**Create Custom Helm Chart**

**What is Helm?**

Helm is a **package manager for Kubernetes**. It simplifies the deployment and management of applications by using **Helm Charts**, which are reusable templates that define Kubernetes resources (e.g., deployments, services, and configurations).

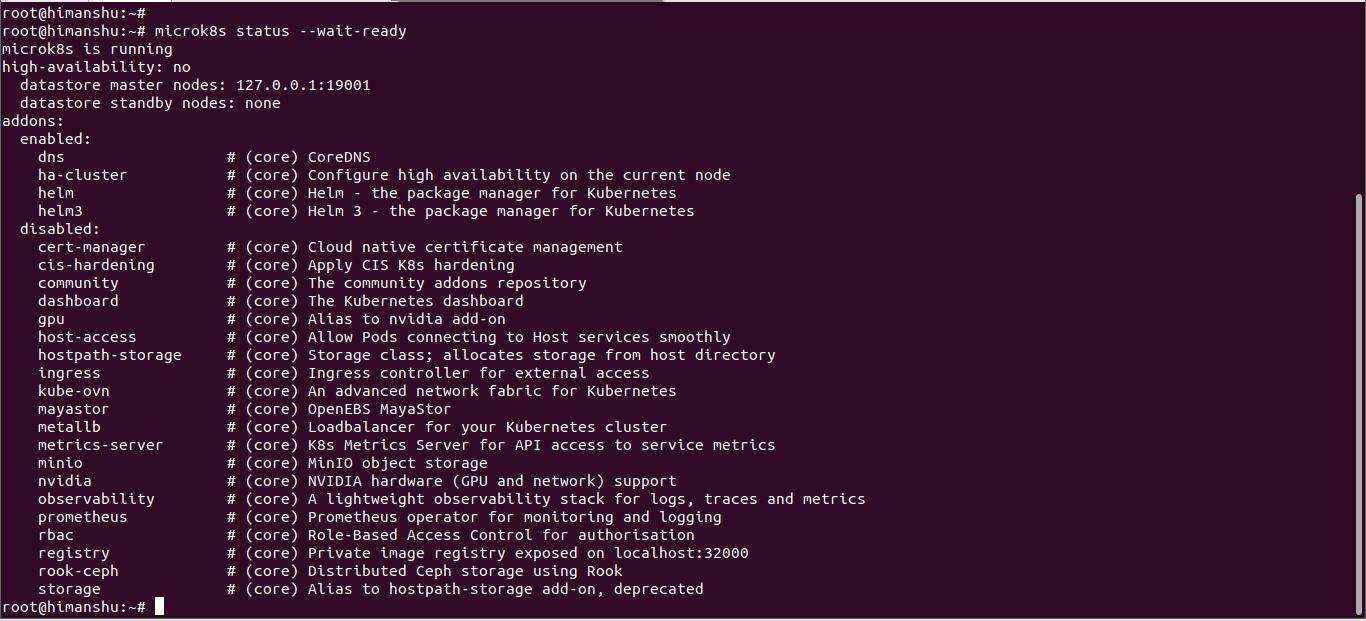
**In this document we see how to create our own helm chart.**

Install MicroK8s on Linux

sudo snap install microk8s --classic

To check

microk8s status --wait-ready



For installing kubectl

If curl is not installed the install it by using command

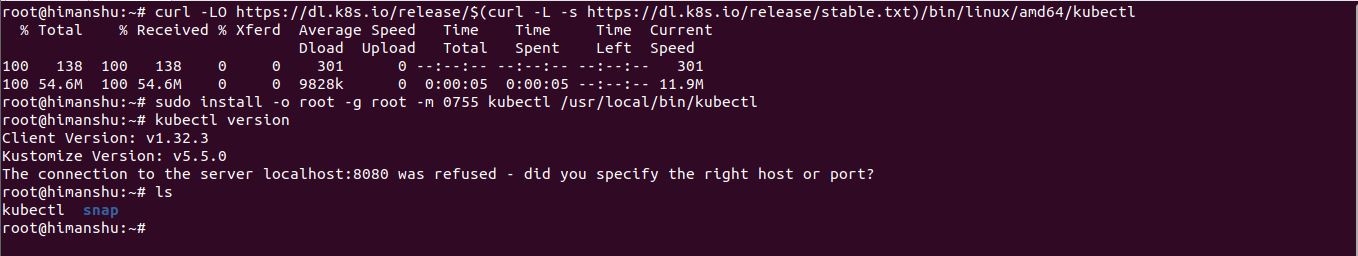
“snap install curl” or “apt install curl”

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

**After installation of kubectl we can check it using by below command**

kubectl version



After this we have to configure kubectl with microk8s.

cd $HOME

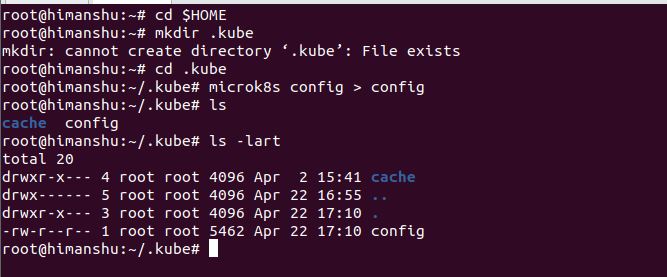
mkdir .kube

cd .kube

microk8s config > config

**To see all file run command**

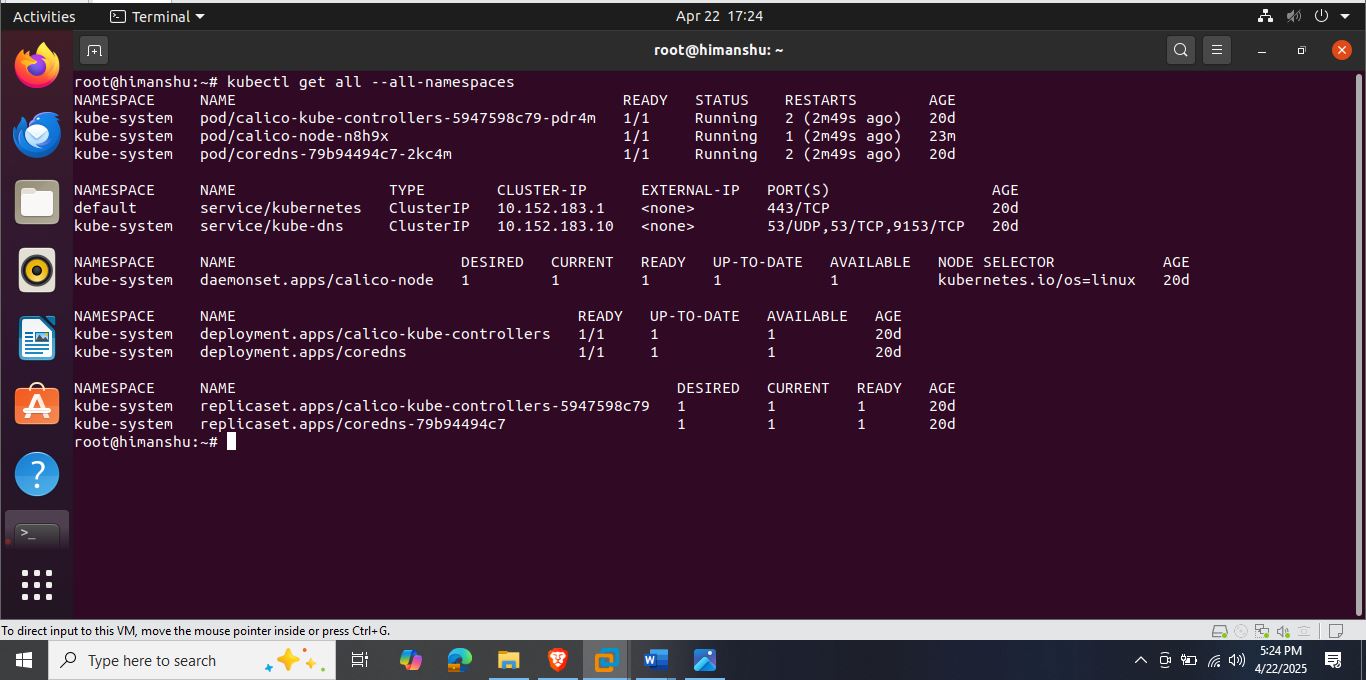
ls -lart



After this restart the system using below command so it wil config all the things.

“sudo reboot” or “ sudo shutdown -r now”

kubectl get all --all-namespaces



**Now for installing the helm use below command.**

curl -O <https://get.helm.sh/helm-v3.16.2-linux-amd64.tar.gz>

tar xvf helm-v3.16.2-linux-amd64.tar.gz

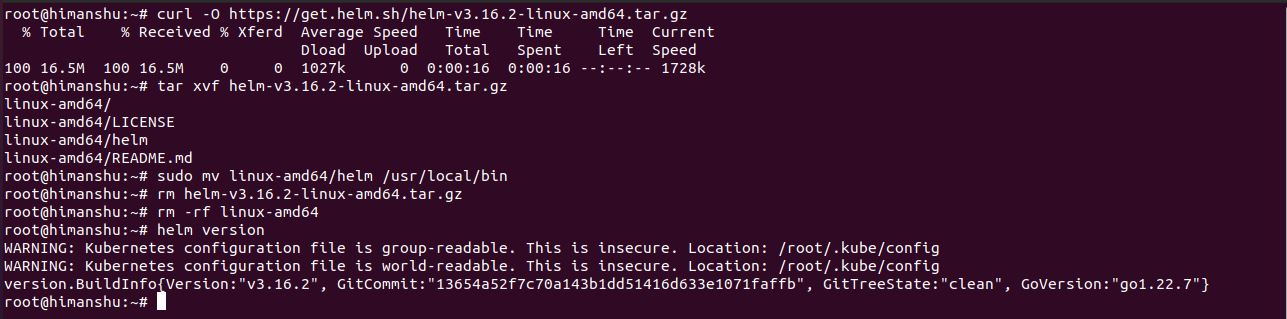
sudo mv linux-amd64/helm /usr/local/bin

rm helm-v3.16.2-linux-amd64.tar.gz

rm -rf linux-amd64

**To check version of helm run below command.**

helm version



**Now our next target is to install git and docker for that use below command.**

sudo apt update

sudo apt install git -y

git –version

**For docker installation**

**Step 1: Update Software Repositories using the following command on the terminal.**

sudo apt update

**Step 2:** Install Docker using the following command

sudo apt install docker.io -y

**Step 3:** Enable and start the docker service by using the following commands.

sudo systemctl enable docker –now

**Step 4:** Check Docker Version.

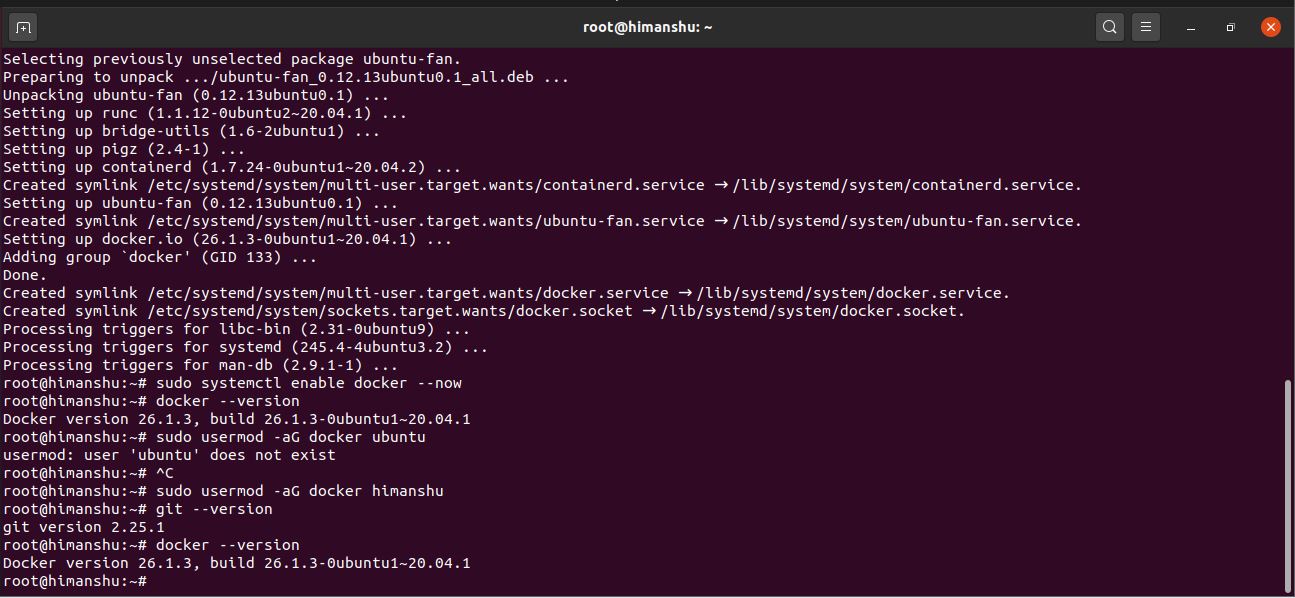
docker –version

**Executing the Docker Command Without Sudo**

We will get a permission denied error as a regular user doesn’t have permission to execute [docker commands](https://www.geeksforgeeks.org/docker-instruction-commands). Now we need to add the user to the required group.

So we need to add an [Ubuntu](https://www.geeksforgeeks.org/how-to-install-ubuntu-on-virtualbox)user to the docker group.

sudo usermod -aG docker $USER



**After that we have to clone our github repository by using the below command**.

git clone <https://github.com/himanshu8987/Python-flask-rest-api-proj.git>

**After that we have to build Docker image**

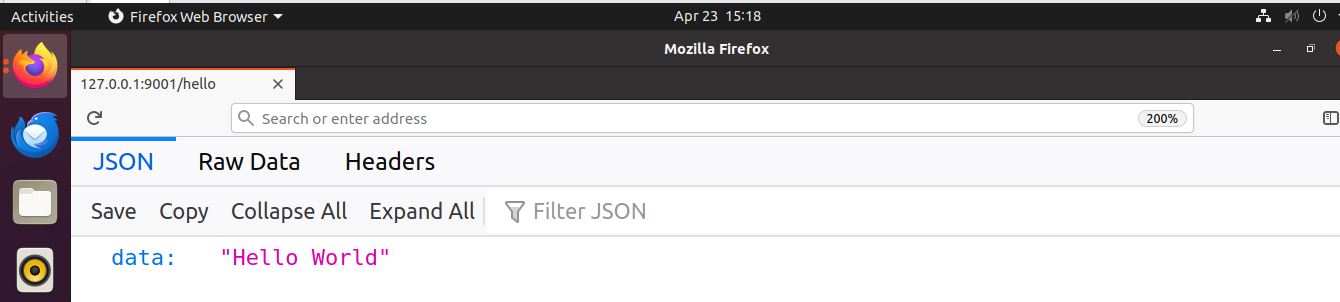
docker build -t python-proj

To list all Docker images on your system, just run:

docker images or docker image ls

**Now before we push our docker image to dockerhub repository we check our image by running it using below command.**

docker run -p 9001:9001 python-proj



So we can see our docker container is up and running. And our next task is push the image to dockerhub repository for that we have to create a repository in dockerhub using web browser and after that login into our dockerhub repository by using command line using below command.

docker login

After that it will ask credential.

A computer screen with white text

AI-generated content may be incorrect.

**Now we have to build and tag according to dockerhub**.

docker tag python-proj himanshu8987/python-flask-rest-api-proj:python-proj

**After that we have to push the image on the dockerhub repository by using command**.

docker push himanshu8987/python-flask-rest-api-proj:python-proj

A computer screen with white text

AI-generated content may be incorrect.

We have to refresh the dockerhub page then we can see that our image is pushed.

A screenshot of a computer

AI-generated content may be incorrect.

**Now we have to start working on helm chart creation for that first we will create a helm chart and then we will update the repository url of this particular docker image so our helm chart can access the particular repository.**

helm create python-flask-rest-api-proj

A screenshot of a computer program

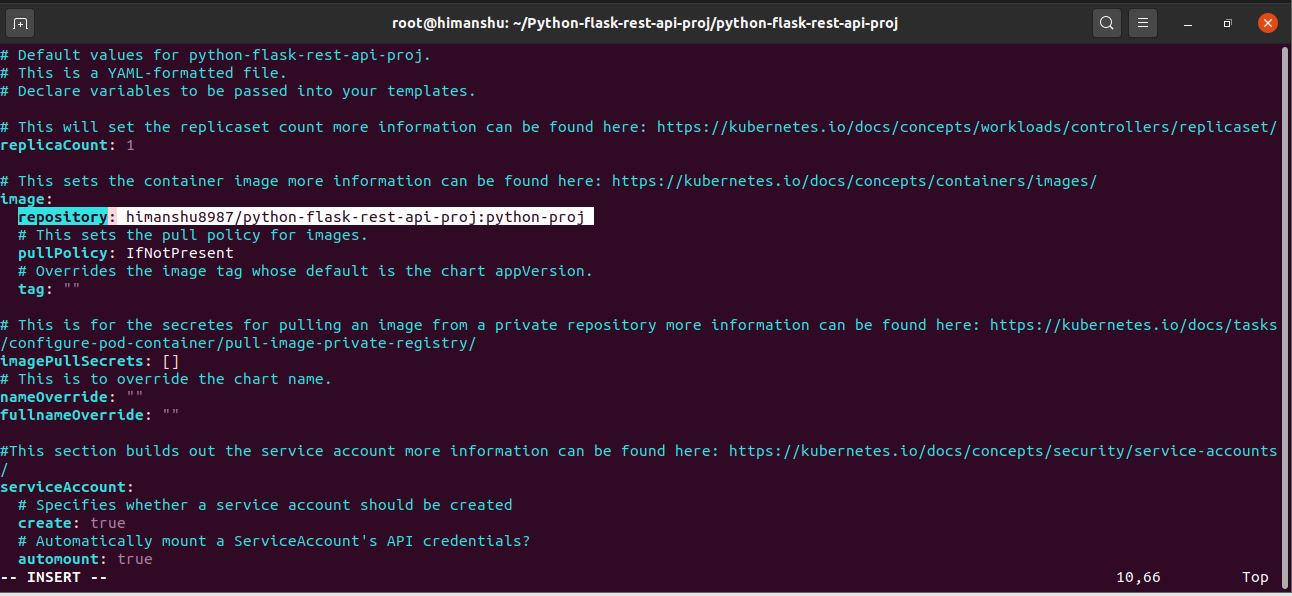
AI-generated content may be incorrect.

Now we have to open Chart.yaml and comment the appversion because we are not using any app.

A screenshot of a computer program

AI-generated content may be incorrect.

Now we have to update the repository in values.yaml.



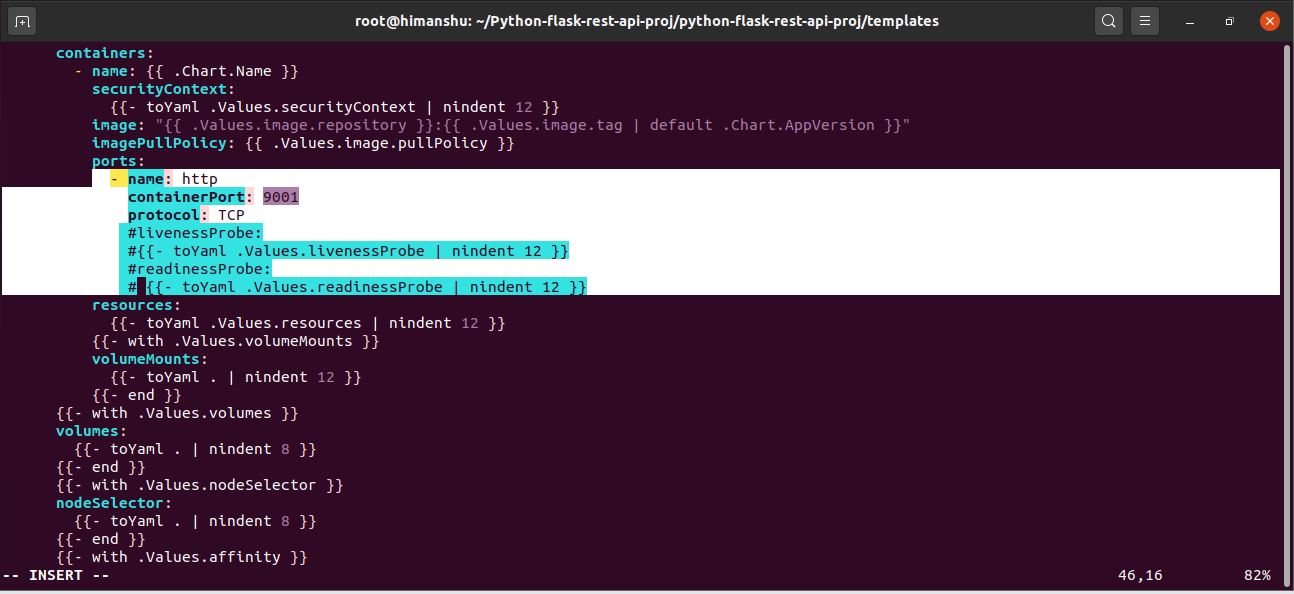
Now we have to specify the service type so that Kubernetes know which kind of service we are expecting from this particular docker image.

For that we have to update in values.yaml and this time we are going to use NodePort so that we can access that particular REST API within our Kubernetes cluster.

A computer screen with text

AI-generated content may be incorrect.

Now we have to update the port number and comment or delete the livenessProbe and redinessProbe and also delete the AppVersion in deployment.yaml.



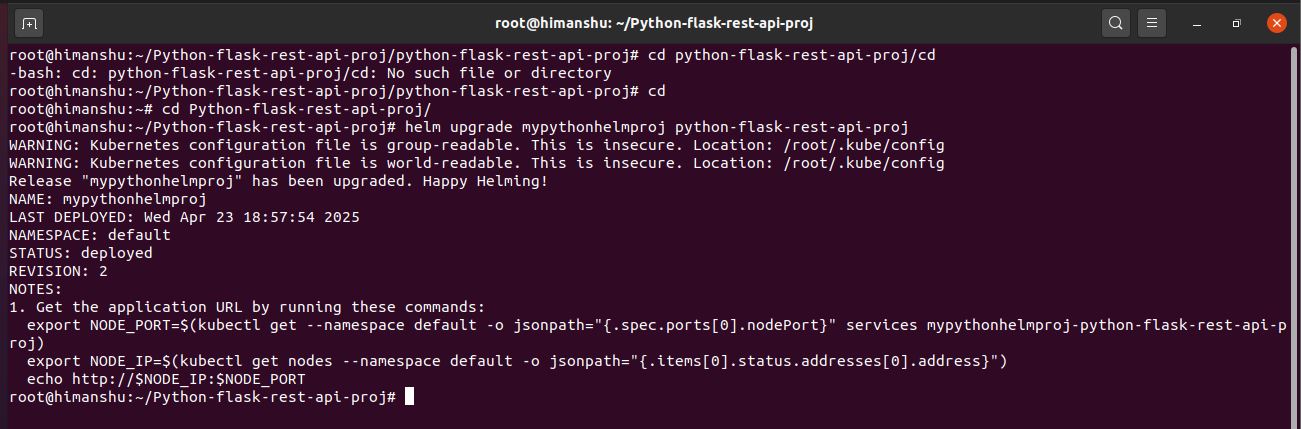
A screenshot of a computer

AI-generated content may be incorrect.

**Now our helm chart is ready and now we can run helm install command and verify that our hello world application is running or not.**

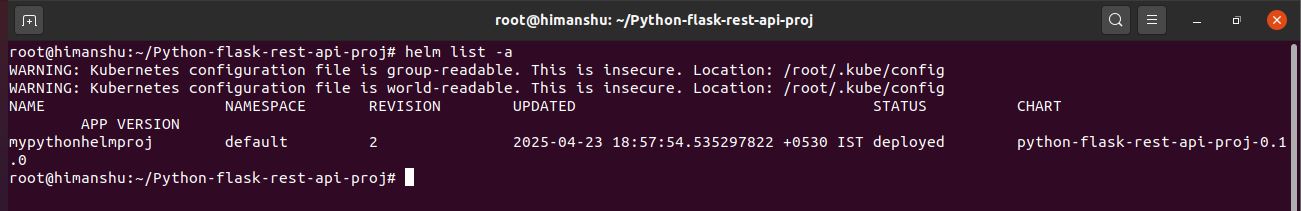
helm install mypythonhelmproj python-flask-rest-api-proj

**And we can see in below snap it shows our chart is deployed in below snap I used “helm upgrade mypythonhelmproj python-flask-rest-api-proj” because i have given wrong repository name that’s why I used upgrade command here.**

****

**Now the next thing we go to verify our deployment and access our hello world endpoint from our Kubernetes cluster before that we check our helm chart list.**

helm list -a # it will show the list of chart

**Now we are going to verify kubectl deployment check by using command**

kubectl get deployments

**And also we have to verify on which port it is running for that we use below command**

kubectl get service

A screen shot of a computer

AI-generated content may be incorrect.

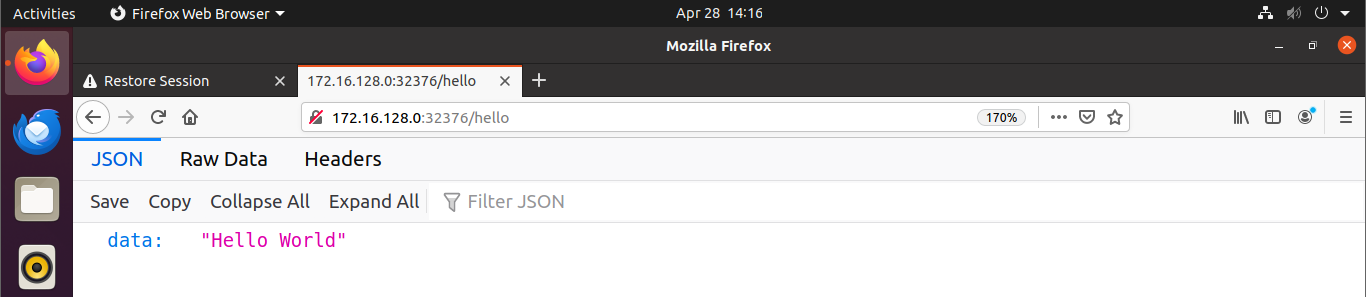
We can see this deployment “mypythonhelmproj-python-flask-rest-api-proj” running on my Kubernetes cluster and service is NodePort we can access it via our virtual machine ip with port number 32376

For ip of our virtual machine we use command “hostname -I” then it will show the ip.

Like example:

root@himanshu:~# hostname -I

172.16.128.0 172.17.0.1 10.1.136.128



We can see in above snapshot we successfully run our application from Kubernetes cluster service.