Distributed and Parallel Computing Technologies



**Fakhra Imran - 2160002**

University of Bedfordshire

Luton, United Kingdom

Table of Contents

[1. Machine Learning-based Orchestration of Containers 4](#_Toc149936094)

[1.1 Introduction 4](#_Toc149936095)

[1.2 Dynamic Scaling 4](#_Toc149936096)

[1.3 Load Balancing 5](#_Toc149936097)

[1.4 Anomaly Detection 5](#_Toc149936098)

[1.5 Traffic Routing 5](#_Toc149936099)

[1.6 Resource management 5](#_Toc149936100)

[1.7 Auto Scaling 5](#_Toc149936101)

[1.8 Security 5](#_Toc149936102)

[1.9 Predictive Maintenance 5](#_Toc149936103)

[1.2 Current Focus – Machine Learning based Predictive/Dynamic Scaling of Kubernetes 5](#_Toc149936104)

[1.3 Road Map – Machine Learning based Predictive/Dynamic Scaling of Kubernetes 5](#_Toc149936105)

[2. Oracle Virtual Box - Installation 6](#_Toc149936106)

[2.1 Documenting the setup of a Linux Virtual Machine from scratch 6](#_Toc149936107)

[2.2 Download Ubuntu ISO Image 7](#_Toc149936108)

[2.3 Ubuntu – Virtual Box Configuration 8](#_Toc149936109)

[3. Ubuntu Master – Installation 10](#_Toc149936110)

[3.1 Start Installation - Ubuntu 10](#_Toc149936111)

[3.2 Language Selection 10](#_Toc149936112)

[3.3 Installation Type: Write changes to disk 10](#_Toc149936113)

[3.4 Installation Type: Normal 11](#_Toc149936114)

[3.5 Select Country 11](#_Toc149936115)

[3.5 Setup Computer Name and Username 11](#_Toc149936116)

[3.5 Completed Ubuntu Installation 12](#_Toc149936117)

[4. Ubuntu Worker – Installation 12](#_Toc149936118)

[4.1 Clone master machine 12](#_Toc149936119)

[5. Networking – Master and Worker Connectivity 12](#_Toc149936120)

[5.1 Bridge Adapter Settings 12](#_Toc149936121)

[5.2 Hostname & IP address 13](#_Toc149936122)

[5.3 Test Connectivity between Master and Worker Node 14](#_Toc149936123)

[6. Docker Installation (Master/Worker Node) 15](#_Toc149936124)

[6.1 Update List of Packages 15](#_Toc149936125)

[6.2 Pre-requisites Packages Installation 15](#_Toc149936126)

[6.3 Add GPG Key for repository 15](#_Toc149936127)

[6.4 Add Docker Repository 15](#_Toc149936128)

[6.5 Docker Installation 15](#_Toc149936129)

[6.6 Verify Docker Version 16](#_Toc149936130)

[6.7 Docker Status 16](#_Toc149936131)

[6.8 Containerd Installation/Restart/Enablement 16](#_Toc149936132)

[7. Kubernetes Installation 16](#_Toc149936133)

[7.1 Swap off Temporary (Master/Worker Node) 16](#_Toc149936134)

[7.2 Swap off Permanently (Master/Worker Node) 16](#_Toc149936135)

[7.3 Let IP Tables See Bridge Traffic 17](#_Toc149936136)

[7.4 Kubernetes Installation – Kubelet Kubeadm kubectl 18](#_Toc149936137)

[7.5 Kubernetes Calico Setup 18](#_Toc149936138)

[7.6 Kubernetes Initialization (Run only on Master Node) 18](#_Toc149936139)

[7.7 Join Master Node - (Run on Workers Node) 18](#_Toc149936140)

[7.8 Verify Worker Node is attached to Master 18](#_Toc149936141)

[8. Application – Creating a JAR file 19](#_Toc149936142)

[8.1 Introduction 19](#_Toc149936143)

[8.2 Add API 19](#_Toc149936144)

[8.3 Multiply API 19](#_Toc149936145)

[8.4 Divide API 19](#_Toc149936146)

[8.5 Minus API 19](#_Toc149936147)

[8.6 Print History and clear History APIs 19](#_Toc149936148)

[8.7 Static Initializer that load Historical data 20](#_Toc149936149)

[8.8 Add Request Count for Each Day 20](#_Toc149936150)

[8.8 Create a JAR file 21](#_Toc149936151)

[9. Docker – Create a Docker File 21](#_Toc149936152)

[9.1 Prepare a Docker file 21](#_Toc149936153)

[9.2 Build an Docker Image 22](#_Toc149936154)

[9.3 Push image to docker hub 22](#_Toc149936155)

[10. Kubernetes - Deployments 23](#_Toc149936156)

[10.1 Kubernetes - Service File 23](#_Toc149936157)

[10.2 Kubernetes - Apply Deployments 23](#_Toc149936158)

[10.2 Kubernetes - Get Deployments 23](#_Toc149936159)

[10.3 Kubernetes - Get Service 24](#_Toc149936160)

[10.4 Kubernetes - Get Pods 24](#_Toc149936161)

[10.5 Kubernetes - Scale Up 24](#_Toc149936162)

[10.6 Test the calculator Service 24](#_Toc149936163)

[11. Machine Learning based Predictive/Dynamic Scaling of kubernetes 24](#_Toc149936164)

[11.1 Difference between CPU metric based scaling (Static) and Predictive Scaling 24](#_Toc149936165)

[11.2 Pre-Trained Models 25](#_Toc149936166)

[11.3 Actual Plan 25](#_Toc149936167)

[11.4 Machine Learning based Python code Using Facebook Prophet 25](#_Toc149936168)

[11.5 Predictive/Dynamic Scaling based on historical data 26](#_Toc149936169)

[12. Challenges Faced 26](#_Toc149936170)

[13. Way Forward Strategy 26](#_Toc149936171)

[14. References 27](#_Toc149936172)

# Machine Learning-based Orchestration of Containers

## Introduction

Containerization is a lightweight application virtualization technology that provides a few focal points, including high environmental consistency, portability over diverse operating systems, and resource isolation. Many standard cloud service providers have broadly grasped container technologies as portion of their distributed system infrastructures to automate application management. Container orchestration has developed as a crucial research problem to handle the automation of deployment, maintenance, autoscaling, and networking of containerized applications. However, the dynamic and diverse nature of cloud workloads and environments significantly complicates orchestration mechanisms.

To address this complexity, container coordination systems utilize machine learning algorithms to demonstrate behavior and predict multi-dimensional performance metrics. These experiences can enhance resource provisioning decisions to adjust to changing workloads in complex environments. This article provides a comprehensive writing review of existing machine learning-based holder coordination approaches. The research is categorized into detailed taxonomies based on common features, and the advancement of machine learning-based container orchestration technologies from 2016 to 2021 is investigated with a focus on objectives and metrics. [1]

In addition to machine learning's role in container orchestration, there's a growing interest in utilizing machine learning for traffic management in orchestration tools like Kubernetes. To optimize different aspects of application traffic handling in Kubernetes, it is significant to consider a few factors and strategies. Implementing machine learning for traffic management includes collecting and analyzing significant data, training models, and integrating them into Kubernetes infrastructure through tools like custom controllers, Kubernetes-native services, or external solutions. Nonstop monitoring and updates of machine learning models are fundamental to adapt to changing traffic patterns and requirements. This intersection of container orchestration and machine learning in traffic management represents an exciting area of inquire about and advancement with potential future directions and open inquire about challenges.

### 1.2 Dynamic Scaling

Use machine learning algorithms to predict traffic patterns and adjust the number of replicas (Pods) for your kubernetes application. This help efficiently handling dynamic levels of traffic

### 1.3 Load Balancing

Implement intelligent load balancing to distribute the traffic on their current load and response times

### 1.4 Anomaly Detection

Employ machine learning models to detect anomalies in incoming traffic, such as sudden spikes and unusual traffic

### 1.5 Traffic Routing

Define dynamic and best path traffic routing strategies

### 1.6 Resource management

Predict resource requirement based on historical data

### 1.7 Auto Scaling

Auto scale factors that consider response time, error rate, and request rate

### 1.8 Security

Machine is helpful to detect DDoS attack, identity malicious traffic, and enhance security by learning patterns

### 1.9 Predictive Maintenance

Use machine learning to predict when certain components or nodes might fail due to increased traffic or other factors. This allow proactive maintenance and avoids downtime.

## 1.2 Current Focus – Machine Learning based Predictive/Dynamic Scaling of Kubernetes

When it comes to traffic management in orchestration tools like Kubernetes, the machine learning can be used to optimize various aspects of application’s traffic handling. In this document focus will be on dynamic scaling based on the historical data using machine learning based pertained models. For this purpose I will be saving request count on daily basis to build historical data. Once that data in collected we will analyse the data and predict traffic request counts. This prediction will be performed using machine learning pre trained model using tools like FbProphet which will help to predict total traffic request count on daily basis and on the base of that we will be scaling up or scaling down the PODs proactively rather than following the static auto scaling mechanism of kubernetes which work on the based on CPU consumption and RAM utilization.

Below are the major factors and strategies that required due considerations.

## 1.3 Road Map – Machine Learning based Predictive/Dynamic Scaling of Kubernetes

Please find below the road map to read machine learning based dynamic scaling.

# Oracle Virtual Box - Installation

## 2.1 Documenting the setup of a Linux Virtual Machine from scratch

Download and install VirtualBox version 7.0.10 on your host machine. This can be done by visiting the VirtualBox website (https://www.virtualbox.org/). Clicking on "Download Virtual Box 7.0" will take you to the next page, and then click on "VirtualBox 7.0.10 Software Developer Kit (SDK) - All Platforms" to initiate the download process.

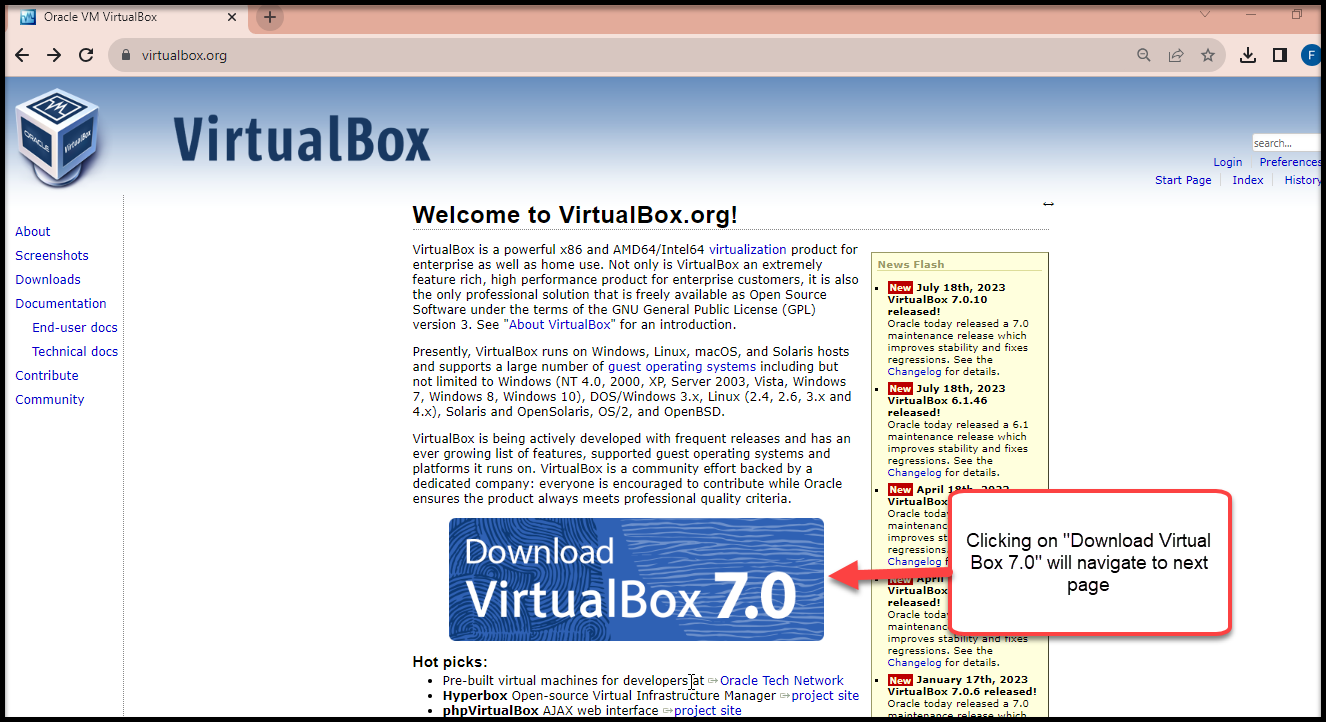


Figure # 2.1 - Selection of the Virtual Box for downloading

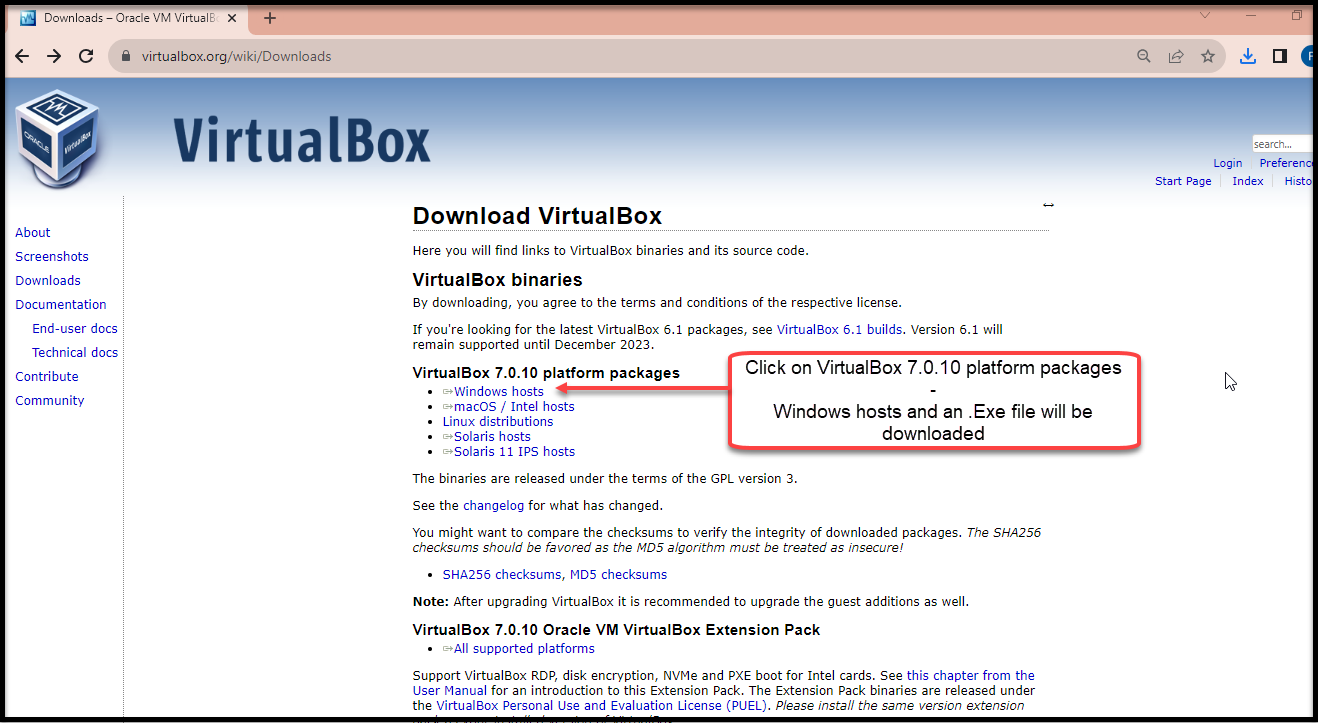


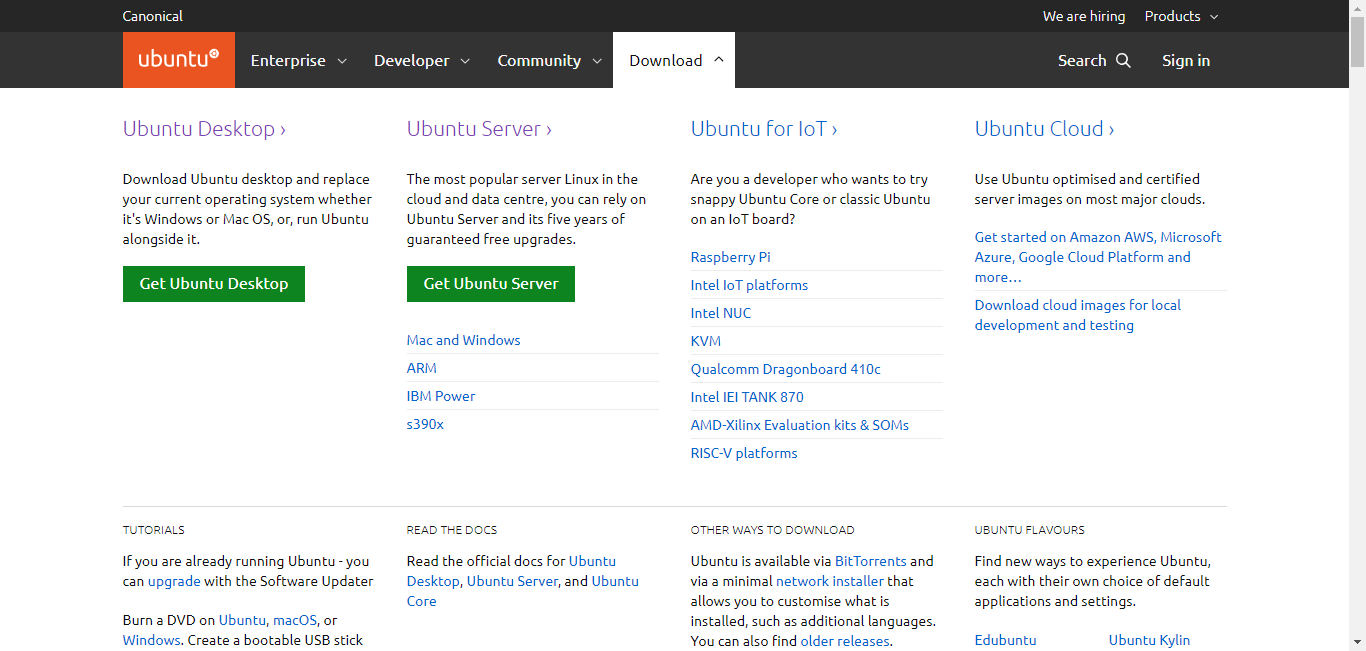
Figure #2.1.1 - Selection of the Virtual Box 7.0.10 platform for windows

1. Install VirtualBox by clicking the 'Next' button and then click the 'Finish' button. This will install VirtualBox version 7.0.0.10 as shown below



Figure # 2.1.2 – Finishing Installation

## 2.2 Download Ubuntu ISO Image



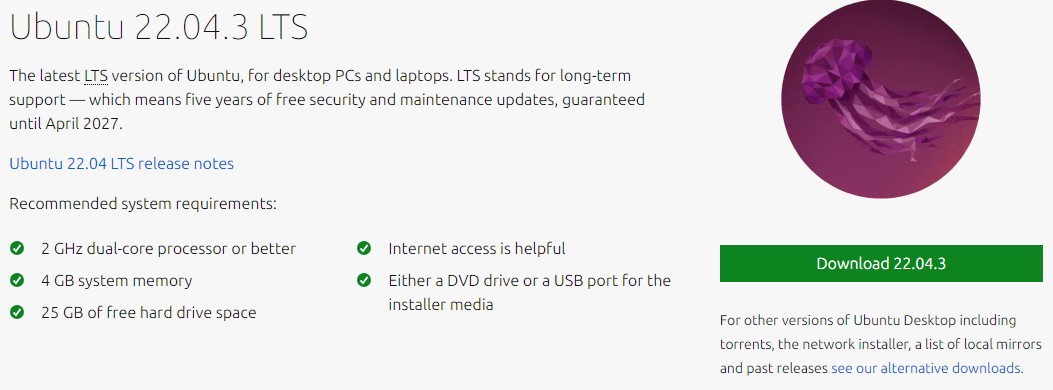


Figure #2.2.1 – Download Ubuntu Client ISO Image

* Click on Ubuntu Desktop button
* Recommendation is to choose latest LTS (Long Term Support) version
* Click on download now button to start downloading the ISO file

## 2.3 Ubuntu – Virtual Box Configuration

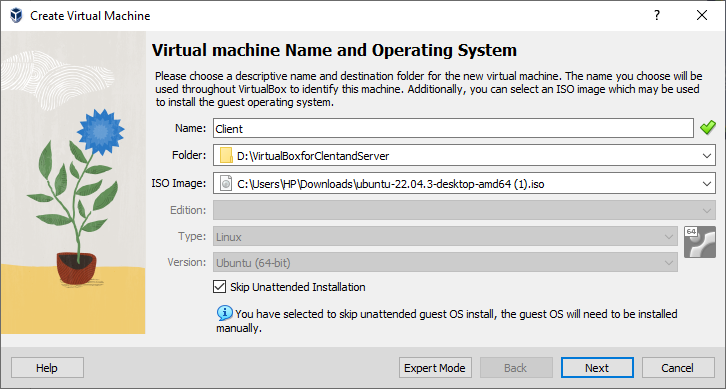


Figure #2.3.1 – Virtual Box Configuration

* Once the ISO image is downloaded open Virtual box and click on New button
* Specify Name: Client
* Specify folder directory
* Select an ISO image for client
* Select Linux as Type
* Select Version Ubuntu (64-bit)

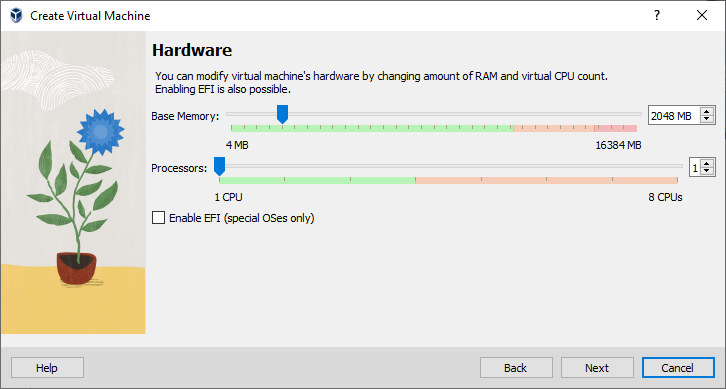


Figure #2.3.2 – Virtual Box Configuration

* In this screen user can select RAM and CPU. In this scenario 2GB RAM and 1 CPU will suffice the requirement.

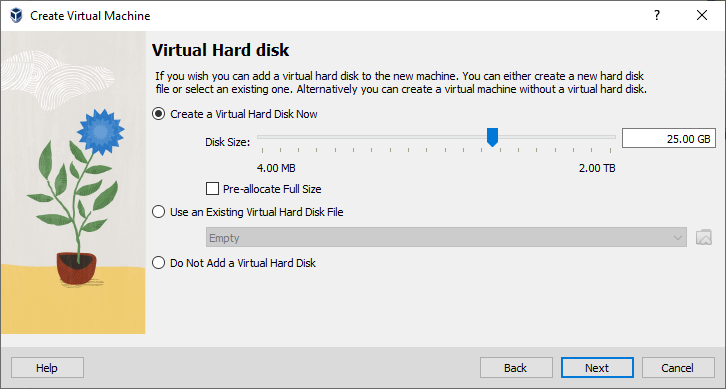


Figure #2.3.3 – Virtual Box Configuration

* User can specify the disk size. In this scenario 25 GB disk space will be allocated for Virtual machine.

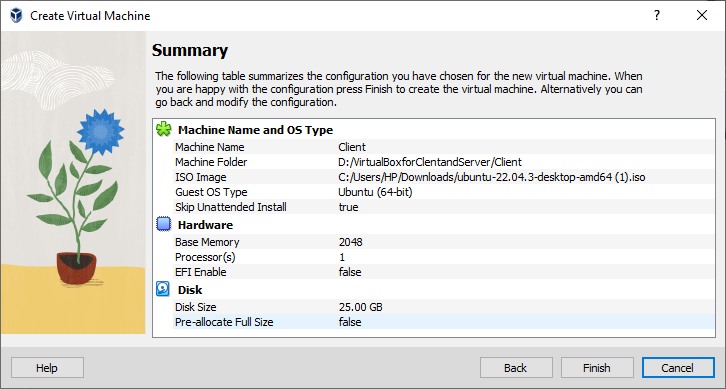


Figure #2.3.4 – Virtual Box Configuration

* Moving to next screen user will see the summarize view of selected option and press finish to complete the VM configurations.

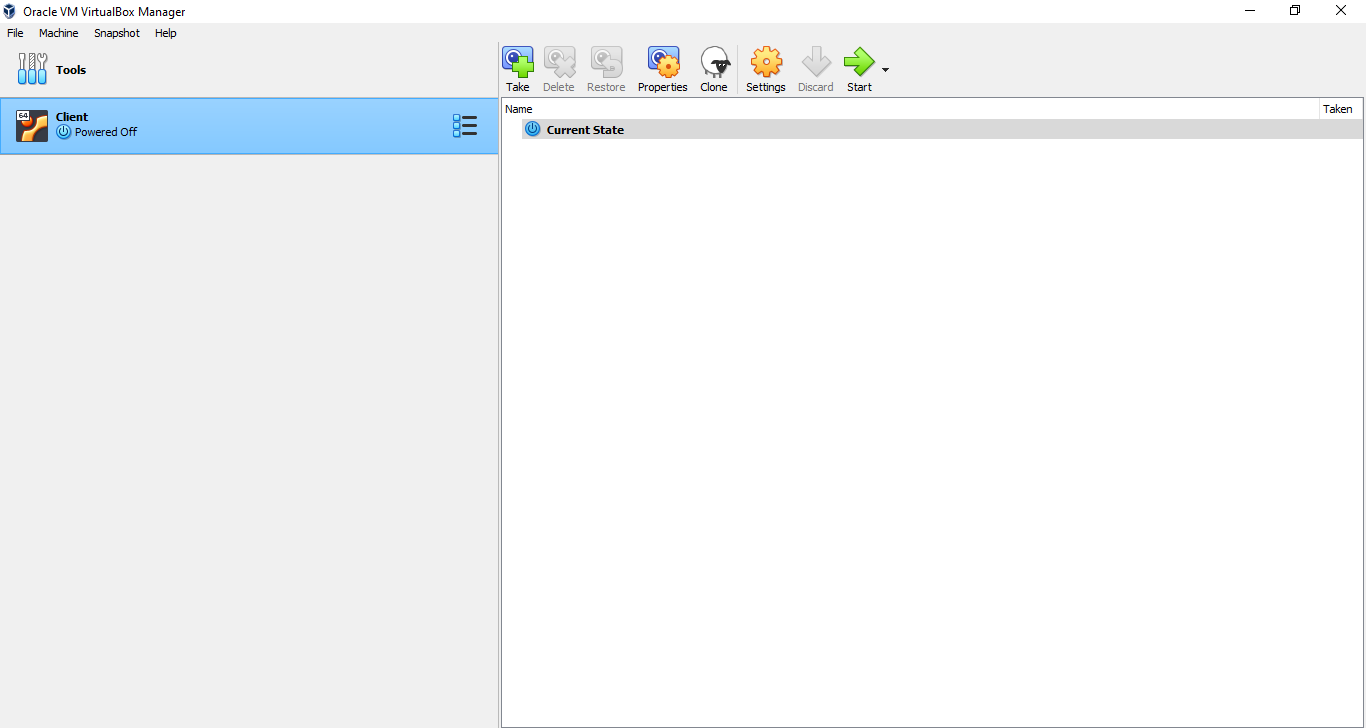


Figure #2.3.5 – Virtual Box Configuration

* User can now click on Start button to initiate the installation of Ubuntu Client machine.

# Ubuntu Master – Installation

## 3.1 Start Installation - Ubuntu

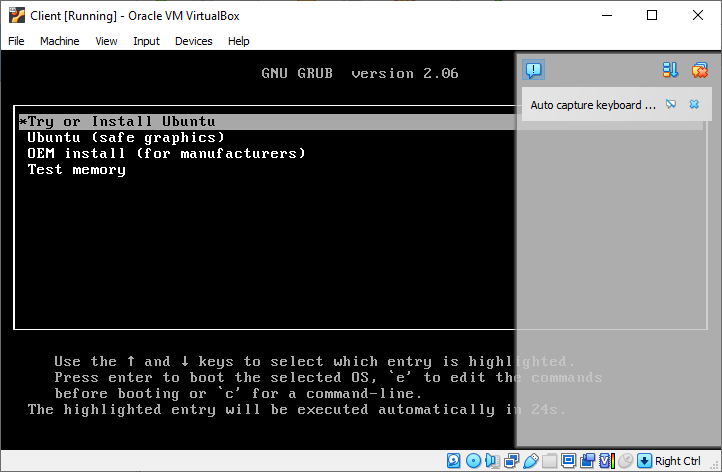


Figure #3.1.1 – Ubuntu Client Installation

* User can select Try or Install Ubuntu option to proceed with installation

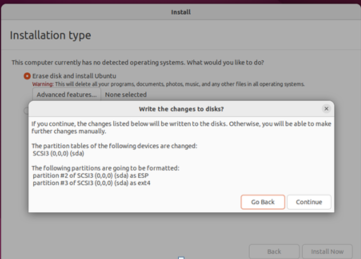
## 3.2 Language Selection

v

Figure #3.1.2 – Ubuntu Installation

* User can select language: English
* User can click on install Ubuntu button to proceed further

## 3.3 Installation Type: Write changes to disk

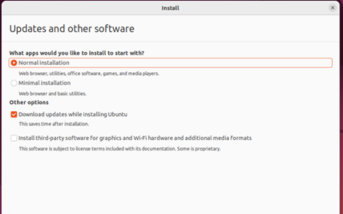


* User can select the Normal Installation option and also allow to download updates while installing.

Figure #3.1.3 – Ubuntu Installation

* User can select Erase disk and install Ubuntu option for New installation. Although precautionary measure should be taken in case of existing installation.

## 3.4 Installation Type: Normal



## 3.5 Select Country

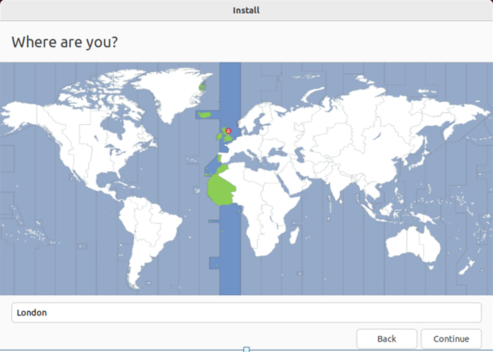


Figure #3.1.4 – Ubuntu Installation

* User will select the current location. In this scenario London is selected.

## 3.5 Setup Computer Name and Username

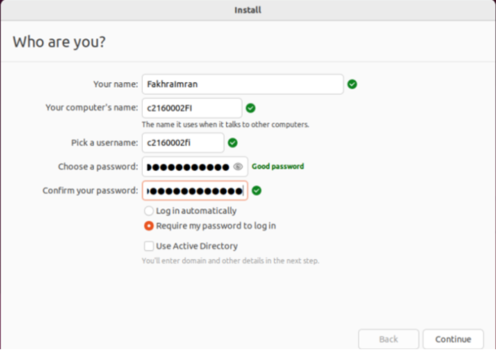


Figure #3.1.5 – Ubuntu Installation

* In this screen user can specify name and compute name.
* We will change the computer name and username later.

## 3.5 Completed Ubuntu Installation

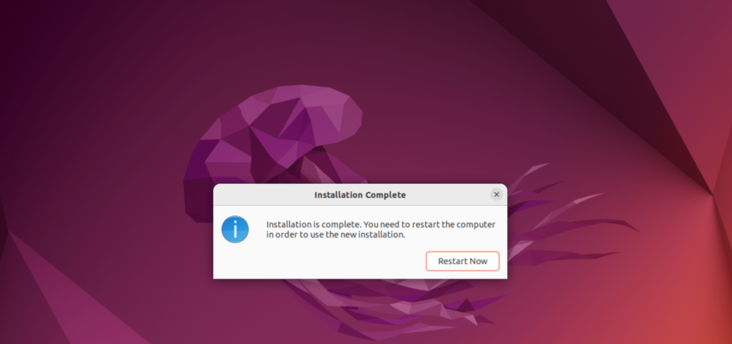
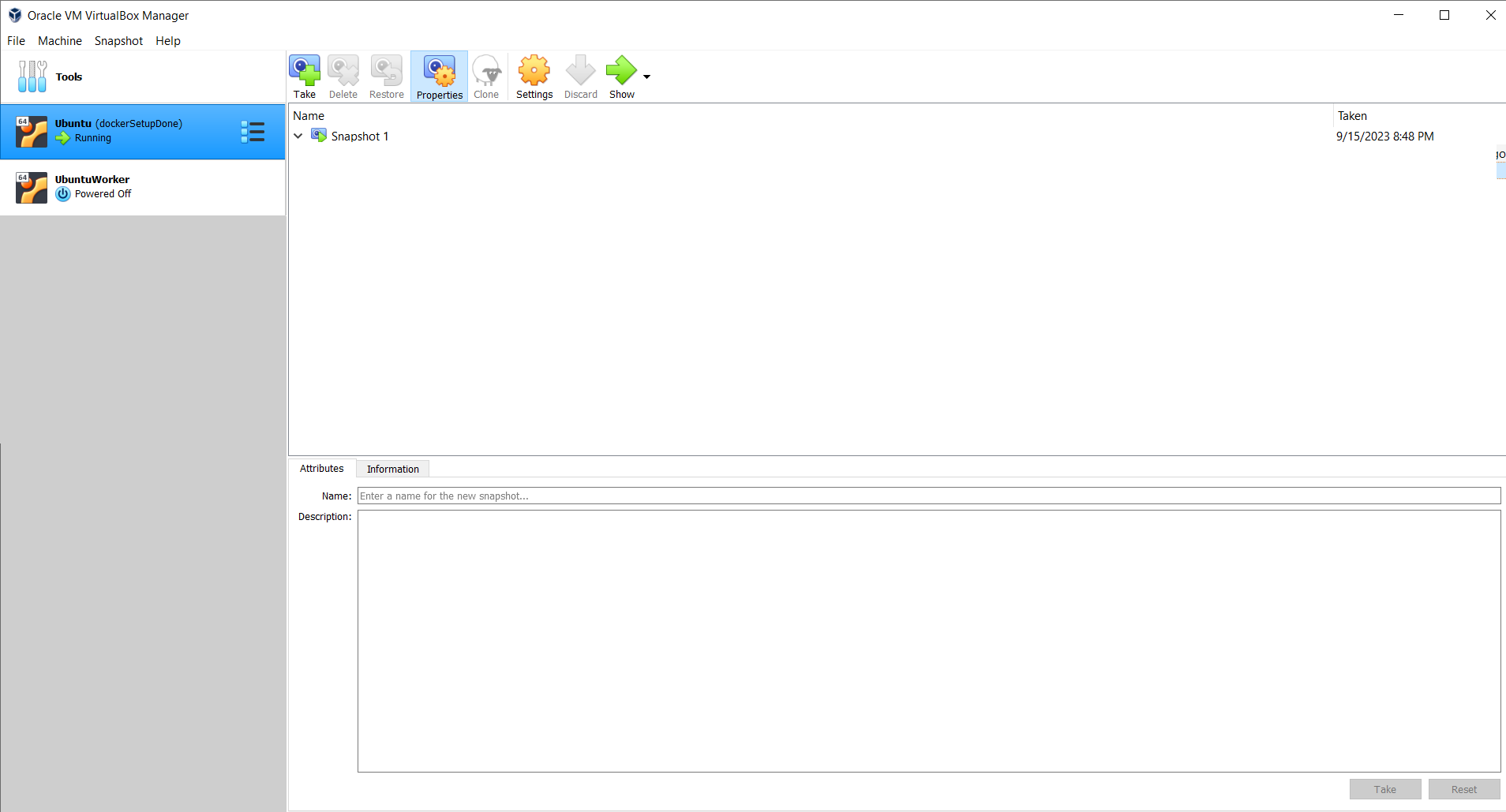


Figure #3.1.6 – Ubuntu Installation

* It will take few minutes to complete and once installation is complete it will notify with restart now option.

# Ubuntu Worker – Installation

## 4.1 Clone master machine



* Right click master machine and click on clone button to make a new clone of Ubuntu Machine
* We will call this newly created machine as worker machine.

# 5. Networking – Master and Worker Connectivity

In order to establish communication between master and worker machine it is necessary to setup bridge adapter.

## 5.1 Bridge Adapter Settings

Following are the list of benefits when you choose Bridge adapter mode for your VMs.

* It can directly communicate with other machines on your local network
* It will help to communicate with your VM as a standalone machine
* It will seemlessly integrate with other machines
* You can access shared resources on the local network.
* You can remotely access other devices
* If you choose to select bridge adapter mode than internet can be accessed directly through your local network gateway or router, just like any other machine or device.
* In future additional machine become part of local network.
* It provide integration, testing, remote access and internet connectivity benefits make it great choice in our scenario.

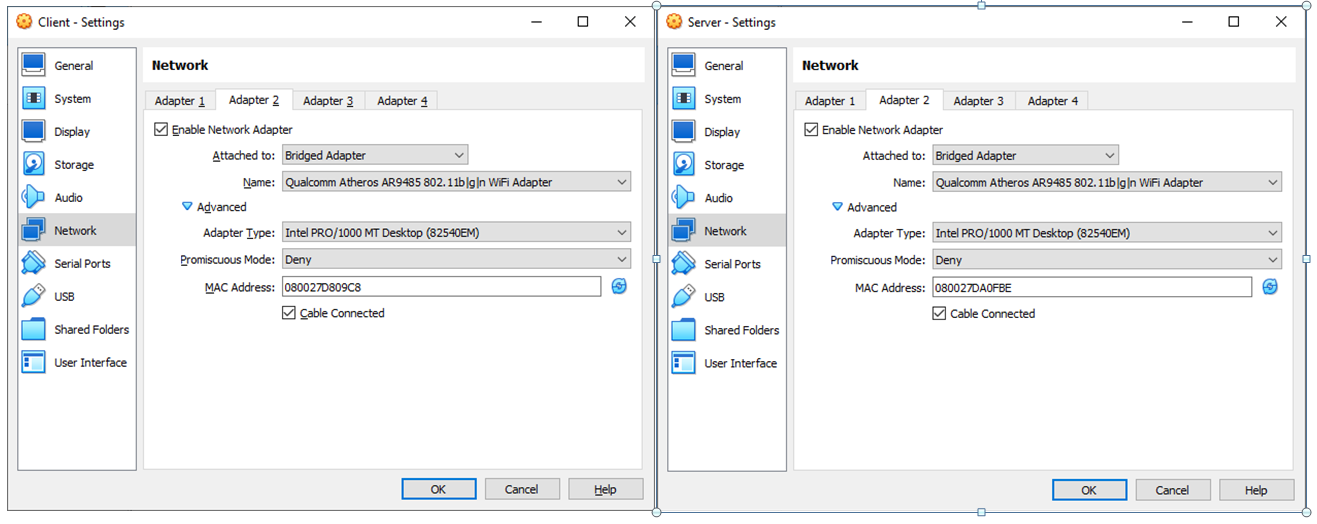


Figure #5.1 – Master and Worker Bridge Adapter settings

.

## 5.2 Hostname & IP address

* Master Node IP Address: 192.168.0.22
* Worker Node IP Address: 192.168.0.26
* Master Node Hostname is : Master
* Worker Node Hostname is: Worker
* Once the bridge adapter is setup both master and worker nodes communicate with each other using hostname, for this /etc/hosts file need to be updated accordingly.

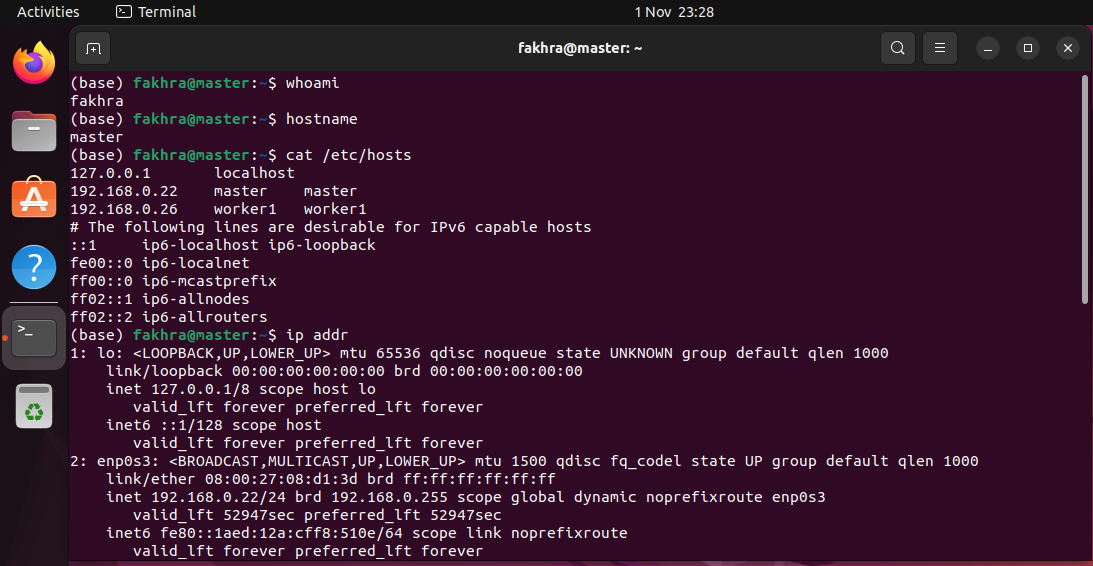


Figure #5.2 – Ubuntu Master IP Address

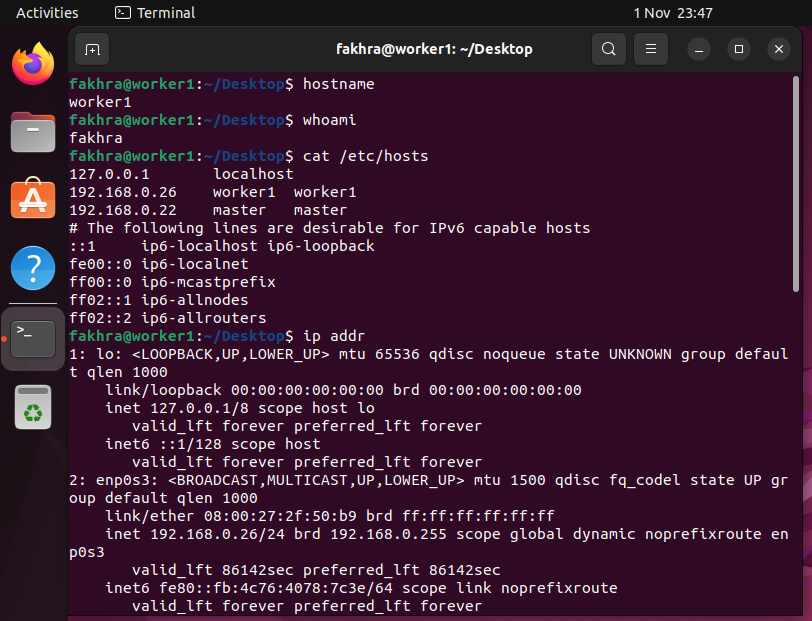


Figure #5.3 – Ubuntu Worker1 IP Address

## 5.3 Test Connectivity between Master and Worker Node

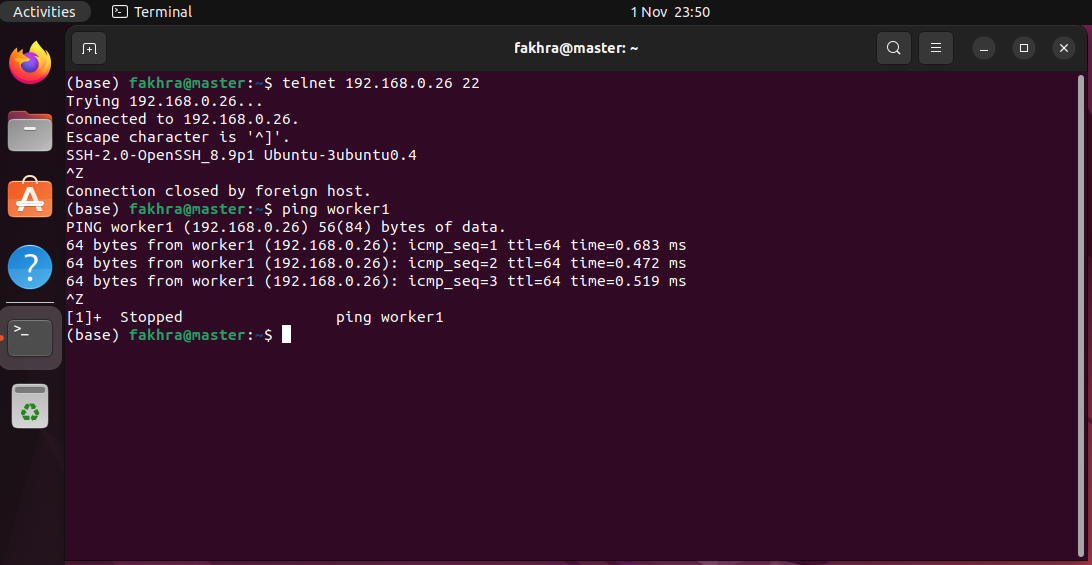


Figure #5.4 – Ubuntu master to worker connectivity using telnet

# Docker Installation (Master/Worker Node)

Underneath are all the steps for Docker Installation (Master/Worker Node) installation, and each step is clarified with screenshot [2].

## 6.1 Update List of Packages

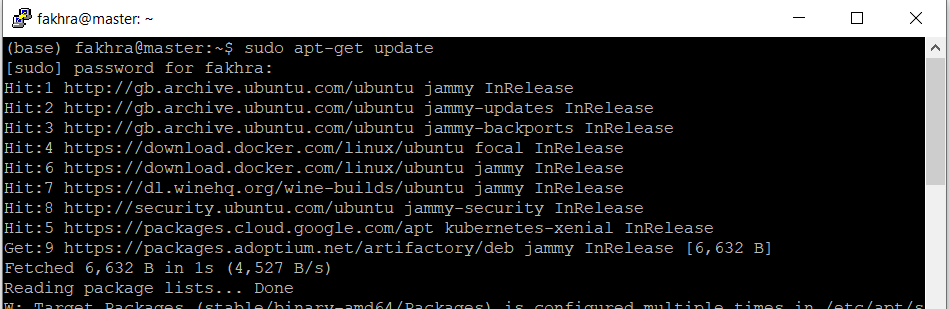
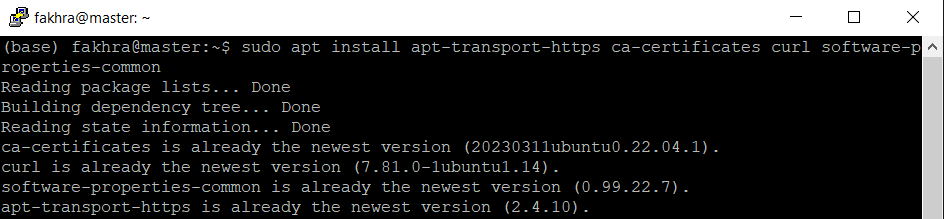
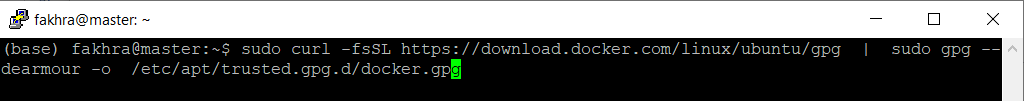


Figure #6.1 – Update list of packages

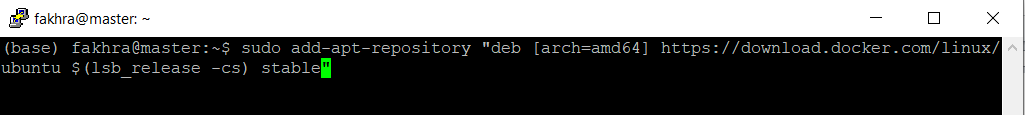
## Pre-requisites Packages Installation



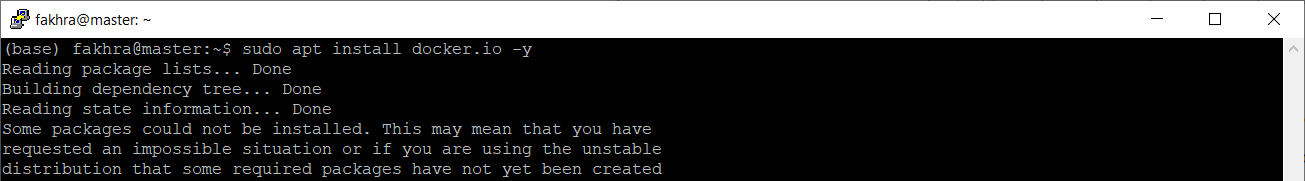
## Add GPG Key for repository



## Add Docker Repository



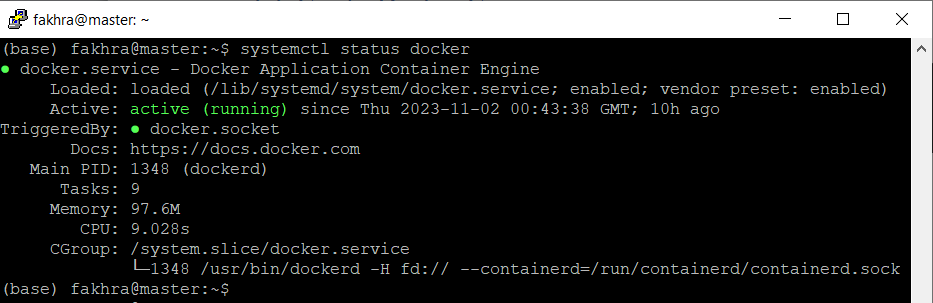
## Docker Installation



## Verify Docker Version



## Docker Status



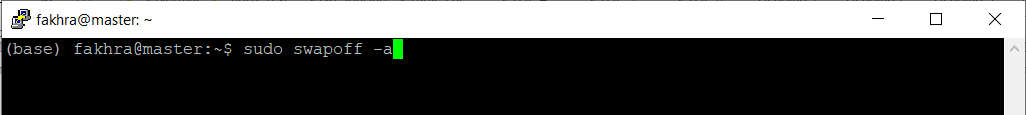
## Containerd Installation/Restart/Enablement

|  |
| --- |
| sudo apt update  sudo apt install –y cointainerd.io  containerd config default | sudo tee /etc/containerd/config.toml >> /dev/null 2>&1  sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml  sudo systemctl restart containerd  sudo systemctl enable containerd |

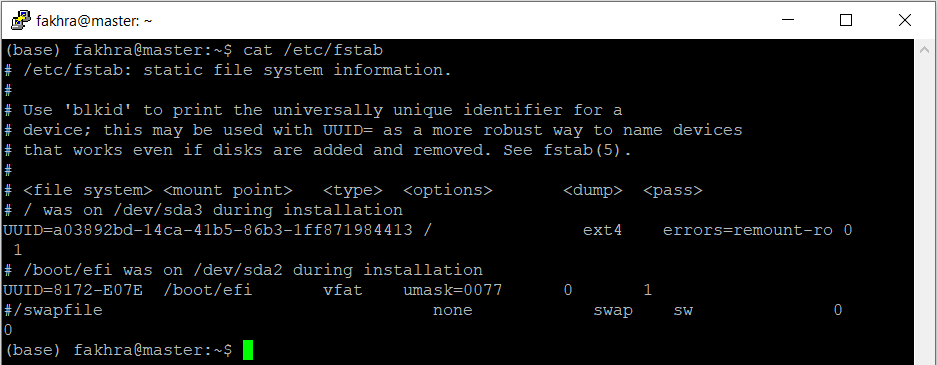
# Kubernetes Installation

Underneath are all the steps for Kubernetes Installation installation, and each step is clarified with screenshot [3].

## 7.1 Swap off Temporary (Master/Worker Node)

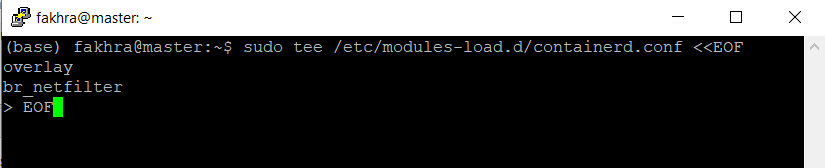


## 7.2 Swap off Permanently (Master/Worker Node)



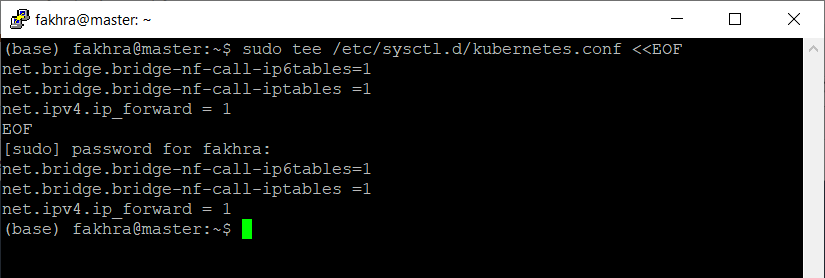
* Permanently disable swap on both nodes (Master/Worker)
* Comment swap file entry in /etc/fstab
* After changing the /etc/fstab file mount and load the changes
  + sudo mount –a

## 7.3 Let IP Tables See Bridge Traffic



For the master and worker nodes to correctly see bridged traffic, you should ensure net.bridge.bridge-nf-call-iptables is set to 1 in your config. First, ensure the br\_netfilter module is loaded.

* sudo modprobe overlay
* sudo modprobe br\_netfilter



* sudo modprobe net.bridge.bridge-nf-call-ip6tables=1
* sudo modprobe net.bridge.bridge-nf-call-iptables=1
* Once done you need to load the changes with below command
  + sudo sysctl --system

## 7.4 Kubernetes Installation – Kubelet Kubeadm kubectl

|  |
| --- |
| curl –s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -  sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"  sudo apt update  sudo apt install –y kubelet kubeadm kubectl  sudo apt-mark hold kubelet kubeadm kubctl |

## 7.5 Kubernetes Calico Setup

|  |
| --- |
| curl <https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifest/calico.yaml> -O  https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml  kubectl apply -f calico.yaml |

## 7.6 Kubernetes Initialization (Run only on Master Node)

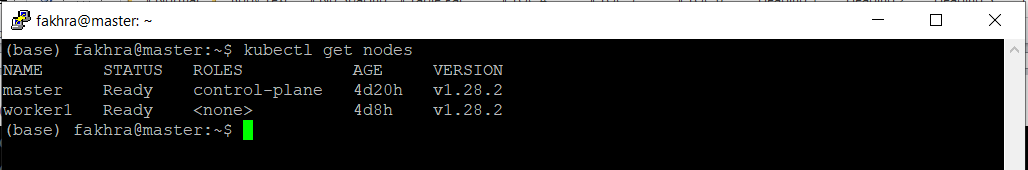
|  |
| --- |
| sudo kubeadm init --pod-network-cidr=10.10.0.0/16 --control-plane-endpoint=master  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  sudo chown $(id -u):$(id -g) $HOME/.kube/config  export KUBECONFIG=/etc/kubernetes/admin.conf |

* After init command it give join command token for the all worker nodes to join the master node to form a cluster

## 7.7 Join Master Node - (Run on Workers Node)

|  |
| --- |
| kubeadm join master:6443 --token go3cyd.uzw221d2wkbyz116 \  --discovery-token-ca-cert-hash sha256:1bb98806c417f96da432cedcbf6c33dbc71b705963ac77b5bbd62c6738261062 |

## 7.8 Verify Worker Node is attached to Master

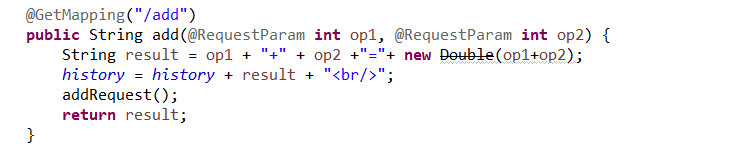


# Application – Creating a JAR file

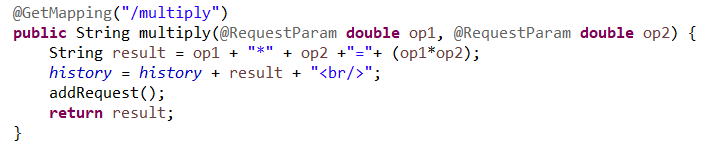
## 8.1 Introduction

In order to deploy java based application on kubernetes and to scale it up and down dynamically I will take calculator rest APIs that performed Add, multiple, divide and subtract operations.

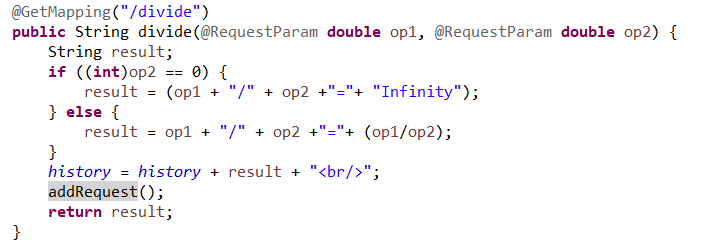
## 8.2 Add API



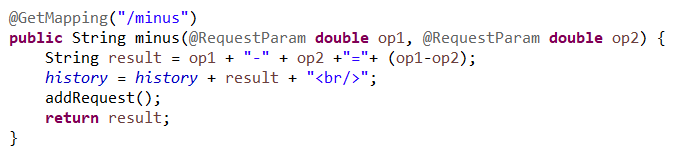
## 8.3 Multiply API



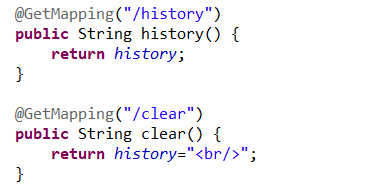
## 8.4 Divide API



## 8.5 Minus API

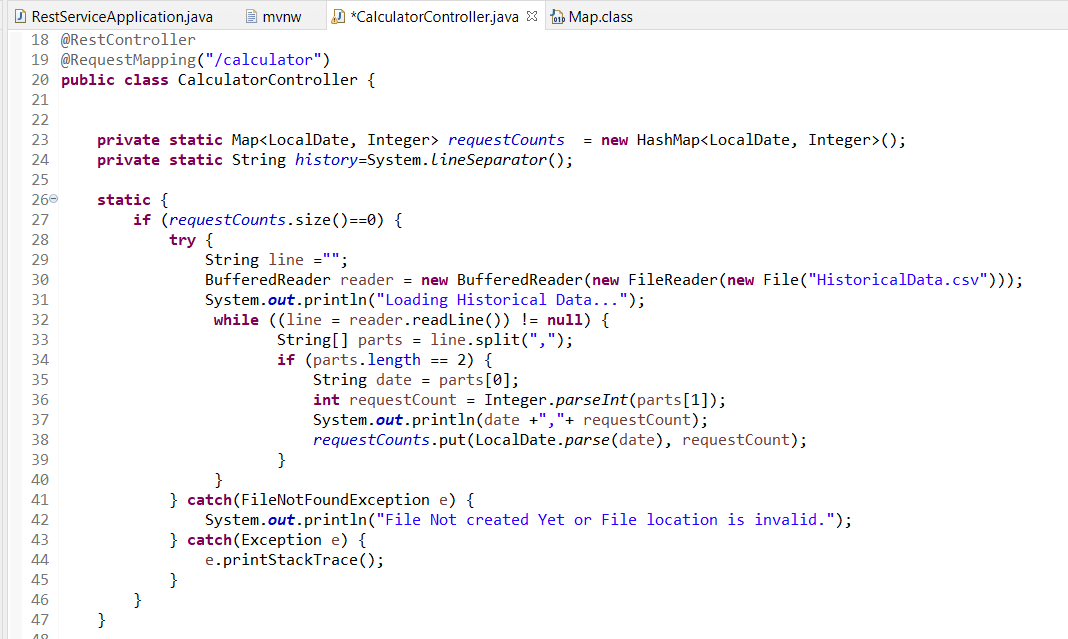


## 8.6 Print History and clear History APIs



## 8.7 Static Initializer that load Historical data

* This function loads the historical data that was saved on daily basis.
* Any single API called is counted and save on daily basis in a file name “HistoricalData.csv”.
* The information saved in HistoricalData.csv file is “Date” and “RequestCounts”
* On the base of Historical data saved, the system will make prediction and proactively scale up or scale down the application PODs, this behavior override the default static CPU or memory based scaling mechanism



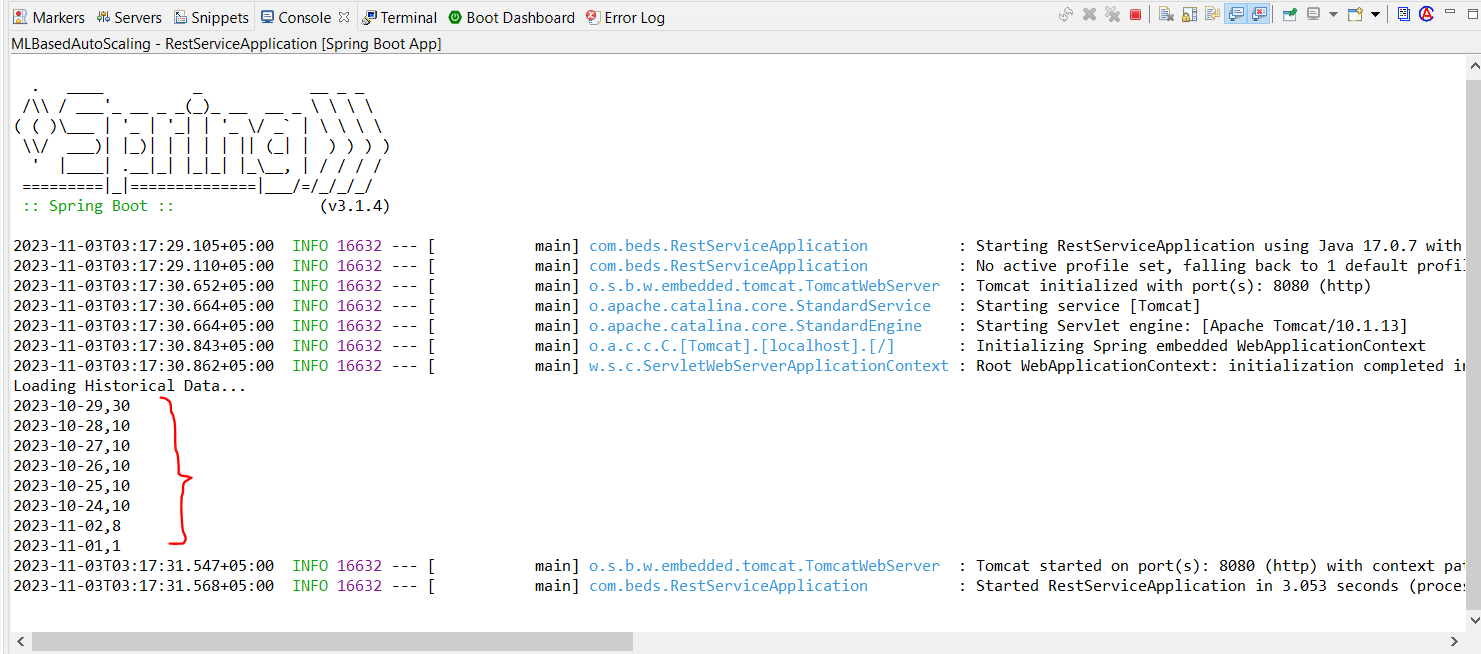
## 8.8 Add Request Count for Each Day

* This function sum total request count on daily basis and keep information in “HistoricalData.csv"
* Each single request is counted and saved it against the current date.
* This function is called for each call made to add, multiply, divide and subtract operation

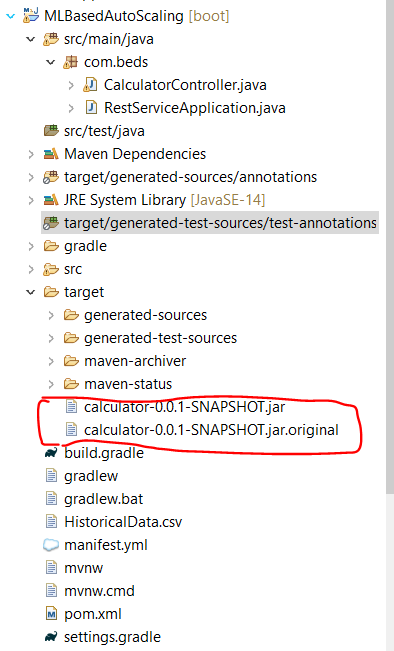


## 8.8 Create a JAR file

* Click on project
* Right click and select “Run As” and then select “Spring Boot App” to compile and run the application



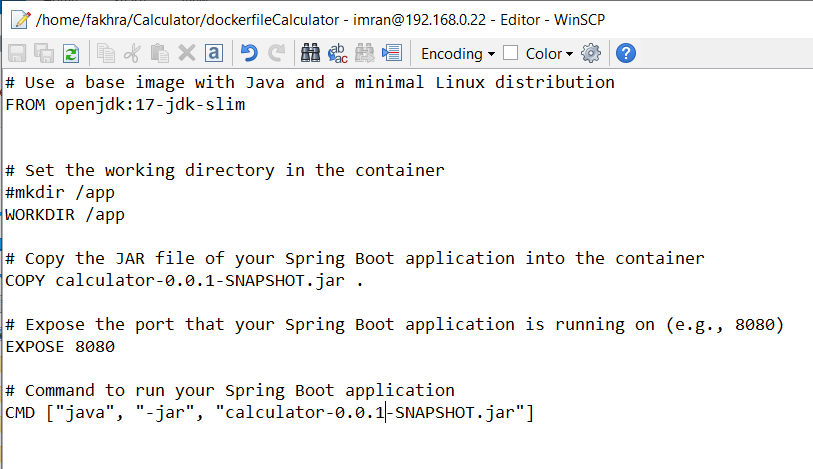
* Once it is run successfully it will create a jar file at following location
* Created jar file location



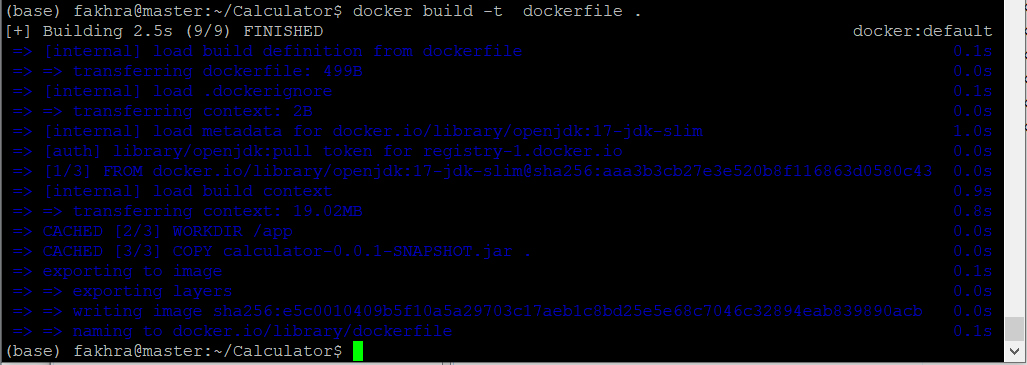
# Docker – Create a Docker File

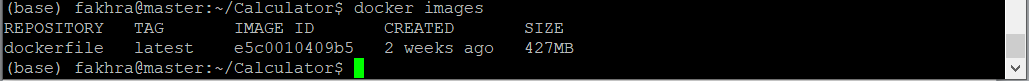
## 9.1 Prepare a Docker file

* This file is used to create a docker image.
* That image will be pushed to dockerhub.com
* Once uploaded it will be pulled and deployed on kubernetes.
* Let’s create a docker file.



## 9.2 Build an Docker Image



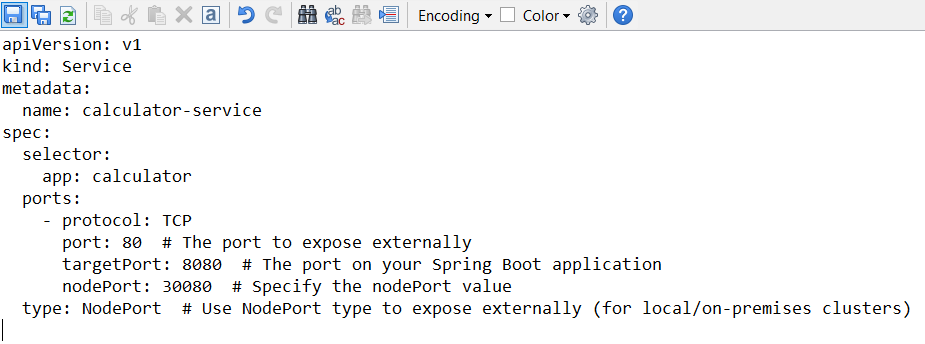
* Run below command to see the created image

## 9.3 Push image to docker hub

|  |
| --- |
| docker login https://hub.docker.com/  docker images  docker tag calculator:latest fakhrakhan/calculator:latest  docker images  docker push fakhrakhan/calculator:latest |

# Kubernetes - Deployments

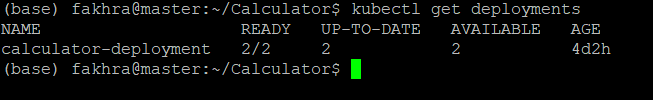
## 10.1 Kubernetes - Service File



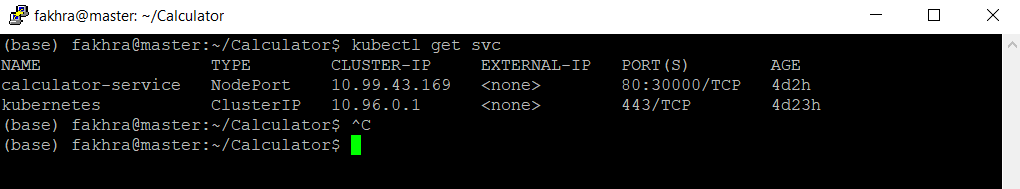
## 10.2 Kubernetes - Apply Deployments

|  |
| --- |
| kubectl apply -f deployment.yaml  kubectl apply -f service.yaml |

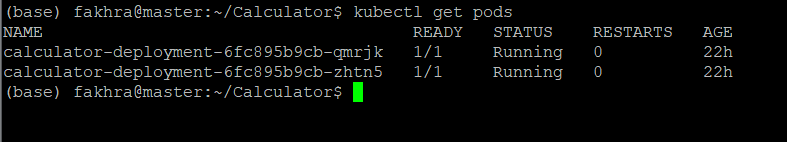
## 10.2 Kubernetes - Get Deployments



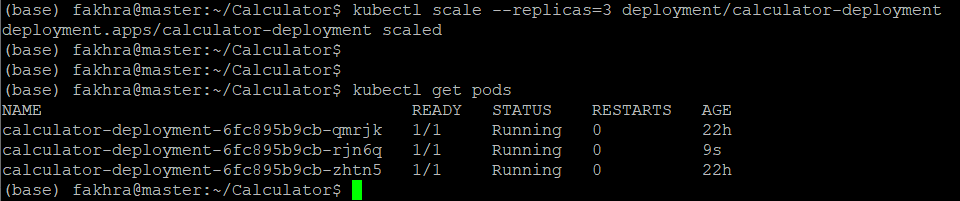
## 10.3 Kubernetes - Get Service



## 10.4 Kubernetes - Get Pods



## 10.5 Kubernetes - Scale Up



* This will scale the calculator service from 2 PODs to 3 PODs, but this process is done manually.
* Now I have setup the request counts and based on the historical data and request count I will proactively scale the application based on the prediction done by pre-trained machine learning models.
* Pre-trained machine learning model will take HistoricalData.csv file as input and predict current date traffic and on the base of that I will make the decision to scale up the application accordingly.

## Test the calculator Service



# Machine Learning based Predictive/Dynamic Scaling of kubernetes

## 11.1 Difference between CPU metric based scaling (Static) and Predictive Scaling

Kubernetes autoscaling based on CPU metris, it uses simple threshold-based approach to determine when to scale the number of Pods. It compares the current CPU utilization to a predefine threshold limit. Here, the thing is if CPU utilization exceed the threshold on fixed define utilization it scale up the application. Therefore, in order to encounter this limitation machine learning predictive Scaling is used.

|  |  |
| --- | --- |
| **Threshold based Scaling** | **Predictive/Dynamic Scaling** |
| * Traditional kubernetes based on autoscaling on CPU metric * Relies on predefine static threshold * React to changes based on CPU utilization * Doesn’t predict future trends * Sudden spikes and fluctuation might result in over provisiong * It might scale too early or too late * Primary focus on short term changes in CPU utilization | * Machine learning models analyze real time captured historical data * Do prediction based on future trends/traffic patterns * It adopt proactive approach * Machine learning based model distinguish between temporary fluctuation and real traffic increase * Make proactive decision on predictive analysis on historic data * Can analyze long term trends and provide insights and futurn trends and patterns |

## 11.2 Pre-Trained Models

There are many pre-trained time series based forecasting models. The most popular one are Prophet and ARIMA

Due simplicity nature in this predictive forecasting FB Prophet tool can be utilized to achieve the desire result.

The Prophet is a good choice. Prophet, developed by Facebook, is designed to be user-friendly and is particularly well-suited for forecasting with daily observations that display patterns on different time scales

## 11.3 Actual Plan

* Actual plan was to install fbprophet
* Use python script given below to predict the current date request counts
* The HistoricalData.csv file will be used as an input that is being used to capture the historical data which contains date and request counts.
* See Section: [8.8 Add Request Count for Each Day](#_8.8_Add_Request)
* Provide real time historical data to Prophet model instead of Pandas dummy data
* Model will initialize the data and predict future request count on daily basis
* On the base of predicted request count shell script code will be execute on master node of kubernetes cluster to execute kubectl scale up or down command to increase or decrease kubernetes Pods to handle expected traffic on daily basis.

## 11.4 Machine Learning based Python code Using Facebook Prophet

|  |
| --- |
| from fbprophet import Prophet  import pandas as pd  # Create a dataset with 'ds' (date) and 'y' (value) columns, including the last 7 days  data = pd.DataFrame({  'ds': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05', '2023-01-06', '2023-01-07'],  'y': [1, 2, 3, 4, 5, 6, 7]  })  # Initialize and fit the Prophet model  model = Prophet()  model.fit(data)  # Create a DataFrame for future dates, including the next day '2023-01-08'  future = model.make\_future\_dataframe(periods=1)  # Generate forecasts  forecast = model.predict(future)  # Extract the expected value for '2023-01-08'  expected\_value = forecast[forecast['ds'] == '2023-01-08']['yhat'].values[0]  print(f"Expected value for '2023-01-08' based on the Historical Data: {expected\_value:.2f}") |

## 11.5 Predictive/Dynamic Scaling based on historical data

* Based on results gathered from above python script we can use kubernetes scaling command to increase or decrease number of pods to handle the expected traffic on specific date.
* Command for scaling: “kubectl scale –replicas=<Expected Value> deployment/calculator-deployment”
  + See Section: [10.5 Kubernetes - Scale Up](#_10.5_Kubernetes_-)
* Source Code Line: forecasted\_value = forecast[forecast['ds'] == '2023-01-04']['yhat'].values[0]
  + The above command give forecast value for the current date which is being used to scale up or scale down the pods to handle expected traffic as an proactive measures.

# Challenges Faced

During the installation of 'fbprophet,' several challenges were encountered. These challenges primarily stemmed from a group of dependent libraries that were found to be incompatible with Python version greater than 3.10. Consequently, the installation process for 'fbprophet' was not completed.

Solution:

* As an alternative solution would be new docker based container with required python library 3.7 or 3.8 can be created to install fbProphet tool with in the container.

# Way Forward Strategy

* Apart from “pip install” command there are other tools available to install FBProphet.
* Tools like “conda” can be utilized to install FBProphet, but this required to install “miniconda” or “anaconda” tools
  + conda install -c conda-forge fbprophet
  + Whether you should install Anaconda or Miniconda depends on specific needs and the size of the environment you want to set up
* There are multiple pre-trained models available as open source in the market.
  + Exploration of those tools with respect to time series can by handy here to achieve the desired results
* There are numerous option to explore further to achieve the desired time series based result set.

# References

[1] Zhiheng Zhong, M. A. (2023). Machine Learning-based Orchestration of Containers: A Taxonomy. *Machine Learning-based Orchestration of Containers: A Taxonomy*. doi:https://doi.org/10.1145/3510415

[2] Turnbull, J. (2014). *The Docker Book* (Vol. v1.2.0 ). the Creative Commons. Retrieved March 15, 2023, from https://dockerbook.com/

[3] Jakir Mehemood Patel, J. M. (2016). *Kubernetes Essential Practical Guide to Learn Kubernetes.* Retrieved Ocotober 24, 2023