**Docker course-end project:-**

**Create a Docker Image and Deploy It to Kubernetes**

**Description**

You are working as a DevOps engineer in an IT firm. You have been asked to create a Redis-based Docker image and deploy it on a Kubernetes cluster.

**Background of the problem statement:**

**Your organization wants to use Redis in a Kubernetes cluster for the data storage and caching purpose. The development team has asked you to create a Redis-based Docker image using a Dockerfile and deploy this image on a Kubernetes cluster.**

**You have also been asked to publish this image on your organization's Docker Hub account so that other team members can also access this image**.

**You must use the following:**

* Docker CLI: To create the Docker image using a Dockerfile
* Docker Hub : To publish the image
* Kubectl : To deploy the image on Kubernetes cluster

**Following requirements should be met:**

* Follow the above-mentioned specifications
* Make sure you create an account on Docker Hub to push the Docker image
* Document the step-by-step process involved in completing this task.

**Requirements**

* Ubuntu Machine
* Dockerhub account
* Minikube

**Project made by** - **Chaitra Boregowda**

**Updating Ubuntu Machine**

**sudo apt update -y**

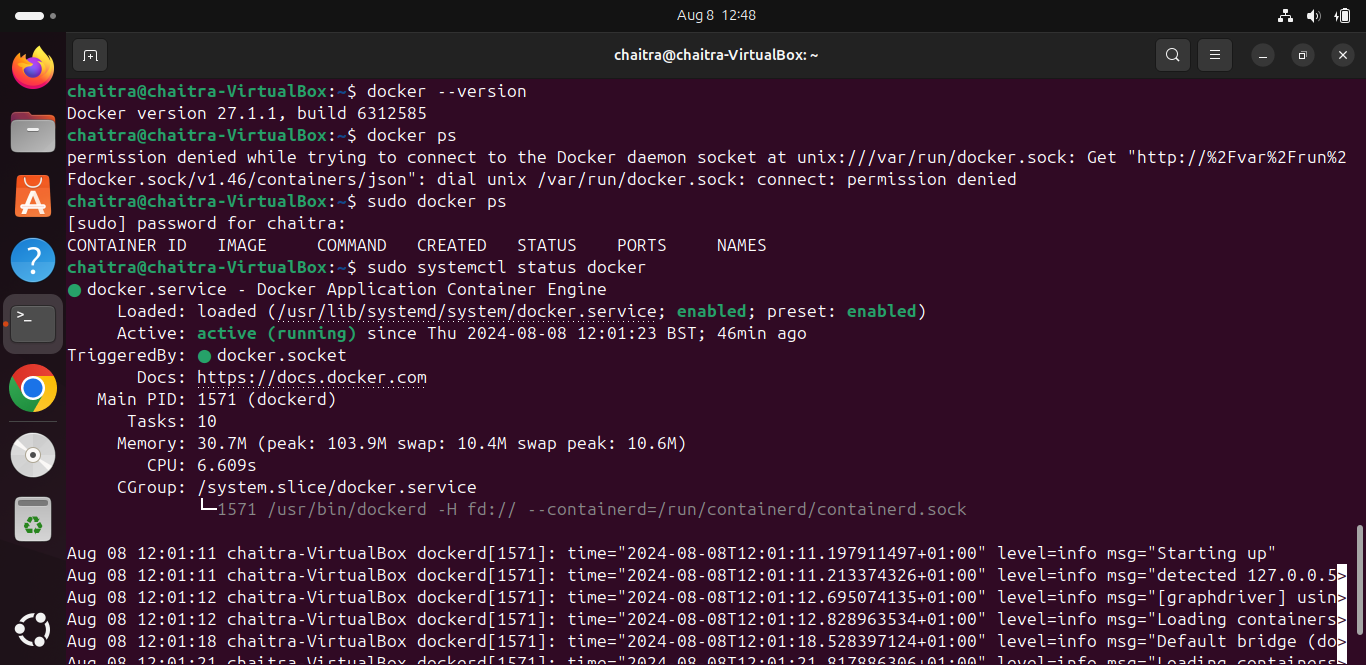
**Installing Docker**

**sudo apt install docker.io -y**

**Check the docker version and the status of the docker**

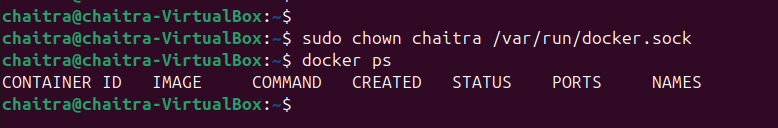
**docker --version**

**sudo systemctl status docker**



**Assigning permissions to docker**

**sudo chown chaitra /var/run/docker.sock**



**Docker Image Creation**

**Step1 - Dockerfile Creation**

* Create a Dockerfile specifying the instructions for building the Redis Docker image.
* Include necessary dependencies, configurations, and commands to set up Redis within the container.

Vi Dockerfile

**DockerFile**

**# Use the official Redis image as the base**

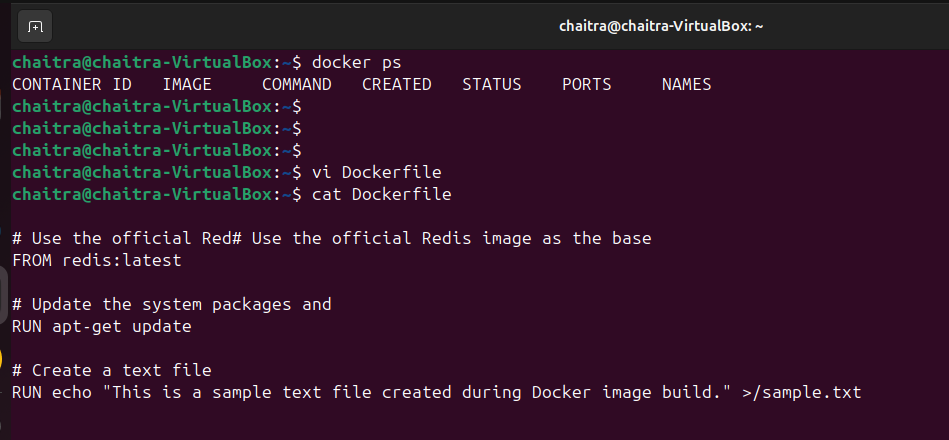
**FROM redis:latest**

**# Update the system packages and**

**RUN apt-get update**

**# Create a text file**

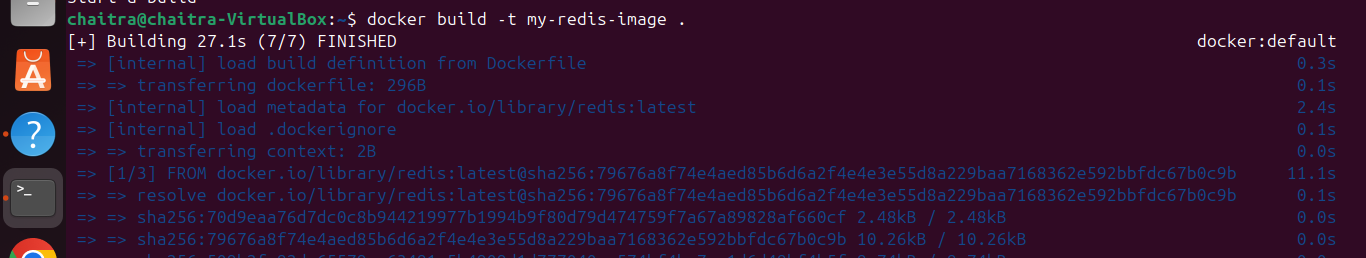
**RUN echo "This is a sample text file created during Docker image build." >/sample.txt**

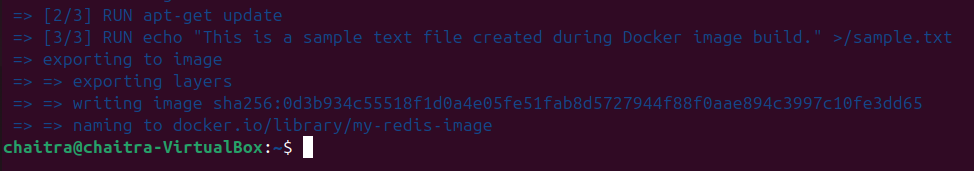


**Step 2 - Build Docker Image**

* Use the Docker CLI to build the Docker image using the created Dockerfile
* Execute the following command in the directory containing the Dockerfile

docker build -t my-redis-image .





**Check for the redis docker image**

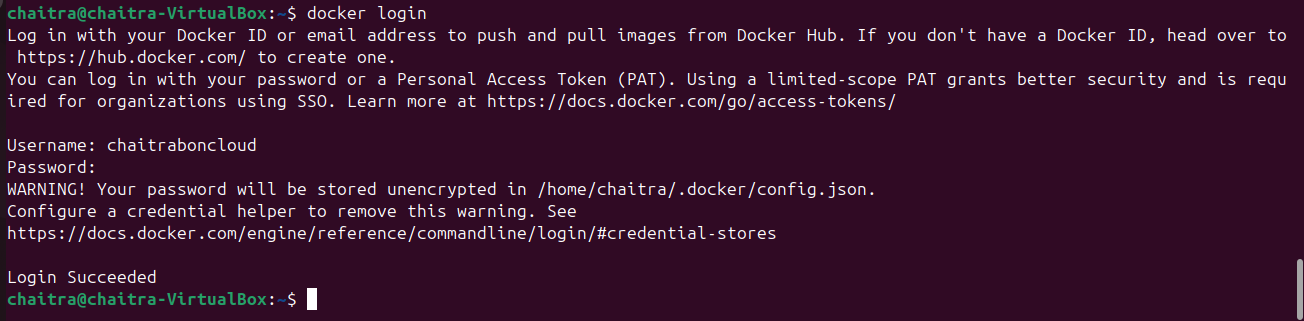
****

**Publishing Docker Image to Docker Hub**

**Step1- Docker Hub Account Setup**

* Create an account on Docker Hub if not already done.
* Log in to the Docker Hub account using the Docker CLI.

**docker login**

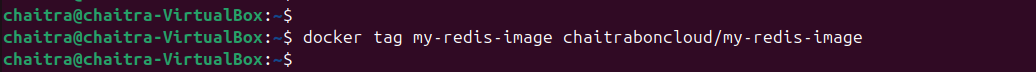


* Provide your Docker Hub credentials when prompted.

**Step2 - Tag Docker Image**

* Tag the locally built Docker image with your Docker Hub username and repository name:

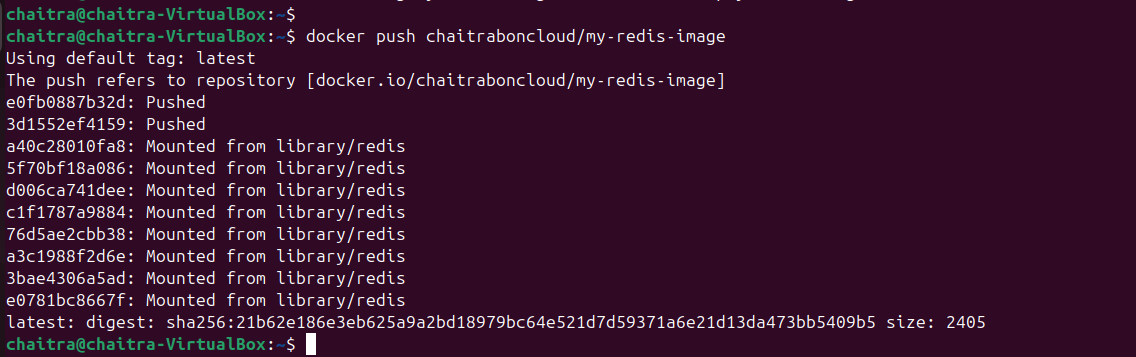
**docker tag my-redis-image chaitraboncloud/my-redis-image**

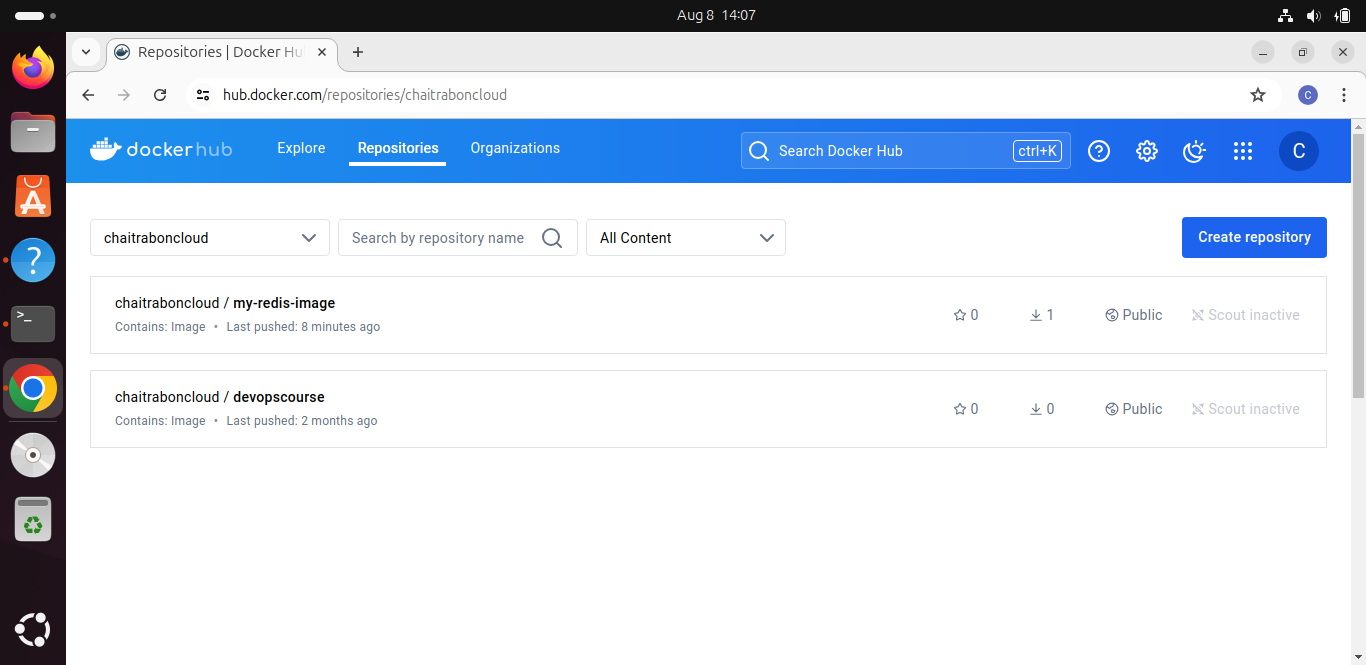


**Step 3- Push Docker Image to Docker Hub**

* Push the tagged Docker image to Docker Hub

**docker push chaitraboncloud/my-redis-image**

****

****

**Minikube for Kubernetes Cluster Setup**

I am installing minikube for Kubernetes cluster setup you can go to its official documentation and setup minikube.

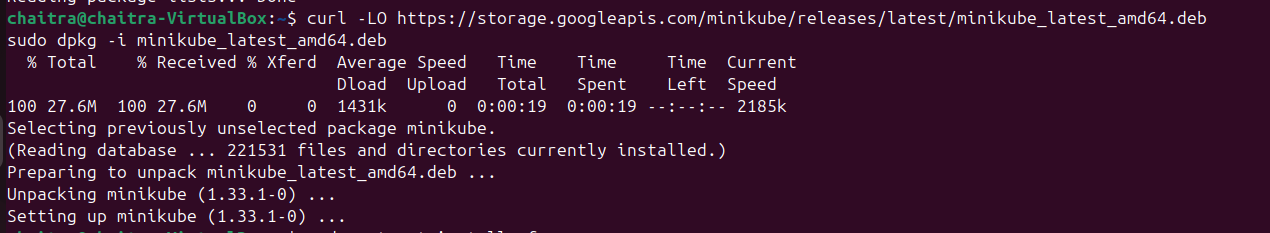
**Step1 : Update your system**

**sudo apt-get update**

**Step2 : Install the minikube**

**curl -LO <https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb>**

**sudo dpkg -i minikube\_latest\_amd64.deb**



**Step3 : Start Minikube with Docker Driver**

Since Docker is already installed, you can use it as the driver for Minikube:

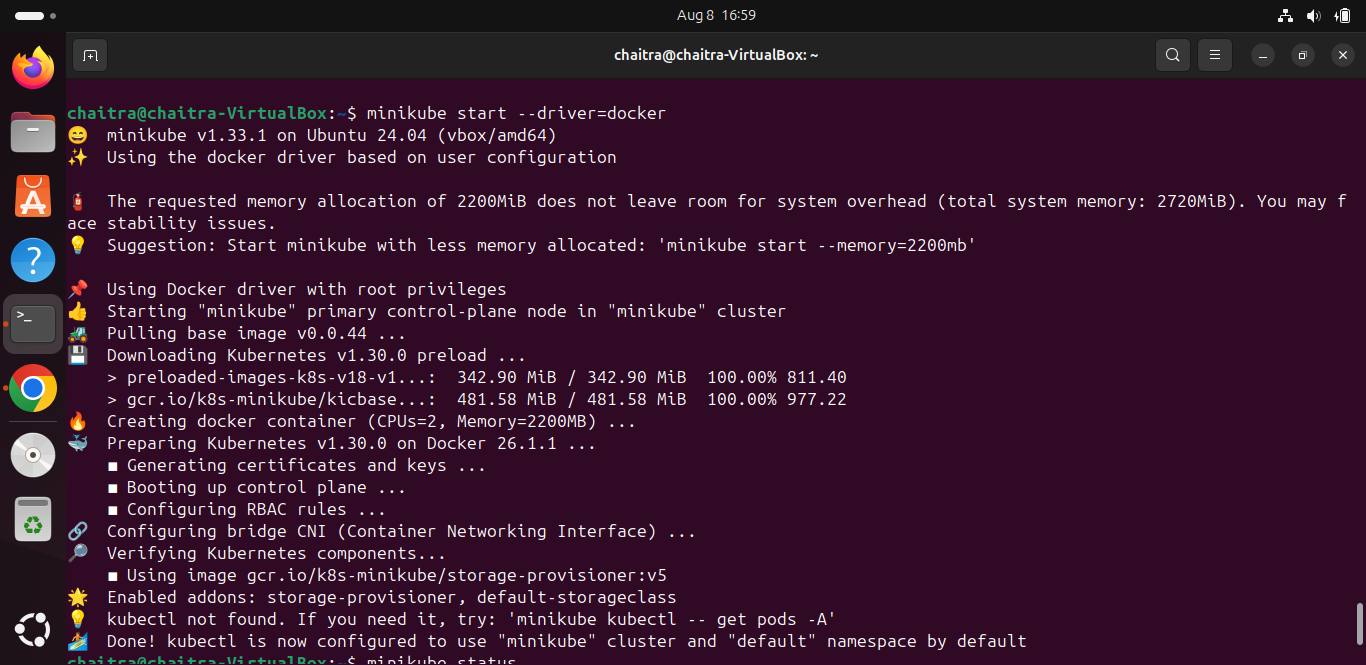
**minikube start --driver=docker**

This command tells Minikube to use Docker to create and manage the Kubernetes cluster.

**Step4 : Verify Minikube Installation**

Check the status of Minikube to ensure it’s running correctly:

**minikube status**

****

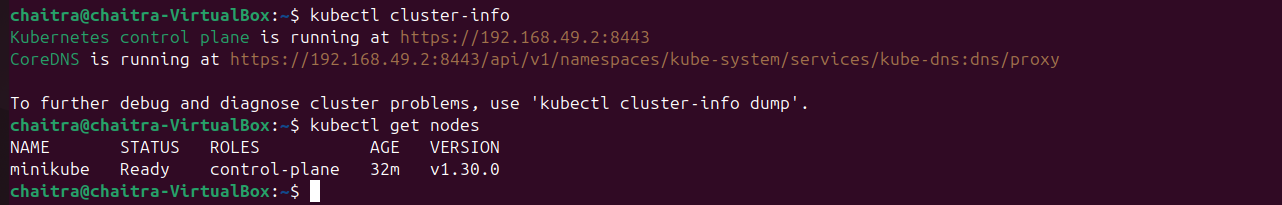
****

**Step5 : Check Kubernetes Cluster Information**

Use kubectl (which should be installed separately) to interact with your Kubernetes cluster:

**kubectl cluster-info**

**kubectl get nodes**

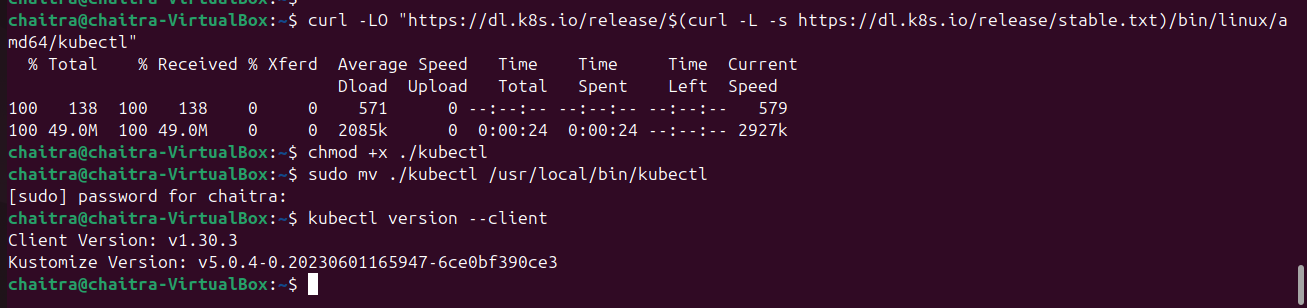


If you don't have kubectl installed, you can install it with the following commands:

**curl -LO "https://dl.k8s.io/release/$(curl -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"**

**chmod +x ./kubectl**

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

****

**Deploying Redis on Kubernetes**

**Step1- Kubernetes Cluster Setup**

* Ensure access to a Kubernetes cluster where Redis will be deployed.
* Configure kubectl to connect to the desired Kubernetes cluster.

**vi deployment.yml**

****

**deployment.yml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: redis-deployment**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**app: redis**

**template:**

**metadata:**

**labels:**

**app: redis**

**spec:**

**containers:**

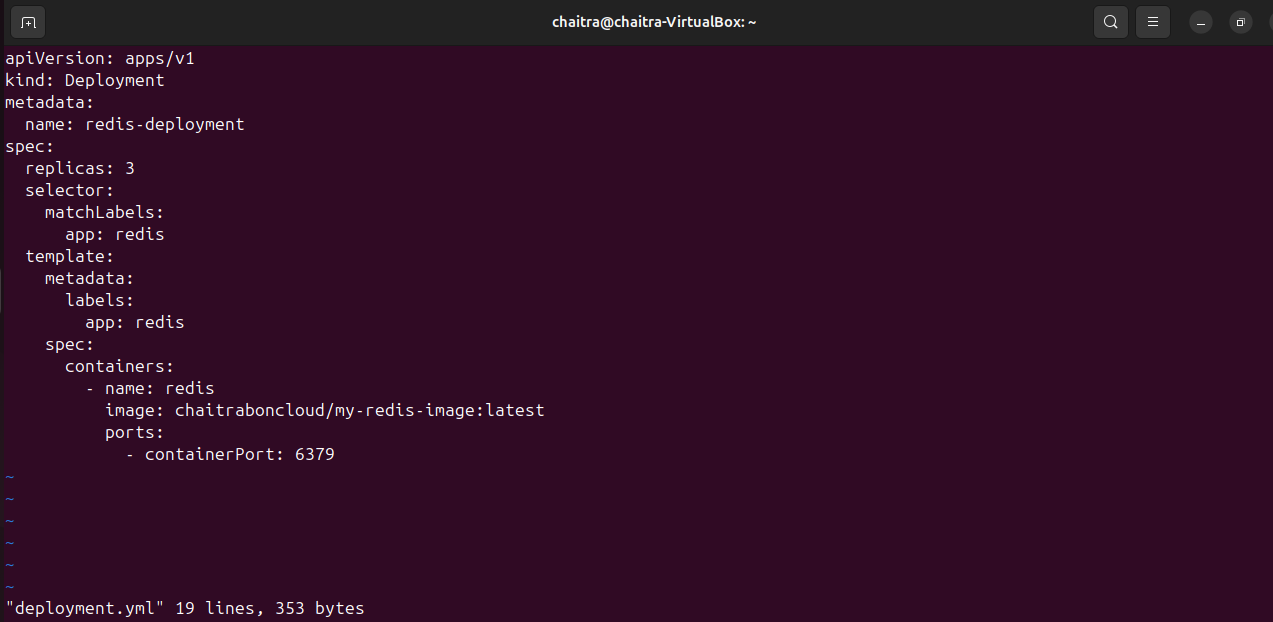
**- name: redis**

**image: chaitraboncloud/my-redis-image:latest**

**ports:**

**- containerPort: 6379**

* Create a Kubernetes Deployment YAML file specifying the deployment configuration for Redis

****

**Step2- Deploy Redis on Kubernetes**

* Apply the Deployment YAML to deploy Redis on the Kubernetes cluster

**kubectl apply -f redis-deployment.yml**

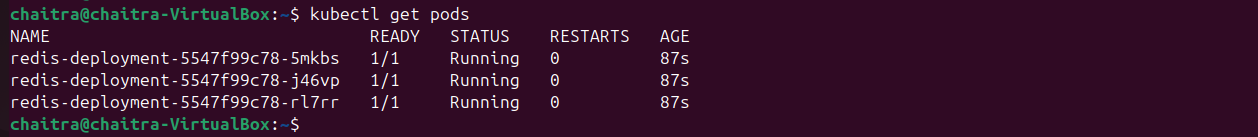
****

**PODS**

**Definition**: A pod is the smallest and simplest unit in the Kubernetes object model that you can create or deploy. It represents a single instance of a running process in your cluster and can contain one or more containers that share the same network and storage.

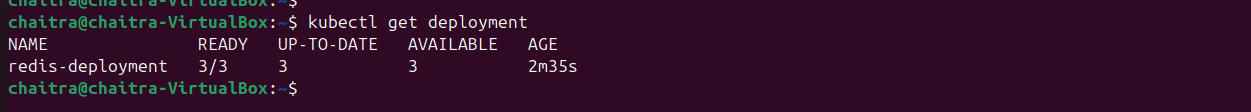
**Purpose:** A pod is the actual instance of your application running in Kubernetes. For example, in your Redis deployment, each pod would run a Redis container.

**kubectl get pods**



**Deployment Object**

**kubectl get deployment**

****

**Summary**

**Pulled**: Got the base Redis image from Docker Hub.

**Built and Tagged**: Created and tagged a customized version of this image.

**Pushed**: Uploaded this custom image to your Docker Hub account.

**Deployed**: Used Kubernetes to deploy the custom Redis image to a Kubernetes cluster.

**Conclusion**

By following the above steps, we will successfully create a Redis-based Docker image, publish it on Docker Hub, and deploy it on a Kubernetes cluster. This setup enables efficient data storage and caching within the Kubernetes environment, fulfilling the organization's requirements for enhancing application performance and scalability.