**DOCKER**

1. **Initial GCP Setup –** 
   1. Create one free trial account on GCP and Create a project in GCP.
   2. Now create a VM instance and SSH into that VM.
   3. Upload jar file in VM

**2. Setup Docker VM –**

2.1 Before installing docker, we need to add couple of dependencies.

sudo apt-get -y install apt-transport-https ca-certificates curl software-properties-common

2.2 Before installing Docker repo, we need to add a gpg key from docker.com for authentication reasons

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -

2.3 Now, we can finally add Docker repository

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb\_release -cs) stable"

* 1. Now, just to run a simple update and to install docker-ce

sudo apt-get update

sudo apt-get install docker-ce

* 1. To eliminate the need to write sudo at every command, add your current user to docker group

sudo usermod -aG docker $USER

* 1. Docker is successfully installed and now to install docker-compose follow the below commands

sudo curl -L "https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

2.7 Now, docker and docker-compose is installed on your VM and to check if they are successfully installed check the version with docker –v for docker version and docker-compose –v for docker-compose version.

**4. What is Docker?**

Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

## **Features of Docker**

* Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
* With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
* You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
* Since Docker containers are pretty lightweight, they are very easily scalable.

### **What are containers?**

The industry standard today is to use Virtual Machines (VMs) to run software applications. VMs run applications inside a guest Operating System, which runs on virtual hardware powered by the server’s host OS.

VMs are great at providing full process isolation for applications: there are very few ways a problem in the host operating system can affect the software running in the guest operating system, and vice-versa. But this isolation comes at great cost — the computational overhead spent virtualizing hardware for a guest OS to use is substantial.

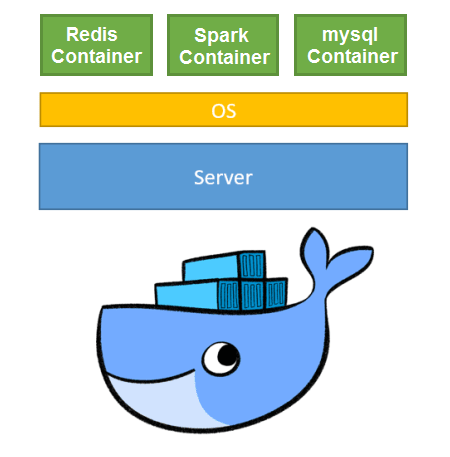
Containers take a different approach: by leveraging the low-level mechanics of the host operating system, containers provide most of the isolation of virtual machines at a fraction of the computing power.

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run. This decoupling allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer’s personal laptop. This gives developers the ability to create predictable environments that are isolated from rest of the applications and can be run anywhere.

**3. Creating container** **–**

3.1 Docker Compose : Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

Following is the structure that is implemented :



3.2 To create container upload docker-compose.yml file to your VM and to run the docker compose execute the following command

docker-compose up –d

3.3 Now all the containers mentioned in the docker compose files are up and running and to run the spark job on spark container open the terminal of spark container

docker exec –it <sparkContainer> bash

cd ~/../sparkJob

spark-submit –class test <jarFile>