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

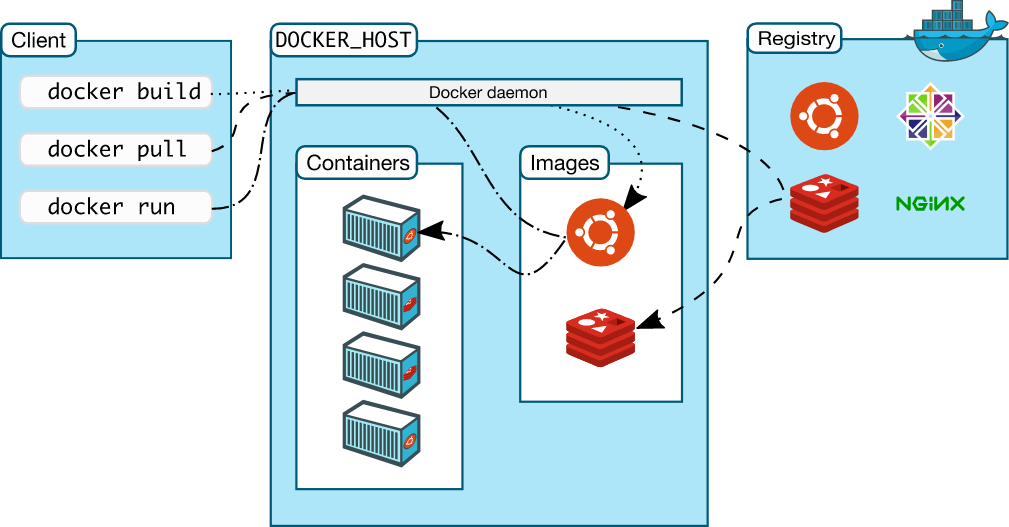
Docker’s container-based platform allows for highly portable workloads. Docker containers can run on a developer’s local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.

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

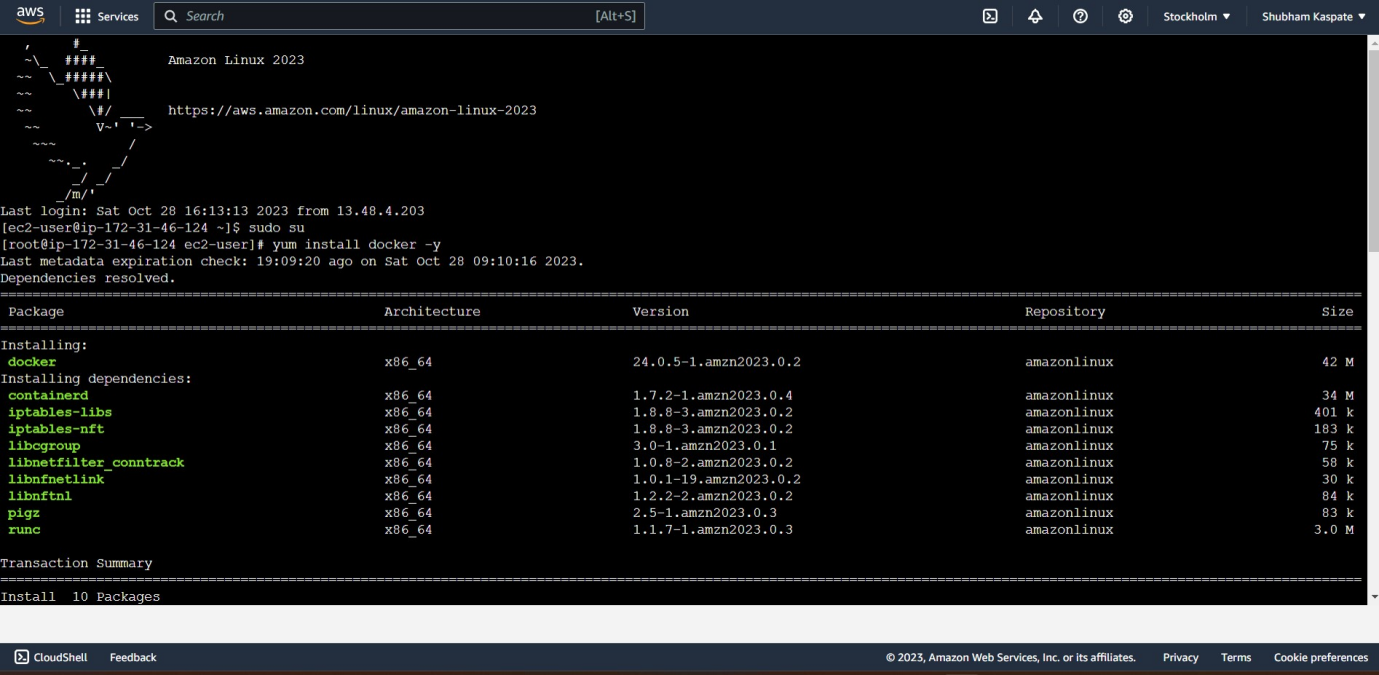
Docker is lightweight and fast. It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your server capacity to achieve your business goals. Docker is perfect for high density environments and for small and medium deployments where you need to do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

**Architecture of Docker:**

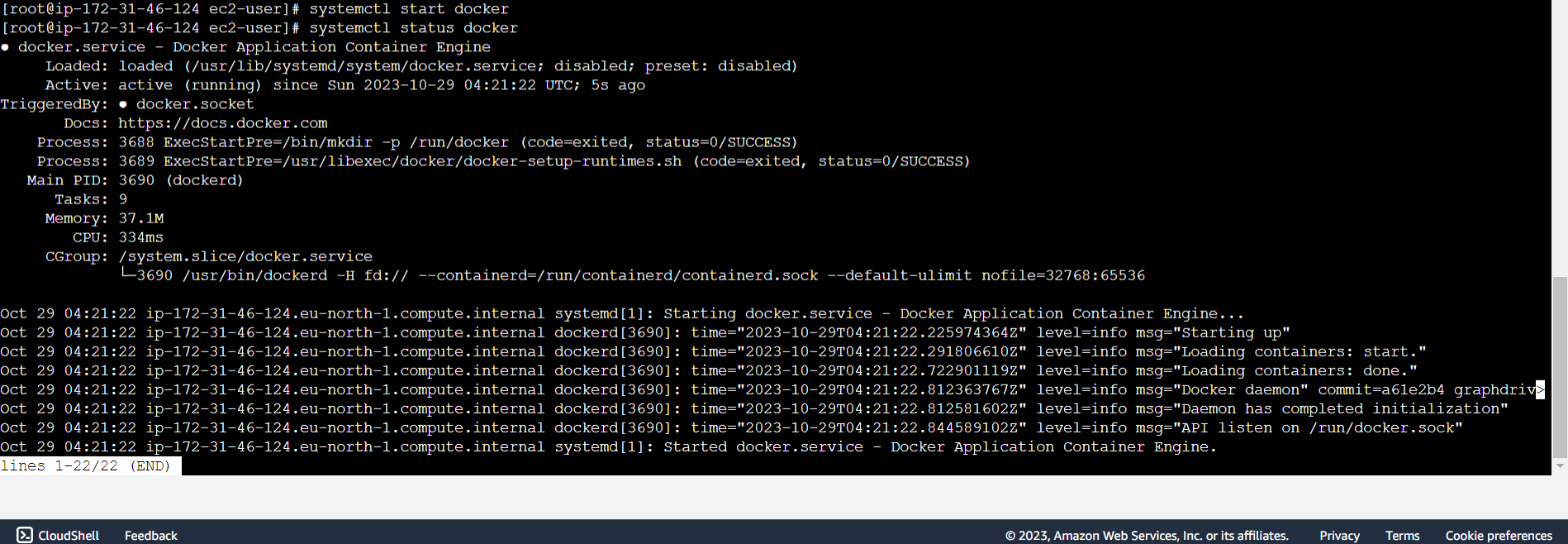


**Practical:**

Step 1: Installation of Docker



Step 2: Start the Docker Engine.

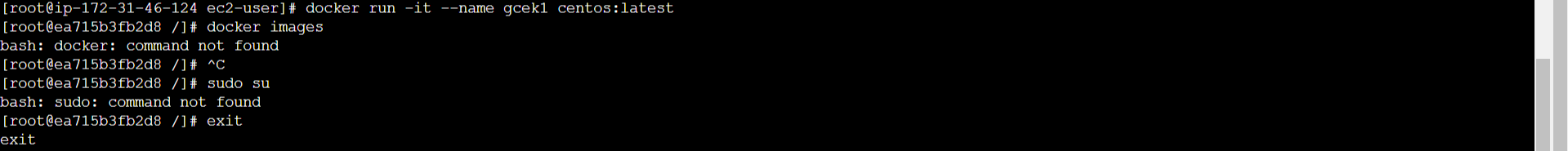


Step 3: Downloading image of OS.



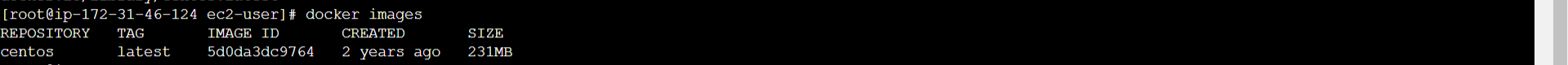
Step 4: Creating the docker container using command.

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



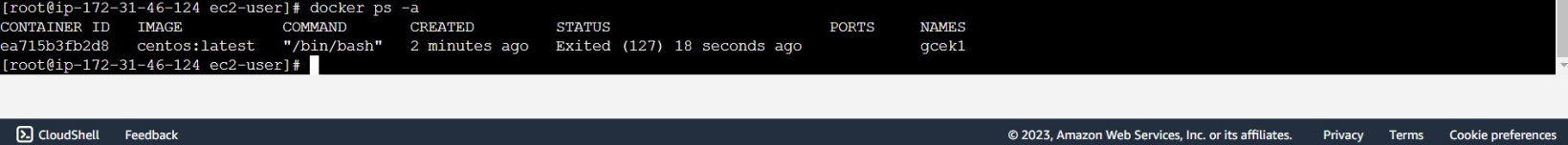
Step 5: To see all images and Conatiners running and stop.

1.)To view all Images downloaded



2.) To view all Running and all containers.

|  |
| --- |
| $ docker ps |
| $ docker ps -a |

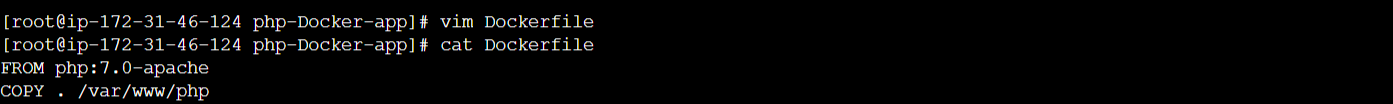


Step 6: Images can be created by own in two methods by commit method and using Dockerfile.

1. Create the php-Docker-app directory. 
2. Create the index.php file

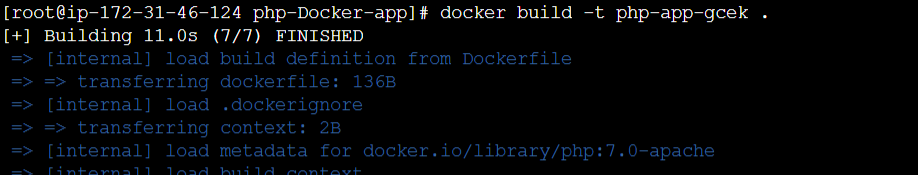


1. Create the Dockerfile in current folder.

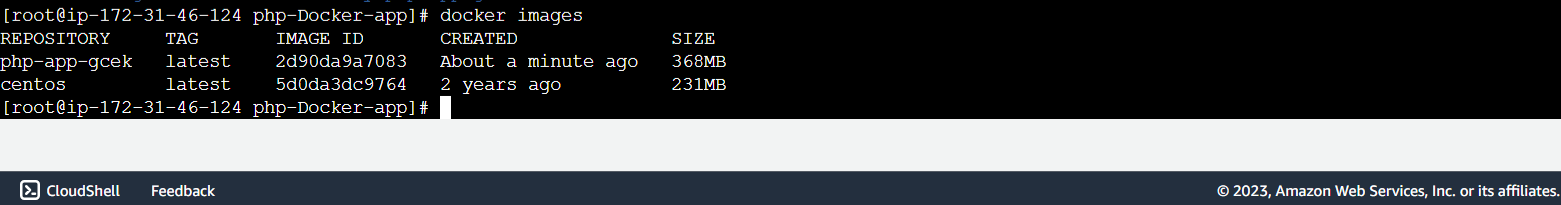


1. Create the image using Docker build.

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



1. See image created using docker images.



**Conclusion:**

Thus, I have successfully installed and configured Docker on Linux Operating system.