Syllabus is for container Technology = Docker community based on DCA (Docker Certified Associate) In DCA exam questions are coming from Kubernetes also

EX-180 = Red Hat Certified Specialist in Containers and Kubernetes exam. EX-280 = Red Hat Certified Specialist in OpenShift Administration exam. CKA is 32K but there will be discount offers for 30% or 60% offers during good Friday etc. CKA exam pass marks is 66 out of 100 marks. CKAD and CKS are extensions of Kubernetes. K8S is a GOOGLE initiative. Linux Foundation offers CKA, CKAD, CKS

Openshift is redhat product and it licenses open source. Openshift is an enterprise orchestrator based on K8S. Kubernetes is Opensource.

Doubts session OR Q&A = Tue/Wed = 12pm-1pm Afternoon. Ram sir cheatsheet and official course content also to be shared

E-learning portal = official curriculum and cheat sheet = 3 years time access

Cloud PAAS models based on kubernetes = EKS, GKE, AKS

DAY ONE = Physical servers and virtualization. Hardware based hypervisor is a software and using this we can run multiple guest OS on a single hardware

What is type 1 and type 2 virtualization. Physical server without OS type virtualization is type one for example vmware ESXi. VMWARE workstation is desktop level virtualization is called type 2 that requires operating system to run virtualization.

Type 2 hypervisors is that Type 1 runs on bare metal and Type 2 runs on top of an operating system. Type 1 is for server and type 2 is for students in general.

Dual boot we cannot use both operating systems in parallel. Its single hard disk with two OS in single partition but cant run both in parallel. In virtualization we can use multiple OS in parallel in a single hardware

OS level resources were getting wasted, delay in provisioning were some of the key challenges = Due to conflict in libraries and dependencies its not possible to install multiple applications inside same OS and even handle the upgrades etc.

Each software has its own library or shares common libraries with OS. We want to run multiple environments in a single OS but with library isolation to avoid conflict in libraries

Why a container is required? To run multiple applications without library conflict. What is kernel? Default network of containers? What is drawback of container?

So from hardware virtualization we came to OS virtualization or container based virtualization

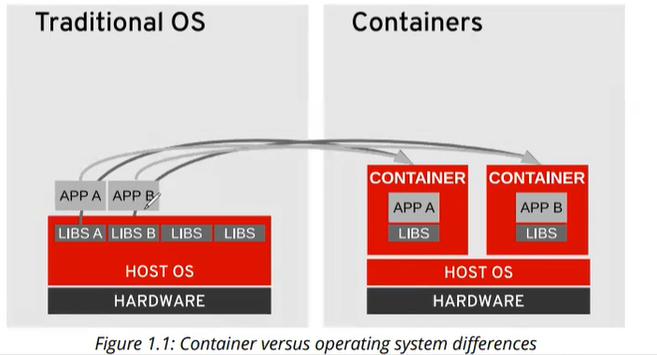
For example paytm has a lot of features = architecture models used in paytm??

One service can be run on one container. This is called micro-service architecture

Kernel is same for both containers and host since they use same kernel libraries. Only application library is isolated from container host which was the problem statement for which containerization was developed. After kernel patching both OS and container used new updated kernel features.

Each container also has a minor footprint of operating system. A linux host can run only linux containers. A windows host can only have windows containers. Each container has its own virtual NIC. 172.17.0.0/16 is default vnet which provides ip to each container.

Rocket, DrawBridge, Docker, podman are some examples of container runtime engines used in OS virtualization to optimize provisioning for delivery products at scale. Without OS we cannot run container technology. Docker has an enterprise edition EE and a community edition CE



Quick deployment, multi environment deployment and re-usability are some advantages of using containers. One container can have one application

Container is created using an image. Runtime of the image loaded in memory is container, if not its called as an image. A running image is called as a container

Container migration to a different platform in cloud. Do you migrate images or containers? You can only migrate the images and the applications running on it.

Vmware images could run only in vmware but a container image can run anywhere with only limitation that windows containers can run only on windows and linux images can run only on linux OS. OS should be the same and the release doesn’t matter.

Application can consume entire memory and disk allocated to the container unlike OS which had its own memory and disk utilization

Live containers migration refers to the process of moving application between different physical machines or clouds without disconnecting the client.

Before containerization the application library was installed inside OS and after containerization the application library is installed inside the container. Library is isolated. Software is distributed easily and deployed pro-efficiently because images are portable.

A monitoring tool running inside container can monitor only container processes within namespace. This is how Prometheus works.

**CONTAINER ARCHITECTURE AND KEY TERMINOLOGIES**

What is OCI? Open Container Initiative is a global containerization standard for any image that can be containerized.

An **IMAGE** is a lightweight contains your application and app dependencies. A running image or an application during its runtime is called a container. A stopped app is called an image. An image loaded in memory is called container and when its offline we call it as an image.

A container works on thin provisioning concept if there is no orchestration tool and if there is no memory/CPU reservation during creation of container then it will use memory of host as per application inside container usage.

**CONTAINER HOST** is the host running the container. If the container host is running docker then its called as docker host. Its also called as DOCKER HOST if using DOCKER or PODMAN host if using PODMAN. It can also be called as a container engine.

**CONTAINER IMAGE** = always go for community based images instead of custom images. Community based images are ready to use images

**IMAGES REGISTRY SERVER** = Community based/internet based registry/external registry. REDHAT uses redhat.access.redhat and quay.io. GOOGLE uses grc.io. DOCKER uses docker.io and docker hub. Enterprise product registry needs a license and OPEN source registry is free. User based registry also called as local registry or private registry keeps a copy of the images hosed publically because everytime the internet traffic going to public registry can be saved and can be downloaded locally from local intranet or local vnet.

**CONTAINER** = It is running mode of image. A container creates an isolated environment for any application. It creates runtime dependency for any application

Using one image we can deploy multiple containers

**NAMESPACES** = Each container during running stage it has a separate NIC, separate FS, separate storage, separate process. To isolate application we need these virtual resources and using these resources app is isolated. This is called container namespace. IP and STORAGE namespaces can be shared between containers

**CGROUPS** = It restricts container cpu/memory uses. CGROUP stands for container group to put an uppercap limit on virtual resources used by containers

**CONTAINER RUNTIME** =

Images uses storage driver called OVERLAY2 which is the FS utilized by docker. Aquasec/trivy is used to detect for malicious images prior deployment

What is rollout container? V1 of app and v2 is out. Is v2 compatible with previous v1. So then you will do container + v2. Image update. Update container: +2

WORDPRESS is a web development tool

LAB SETUP AND INSTALLATION

Container is OS virtualization and OS is required for containerization

Hypervisor can be installed on bare metal and on top of windows OS you can install desktop level virtualization.

Overlay2 is the storage driver used by docker

CONTAINER RUNTIME IS CONTAINERD

Command= docker pull ubuntu

Command= docker images - - - >shows ubuntu image as 77MB

Images are of two types = community based and custom images. In community image there are two types = service image and raw image

Nginx, mysql are service images. Ubuntu is a raw image.

Service images are ready to use. Raw images are minimum OS footprint

Service images have library files packed to deploy the service and its ready to use

Without raw image you cannot build a service image

IMAGE ID is same for a single image downloaded into multiple K8s clusters

IF you pull a raw image and the output says that it already exists this means another service image is already using this raw image

Service images are also called as daemon images

Command = docker history IMAGE ID

/var/lib/docker/containers is the path

Docker run versus docker exec?

Docker run versus docker attach

Docker run-it versus docker run -d

Docker attach is only for raw images

Docker run -e - - - > e is used to pass variables because image cannot run without variables

Image runtime exceptions arent documented and you need to pass the variables and for any image you need to know the exceptions from internet. E is for extra variable. You can pass both mandatory and optional extra variables

These logs can be configured to be persistent using elastic search which is a metric based tool

If you run interactive on a service container then bash will run. Use -d for daemon containers usages.

All images are pulled from docker community registry using hub.docker.com

DOCKER IMAGE BUILD PROCESS

Two ways = manual method and automatic method using dockerfile

To manually create the docker image first you need a raw image. Then you need to login to that image by launching it in the container aka deploy the image inside the container. Third step is to install software library inside the container. Now since you have installed software inside the container and the bottomline is to build an image in step 4 we need to commit container changes. Commit will convert the upper directory changes into a new image and using this image you can deploy a new container. This is how in a new container you will get the software installed in your container.

To save permanently the changes made in upper directory then before deleting the container commit the container so that the container changes can be persistent else by default with deletion of container its upper directory will be lost

When we commit the container then it makes a new image with software changes

Now we can launch a new container using a new image

When you commit a container it can be in running state or in stopped state

To terminate a container use docker rm containername

To delete an image use command = docker rmi imagenmae or imageID

To commit use command = docker commit containername and imagename

Now after you commit you cannot delete the image because of child dependency. Raw image is the parent and the committed image is the child. Child image still has the parent image layer and the OS footprint is still referenced with the parent image mounted in its image at container runtime

The newly committed image total size is both raw image and committed image

What is docker system df command do?

Run the command docker history imagename = it will show the parent ID size and the commiited image file.

So whenever you need to share this committed image then you need both raw image and the new committed image

Using docker save command you can save the image child and parent in a .tar image and you can use docker load command to load the image. Now if you run docker images command you can see only one image merged both parent and child

To transfer image without docker registry you can use docker save and docker load commands.

Docker save apache2 > apache.tar

Docker load -i apache.tar

Docker images

Image ID is also called as commit ID

Docker rmi apache2 and then run docker images. Now it will show only ubuntu image.

Service image use docker run -d and raw image use docker run -it

One single parent image can have multiple child images and multiple child images can have a same parent

/var/lib/docker/overlay2

What is a dangling image?

Ls /imageid/diff

Cat lower

Each docker image has a linked file called l (el alphabet)

Commit doesn’t capture the external volume changes. Commit contains the changes only in the upper directory

DOCKER VOLUME

Container OS disk is its upper directory that is the writing layer but its not persistent because once container is deleted its upper layer directory is also deleted

To prevent data loss of container use persistent volume and persistent volume claim

Lsblk shows raw disk and you cannot write data on a raw disk unless it is partitioned with a file system or formatted with a file system and a mount

After you partition the disk or without partition if you directly format the raw disk using file system then you need to mount the disk

Now pull the image and use -v for volume for persistent mount

What is docker volume ls?

Docker native volumes is created by using command = docker volume create vol1 are created in /var/lib/docker/volumes - - > this is docker native place for volumes

Native volumes are thin provisioned

What is host path?

https://docs.docker.com/storage/storagedriver/overlayfs-driver/

EXPLORE CONTAINER FILE SYSTEM

BUILD A CONTAINER IMAGE

Where are images stored and where is a library stored?

Stop and remove all docker containers = docker rm -f $(docker ps -a -q)

If you only want to do the running ones, remove -a. ps references to process status

To remove all docker images = docker rmi $(docker images -a -q). a is all and q is quiet

Image is read only file system and cannot be modified. FYI JBOSS is middleware

Every container has its own writing layer called writing namespace

Container associated writing namespace of container is removed when you remove the container

If you do a touch a file inside /diff of the container image then you can see it in the container also but if you a file inside container you cannot see it in the image because it will use another writing layer called -init in /var/lib/docker/overlay2

You cannot delete an image when the container is runtime

Upper and lower directory = lower is image read only FS and upper layer is writable namespace of the container

Change the working directory from /var/lib/docker/overlay2 to another mount and this requires downtime

Overlay2 storage/container file system works on COW concept – copy on write

Overlay2 drawback is COW, what problems can it create = it creates slowless, how to get rid of COW2 is by using storage containers

How to maintain large files inside image to get rid of COW disadvantage?

Once lower directory has a new file then it copies to higher directory and then writes on top of it.

Every container has its own writing layer. Container1 created from image1 has its own writing layer and container2 created from image1 has its own writing area

All containers are continuously reading the image file system and when you put file in lower layers then all containers can see it. COW comes when image files are modified at container level using interactive logon. ISOLATION fails when you copy the file from image FS to container writable FS.

Each container has its own writable file system namespace

Each container has a lower directory that is the image read only file system

If we know the container upper layer directory then we can check and remove the file that is locked out if we aren’t able to login to the container using interactive login command to use killsignal.

First use docker container inspect container | grep -I diff to check the layers. Next use ls /absolutepath to the diff folder. Use rm -f to remove the file or copy the file to another location and restart the container

How is docker volume different than docker commit? The main reason is portability; volumes are not supposed to be part of the image, and are stored outside the image.

Docker commit is used for use case scenario for data type of software data and volume is used for user data.

Commit is not used frequently else image size will grow. Even if image is lost data should not be lost and this is user data that should be stored over volume.

How to send a single file used in container to developer? In this use case scenario we can send the file from the upper directory and once committed we can provide from lower directory also. To copy the file from container to another destination use docker cp command = docker cp test:/etc/nginx/nginx.conf /tmp/ (Syntax is docker cp name of the container:source path destination path). Note that /tmp data will be gone with reboot

Same docker cp can be used to copy file from host to docker container also. Command = docker cp run.sh test:/ Check it using command = docker exec -it test bash and ls

You can also choose file to be kept on a volume and then copy the file to or from volume. Use volume for copying large files in Gb.

If you want to copy data between containers then you need to share the volumes. Because scp and rsync command wont work between containers.

DOCKER NETWORKING / CONTAINER NETWORKING

CONFIGURATION EXPECTATIONS = By default 3 configuration settings are out of the box = Each container should have unique IP. Containers should communicate with each other. Container should be able to talk to other hosts. Additional configuration to be done are = External users should be able to reach the container. Integrate LB across containers to distribute inbound round robin traffic. DNS for container for lookups approach because container gets a new IP after every restart.

NOTE that container network technology by default works on bridge network concept. Bridge is a software defined network which works at L2 switch for container aka virtual switch and this is required to isolate the network and achieve docker network isolation from host network. Docker network is running on bridge network and this is the reason we can deploy multiple containers in a host machine. Without bridge network cannot be isolated. Bridge plugin should be installed. Docker system by default already has created one bridge network during installation with 172.17.0.0/16 subnet range

Docker container inspect test | grep -i ipaddress //shows container ip address allocation automatically

Now deploy new container using raw image = docker run -it –name=con2 centos:7

Docker bridge commands =

Docker network ls =shows 3 networks host, bridge and none

Docker network inspect bridge | grep -i subnet //shows logical subnet

Docker network inspect bridge | grep -i name //shows containers associated with this subnet

Run apt update command first and then apt install net-tools -y. Run the command ifconfig

Run the command route -n //every bridge network has an external port to join it to host network. Gateway is the entry/exit point of the container.

By default one container in one host cannot ping another container in another host due to L2 bridged network. InK8s we can do it by default because we use L3 switch. By default the bridge gateway IP is 172.17.0.1 which his a virtual switch that has a bridge to the host machine. The docker containers will get IP’s assigned from 172.17.0.2 onwards.

Every container has its resolv.conf file.

Now lets create a custom bridge network. In a large enterprise organization one network is divided into subnetworks for security to allow department users to ping each other and not allow outside department to access each others network and have team specific network isolated within the team. This can be achieved using network switch and VLANs

Create multiple bridge networks

Docker network create –subnet 192.168.11.0/24 –driver bridge frontend //Note that don’t use an existing network already in use.

Docker network ls

Ifconfig

Docker network create –subnet 192.168.10.0/24 –driver bridge backend

Docker network ls

Docker network inspect frontend | grep -i subnet

Docker network inspect backend | grep -i subnet

//run a FE container in a 2 tier architecture= docker run -d –name=web –network frontend nginx

//deploy a BE database in a 2 tier architecture = docker run -d –name=db –network backend nginx

Docker exec -it web bash

Ifconfig

Apt update -y; apt install net-tools -y

Ifconfig

To check gateway use the command = route -n

Exit //come out of container

//run APP container in 3 tier architecture = docker run -d –name=app –network frontend nginx

Docker network connect backend app //to bridge both frontend and backend networks

Docker network disconnect backend app

If you don’t provide –subnet option while creating a docker container then it will goto default bridge. For isolated container run docker run -d –nameweb25 –network none nginx. IF you want to change the default none then use docker network disconnect and then use docker network connect else a single container cannot be connected to multiple networks

Docker custom network bridges are one time configuration and after reboot of host also they will remain as persistent configuration

Bridge networks are hot swappable and can be allocated during docker create and/or docker run also. During runtime it can be swapped, attached or detached

docker images //shows all docker images pulled on the docker host machine inventory where all images are stored. Image name is called as repository name.

docker pull nginx //pull community images from public registry servers like dockerhub because we installed docker CE community edition and this has a default registry server linked to dockerhub if we don’t put docker registry URL. This command output shows we pull all the FS layers of the image from the repository. 4 status messages –pulling FS layer, downloading, extracting, pull complete

docker pull docker.io/nginx //pull from specific docker registry

docker pull docker.io/nginx:1.16 //pull specific version

Every image has a commit ID when built and committed into public repository and when we pull image we get a unique identifier called Image ID

ls /var/lib/docker

docker info

Docker info | grep -i root //shows docker root directory location as /var/lib/docker

docker search docker.io/mysql

Every image that goes through a change gets a new commit ID and you can see all those commit ID’s being pulled when you pull an image from dockerhub

Docker system df //shows total size in MB utilized

An offline image is what we call when we have pulled the image but haven’t created a container from that image yet

While a container repository is a collection of related container images used to manage, pull and push images, a container registry is a collection of repositories made to store container images. Container registries can store container images as well as API paths and access control rules.