Contents

[A. Download & Install VM 2](#_Toc30518180)

[1. Download oracle VM virtual box 2](#_Toc30518181)

[2. Download Ubuntu VM image 2](#_Toc30518182)

[3. Create new VM 2](#_Toc30518183)

[B. VM Configuration & Install required libraries 8](#_Toc30518184)

[1. Dragging and Dropping Files in VirtualBox 9](#_Toc30518185)

[2. Installing the VirtualBox Guest Additions 10](#_Toc30518186)

[3. Configuring Shared folders 13](#_Toc30518187)

[4. Configure proxy settings 16](#_Toc30518188)

[C. Install required software using APT 18](#_Toc30518189)

[1. Install aws software 19](#_Toc30518190)

[2. Install k8s 19](#_Toc30518191)

[3. Install serverless 19](#_Toc30518192)

[4. Install aws kubectl 19](#_Toc30518193)

## Download & Install VM

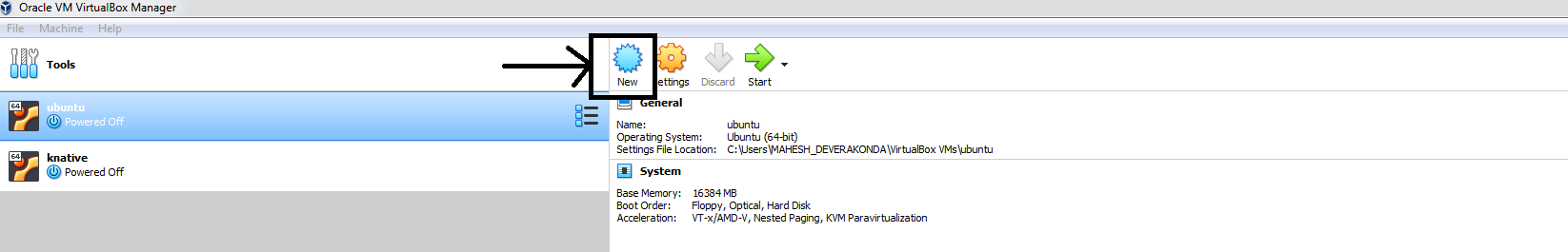
1. Download oracle VM virtual box

<https://download.virtualbox.org/virtualbox/6.1.2/VirtualBox-6.1.2-135663-Win.exe>

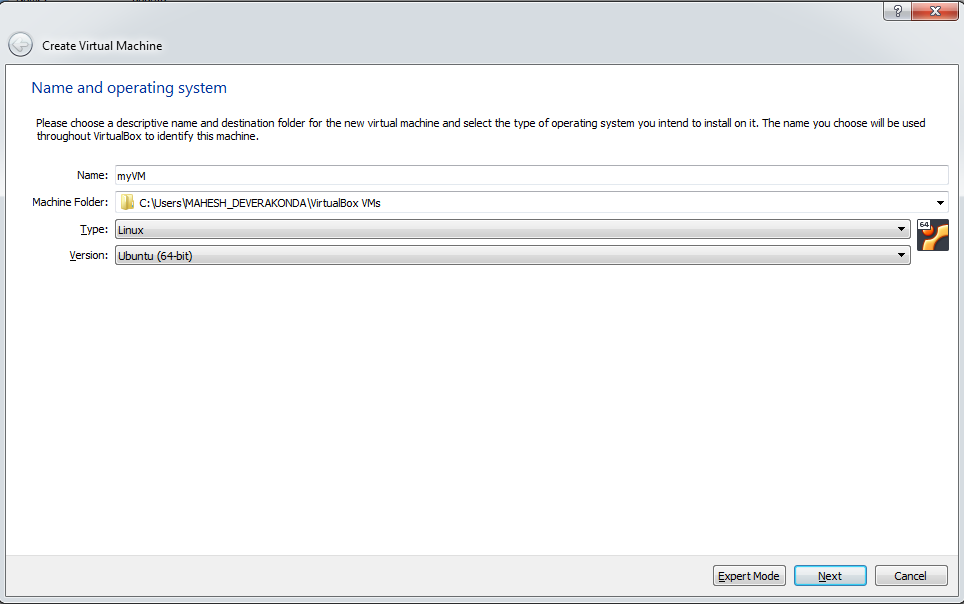
1. Download Ubuntu VM image

<http://releases.ubuntu.com/18.04/ubuntu-18.04.3-desktop-amd64.iso>

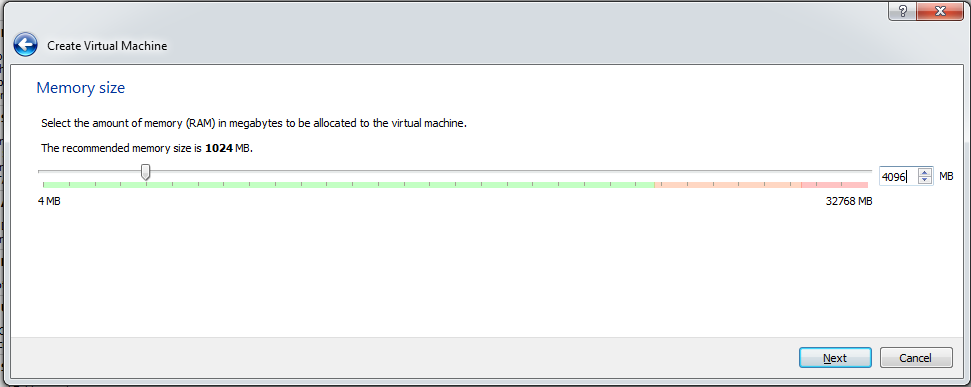
1. Create new VM
2. Click on “New”



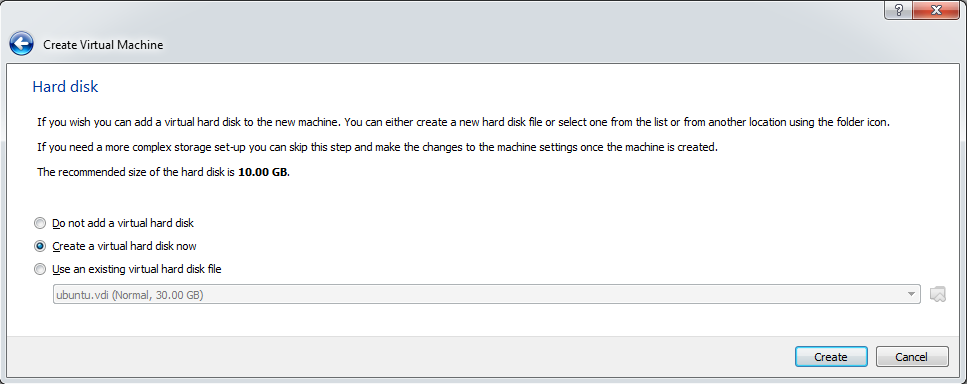
1. Provide details as below –



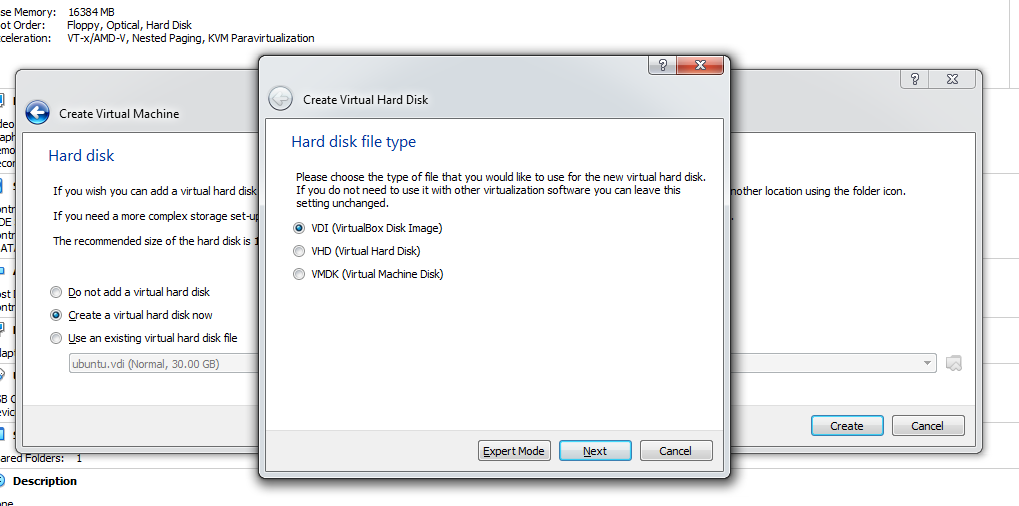
1. Specify “Memory Size” as 4096 MB (preferred) or 2048 MB, depending on the availability of resources.



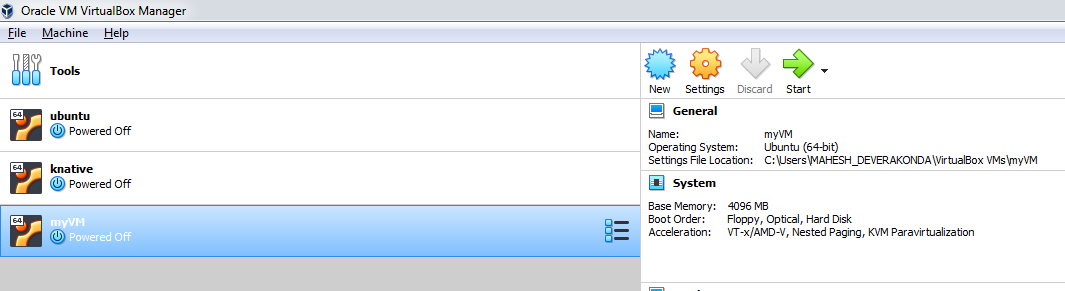
1. Create a virtual hard disk as below and click on “Create”.



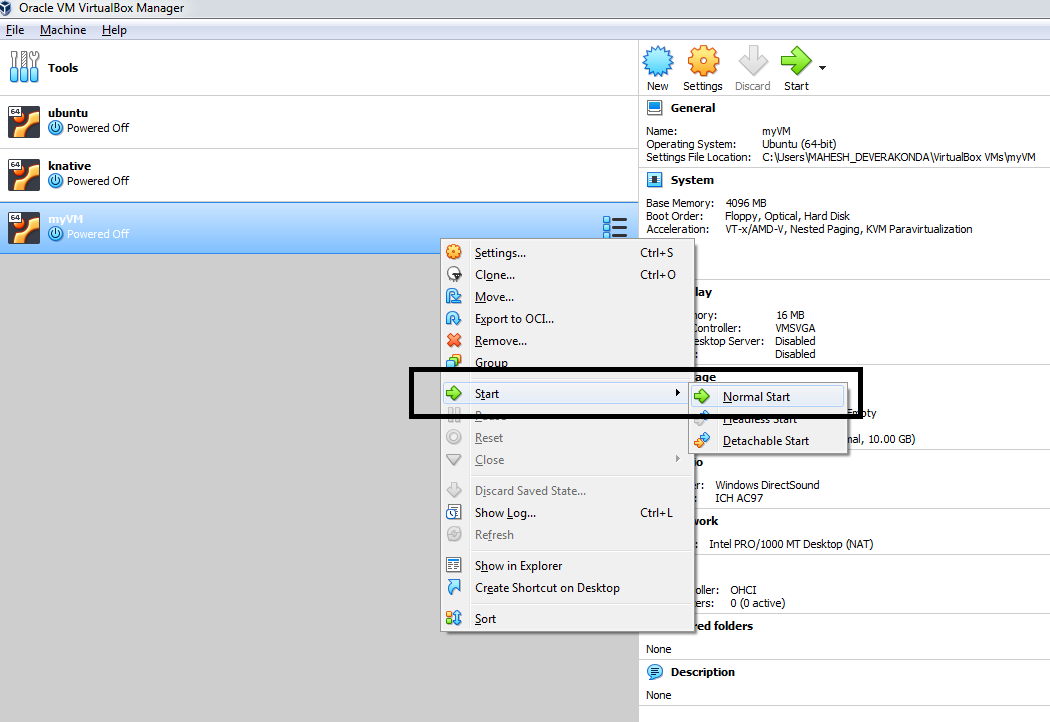
1. Select VDI, click “Next” and rest of the options, leave default and click on “Next”.



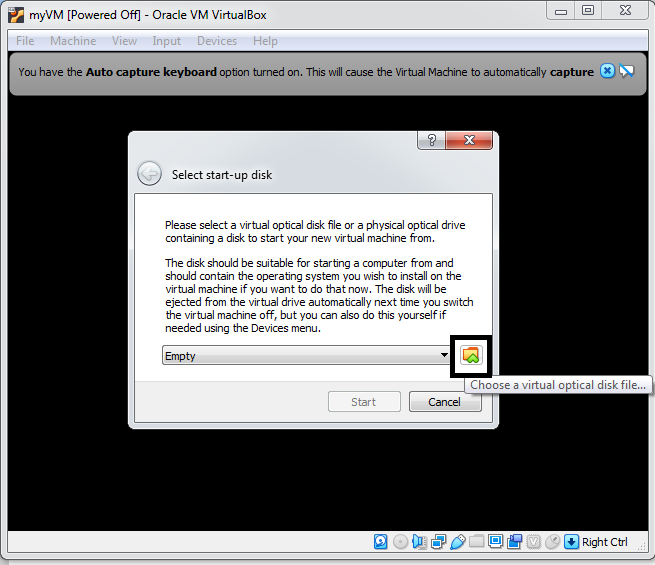
1. After following above steps, one VM with name “myVM” will be created as below.

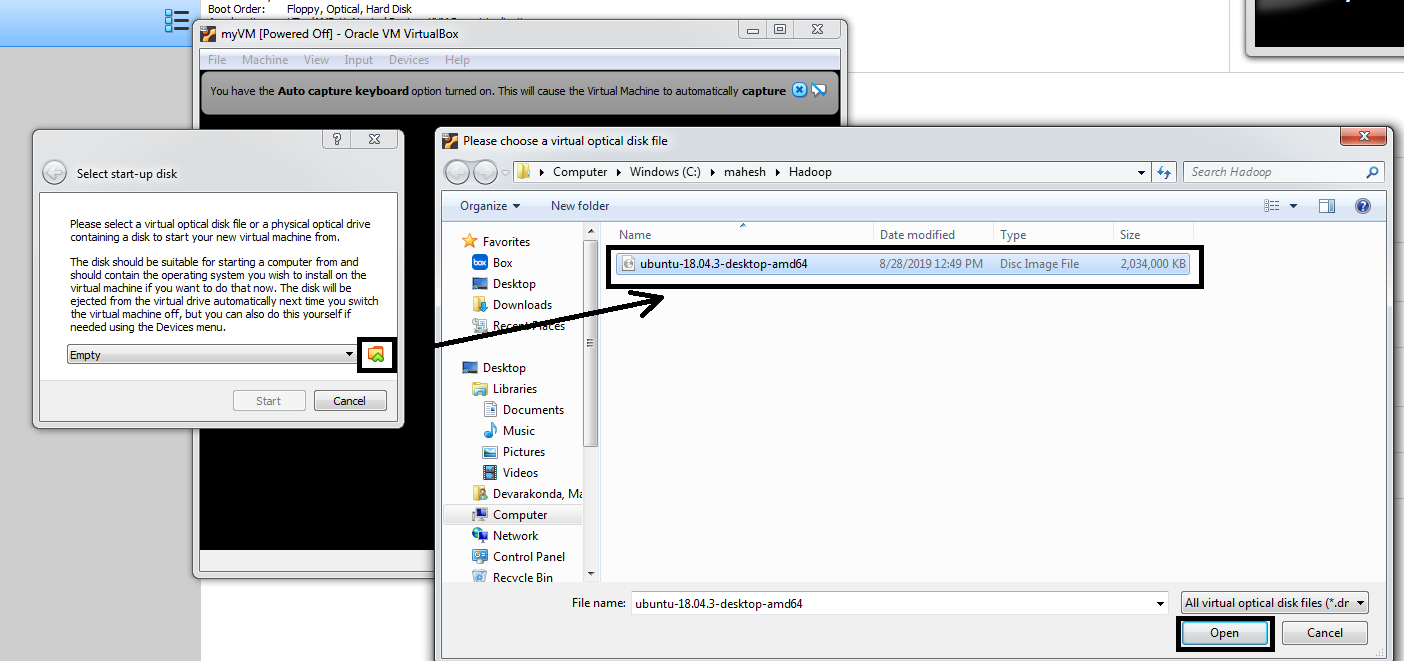


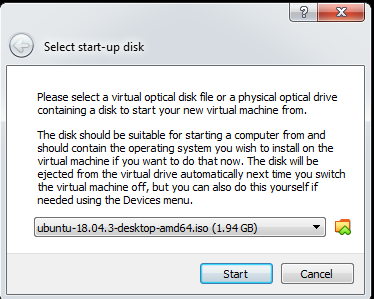
1. Once VM is create, start it.



1. Choose the downloaded Ubuntu VM image and click “Start”. This process may take few seconds.



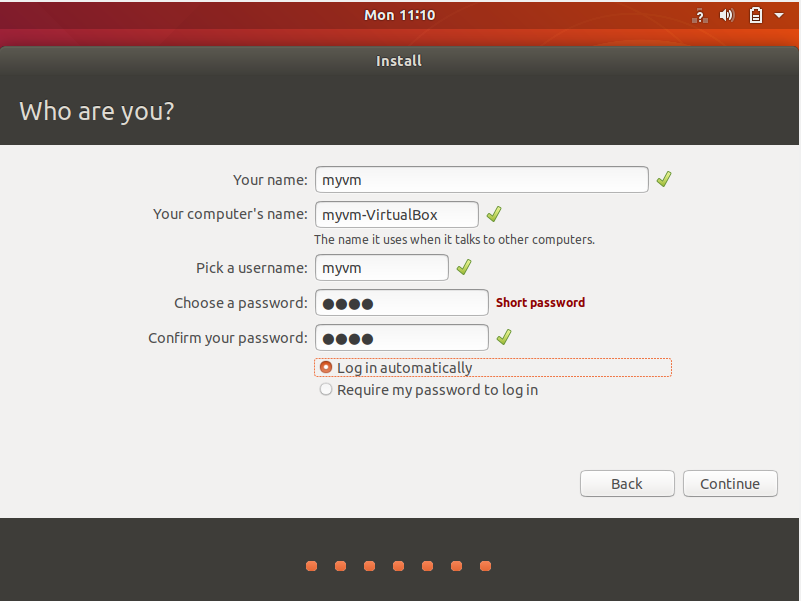


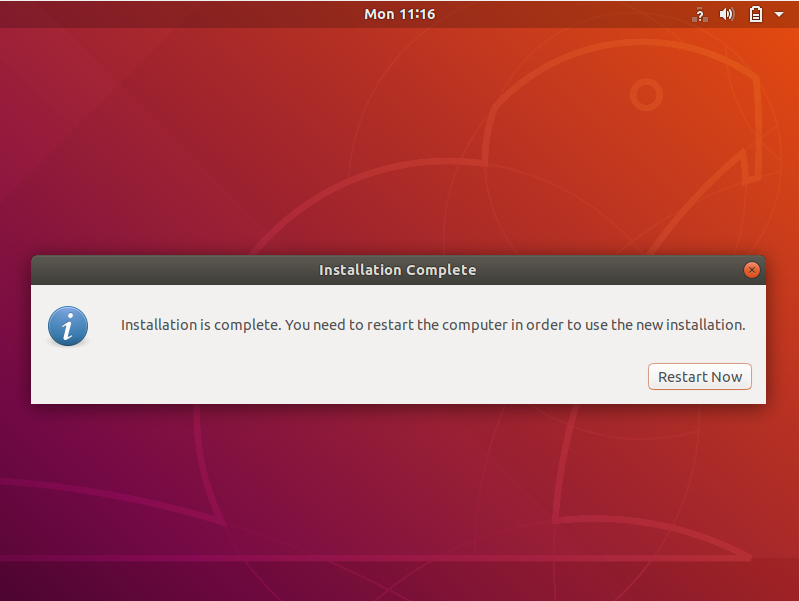


1. Click on “Install Ubuntu”. This will take few minutes. If any choices are asked, go with the default selection.



1. Select your country for the time zone of the VM.
2. Provide details for your user and passwords and click “Continue”. This may take some time. Once VM is installed, “Restart VM”.





## VM Configuration & Install required libraries

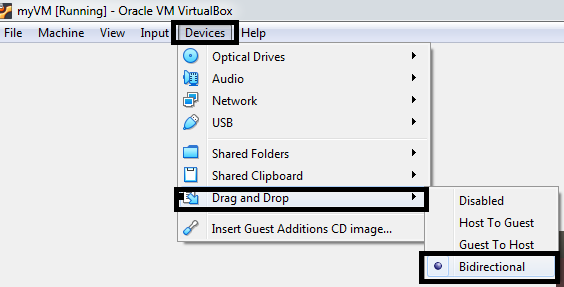
We will see this screen after reboot



1. Dragging and Dropping Files in VirtualBox

If you only need to transfer a few files quickly, you can simply drag and drop the files in. This means that you will be able to drag files from the host to the guest and from the guest to the host. Once bidirectional drag and drop is checked, you should be able to begin dragging and dropping files

1. On the top bar of the running guest machine, click on *Devices and click on Drag and Drop* and make sure that *Bidirectional* is selected

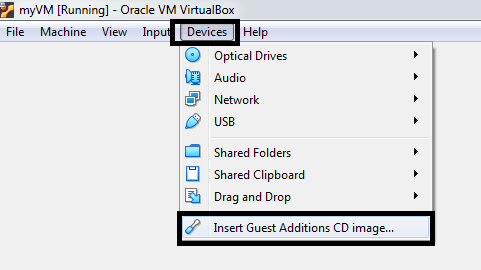


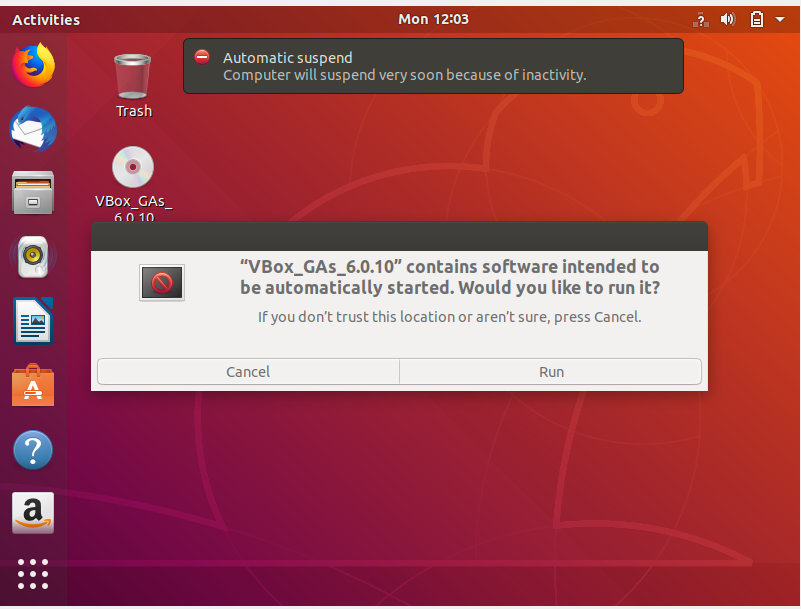
**NOTE:**Sometimes when dragging files *into***the course VM**, you may not be able to drag into the file browser directly. If you encounter this issue, you should drag your files onto the *Desktop* and move the files around from there. You should see the cursor change when it is ready to drop files

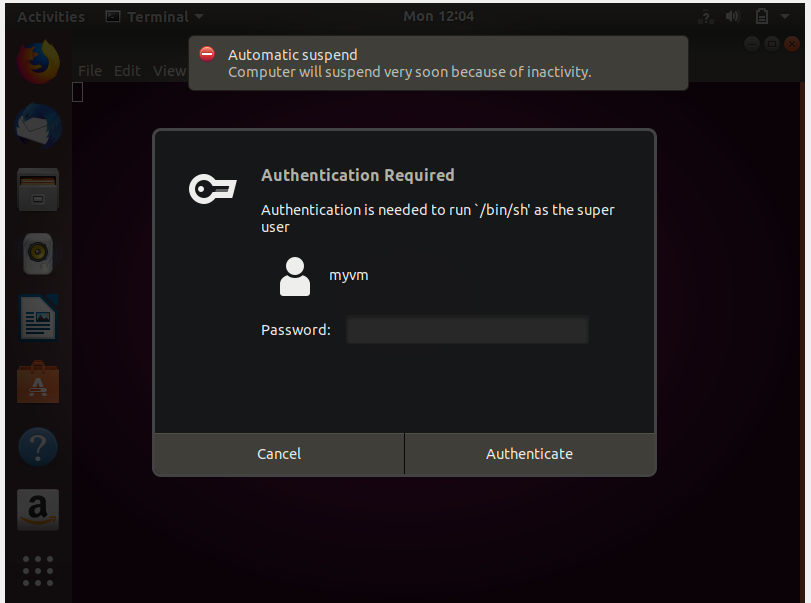
1. Follow the same steps for Shared Clipboard (if required)
2. Installing the VirtualBox Guest Additions

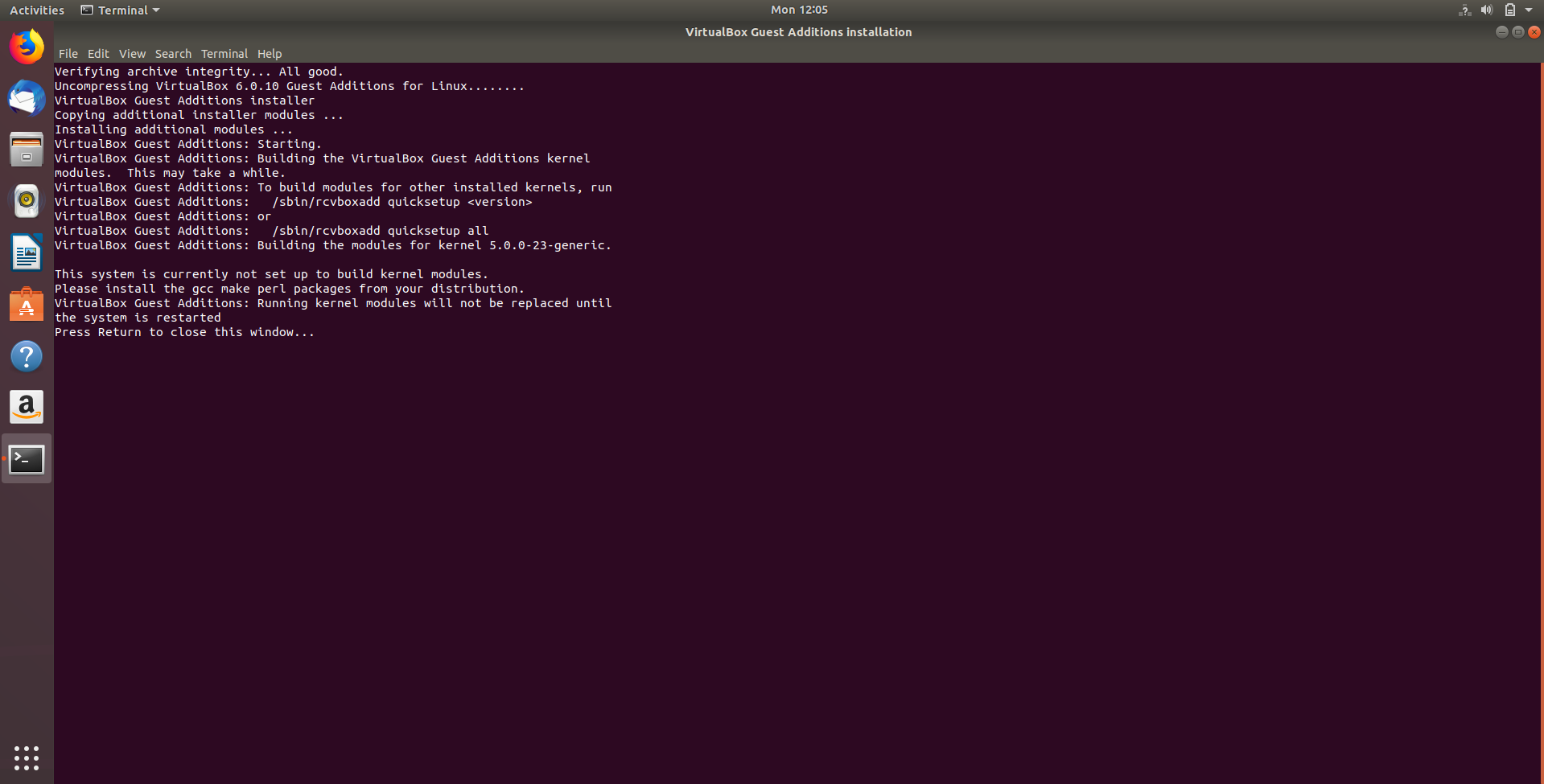
The VirtualBox Guest Additions consist of device drivers and system applications that optimize the operating system for better performance and usability.

Select Devices and Click on Insert Guest Additions CD image. Then we get prompt of VBOX\_Gas. Click on Run. It will ask for authentication. Once it is done then we will see full screen of VM. Then close terminal





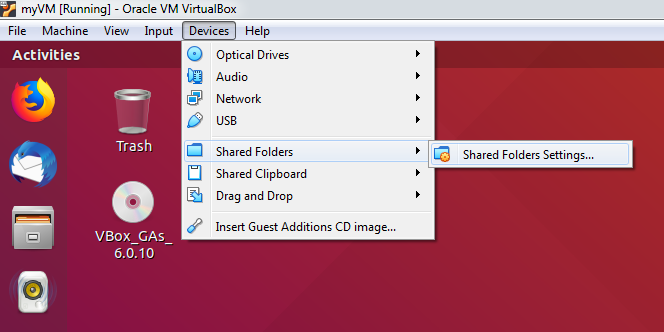




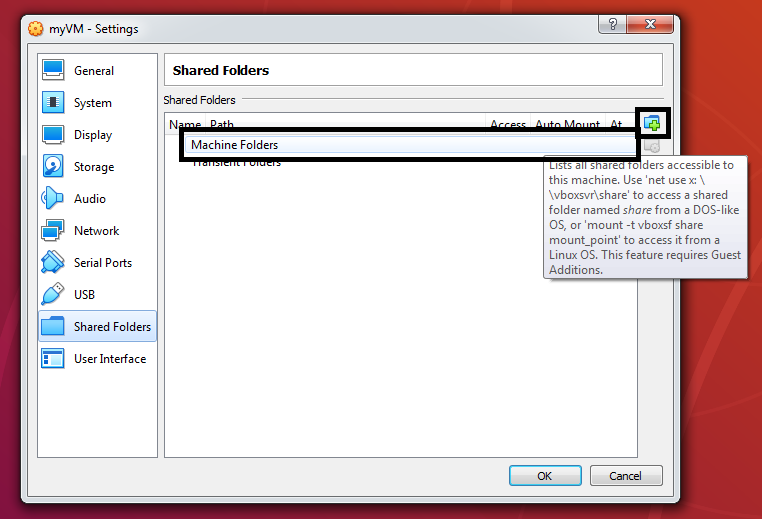
1. Configure Shared folders

Creating a shared folder between the guest and the host allows you to easily manage files which should be present on both machines

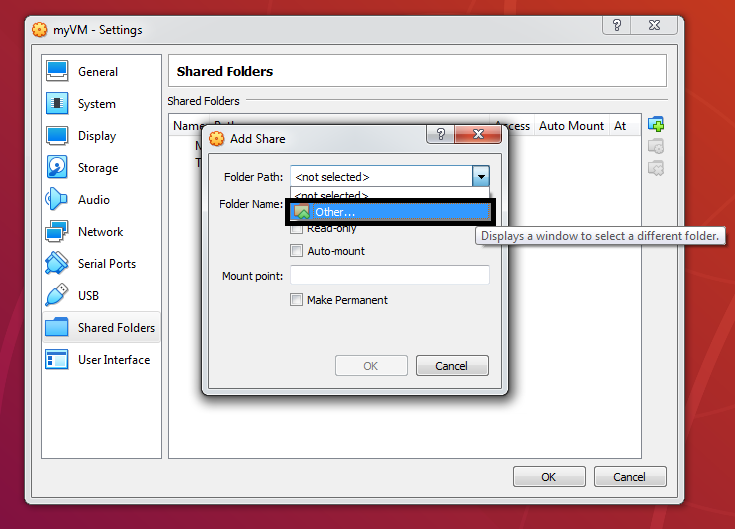
1. Create a virtual shared folder. You can do this by going to **Devices** and clicking on**Shared Folders – Shared Folders Settings**.



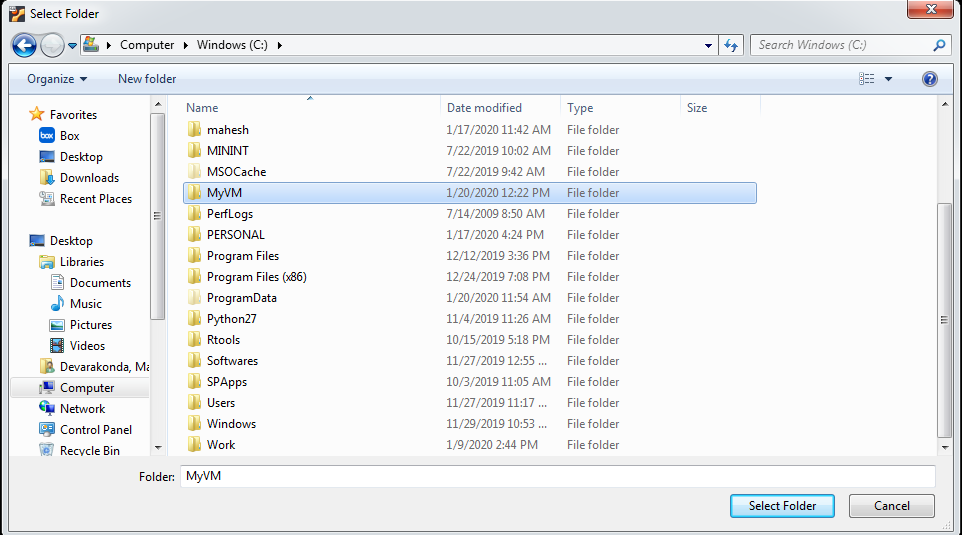
1. Now click on the**Add New Shared Folder** button on the right



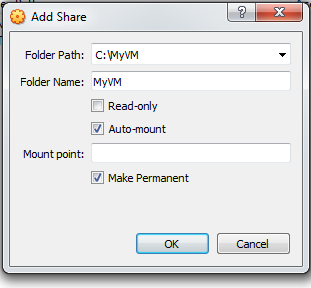
1. In the **Folder Path** box, click the down arrow and then click **Other**.



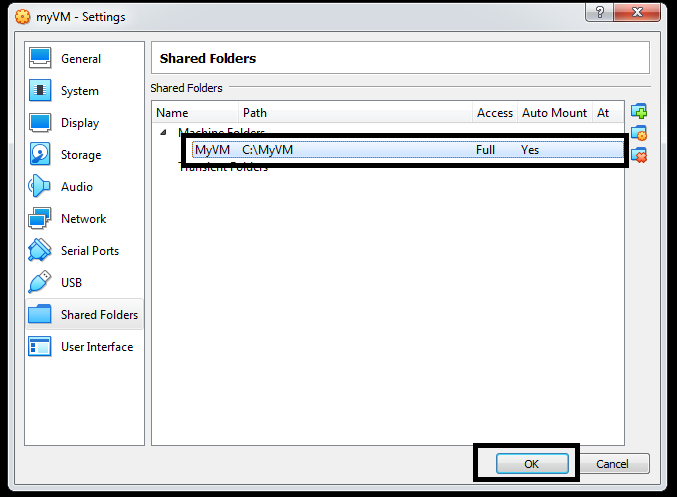
1. Browse to the folder on the host OS that you would like to share with the guest OS.



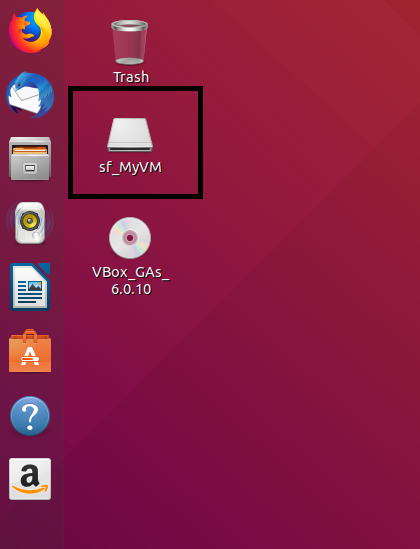
1. In the Folder Name box, give your share a name. You can choose to make it read-only, auto-mount the folder after restarts, and make the shared folder permanent.



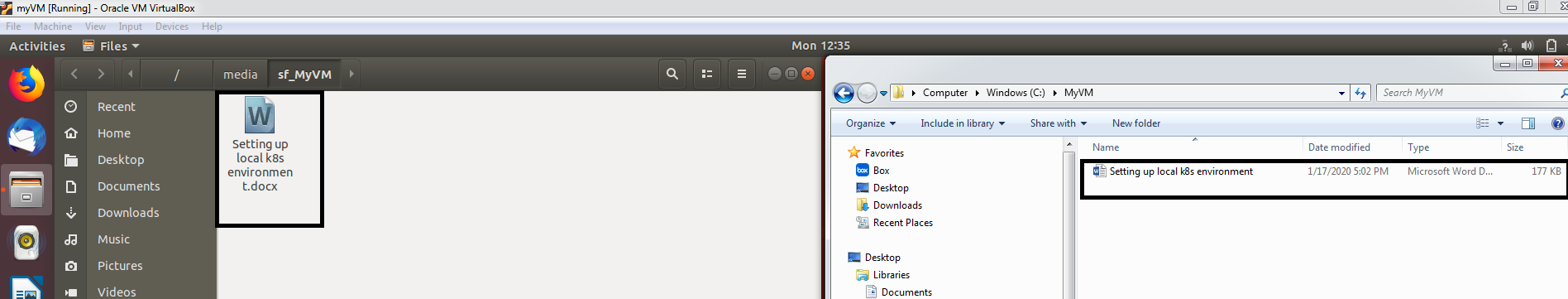
1. Click OK and the new shared folder will show up under **Machine Folders** if you chose to make it **Permanent**, otherwise it will show under **Transient Folders**.



1. We will see the sf\_folder\_name on the desktop



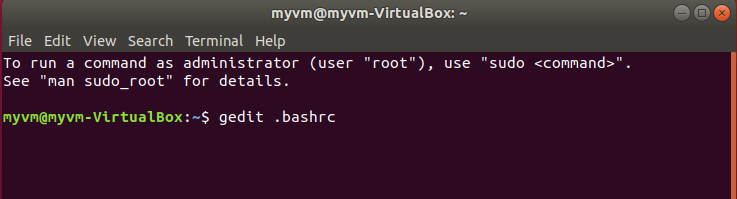
1. In Host OS, if we place any file in the shared folder is automatically reflected in the guest OS



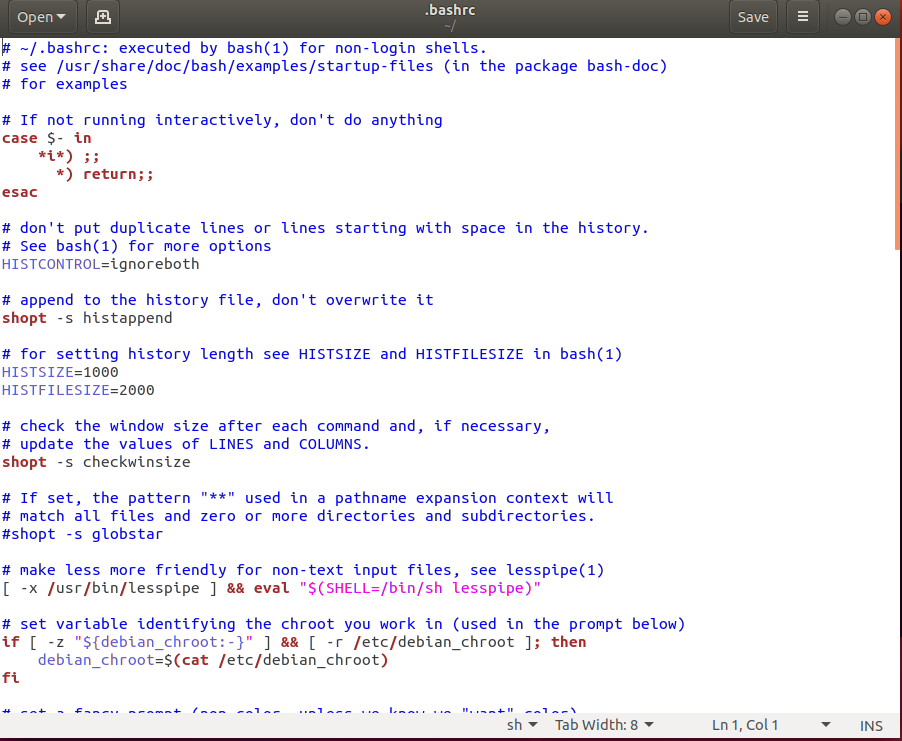
1. Configure proxy settings

Set proxy server in your bashrc and terminal. If you use sudo or bash sudo please do in that shell as well.

1. Open Terminal & type “gedit .bashrc” then click enter



1. You will see the .bashrc document as below



1. Go to bottom of the file and add proxy settings.

set HTTP\_PROXY = "http://arun\_rapaka:Welcome7%21@corp-eq5-proxy.mhc:8080"

export HTTPS\_PROXY=$HTTP\_PROXY

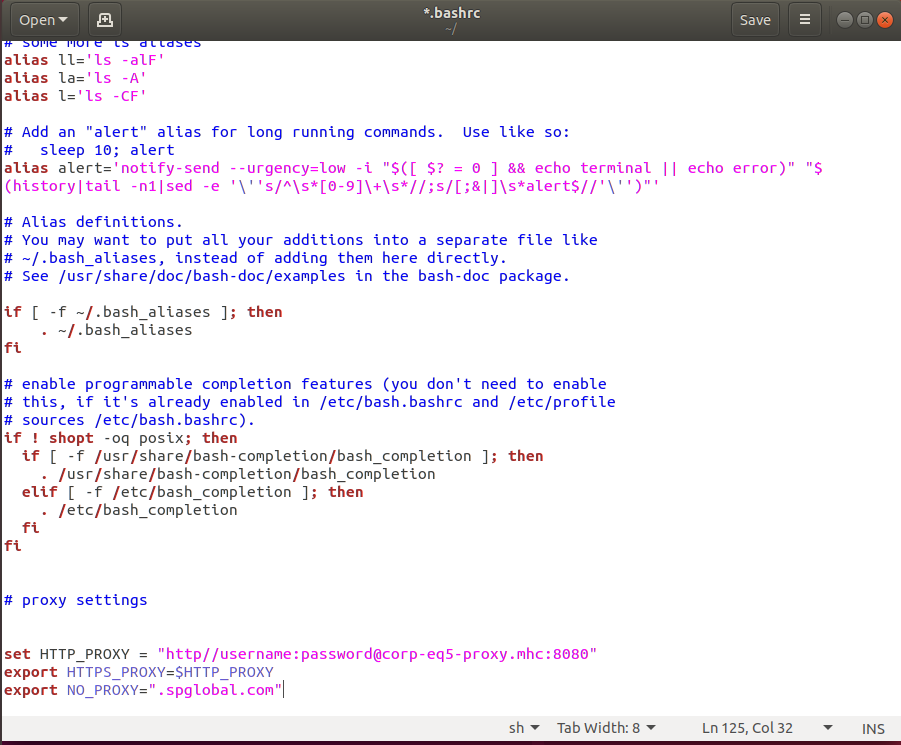
export NO\_PROXY=".spglobal.com"

Or try with this:

export HTTP\_PROXY=http:// username:password@sqpx.mge.spratingsvpc.com:10041

export HTTPS\_PROXY=$HTTP\_PROXY

export NO\_PROXY=".spglobal.com"



**Note:**

Username is your system username.

Password is your system password. If password is having any special characters replace those special characters with URL encoder characters

Ex: Hello3! 🡺 Hello3%21

Use this url : <https://www.w3schools.com/tags/ref_urlencode.ASP> to encode special characters

1. Save your settings.
2. The proxy settings will be applied the next time you start a session, by logging into the server or opening a new Terminal window from a Desktop.
3. To force apply your new proxy settings in the current Terminal session, execute the source command against your bash profile.

Type “source .bashrc” then click enter

## Install required softwares

Open Terminal & Type below commands

sudo apt update

sudo apt install docker.io

sudo apt install yum

sudo apt install python3

sudo apt install python3-pip

sudo apt install curl

sudo apt install npm

sudo apt install -y dkms build-essential

### Install aws software

pip3 install awscli

### Install k8s

**Using snap to install microk8s**

snap install microk8s --classic --- connect to vpn(no need of vpn if installing from home)

Install full k8s using kadm instructions from kubernetes site.



### Install serverless

sudo snap install --edge node --classic

sudo npm install npm@latest

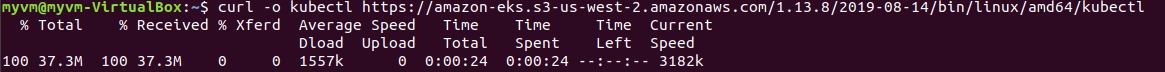
sudo npm install -g serverless

sudo npm install snap

### Install aws kubectl

Refer this URL <https://kubernetes.io/docs/tasks/tools/install-kubectl/>

curl -o kubectl <https://amazon-eks.s3-us-west-2.amazonaws.com/1.13.8/2019-08-14/bin/linux/amd64/kubectl>



chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version –client



Copy STS.py file to home

sudo apt-get install python3-lxml

 sudo apt-get install python3-boto3

sudo apt install snapd

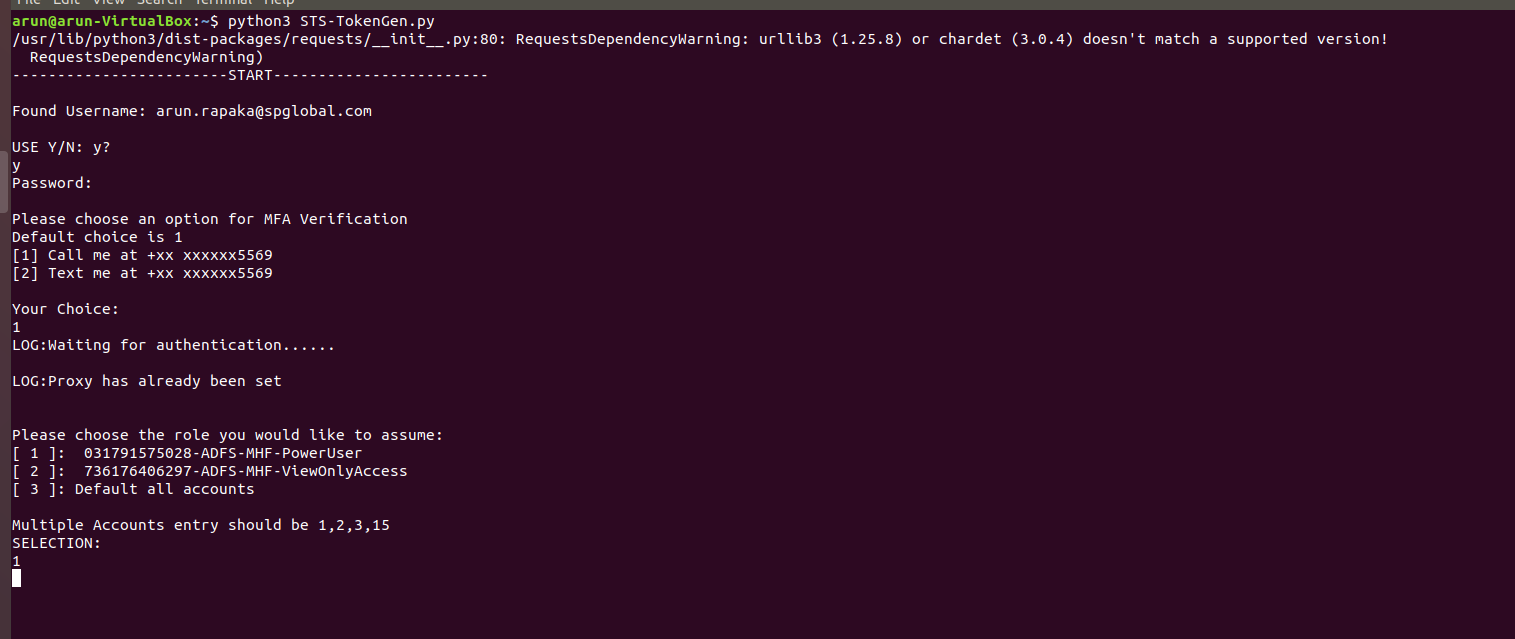
sudo snap install aws-cli --classic

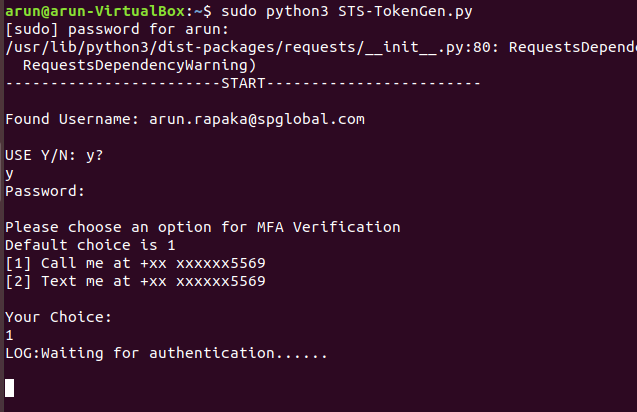
sudo python3 STS-TokenGen.py

in the username.txt enter name [arun.rapaka@spglobal.com](mailto:arun.rapaka@spglobal.com)

sudo gedit /etc/resolv.conf

connect to vpn





sudo apt install snapd

sudo snap install aws-cli --classic

pip3 install awscli

sudo bash

python3 STS

export AWS\_PROFILE=031791575028-ADFS-MHF-PowerUser

aws s3 ls --profile 031791575028-ADFS-MHF-PowerUser

aws eks --region us-east-1 update-kubeconfig --name isg-eks-poc-cluster

set nameserver in gedit /etc/resolv.conf

docker login docker.msbx.spratingsvpc.com

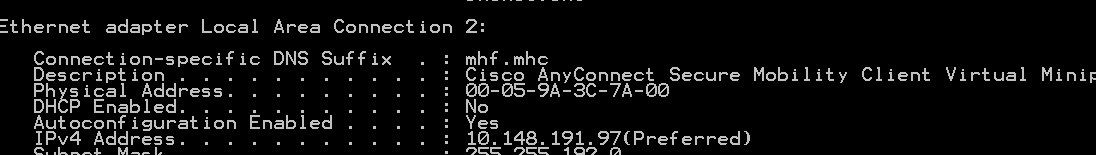
sudo gedit /etc/resolv.conf

nameserver 127.0.0.53

options edns0

search mhf.mhc

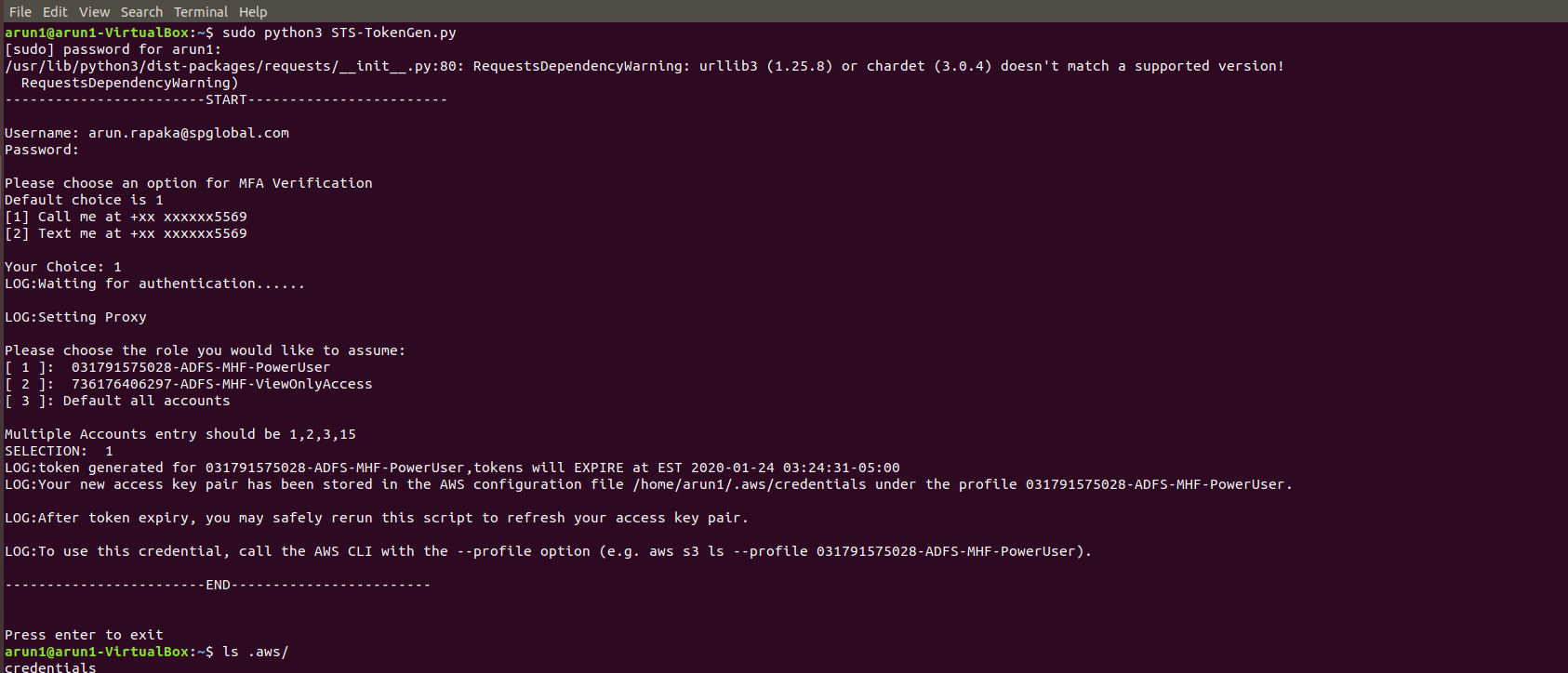
nameserver 10.148.191.97

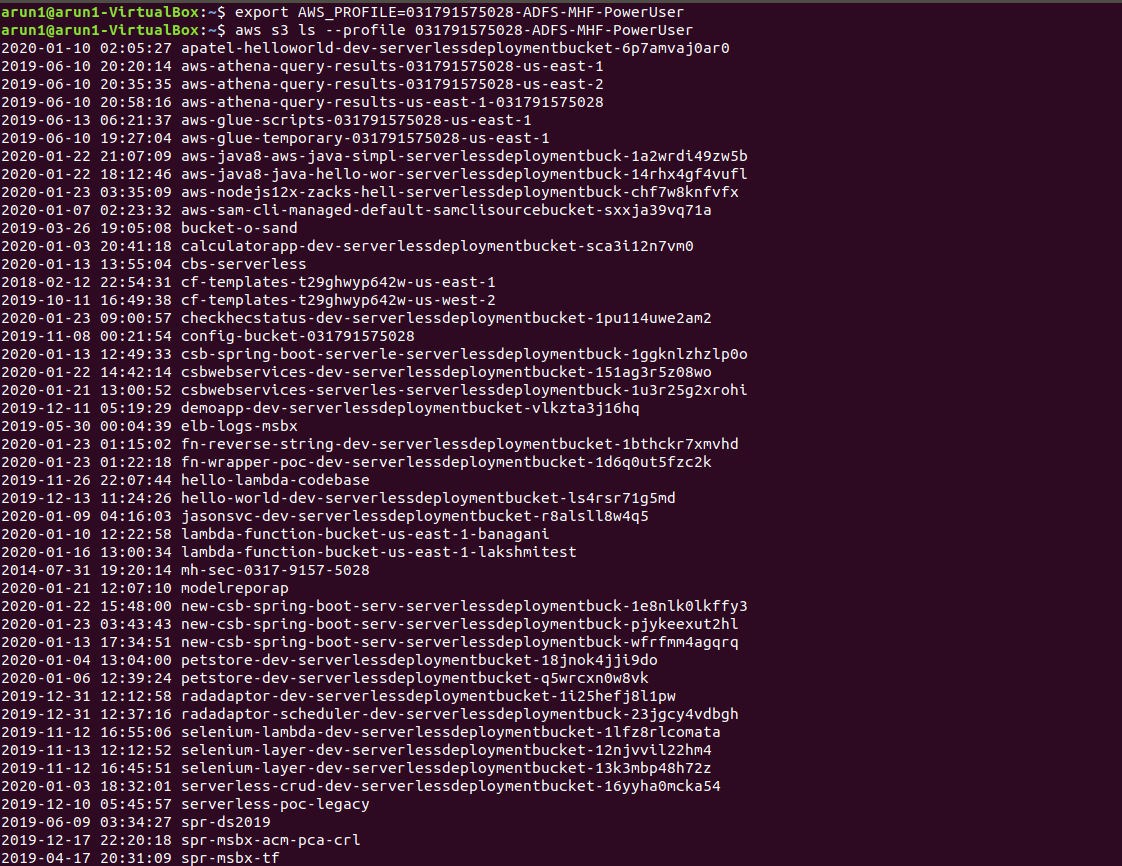


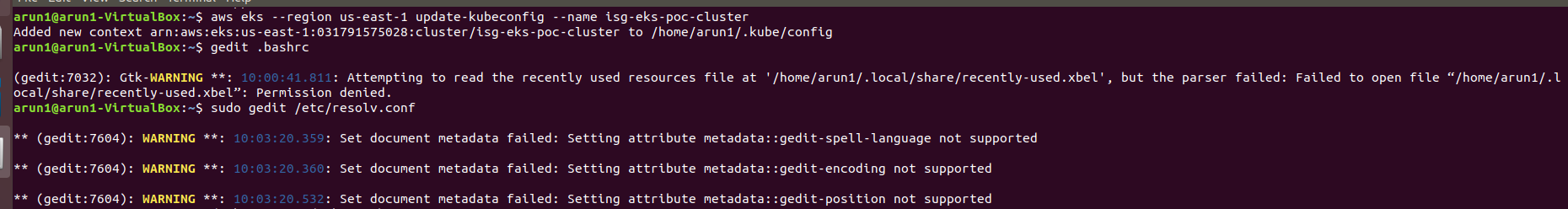
Sudo bash

Id

Td







kubectl apply -f [https://raw.githubusercontent.com/google/metallb/v0.7.3/manifests/meta llb.yaml](https://raw.githubusercontent.com/google/metallb/v0.7.3/manifests/meta%20llb.yaml)

kubectl apply -f <https://raw.githubusercontent.com/google/metallb/v0.7.3/manifests/metallb.yaml>

1. docker login docker.msbx.spratingsvpc.com
2. Build & Push
3. cd /media/sf\_VMshared/workspace/ratingsMS/ 🡨 this is my share folder
4. build the docker image, args passed in b/c my docker file asks for JAR\_FILE as an ARG

cd /media/sf\_VMshared/workspace/ratingsMS/getscenarioid/

docker build . -t docker.msbx.spratingsvpc.com/isg/knativegetscenarioid:v1 --build-arg JAR\_FILE=build/target/getscenarioid.jar ­

docker build . -t docker.msbx.spratingsvpc.com/isg/arunhelloworld:v1 --build-arg JAR\_FILE=build/target/arunhelloworld.jar

1. Optional: test your container in the vm

docker run docker.msbx.spratingsvpc.com/isg/knativegetscenarioid:v1 –p 8080:8080

docker run docker.msbx.spratingsvpc.com/isg/arunhelloworld:v1 –p 8080:8080

1. push the image to our private repo

*docker push docker.msbx.spratingsvpc.com/isg/*knativegetscenarioid*:v1*

*docker push docker.msbx.spratingsvpc.com/isg/*arunhelloworld*:v1*

1. run the deploy command

kubectl apply -f /media/sf\_VMshared/workspace/ratingsMS/getscenarioid/knative.yaml

kubectl apply -f /media/sf\_VMshared/workspace/ratingsMS/arunhelloworld/knative.yaml

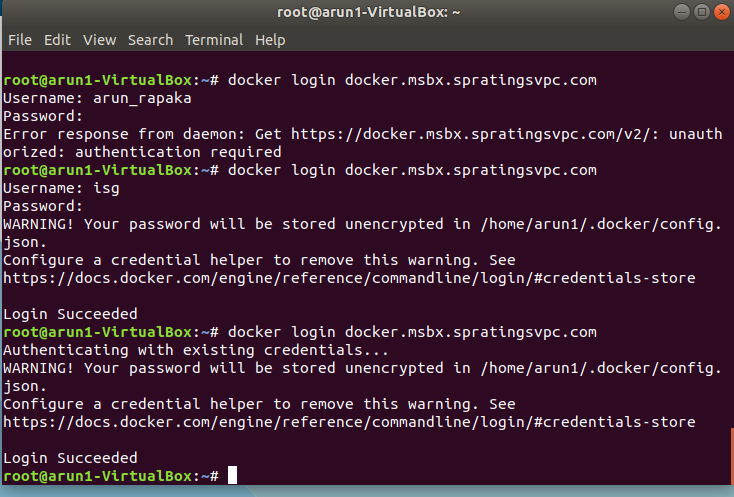
1. now check with these commands:

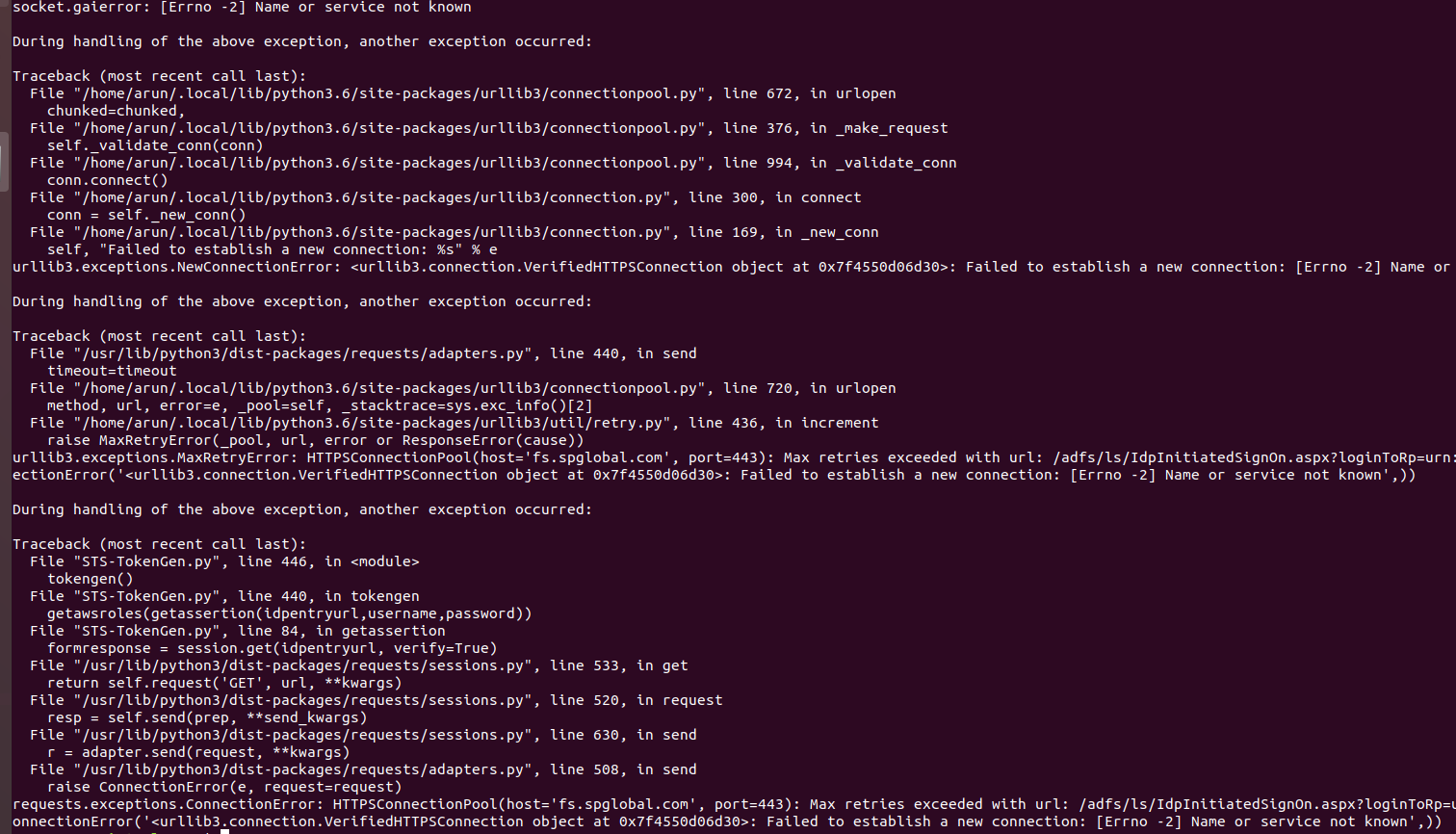
kubectl get ksvc 🡨 this will list all knative services

kubectl get pods 🡨 see if your pods are ready

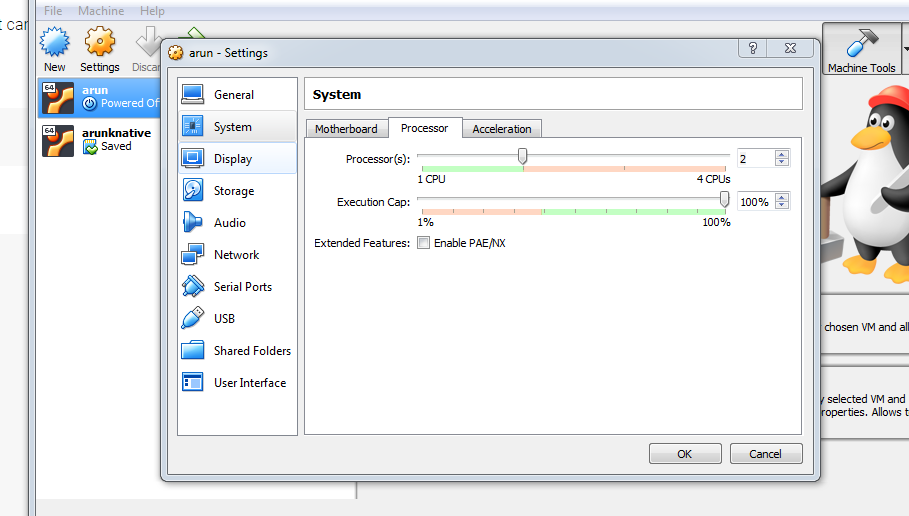
kubectl get ksvc hello 🡨 this will print out the url to your svc.

In browser docker.msbx.spratingsvpc.com





If you get the above error then restart the instance



 snap remove microk8s

microk8s removed. If already installed.

 kubectl get nodes

error: the server doesn't have a resource type "nodes"

 sudo snap install kubectl --classic

kubectl 1.17.2 from Canonical\* installed

 curl -O https://storage.googleapis.com/minikube/releases/latest/minikube\_1.6.2.deb

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 14.5M 100 14.5M 0 0 145k 0 0:01:42 0:01:42 --:--:-- 315k

 sudo dpkg -i minikube\_1.6.2.deb



 sudo minikube start --vm-driver=none

 kubectl create deployment hello-node --image=docker.msbx.spratingsvpc.com/isg/knativegetscenarioid:v1

kubectl create deployment hello-node --image=docker.msbx.spratingsvpc.com/isg/arunhelloworld:v1

error: unable to read client-key /home/knative/.minikube/client.key for minikube due to open /home/knative/.minikube/client.key: permission denied

 sudo chown -R $USER $HOME/.kube $HOME/.minikube

docker login docker.msbx.spratingsvpc.com

 kubectl create deployment knative-getscenarioid --image=docker.msbx.spratingsvpc.com/isg/knativegetscenarioid:v1

kubectl create deployment arunhelloworld --image=docker.msbx.spratingsvpc.com/isg/arunhelloworld:v1

deployment.apps/mahesh-hello created

 kubectl expose deployment knative-getscenarioid --type=NodePort --port=8080

kubectl expose deployment arunhelloworld --type=NodePort --port=8080

service/mahesh-hello exposed

 minikube service knative-getscenarioid

minikube service arunhelloworld

|-----------|--------------|-------------|------------------------|

| NAMESPACE | NAME | TARGET PORT | URL |

|-----------|--------------|-------------|------------------------|

| default | mahesh-hello | | http://10.0.2.15:31622 |

|-----------|--------------|-------------|------------------------|

Opening service default/knative-getscenarioid in default browser...