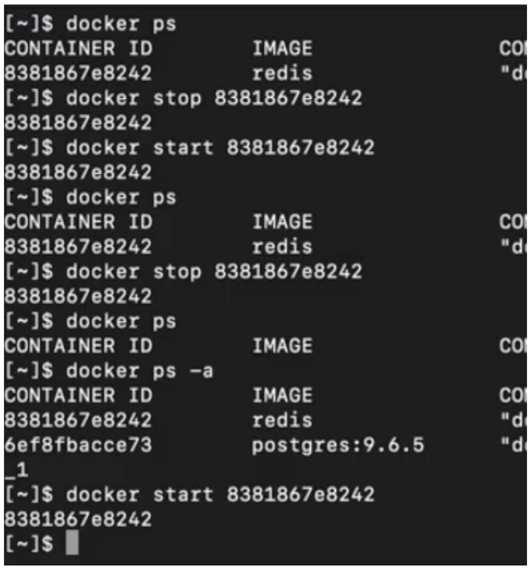
Docker commands:

- Docker run <image name> //will search for docker image locally before searching docker hub, starts new image in container, use this for new containers
  - -d detached mode (in background)
- Docker pull <image> //downloads local copy
- Docker ps //lists all running containers
  - -a //lists all containers, active or not
- Docker stop <image ID> //terminates the container
- Docker start <container ID> //start one or more stopped containers, use this for starting old containers
- Docker run <image>:<tag> 

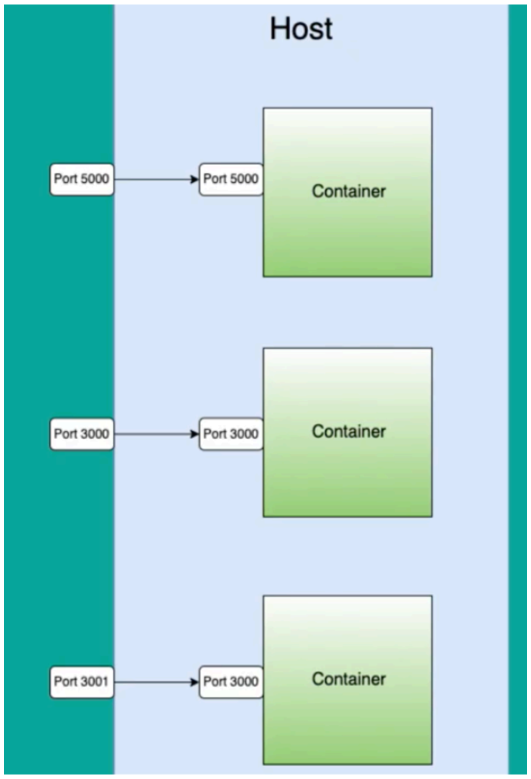
```
[~]$ docker run redis:4.0
```

Multiple versions of the same image can run in parallel without conflict

Wednesday, September 30, 2020



```
[~]$ docker ps
CONTAINER ID        IMAGE               CO
8381867e8242        redis               "d
[~]$ docker stop 8381867e8242
8381867e8242
[~]$ docker start 8381867e8242
8381867e8242
[~]$ docker ps
CONTAINER ID        IMAGE               CO
8381867e8242        redis               "d
[~]$ docker stop 8381867e8242
8381867e8242
[~]$ docker ps
CONTAINER ID        IMAGE               CO
[~]$ docker ps -a
CONTAINER ID        IMAGE               CO
8381867e8242        redis               "d
6ef8fbacce73        postgres:9.6.5      "d
_1
[~]$ docker start 8381867e8242
8381867e8242
[~]$
```



Containers are allowed to listen on the same ports because they're entirely virtual, as long as the host's ports between each container is different (See picture above for reference).

Containers remain unreachable if you don't bind a host port with a container port

Docker run -p <host port>:<container port> //binds host port with container port