<https://github.com/jleetutorial/dockerapp>

<https://www.youtube.com/watch?v=VlSW-tztsvM&t=684s>

Docker technology is one implementation of container based virtualization technologies.

Introduction to virtualization technology:

**Pre-virtualization** => Physical server -> Host operating system -> Bins/libs -> Apps

Problems: 1. Huge cost (and low utilization of resources), 2. Slow deployment, 3. Hard to migrate

Rescue: **Hypervisor based virtualization** => Physical server -> Host operating system -> Hypervisor

1. VM1[ Guest OS (ubuntu) -> binaries/libraries -> app1]
2. VM2[Guest OS (Debian) -> binaries/libraries -> app2] etc…

Hypervisor providers => VMWare and virtualbox, but now-a-days virtual VMs are not in physical server they are getting deployed in cloud providers AWS, Microsoft Azure, google cloud…

Problems: 1. Cost effective 2. Easy to scale and Limitations: 1. Kernel resource duplication (if all VMs have same OS, then lot of duplication of kernel related files) 2. Application portability issue (VMs are not good at portability)

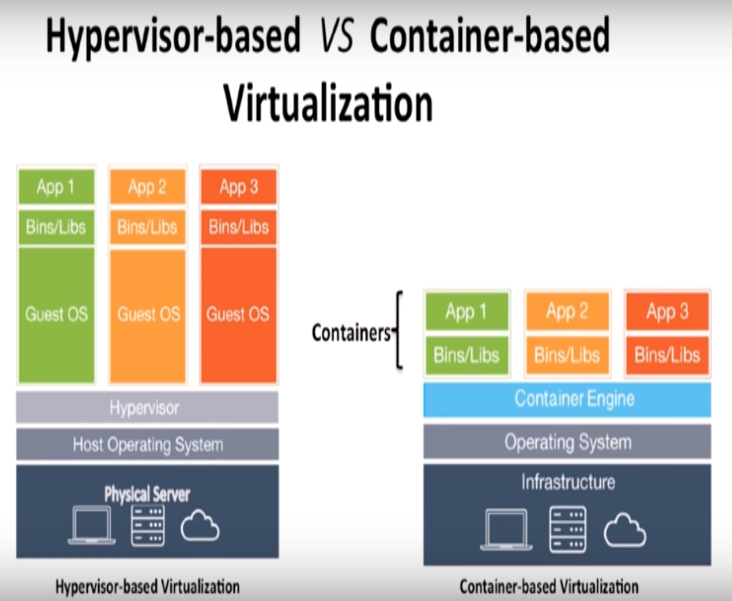
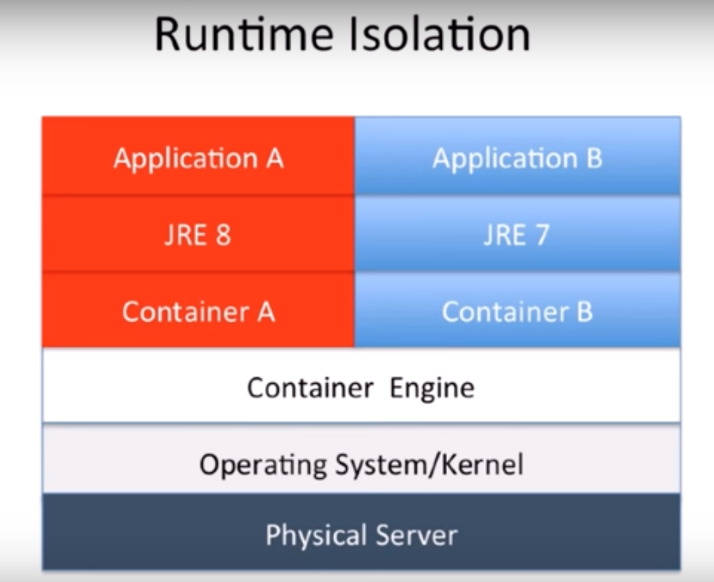
Rescue: Container based virtualization comes to resolve these above hypervisor based virtualization issues.

**Container based virtualization (replication of kernels, no own copy of kernel)**: physical server -> operating system -> container engine => container1 (bins/libs, jre7 -> app1) and container2 (bins/libs, jre8 -> app2) etc…

**Runtime isolation:**

Why cannt we run all applications in one single VM? If there is a need to have a different JRE’s for different applications, then its not possible to put in single VM, but containers can handle this easily.

Benefits: 1. Cost effective (requires lesser resources compared to VMs), 2. Fast deployment (booting time is very less) 3. Guaranteed portability

Docker server-client architecture:

1. Docker Clients (build, pull, run etc…) => 1. Docker CLI and 2. Kitematic GUI

Docker client can also connect to a remote docker daemon/engine.

1. Docker Host (docker daemon/engine/server-per system process-building, running and distributing containers, images, containers etc..)

Docker daemon doesn’t run on any non-linux platforms natively because it uses linux specific kernel features. Window/MacOS -> docker engine is running on a linux VM created on top of it.

1. Registry

**Docker installation:**

Docker toolbox installation is for windows < 10 and mac OS x 10.10.3 or lesser or non-linux, otherwise we have direct docker installable package to take care.

Like JVM, Docker installation is specific to an operating system, hence we have different distributions specific to

Linux -> Ubuntu, Debian, redhat, suse, fedora, centos etc..

Windows -> docker for windows or docker toolbox

Mac OS -> docker for mac or docker toolbox

After installation, use “**docker info**” to check the docker system information.

**Note:** All docker commands output is captured in the below attachment.

“docker-machine ls” and “docker version” also captured.

**Docker concepts:**

1. Images

Images are read only templates used to create containers.

Images are created with the docker build command either by us or by other docker users and push them into registry. All images are stored in docker registry/hub.

Images are composed by layers of other images.

1. Containers

If an image is a class, then container is an instance of the class – a runtime object.

Containers are lightweight and portable encapsulation of an environment in which to run applications. Containers are created from images, inside a container, it has all the binaries and dependencies needed to run the application.

1. Registry and Repositories. Registries -> to store our images, public registry/docker hub or we can maintain our own registry. Inside a registry, images are stored as repositories. Docker repository is a collection of different version of the same image name with different tags/versions.
2. Docker Hub -> public registry with lots of images stored in it.

Why use official images? 1. Clear documentation, 2. Dedicated team for reviewing image content and 3. Security updates in timely manner 4. More support from the community

During run command, if we don’t mention the tag (or version), then the image will from the default, which is usually the latest version of it (eg mysql).

**Deep dive into docker containers:**

Foreground vs background:

-d => detached mode to run the container in background

The container started in background mode will exit when the root process used for starting exits.

In foreground mode, the process in the container is attached to the console and process’s stdin, stdout and stderr are output to the console.

**Docker port mapping and docker logs:**

-p host\_port:container\_port

docker run -it --rm -p 8888:8080 tomcat:8.0

**Volumes:**

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.

