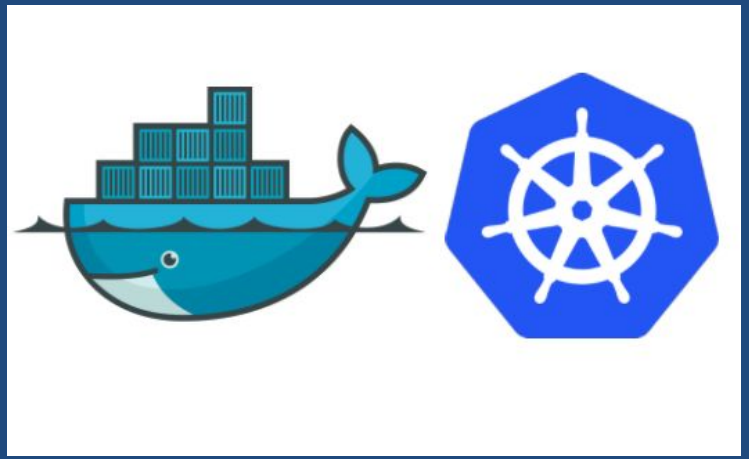
Docker Introduction



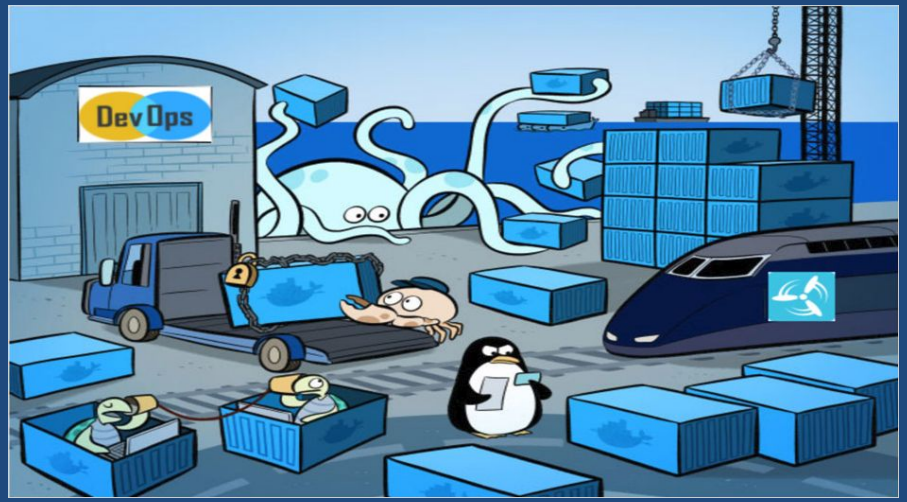
Session 2

9th June 2018

Rama Krishna Bhupathi

Docker

Docker is a platform for developers and sysadmins to develop, deploy, and run applications with containers. The use of Linux containers to deploy applications is called containerization. Containers are not new, but their use for easily deploying applications is.



Docker

Containerization is increasingly popular because containers are:

- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly.

Images and Containers

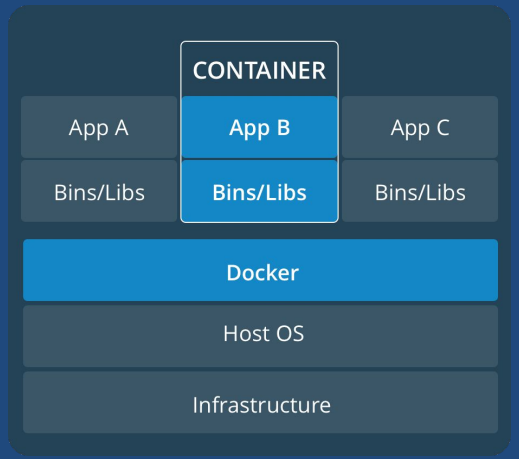
A container is launched by running an image. An image is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files.

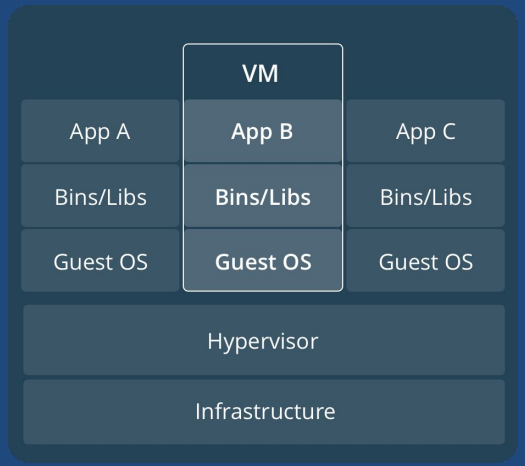
A container is a runtime instance of an image..what the image becomes in memory when executed (that is, an image with state, or a user process). You can see a list of your running containers with the command, docker ps, just as you would in Linux.

Containers and VMs

A **container** runs *natively* on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight.

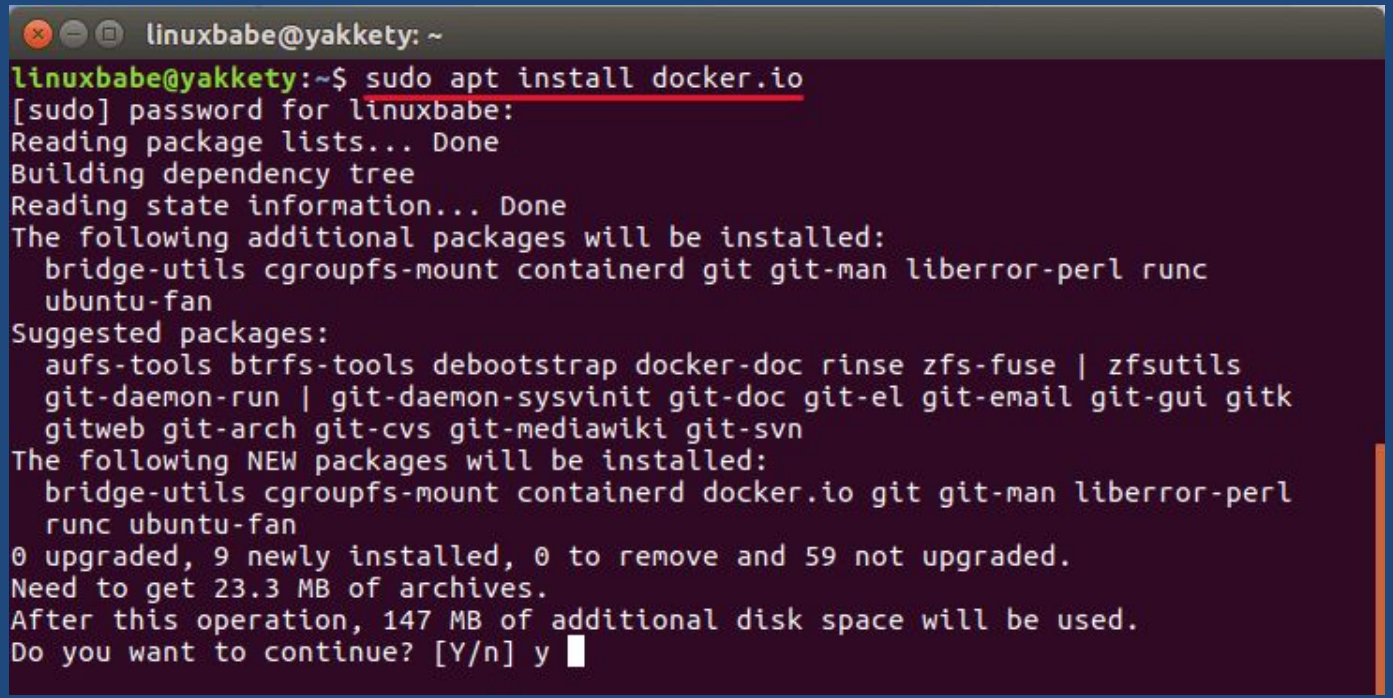
By contrast, a **virtual machine** (VM) runs a full-blown “guest” operating system with *virtual* access to host resources through a hypervisor. In general, VMs provide an environment with more resources than most applications need.





Docker Installation

https://docs.docker.com/install/



Docker works on MacOS , Windows and all Linux flavors

Docker Installation/Configuration

The docker daemon always runs as the root user, and since Docker version 0.5.2, the docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user root, and so, by default, you can access it with sudo.

However we will tweak it so we do not always need to use sudo to run docker commands.

sudo groupadd docker

sudo gpasswd -a $USER docker

sudo service docker start

Docker Basics

Try the following…

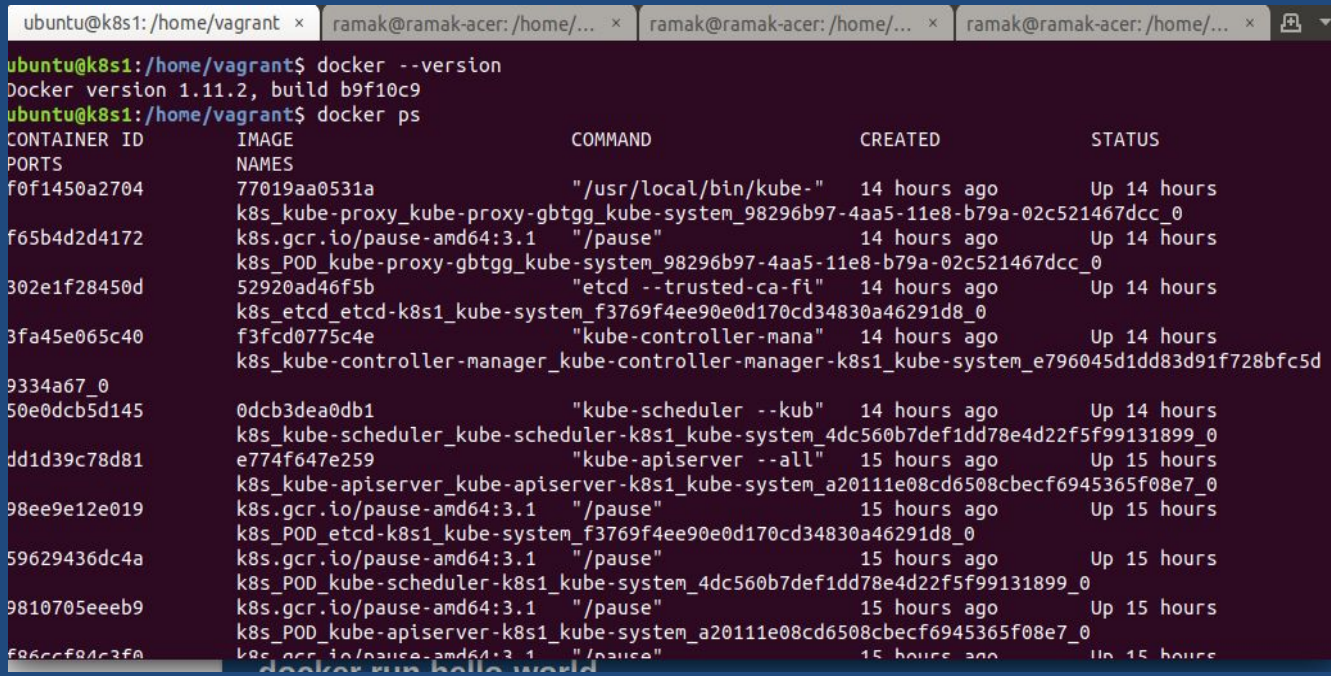
docker --version

docker info

docker ps

docker run hello-world

docker images



Docker Basics

Let us try something more...run the following

```
docker search redis  # what and where is it searching?
docker run -d redis  # what is it doing ?

docker ps  # what is it doing ?

docker stats  # what do you get?

docker inspect  <containerid> # what do you get?

docker logs  # what do you get?
```

Docker Hands-on-exercises

Goto this URL https://www.katacoda.com/courses/docker

Sign up or Sign in.

Run the commands from the previous side.

Docker Hands-on-exercises

1)Let us deploy and run *redis* database image.

Stop the container using docker stop <containerID>

Delete the docker container using docker rm <ContainerID>

Delete the docker image using docker rmi <imageID>

```
Access the directory and check the file system  . This is where Docker stores all the config information /var/lib/docker
```

Docker : Install a webserver on a container

Create the required Files

Run the following

```
docker build -t webserver-image:v1 .
docker images
docker run -d -p 8880:8880 webserver-image:v1
```

Dockerfile

https://docs.docker.com/get-started/#test-docker-installation

Dockerfile defines what goes on in the environment inside your container like Packages to install , download , configure and run.