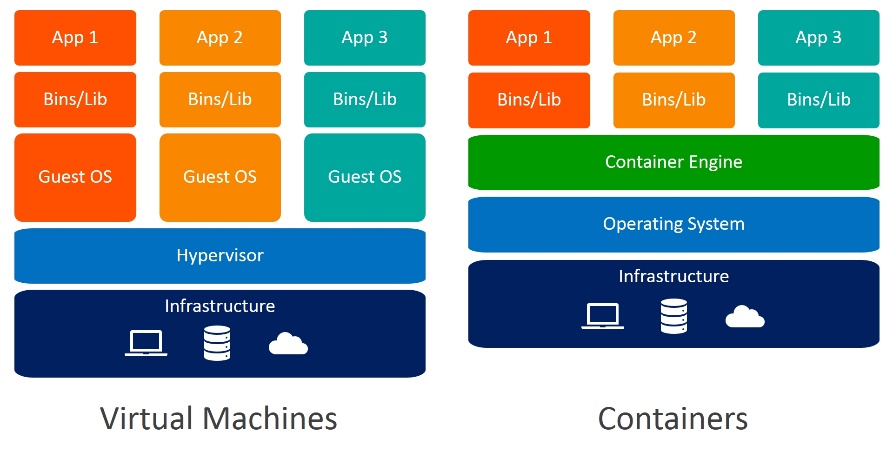
**Docker**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Virtualization** | **Containerization** |
| Isolation | Provides complete isolation from the host operating system and the other VMs. | Provides lightweight isolation from the host and other containers, but doesn’t provide as strong a security boundary as a VM. |
| Speed | Virtual Machines need to start the entire operating system, which includes the full boot process. This will also include the startup of the services, and it will take much longer. | The container starts immediately, since the operating system is already up and running, so the application will start up without any noticeable delay. |
| Security | Virtualization provides complete isolation for all the applications it hosts and hence is more secure. | Based on the configuration, Containers provide only process-level isolation. So security risks are more when compared to virtual machines. |
| Operating System | Runs a complete operating system including the kernel, thus requiring more system resources such as CPU, memory, and storage. | Runs the user-mode portion of an operating system, and can be tailored to contain just the needed services for your app using fewer system resources. |
| Application Deployment | Deploy individual VMs by using Hypervisor software. | Deploy individual application containers by using Docker or deploy multiple containers by using an orchestrator such as Kubernetes. |
| Application Load Time | Virtual Machines simulate the entire operating environment and hence take more time to load. | Containerized applications encapsulates application code together with the required libraries and dependencies needed to run it, they are generally smaller in size as compared to virtual machines.This reduces the loading time of containers. |
| Persistent storage | Use a Virtual Hard Disk (VHD) for local storage for a single VM or a Server Message Block (SMB) file share for storage shared by multiple servers. | Use local disks for local storage for a single node or SMB for storage shared by multiple nodes or servers. |
| Load balancing | Virtual machine load balancing is done by running VMs in other servers in a failover cluster. | An orchestrator can automatically start or stop containers on cluster nodes to manage changes in load and availability. |
| Networking | Uses virtual network adapters. | Uses an isolated view of a virtual network adapter. Thus, provides a little less virtualization. |



Docker File:

* create a file with name DockerFile 🡪touch dockerfile🡪vim dockerfile🡪i🡪FROM ubuntu-🡪RUN apt-get update-🡪to run some output in the command line add 🡪CMD [“Echo”, ”hello world from the first docker image”] 🡪esc->:wq
* cat dockerfile
* docker build (location of the dockerfile) OR if u are already on the location (docker build -t myImage1:1.0 .)
* docker images
* docker run (image id)

Docker Compose:

* Tool for defining and running multi container docker applications.
* We use yaml file to configure application services(docker-compose.yml)
* We can start all the services with a single command :docker compose up
* We can stop all the services with a single command :docker compose down
* We can scale up selected services.( docker-compose up -d --scale database=4) will create 4 instances in the container.

Docker Volume:

USE:

* Decoupling container from storage
* Share volume (Storage/data) among different containers
* Attach volume to container
* On deleting container volume does not delete

To create docker volume(docker volume create \_\_\_\_(any name))

To inspect (docker volume inspect \_\_\_(same name used to create))

To remove (docker volume rm (and the name))

Example with Jenkins

* Docker pull Jenkins/Jenkins
* docker run --name MyJenkins1 -v myvoll:/var/jenkins\_home -p 8080:8080 -p 50000:50000 jenkins/jenkins

(34fd53aa7d8844a7af26e66b95af7b35)—password

* Then open new terminal and paste the same command with few changes (docker run --name MyJenkins2 -v myvoll:/var/jenkins\_home -p 9090:8080 -p 60000:50000 jenkins/jenkins)🡪change the container name and the port numbers
* In this process the volume will remain the same even if the container is deleted or removed

BIND MOUNT:

* A file or directory on the host machine is mounted into a container🡪 actually use a physical location instead of the volume we use the same command (docker run --name MyJenkins3 -v C:\Users\preneethaj\Desktop\Preneetha\DVol2:/var/jenkins\_home -p 9191:8080 -p 40000:50000 jenkins/jenkins) change the port numbersand the container name(dc1974d0fbd247f1912ae30fa03a9700)🡪password

TO STOP the CONTAINERS

* docker stop MyJenkins1(the container name)

To see all the containers

* docker ps
* docker ps -a(the container will be available but they are stopped)

To Remove the containers

* docker rm MyJenkins1(container name)

**Kubernetes**

* minikube start --nodes 2 -p multinode --driver=docker
* minikube status -p multimode🡪 to check the status
* OR minikube status
* kubectl get nodes-🡪to get the list of nodes
* kubectl get pod🡪 to get the list of pods
* kubectl get services 🡪 to see the list of services
* kubectl create deployment nginx-depl --image=nginx 🡪 to create container of the image of the nginx(nginx-depl -🡪 name given)
* kubectl get deployment🡪 to see the deployed list
* kubectl get replicaset 🡪will automatically create a replica set to see the replica set
* Pod is actually a abstraction of the container

To edit an image

* Kubectl edit deployment nginx-depl 🡪 will get auto generated configuration file of the deployment with default values
* kubectl apply -f config-file.yaml-🡪to do editing in the config file
* touch filename.yaml
* Insert and insert the basic configuration

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-app

labels:

app: my-app

spec:

replicas: 2

selector:

matchLabels:

app: my-app

template:

metadata:

labels:

app: my-app

spec:

containers:

- name: my-app

image: my-image

env:

- name: SOME\_ENV

value: $SOME\_ENV

ports:

- containerPort: 8080

* After this give the apply command

OR

apiVersion: apps/v1  
kind: Deployment  
metadata:  
name: nginx-deployment  
spec:  
selector:  
matchLabels:  
app: nginx  
replicas: 2 # tells deployment to run 2 pods matching the template  
template:  
metadata:  
labels:  
app: nginx  
spec:  
containers:  
- name: nginx  
image: nginx:1.14.2  
ports:  
- containerPort: 80

* kubectl apply -f <https://k8s.io/examples/application/deployment.yaml>