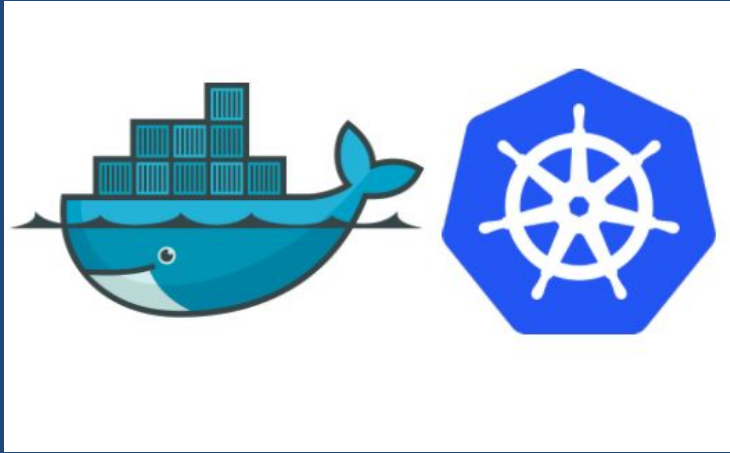


Docker Introduction



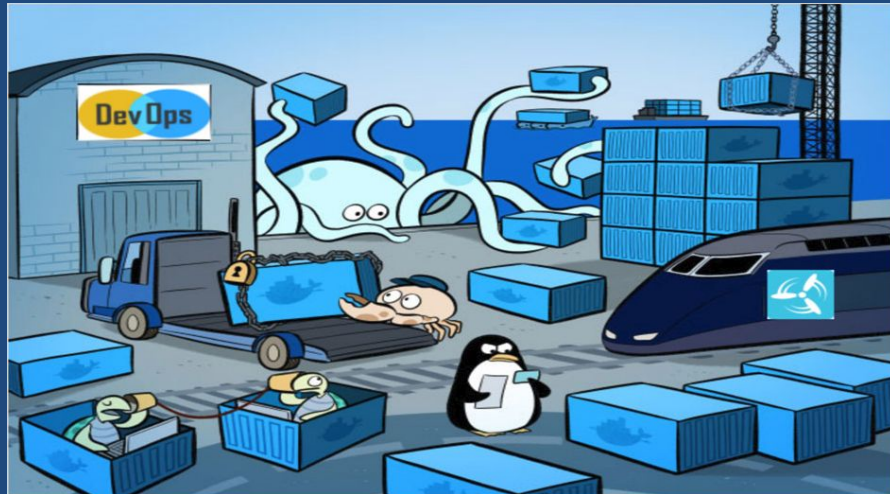
Session 2

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

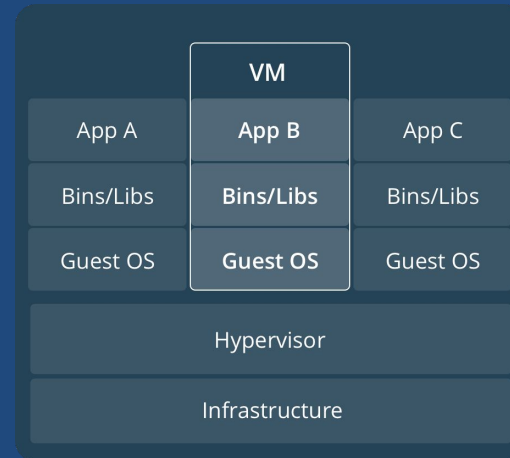
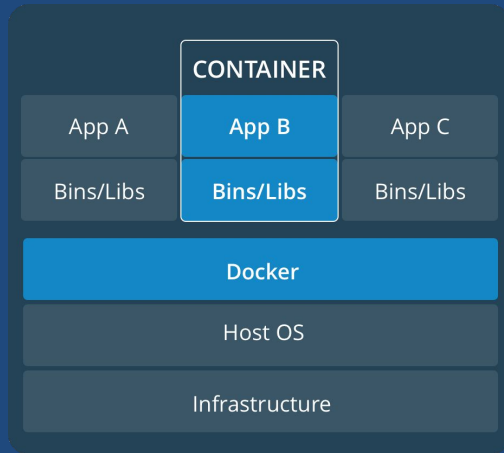
A container is launched by running an image. An image is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files.

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/>

```
linuxbabe@yakkety: ~  
linuxbabe@yakkety:~$ sudo apt install docker.io  
[sudo] password for linuxbabe:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  bridge-utils cgroupfs-mount containerd git git-man liberror-perl runc  
  ubuntu-fan  
Suggested packages:  
  aufs-tools btrfs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils  
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk  
  gitweb git-arch git-cvs git-mediawiki git-svn  
The following NEW packages will be installed:  
  bridge-utils cgroupfs-mount containerd docker.io git git-man liberror-perl  
  runc ubuntu-fan  
0 upgraded, 9 newly installed, 0 to remove and 59 not upgraded.  
Need to get 23.3 MB of archives.  
After this operation, 147 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

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

```
ubuntu@k8s1: /home/vagrant x ramak@ramak-acer: /home/... x ramak@ramak-acer: /home/... x ramak@ramak-acer: /home/... x
ubuntu@k8s1:/home/vagrant$ docker --version
Docker version 1.11.2, build b9f10c9
ubuntu@k8s1:/home/vagrant$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
f0f1450a2704	77019aa0531a	"/usr/local/bin/kube-"	14 hours ago	Up 14 hours
	k8s_kube-proxy_kube-proxy-gbtgg_kube-system_98296b97-4aa5-11e8-b79a-02c521467dcc_0			
f65b4d2d4172	k8s.gcr.io/pause-amd64:3.1	"/pause"	14 hours ago	Up 14 hours
	k8s_POD_kube-proxy-gbtgg_kube-system_98296b97-4aa5-11e8-b79a-02c521467dcc_0			
302e1f28450d	52920ad46f5b	"etcd --trusted-ca-fi"	14 hours ago	Up 14 hours
	k8s_etcd_etcd-k8s1_kube-system_f3769f4ee90e0d170cd34830a46291d8_0			
3fa45e065c40	f3fcd0775c4e	"kube-controller-mana"	14 hours ago	Up 14 hours
	k8s_kube-controller-manager_kube-controller-manager-k8s1_kube-system_e796045d1dd83d91f728bfc5d			
9334a67_0				
50e0dc5d145	0dcb3dea0db1	"kube-scheduler --kub"	14 hours ago	Up 14 hours
	k8s_kube-scheduler_kube-scheduler-k8s1_kube-system_4dc560b7def1dd78e4d22f5f99131899_0			
dd1d39c78d81	e774f647e259	"kube-apiserver --all"	15 hours ago	Up 15 hours
	k8s_kube-apiserver_kube-apiserver-k8s1_kube-system_a20111e08cd6508cbecf6945365f08e7_0			
98ee9e12e019	k8s.gcr.io/pause-amd64:3.1	"/pause"	15 hours ago	Up 15 hours
	k8s_POD_etcd-k8s1_kube-system_f3769f4ee90e0d170cd34830a46291d8_0			
59629436dc4a	k8s.gcr.io/pause-amd64:3.1	"/pause"	15 hours ago	Up 15 hours
	k8s_POD_kube-scheduler-k8s1_kube-system_4dc560b7def1dd78e4d22f5f99131899_0			
9810705eeeb9	k8s.gcr.io/pause-amd64:3.1	"/pause"	15 hours ago	Up 15 hours
	k8s_POD_kube-apiserver-k8s1_kube-system_a20111e08cd6508cbecf6945365f08e7_0			
f86ccf84c3f0	k8s.gcr.io/pause-amd64:3.1	"/pause"	15 hours ago	Up 15 hours

```
docker run hello-world
```



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 # 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 that saw in 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

```
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.

Dockerfile example

- The following is the dockerfile for openssh-server

```
FROM centos:centos6
MAINTAINER cawamata

RUN yum update -y
RUN yum install -y openssh-server
RUN echo 'root:test' | chpasswd
RUN sed -i '/pam_loginuid%.so/s/required/optional/' /etc/pam.d/sshd
RUN /sbin/service sshd start
EXPOSE 22
CMD /usr/sbin/sshd -D
```