===================Docker============================

What is Docker?

It is a too (or set of tools depending on how you look at it)

that packages up an application and all its dependencies in a virtual container so that it can be run on any linux

distribution

It runs containers in form of appliction regardless of knowing operating system distribution.

Does it run on linux only?

Docker can "emulate" Linux within its container space but clients can be installed

on Windows or MacOS

Summary:

Docker offers you the ability to isolate your applications,

Standardize build and deployment process and to create

standard repeatable processes in your

software and infrastructure.

For Windows, since the Docker Engine itself uses "Linux-Specific" kernel features,

it needs to use a lightweight virtual machine to do its works. You then use cilent

to control that virtualized Docker Engine to run work with your containers

When would I use Docker?

There are a lot of reasons to use Docker. Although you will generally hear

about Docker used in conjuction with development and deployments of

applications, there are a ton of examples for use:

Configuration Simplifiction :

In devlopment evironment applications has to deployed in hetoroginious

eviroment with differrent OS, differrent patch leveles, and lots of other dependancies

Using docker we can use application regadless of dependancies.

So if I have Ubuntu Desktop and as developer I need to devlop or test applications in different environments

like Redhat, CentOS or SuSe I can use their docker Containers as separate (or isolated enviromnets from each other) to develop of test my application

Enhance Devloper Productivity :

Server Consolidation and Management:

Application Isolation

Rapid Deployments

Build Management

--------------------------------------

Containers Vs VMs

We all know VMs, It a emulation of computer system type.

A Virtualization sofware like VMware or VirtualBox allows

us to run one operating system withing another using same

physical hardware communicating with hardware using hipervisior.

What is Container?

A container is exactly what you might expect it to be based

on the general definition of the word. It is an entirely isolated set of

pacages, libraries and/or applications that are completely independent from

its surroundings.

Docker Architecture

Docker is client-server application where both the daemon

and client can be run on same system or you can connect

a Docker client with remote Docker Daemon.

Docker clients and daemons communicte via sockets or throuhg

a RESTful API (Represntational State Transfer) - It is a stateless

transfer over HTTP of web page containing an XML file

that describe and includes the desired content

The main components of Docker are:

Daemon

client

Docker.io Registry / or docker HUB

-------------------------------------------------

Installation on CentOS

#yum update

#wget http://ftp.osuosl.org/pub/fodora-epel/6/i386/epel-release-6-5.noarch.rpm

#rpm -Uvh epl---.rpm

#yum update

#yum install docker-io

#service docker start

#chkconfig docker on

Docker commands First image

#docker version

#docker images

#docker search centos

#docker info

Download and install Ubuntu image

#docker pull ubuntu:latest

#docker inspect ubuntu

#docker run -i -t (our current terminal) ubuntu /bin/bash

#docker run -i -t ubuntu /bin/bash

---------------------------------

Working with Multiple images

#docker pull centos:latest

#docker pull training/sinatra (ruby app)

#docker run -i -t training/sinatra /bin/bash

Install Janon pluging for ruby on sinatra imaage

#gem install jason

#exit

------------------------------------

Packaging Customised Container :

Package modified sinatra image and send to hub

#docker commit -m-"Added Ruby JASON module using gem" -a-"MHN Trainig" <imageidofbaseimage> mhntrainig/sintraimage:v1.1

#docker images ==> you will see our newly built image

we can ship this image

Creating new image from scratch :

#mkdir /opt/mhn

#cd /opt/mhn

#touch Dokerfile

#vi Dockerfile

-----EOF----

#This is our custome Docker fie build for sharing

FROM ubuntu:latest

MAINTAINER MHNtranig <mohan@mhntrainig.com>

RUN apt-get update

RUN apt-get install -y ruby ruby-dev

--EOF-------------------

#docker build -t-"ubuntu:latest"

#docker run -i -t ubuntu:latest /bin/bash

============================================================

Running Container Commands with Docker :

###To check if any container is running

#docker ps

#docker run centos:latest /bin/echo 'Hello from your Docker Container'

####Run something as daemon process withing container

#docker run -d ubuntu:latest /bin/bash -c "while true; do echo "Tesing Docker"; sleep 1; done"

(-d means run as daemon in container)

#docker ps

###You will see docker has assigned new name to the container

###Check our daemon is running or not within container

#docker logs <newly-assigned container name>| wc -l

#docker stop <newly-assigned container name>

#docker ps

-----------------------------

Exposing containers with port Redirects :

download tutum/apache-php image

#docker run -t -i tutum/apache-php /bin/bash

#service apache start

#curl http://localhost

#exit

### Now lets start the container as daemon and redirect apache port 80 to port 8080 on Base system

#docker run -d -p 8080:80 tutum/apache-php ==> will get large id

#docker ps

#docker logs

#curl http://localhost:8080

=====================================================

Docker Builds and Deployments:

Docker is process based system

Container Snapshots -

###Everytime you run image it gets new image id while running

###pull new cntos image, run image, run yum update, copy image id and exit

###now we can build new image from modified image

# docker commit <copied imageid> <new docker image name>

# docker images ---> you will see new image created

###now run the new image, install apache, install lynx, copy image id

#docker commit <copied imageid> <new docker image name>

------------------------------------------------

Attach to a Running Container:

Starting Container in backgroupd :

#docker run -t -i -d centos6:withapache /bin/bash

Check IP address of running container

#docker inspect silly\_archimedes (##silly\_archimedes random name assigned to running container)

if you start to access appache using container IP appache is actuall not started

#curl http://172....

Error -could not found page

To start appache

#docker attach silly\_archimedes

#service apache start

#curl http://172....

--------------------------------------------------------

Remove Images ::

#docker image

#docker rmi <imgeid>

erro: faild to remove

You cant remove image directly as there are dependent caitainers

and need to remove dependancy first. since there can be multiple

containers we cant remove dependent conatainers one by one

Solution is

Need to know docer Directory structure :

#ls /var/lib/docker

#cat /var/lib/docker/repositories-devicemapper

file is not readable, To read repositories-devicemapper need to use python + jason plugin

#cat /var/lib/docker/repositories-devicemapper | python -mjson.tool

#docker images

number of images should match with repository

check first 12 digit initail image id

#ls /var/lib/docker/containers

cd /var/lib/docker/containers/<container\_id>

You can see log, config.jason files

To delete image and all its dependancies

#rm -rf /var/lib/docker/containers/<container\_id>

#docker images

--------------------------------------------------

Services That Run on Startup

YOu can call service startup through .bash\_rc file adding

# vi .bash\_rc

/sbin/service httpd start

commit the new immange layer

# docker commit apache:running

Start apache as daemon process on base m/c

#docker run -i -t -d apache:running /bin/bash

Other way of doing by creating a script and call ing script throuh .bash\_rc

<

#!/bin/bash

rm -f /var/run/httpd/\*

/usr/sbin/apachectl -D FOREGROUND

/>

This script will help to run irrespective of having access to /sbin/service command or works on all platforms

----------------------------------------------------------

Building image from scratch:

#mkdir build

#cd build

#vi Dockerfile

