# Experiment No: 3

**Title**: Implementation of Xen Server and Docker

**Aim:** To study and implement Xen Server and Docker

## Theory:

### Xen Server:

Citirix XenServer is a hypervisor platform that enables the creation and management of virtualized server infrastructure. It is developed by Citrix Systems and is built over the Xen virtual machine hypervisor. XenServer provides server virtualization and monitoring services. It is available in a 64-bit hypervisor platform and can be executed on the entire x86 series of processors.

Cirtix XenServer is among the virtualization solutions provided by Citrix Systems, which consolidates a physical server's computing power into multiple virtual machines, all emulating as a standard server. Citrix XenServer is built to provide the operational requirements of a standard server and supports most server operating systems, such as Linux and Windows Server, on guest server machines.

Through its virtual machine monitoring component, Citrix XenServer manages the allocation and distribution of physical server computing resources among virtual machines and administers their performance and use.

### Docker:

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

Docker streamlines the development lifecycle by allowing developers to work in standardized environments using local containers which provide your applications and services. Containers are great for continuous integration and continuous delivery (CI/CD) workflows.

Consider the following example scenario:

- Your developers write code locally and share their work with their colleagues using Docker containers.
- They use Docker to push their applications into a test environment and execute automated and manual tests.
- When developers find bugs, they can fix them in the development environment and redeploy them to the test environment for testing and validation.
- When testing is complete, getting the fix to the customer is as simple as pushing the updated image to the production environment.

Responsive deployment and scaling

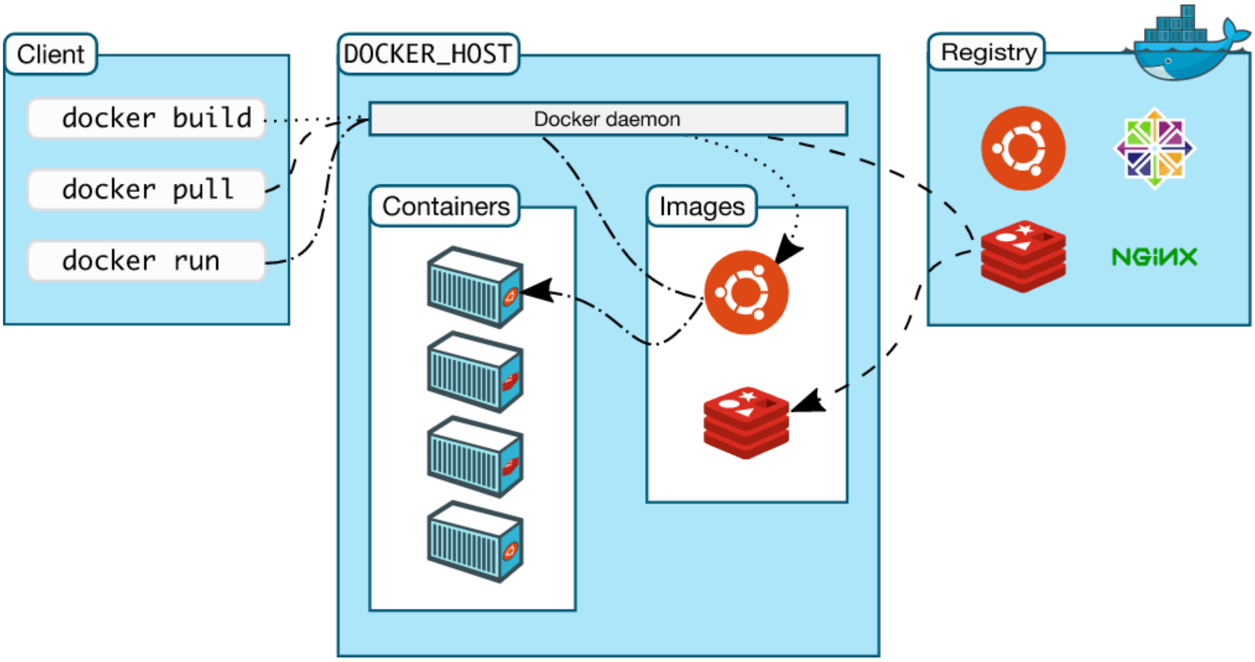
Docker's container-based platform allows for highly portable workloads. Docker containers can run on a developer's local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.

Docker's portability and lightweight nature also make it easy to dynamically manage workloads, scaling up or tearing down applications and services as business needs dictate, in near real time.

Running more workloads on the same hardware

Docker is lightweight and fast. It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your server capacity to achieve your business goals. Docker is perfect for high density environments and for small and medium deployments where you need to do more with fewer resources.

**Architecture of Docker:**




**Practical:**

Step 1: Installation of Docker

```
EC2
[root@ip-172-31-24-83 ~]# yum install docker -y
```

Step 2: Start the Docker Engine.

```
EC2
Dependencies Resolved
[root@ip-172-31-24-83 ~]# docker pull centos:latest
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
[root@ip-172-31-24-83 ~]# systemctl start docker
[root@ip-172-31-24-83 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-10-18 03:46:31 UTC; 5s ago
     Docs: https://docs.docker.com
  Process: 1617 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 1616 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 1620 (dockerd)
    Tasks: 7
   Memory: 20.9M
   CGroup: /system.slice/docker.service
           └─1620 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Oct 18 03:46:30 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:30.765810433Z" level=info msg="ClientConn switching balancer to \"pick_first\...dule=grpc
Oct 18 03:46:30 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:30.801859136Z" level=warning msg="Your kernel does not support cgroup blkio weight"
Oct 18 03:46:30 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:30.802299504Z" level=warning msg="Your kernel does not support cgroup blkio w...t_device"
Oct 18 03:46:30 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:30.802759223Z" level=info msg="Loading containers: start."
Oct 18 03:46:31 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:31.057750847Z" level=info msg="Default bridge (docker0) is assigned with an I... address"
Oct 18 03:46:31 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:31.115174177Z" level=info msg="Loading containers: done."
Oct 18 03:46:31 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:31.130435862Z" level=info msg="Docker daemon" commit=a89b842 graphdriver(s)=o...=20.10.17
Oct 18 03:46:31 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:31.130939931Z" level=info msg="Daemon has completed initialization"
Oct 18 03:46:31 ip-172-31-24-83.ec2.internal systemd[1]: Started Docker Application Container Engine.
Oct 18 03:46:31 ip-172-31-24-83.ec2.internal dockerd[1620]: time="2022-10-18T03:46:31.158264532Z" level=info msg="API listen on /run/docker.sock"
```

Step 3: Downloading image of OS.

```
EC2
[root@ip-172-31-24-83 ~]# docker pull centos:latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
[root@ip-172-31-24-83 ~]#
```

Step 4: Creating the docker container using command.

```
$docker run -it --name gcek1 centos:latest
```

```
EC2
[root@ip-172-31-24-83 ~]# docker pull centos:latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
[root@ip-172-31-24-83 ~]# docker run -it --name gcek1 centos:latest
[root@ba313e334023 /]# ^C
[root@ba313e334023 /]#
```

Step 5: To see all images and Conatiners running and stop.

1.)To view all Images downloaded

```
EC2
[root@ip-172-31-24-83 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
centos       latest    5d0da3dc9764   13 months ago   231MB
[root@ip-172-31-24-83 ~]#
```

2.) To view all Running and all containers.

```
$ docker ps
$ docker ps -a
```

```
EC2
[root@ip-172-31-24-83 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
centos       latest    5d0da3dc9764   13 months ago   231MB
[root@ip-172-31-24-83 ~]# docker ps -a
CONTAINER ID   IMAGE           COMMAND       CREATED         STATUS                          PORTS     NAMES
ba313e334023   centos:latest   "/bin/bash"   3 minutes ago   Exited (0) About a minute ago             gcek1
[root@ip-172-31-24-83 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-24-83 ~]#
```

Step 6: Images can be created by own in two methods by commit method and using Dockerfile.

1.) Create the php-Docker-app directory.

```
EC2
[root@ip-172-31-24-83 ~]# mkdir php-Docker-app
[root@ip-172-31-24-83 ~]# cd php-Docker-app/
[root@ip-172-31-24-83 php-Docker-app]#
```

2.) Create the index.php file

```
EC2
[root@ip-172-31-24-83 ~]# mkdir php-Docker-app
[root@ip-172-31-24-83 ~]# cd php-Docker-app/
[root@ip-172-31-24-83 php-Docker-app]# vim index.php
[root@ip-172-31-24-83 php-Docker-app]# cat index.php
<?php
    echo "Hello Gcekian";
?>
[root@ip-172-31-24-83 php-Docker-app]#
```

3.) Create the Dockerfile in current folder.

```
EC2
[root@ip-172-31-24-83 ~]# mkdir php-Docker-app
[root@ip-172-31-24-83 ~]# cd php-Docker-app/
[root@ip-172-31-24-83 php-Docker-app]# vim index.php
[root@ip-172-31-24-83 php-Docker-app]# cat index.php
<?php
    echo "Hello Gcekian";
?>
[root@ip-172-31-24-83 php-Docker-app]# vim Dockerfile
[root@ip-172-31-24-83 php-Docker-app]# cat Dockerfile
FROM php:7.0-apache
COPY . /var/www/php
[root@ip-172-31-24-83 php-Docker-app]#
```

4.) Create the image using Docker build.

```
$ docker build -t php-app-gcek .
```

```
[root@ip-172-31-24-83 php-Docker-app]# docker build -t php-app-gcek .
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM php:7.0-apache
7.0-apache: Pulling from library/php
177e7ef0df69: Extracting [==========================================>        ]  19.73MB/22.49MB
9bf89f2eda24: Download complete
350207dcf1b7: Download complete
a8a33d96b4e7: Download complete
c0421d5b63d6: Download complete
f76e300fbe72: Download complete
af9ff1b9ce5b: Download complete
d9f072d61771: Download complete
37007e292198: Download complete
8ba923990f24: Download complete
98af8902979a: Download complete
f1548c2cd376: Download complete
e1062fd0605a: Download complete
```

5.) See image created using docker images.

```
EC2
[root@ip-172-31-24-83 php-Docker-app]# docker images
REPOSITORY     TAG          IMAGE ID       CREATED          SIZE
php-app-gcek   latest       b8f6f724768f   39 seconds ago   368MB
centos         latest       5d0da3dc9764   13 months ago    231MB
php            7.0-apache   aa67a9c9814f   3 years ago      368MB
[root@ip-172-31-24-83 php-Docker-app]#
```

**Conclusion:**

Thus, I have successfully installed and configured Docker on Ubuntu operating system.