جامعة عجمان
AJMAN UNIVERSITY

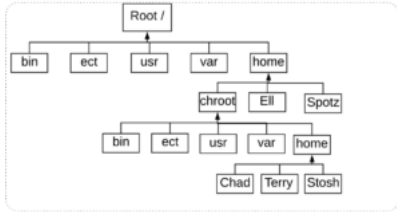
Cloud Technologies - Containers

Mohamed Al Solh

alsolh@alsolh.com

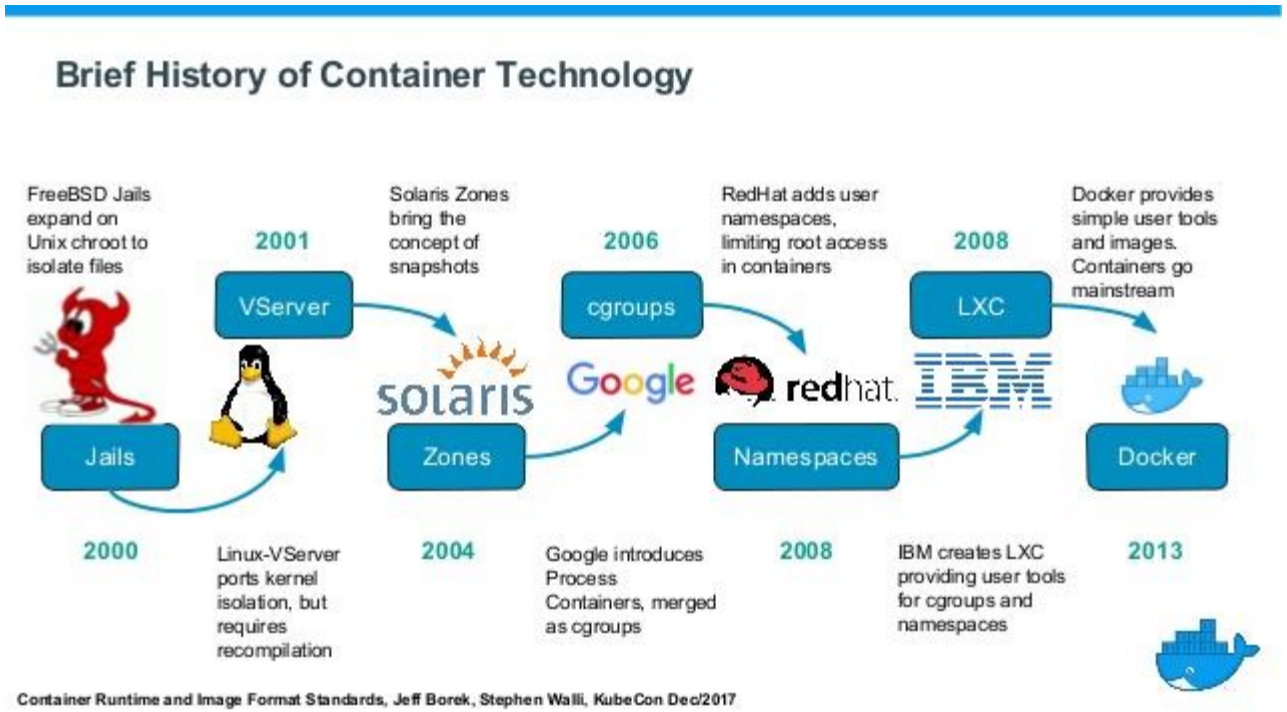
Early History That Led to Containers

- 1960s – it was common to have terminals connected to mainframes, one user was able to bring the whole server down when he runs an application on the mainframe
- 1979 – chroot command introduced to change root directory for a process
- 1990s - Bill Cheswick, a computer security and networking researcher have used chroot to study hackers behaviour, which gave the idea to use jail command



Evolution of Container Technology

- Jails were the first implementation in FreeBSD distribution using chroot
- LXC are still widely used and you can store files and install applications inside them, they are based on cgroups and namespaces, you can even run docker inside them
- Each docker container is designed to run a single application/process unlike LXC which can be considered as a lightweight VM but still uses the same kernel of the OS
- Early versions of docker used to use LXC

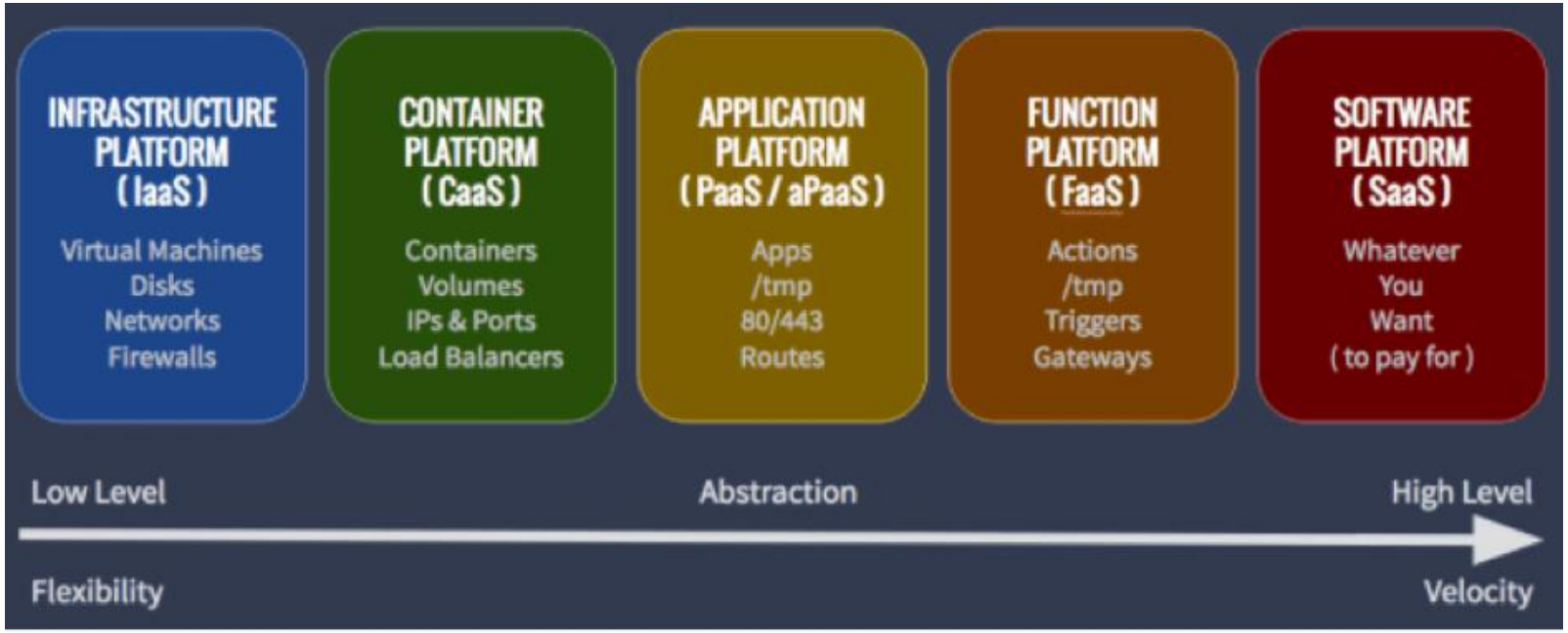


Why to use a container

- **Less overhead.** Containers require less system resources than traditional or hardware virtual machine
- **Increased portability. Applications** running in containers can be deployed easily to multiple different operating systems
  - Containers allow a developer to package up an application with all of the artifacts it needs
- Isolating applications through containers
- Containers provide a consistent experience, as developers and system administrators move code from development environments into production in a fast and replicable way

https://www.netapp.com/us/info/what-are-containers.aspx

Where containers stand in cloud terms

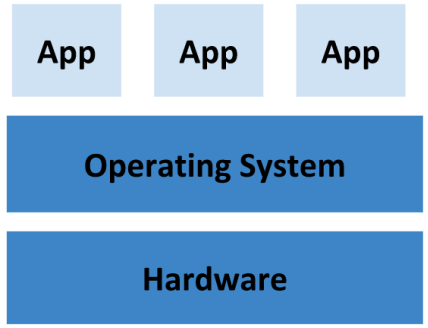


Cloud Platforms, their interfaces, and the scale of abstraction

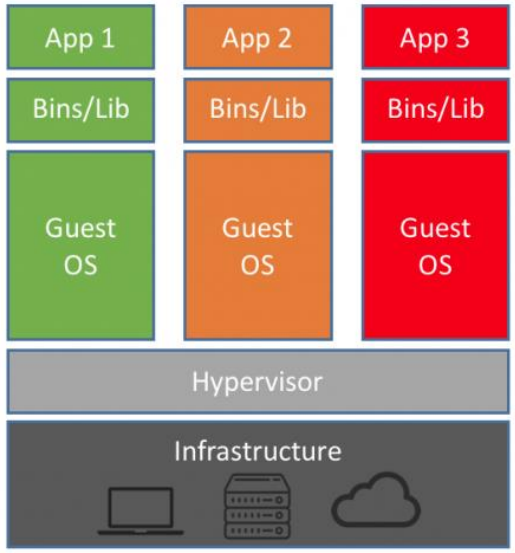
Containers vs VMs

VMs	Containers
Heavyweight	Lightweight
Limited performance	Native performance
Each VM runs in its own OS	All containers share the host OS
Hardware-level virtualization	OS virtualization
Startup time in minutes	Startup time in milliseconds
Allocates required memory	Requires less memory space
Fully isolated and hence more secure	Process-level isolation, possibly less secur

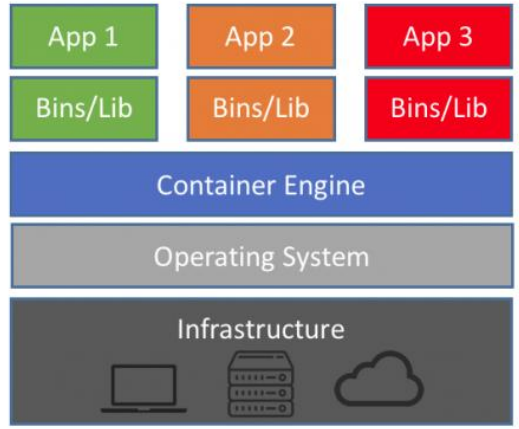
https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm
https://blog.netapp.com/blogs/containers-vs-vms/



Traditional Deployment



Machine Virtualization



Containers

Limitations of a container

- All containers are running inside the host system’s kernel and not with a different kernel
- A container is not a full virtualization stack like Xen, KVM or libvirt
- Security depends on the host system

https://containerjournal.com/topics/container-ecosystems/what-are-containers-and-why-do-we-need-them/

Why docker

- Easy to start with available tools for desktop
- Most github opensource projects have a guide on how to run their project in docker
- Huge repository of ready made images (docker hub) https://hub.docker.com/

Docker Server Installation

- If you are on windows, install docker desktop https://www.docker.com/products/docker-desktop
- Windows home – install **vmware workstation player** and create an ubuntu **virtual machine and install docker ce (community edition)**
- If linux, follow https://docs.docker.com/engine/install/debian/, change the guide based on your distribution
- Docker run hello-world
- Docker Commands
  - https://dzone.com/articles/docker-images-and-containers
  - Docker log
  - Docker ssh login
  - Docker ps –a
  - -d option

Docker Cheat Sheet

- https://github.com/wsargent/docker-cheat-sheet

Why Kubernetes (k8s)

- You are now able to run a docker image, you will face challenges if you are running a production application, k8s does below:
  - Better management to limit resources for pods
  - Scale up and scale down easily with rules
  - load balancing
  - **Self-healing** Kubernetes restarts containers that fail, replaces containers, kills containers that don’t respond to your user-defined health check, and doesn’t advertise them to clients until they are ready to serve
  - **Secret and configuration management -** Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys

Install K8s & 1 Sample Scenario

- https://www.techrepublic.com/article/how-to-add-kubernetes-support-to-docker-desktop/
- Scale
  - https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-interactive/

Portainer

- You will use portainer to easily manage your docker containers running on your machine
- Portainer is another docker container running in your docker desktop environment
- Alternatively you can keep using command line but the web UI of portainer is very intuitive for development use
- https://gist.github.com/SeanSobey/344edd228922ffd4266ae7d451421ab6

Dockerfile & Docker Compose

- Dockerfile https://dzone.com/articles/all-about-hibernate-manytomany-association
  - https://dzone.com/articles/understanding-dockerfile
- Docker Compose https://dzone.com/articles/docker-with-spring-boot-and-mysql-docker-compose-p

- Other Guides
  - https://dzone.com/articles/build-package-and-run-spring-boot-apps-with-docker
  - https://www.baeldung.com/dockerizing-spring-boot-application
  - https://dzone.com/articles/a-start-to-finish-guide-to-docker-with-java
  - https://dzone.com/articles/all-about-hibernate-manytomany-association

Home Work

- Install Docker Desktop
- Install Portainer
- Deploy the following docker images
  - CouchDB
  - Mosquitto
- Go through this tutorial to be more familiar with Dockerfile and docker compose as we will use them in upcoming homeworks which build your project
  - https://www.baeldung.com/dockerizing-spring-boot-application

Your Project from homeworks