Redis Kubernetes Deployment: Docker Image Creation and Publishing

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 a 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

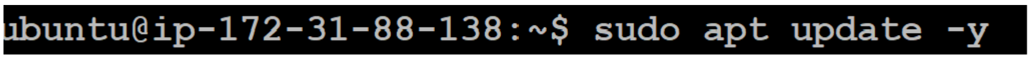
- Ubuntu Machine
- Dockerhub account
- Minikube

Project made by -

Prerana Maurya

Updating Ubuntu Machine

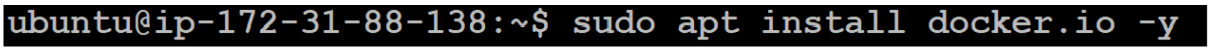
sudo apt update -y



```
ubuntu@ip-172-31-88-138:~$ sudo apt update -y
```

Installing Docker

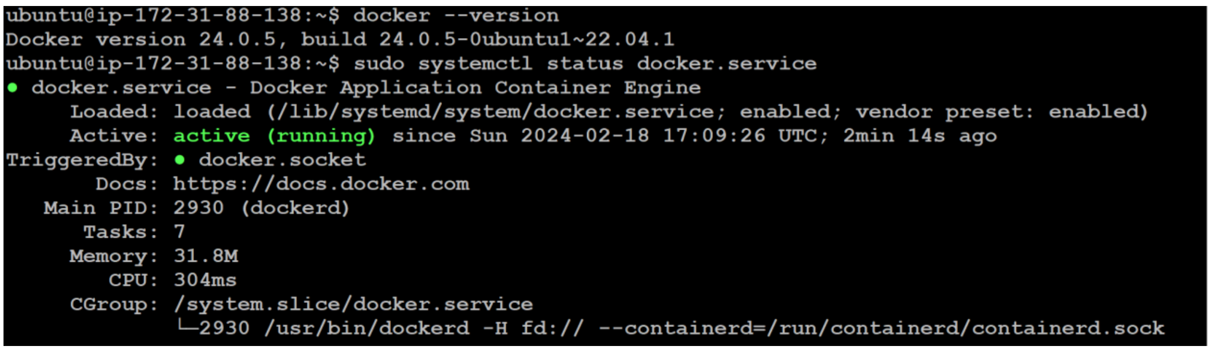
sudo apt install docker.io -y



```
ubuntu@ip-172-31-88-138:~$ sudo apt install docker.io -y
```

docker –version

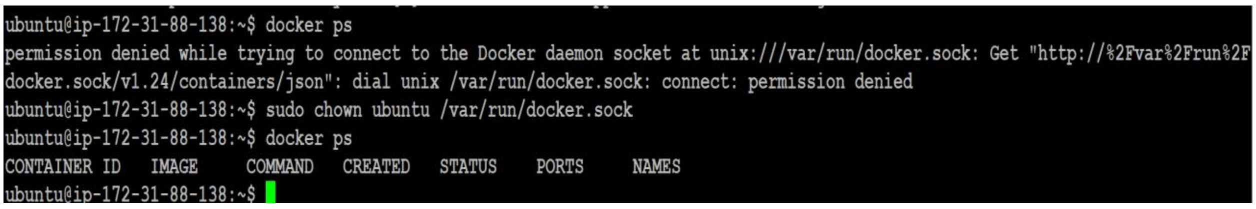
sudo systemctl status docker.service



```
ubuntu@ip-172-31-88-138:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
ubuntu@ip-172-31-88-138:~$ sudo systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-02-18 17:09:26 UTC; 2min 14s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2930 (dockerd)
      Tasks: 7
     Memory: 31.8M
        CPU: 304ms
     CGroup: /system.slice/docker.service
             └─2930 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Assigning permissions to docker

sudo chown ubuntu /var/run/docker.sock



```
ubuntu@ip-172-31-88-138:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2F
docker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-88-138:~$ sudo chown ubuntu /var/run/docker.sock
ubuntu@ip-172-31-88-138:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-88-138:~$
```

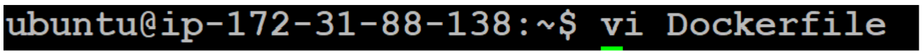
Docker Image Creation

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.

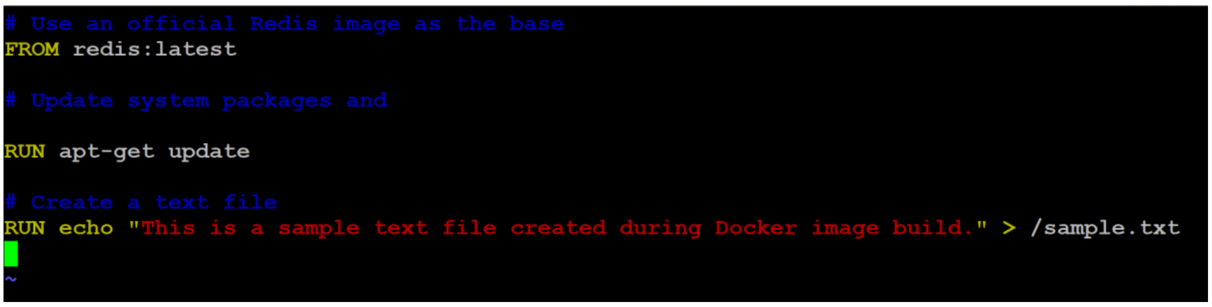
  Vi Dockerfile



```
ubuntu@ip-172-31-88-138:~$ vi Dockerfile
```

Dockerfile

```
# Use an official Redis image as the base
FROM redis:latest


# Update 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
```

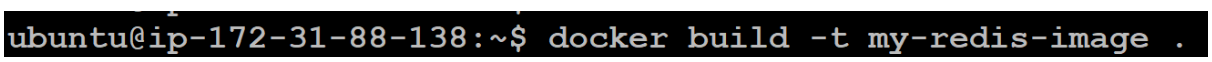


```
# Use an official Redis image as the base
FROM redis:latest

# Update 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

~
```

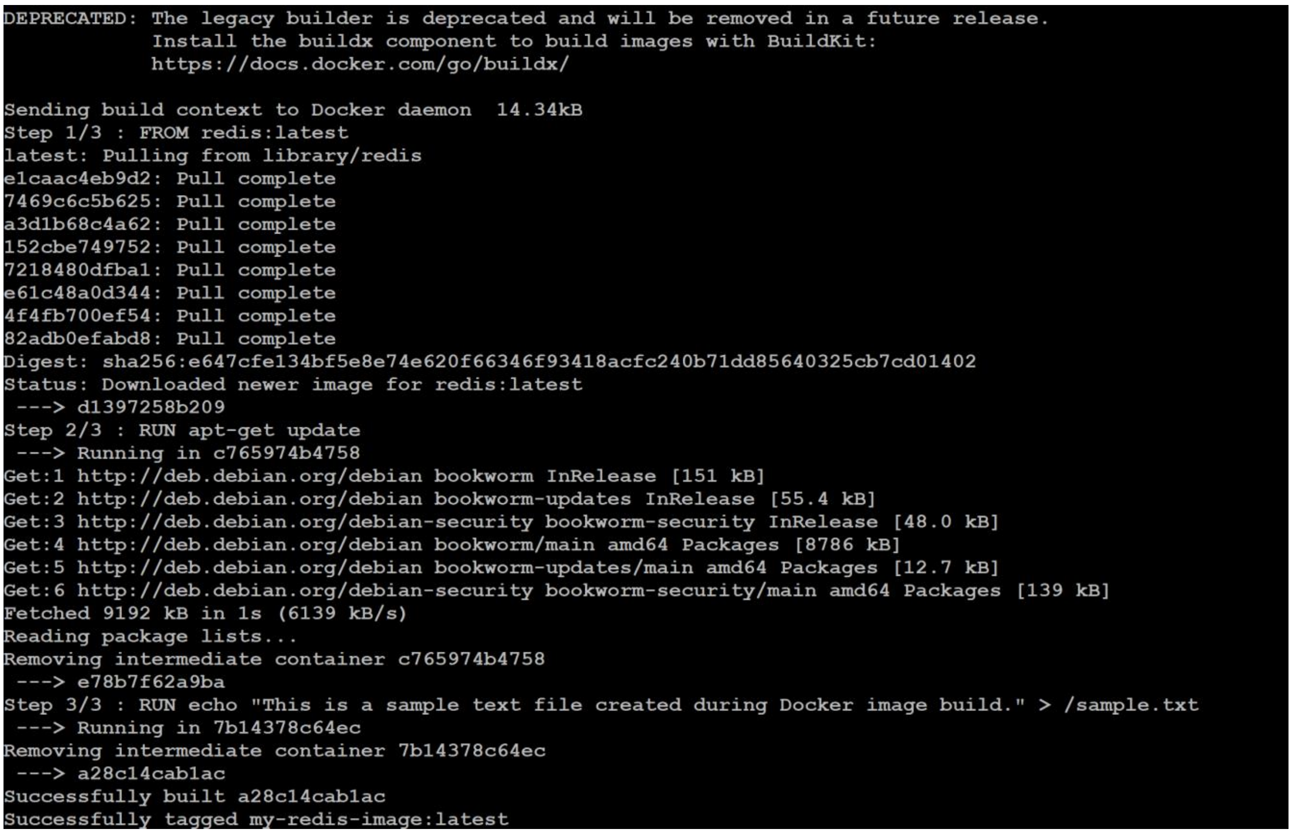
Step 2 - Build Docker Image

- 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 .
```



```
ubuntu@ip-172-31-88-138:~$ docker build -t my-redis-image .
```



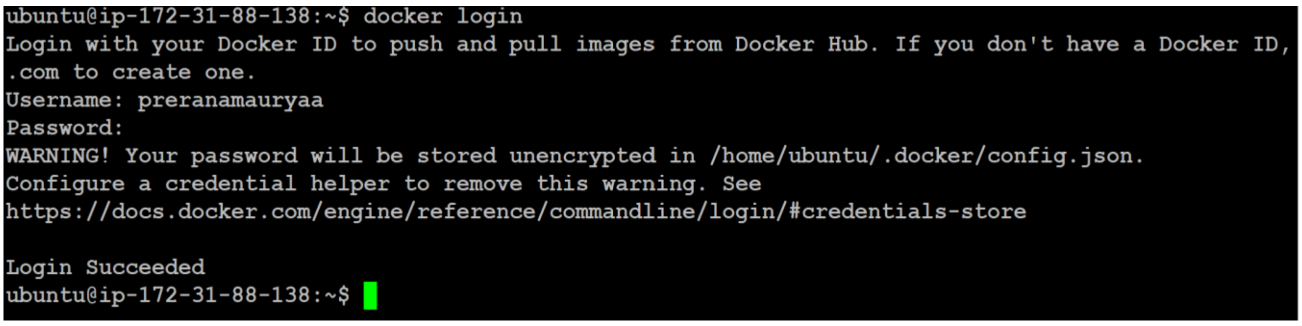
```
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  14.34kB
Step 1/3 : FROM redis:latest
latest: Pulling from library/redis
e1caac4eb9d2: Pull complete
7469c6c5b625: Pull complete
a3d1b68c4a62: Pull complete
152cbe749752: Pull complete
7218480dfba1: Pull complete
e61c48a0d344: Pull complete
4f4fb700ef54: Pull complete
82adb0efabd8: Pull complete
Digest: sha256:e647cfe134bf5e8e74e620f66346f93418acfc240b71dd85640325cb7cd01402
Status: Downloaded newer image for redis:latest
 ---> d1397258b209
Step 2/3 : RUN apt-get update
 ---> Running in c765974b4758
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8786 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [12.7 kB]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [139 kB]
Fetched 9192 kB in 1s (6139 kB/s)
Reading package lists...
Removing intermediate container c765974b4758
 ---> e78b7f62a9ba
Step 3/3 : RUN echo "This is a sample text file created during Docker image build." > /sample.txt
 ---> Running in 7b14378c64ec
Removing intermediate container 7b14378c64ec
 ---> a28c14cab1ac
Successfully built a28c14cab1ac
Successfully tagged my-redis-image:latest
```

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
```



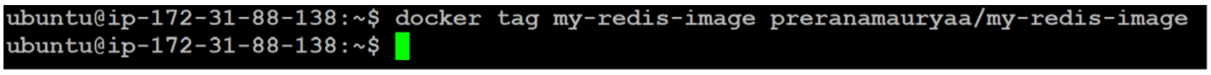
```
ubuntu@ip-172-31-88-138:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID,
.com to create one.
Username: preranamauryaa
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-88-138:~$
```

- 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 preranamauryaa/my-redis-image

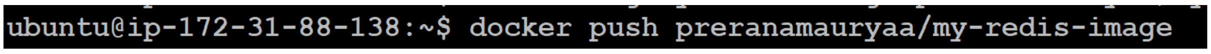


```
ubuntu@ip-172-31-88-138:~$ docker tag my-redis-image preranamauryaa/my-redis-image
ubuntu@ip-172-31-88-138:~$
```

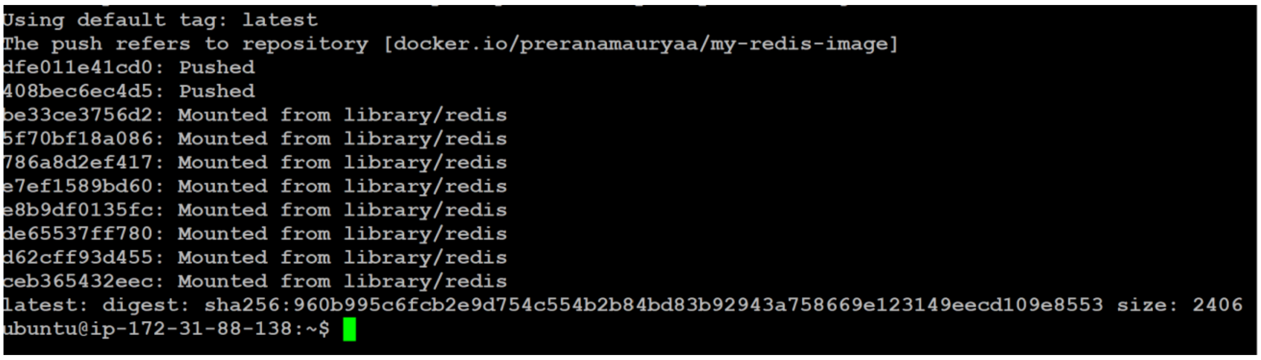
Step 3- Push Docker Image to Docker Hub

- Push the tagged Docker image to Docker Hub

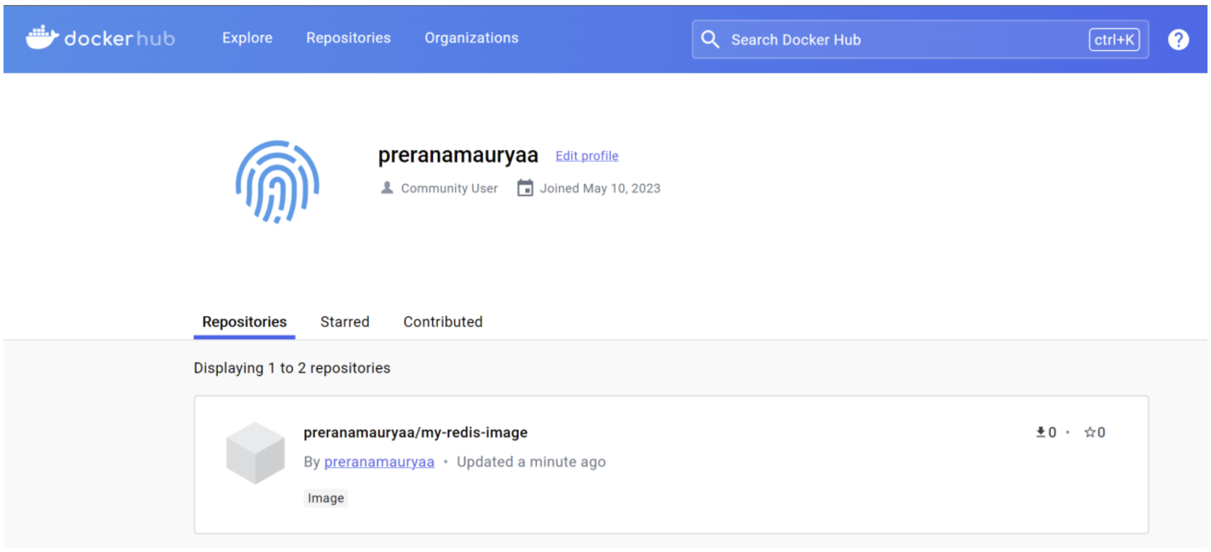
docker push preranamauryaa/my-redis-image



```
ubuntu@ip-172-31-88-138:~$ docker push preranamauryaa/my-redis-image
```



```
Using default tag: latest
The push refers to repository [docker.io/preranamauryaa/my-redis-image]
dfe011e41cd0: Pushed
408bec6ec4d5: Pushed
be33ce3756d2: Mounted from library/redis
5f70bf18a086: Mounted from library/redis
786a8d2ef417: Mounted from library/redis
e7ef1589bd60: Mounted from library/redis
e8b9df0135fc: Mounted from library/redis
de65537ff780: Mounted from library/redis
d62cff93d455: Mounted from library/redis
ceb365432eec: Mounted from library/redis
latest: digest: sha256:960b995c6fcb2e9d754c554b2b84bd83b92943a758669e123149eecd109e8553 size: 2406
ubuntu@ip-172-31-88-138:~$
```



preranamauryaa Edit profile
Community User Joined May 10, 2023

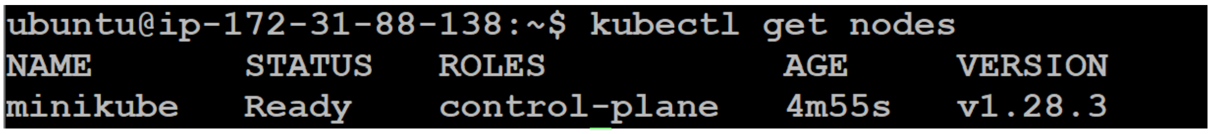
Repositories Starred Contributed

Displaying 1 to 2 repositories

preranamauryaa/my-redis-image
By preranamauryaa • Updated a minute ago
Image
0 • 0

Minikube for Kubernetes Cluster Setup

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



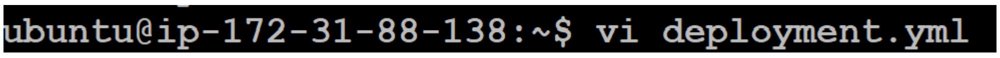
```
ubuntu@ip-172-31-88-138:~$ kubectl get nodes
NAME        STATUS    ROLES            AGE     VERSION
minikube    Ready     control-plane    4m55s   v1.28.3
```

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.

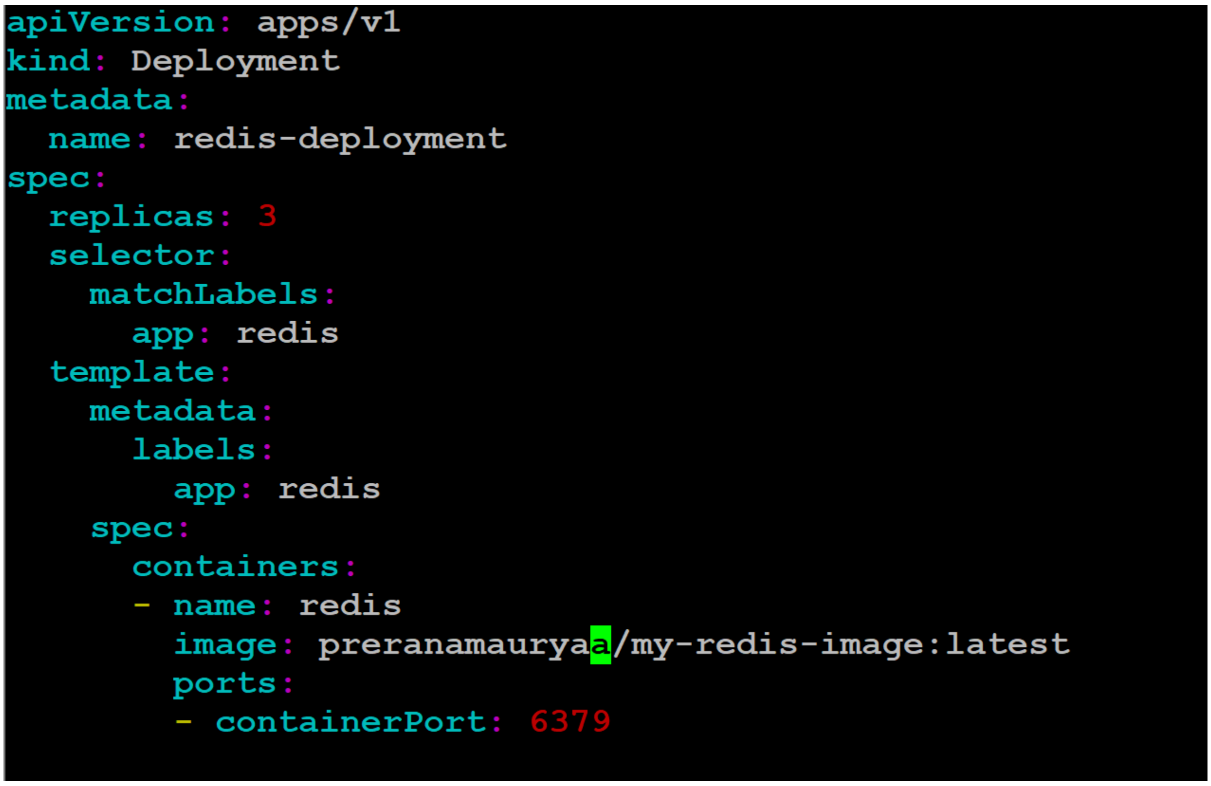
vi deployment.yml



```
ubuntu@ip-172-31-88-138:~$ 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: preranamauryaa/my-redis-image:latest
        ports:
          - containerPort: 6
```



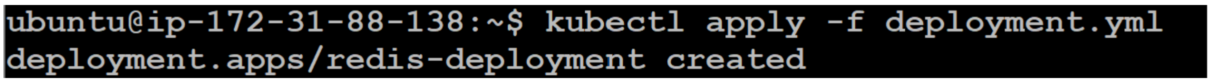
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: preranamauryaa/my-redis-image:latest
        ports:
        - containerPort: 6379
```

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

Step2- Deploy Redis on Kubernetes

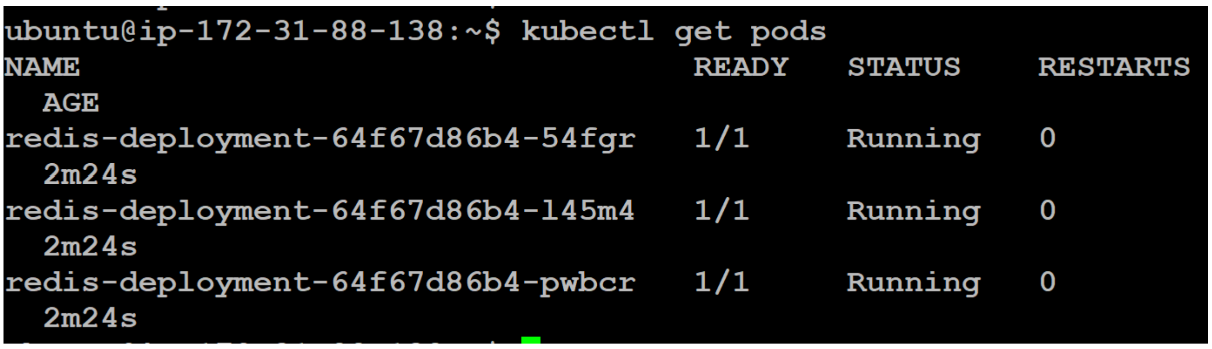
- Apply the Deployment YAML to deploy Redis on the Kubernetes cluster

kubectl apply -f redis-deployment.yaml



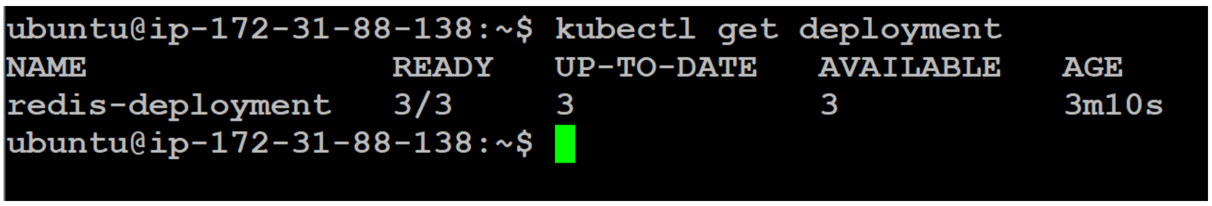
```
ubuntu@ip-172-31-88-138:~$ kubectl apply -f deployment.yml
deployment.apps/redis-deployment created
```

Pods



```
ubuntu@ip-172-31-88-138:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS
  AGE
redis-deployment-64f67d86b4-54fgr   1/1     Running   0
  2m24s
redis-deployment-64f67d86b4-145m4   1/1     Running   0
  2m24s
redis-deployment-64f67d86b4-pwbcr   1/1     Running   0
  2m24s
```

Deployment Object



```
ubuntu@ip-172-31-88-138:~$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
redis-deployment   3/3     3            3           3m10s
ubuntu@ip-172-31-88-138:~$
```

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.