Dockerized Redis Deployment on Swarm Cluster

by

Anjali Lalwani

Problem Statement

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 Swarm cluster.

Background of the problem statement:

Your organization wants to use Redis in a Swarm 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 Swarm 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 and deploy it on Swarm cluster**
- **Docker Hub: To publish the image**

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

Goal

The goal of the project is to enhance the organization's infrastructure by adopting Dockerized Redis deployment on a Swarm cluster, promoting collaboration, and ensuring efficient data management.

The subsequent sections in this document highlight the steps required to achieve this objective.

.

STEPS:

Step 1: Install Docker

Ensure that Docker is installed on your machine. You can use the following steps to install and verify the installation on your machine.

1.Use the following command to update the apt package:

```
sudo apt-get update
```

*2.*Use the following command to install packages to allow the apt to use a repository over HTTPS.

```
sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg \
lsb-release
```

3.Use the following command to add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4.Use the following command to set up a stable Docker repoistory.

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

*5.*Use the following commands to install docker.

```
sudo apt-get install docker-ce
```

docker —version

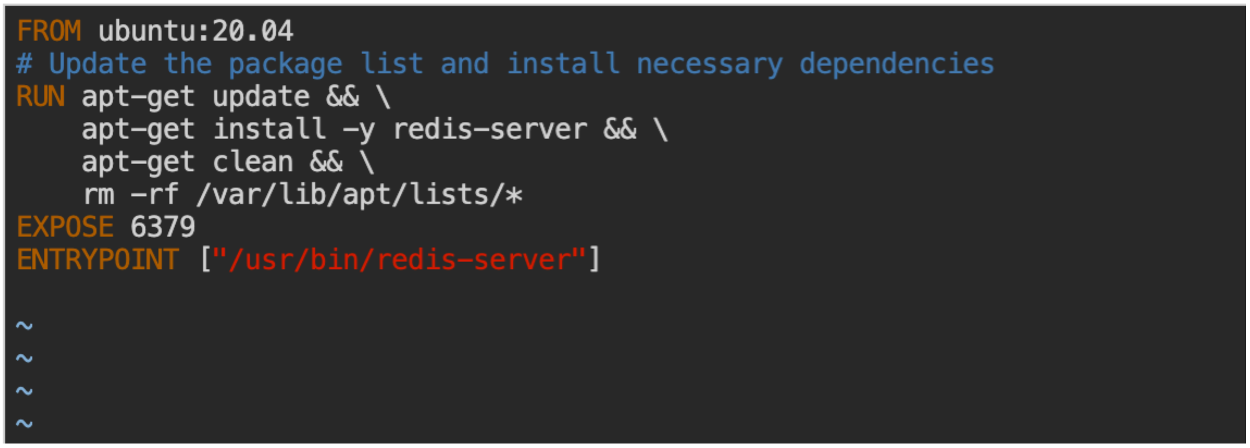
6. Verify the docker installation by running a container.

sudo docker run hello-world

Step 2: Create a Dockerfile

Go to your project directory and create a file called ‘Dockerfile’. Enter the following code inside it to create a redis image.

```
FROM ubuntu:20.04
# Update the package list and install necessary dependencies
RUN apt-get update && \
    apt-get install -y redis-server && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
EXPOSE 6379
ENTRYPOINT [“/usr/bin/redis-server”]
```

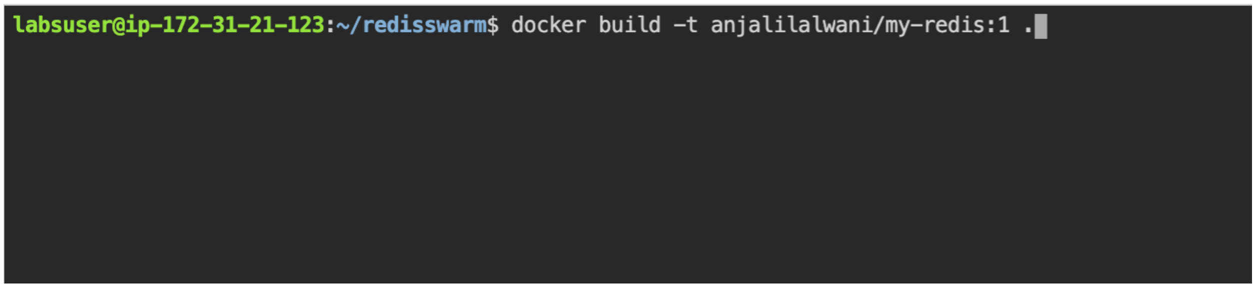


```
FROM ubuntu:20.04
# Update the package list and install necessary dependencies
RUN apt-get update && \
    apt-get install -y redis-server && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
EXPOSE 6379
ENTRYPOINT ["/usr/bin/redis-server"]

~
~
~
~
```

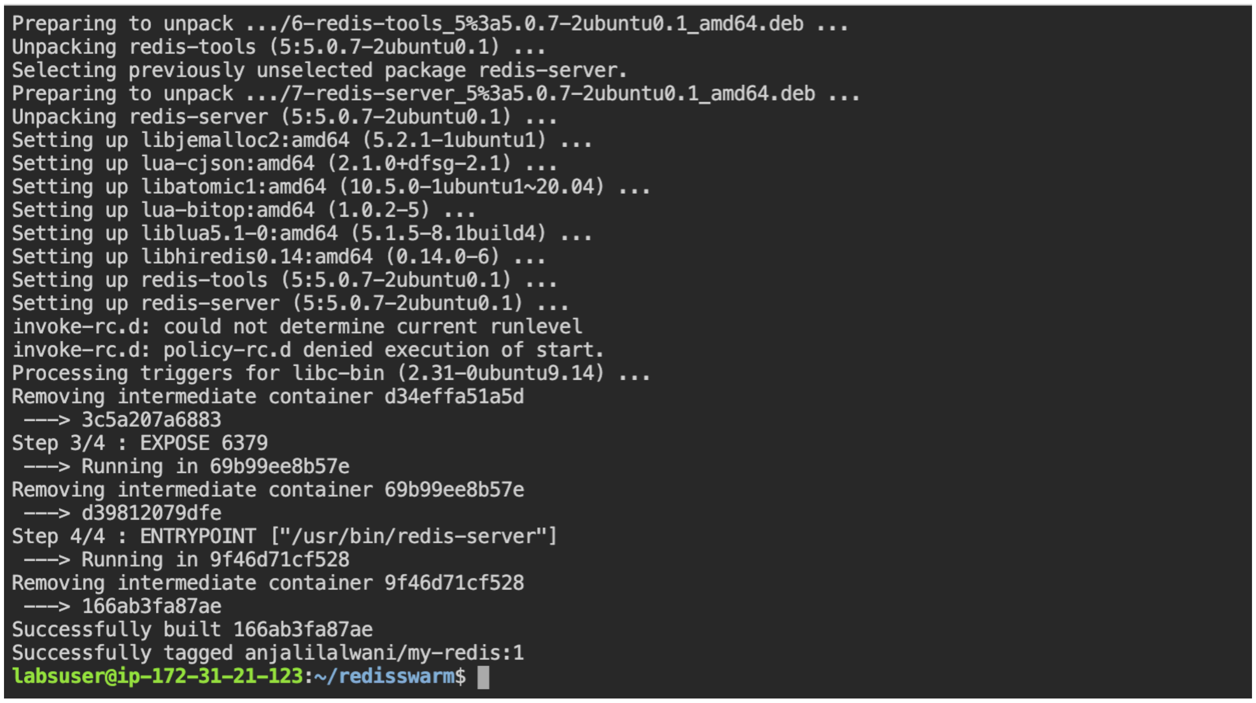
Step 3: Build the Dockerfile and create a Redis Image.

From your project directory, run the following commands to build a redis image.

docker build -t anjalilalwani/my-redis:1 .



```
labsuser@ip-172-31-21-123:~/redisswarm$ docker build -t anjalilalwani/my-redis:1 .
```

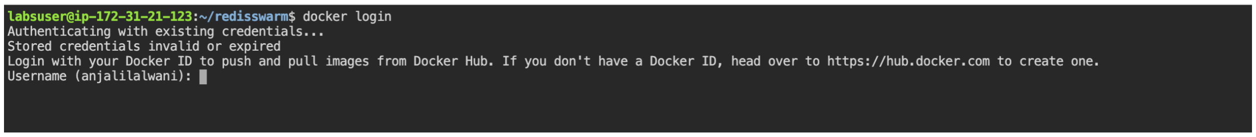


```
Preparing to unpack .../6-redis-tools_5%3a5.0.7-2ubuntu0.1_amd64.deb ...
Unpacking redis-tools (5:5.0.7-2ubuntu0.1) ...
Selecting previously unselected package redis-server.
Preparing to unpack .../7-redis-server_5%3a5.0.7-2ubuntu0.1_amd64.deb ...
Unpacking redis-server (5:5.0.7-2ubuntu0.1) ...
Setting up libjemalloc2:amd64 (5.2.1-1ubuntu1) ...
Setting up lua-cjson:amd64 (2.1.0+dfsg-2.1) ...
Setting up libatomic1:amd64 (10.5.0-1ubuntu1~20.04) ...
Setting up lua-bitop:amd64 (1.0.2-5) ...
Setting up liblua5.1-0:amd64 (5.1.5-8.1build4) ...
Setting up libhiredis0.14:amd64 (0.14.0-6) ...
Setting up redis-tools (5:5.0.7-2ubuntu0.1) ...
Setting up redis-server (5:5.0.7-2ubuntu0.1) ...
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
Removing intermediate container d34effa51a5d
 ---> 3c5a207a6883
Step 3/4 : EXPOSE 6379
 ---> Running in 69b99ee8b57e
Removing intermediate container 69b99ee8b57e
 ---> d39812079dfe
Step 4/4 : ENTRYPOINT ["/usr/bin/redis-server"]
 ---> Running in 9f46d71cf528
Removing intermediate container 9f46d71cf528
 ---> 166ab3fa87ae
Successfully built 166ab3fa87ae
Successfully tagged anjalilalwani/my-redis:1
labsuser@ip-172-31-21-123:~/redisswarm$
```

Step 4: Login to Docker Hub

Run the following command in the terminal to log in to your Docker Hub account. Enter your credentials for Docker Hub when prompted.

docker login

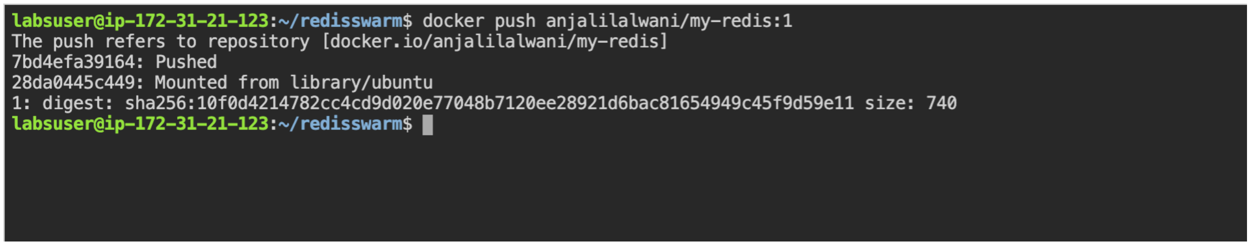


```
labsuser@ip-172-31-21-123:~/redisswarm$ docker login
Authenticating with existing credentials...
Stored credentials invalid or expired
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username (anjalilalwani): 
```

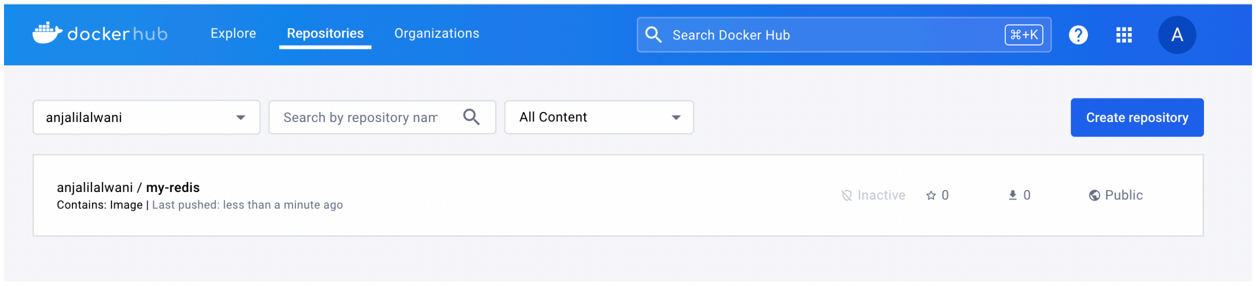
Step 5: Push your Docker Image to Docker Hub

Run the following command to push the redis image to your Docker Hub account.

docker push anjalilalwani/my-redis:1



```
labsuser@ip-172-31-21-123:~/redisswarm$ docker push anjalilalwani/my-redis:1
The push refers to repository [docker.io/anjalilalwani/my-redis]
7bd4efa39164: Pushed
28da0445c449: Mounted from library/ubuntu
1: digest: sha256:10f0d4214782cc4cd9d020e77048b7120ee28921d6bac81654949c45f9d59e11 size: 740
labsuser@ip-172-31-21-123:~/redisswarm$ 
```

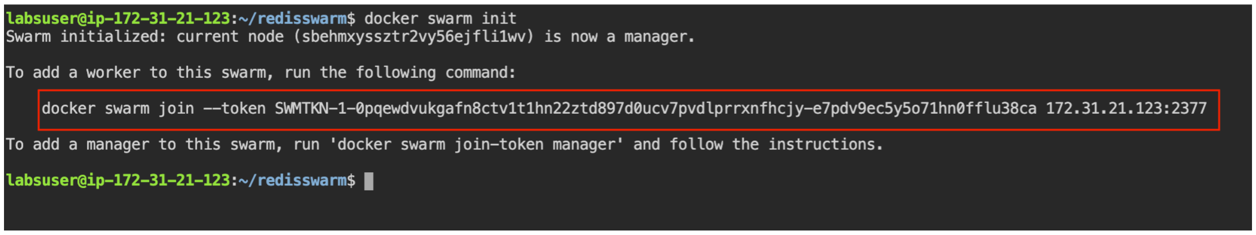


anjalilalwani / **my-redis**
Contains: Image | Last pushed: less than a minute ago
Inactive ☆ 0 ⬇ 0 Public

Step 6: Initialize Swarm

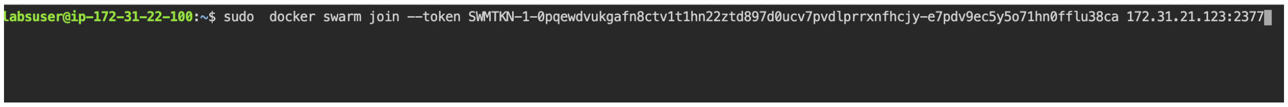
To initialize the docker swarm, run the following command on the manager nodes.

docker swarm init



```
labsuser@ip-172-31-21-123:~/redisswarm$ docker swarm init
Swarm initialized: current node (sbehmxyssztr2vy56ejfli1wv) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0pqewdvukgafn8ctv1t1hn22ztd897d0ucv7pvdlprrxnfhcjy-e7pdv9ec5y5o71hn0fflu38ca 172.31.21.123:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

labsuser@ip-172-31-21-123:~/redisswarm$ 
```

This will initialize the Swarm cluster and provides a command to join other nodes as workers. Use the command provided after the initialization on each worker node.

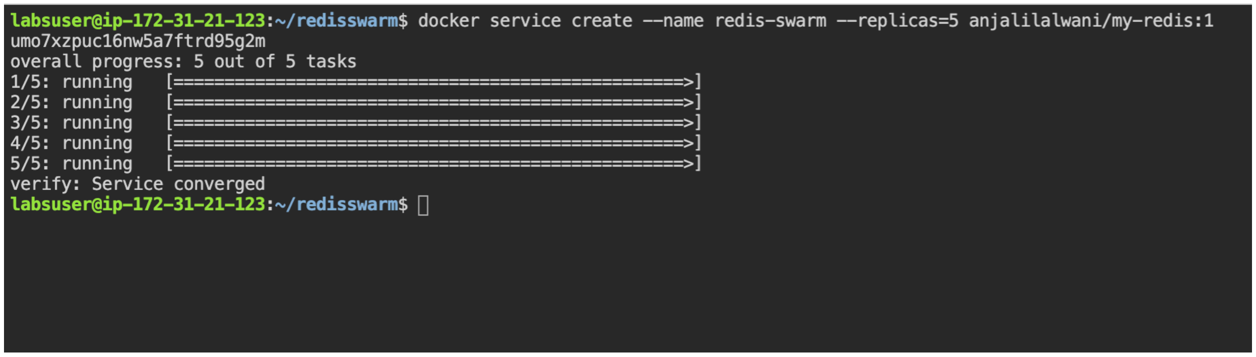


```
labsuser@ip-172-31-22-100:~$ sudo  docker swarm join --token SWMTKN-1-0pqewdvukgafn8ctv1t1hn22ztd897d0ucv7pvdlprrxnfhcjy-e7pdv9ec5y5o71hn0fflu38ca 172.31.21.123:2377
```

Step 7: Deploy Redis Service

Deploy the Redis Service on the swarm cluster using the following command.Enter the number of replicas to created with the —replicas option.

docker service create --name redis-swarm —replicas=5 anjalilalwani/redis-swarm:1

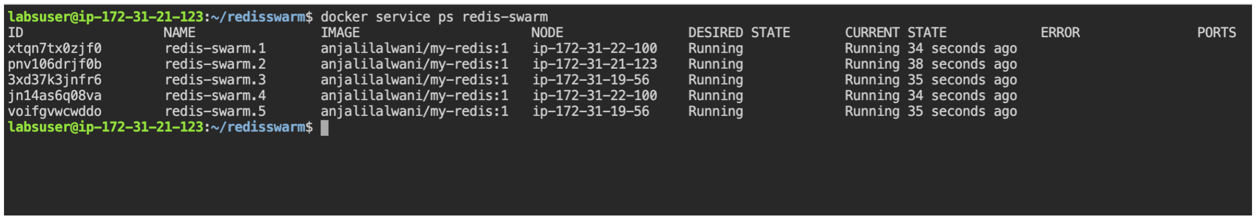


```
labsuser@ip-172-31-21-123:~/redisswarm$ docker service create --name redis-swarm --replicas=5 anjalilalwani/my-redis:1
umo7xzpuc16nw5a7ftrd95g2m
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service converged
labsuser@ip-172-31-21-123:~/redisswarm$ 
```

Step 8: Verify Deployment

Check if the set number of replicas are running and the Redis container is active.

docker service ps redis-swarm

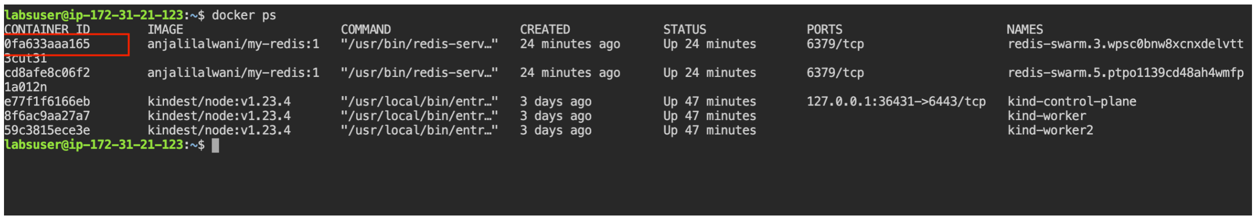


```
labsuser@ip-172-31-21-123:~/redisswarm$ docker service ps redis-swarm
ID             NAME            IMAGE                      NODE              DESIRED STATE   CURRENT STATE            ERROR     PORTS
xtqn7tx0zjf0   redis-swarm.1   anjalilalwani/my-redis:1   ip-172-31-22-100  Running         Running 34 seconds ago
pnv106drjf0b   redis-swarm.2   anjalilalwani/my-redis:1   ip-172-31-21-123  Running         Running 38 seconds ago
3xd37k3jnfr6   redis-swarm.3   anjalilalwani/my-redis:1   ip-172-31-19-56   Running         Running 35 seconds ago
jn14as6q08va   redis-swarm.4   anjalilalwani/my-redis:1   ip-172-31-22-100  Running         Running 34 seconds ago
voifgvwcwddo   redis-swarm.5   anjalilalwani/my-redis:1   ip-172-31-19-56   Running         Running 35 seconds ago
labsuser@ip-172-31-21-123:~/redisswarm$ 
```

Step 5: Access Redis

Run the following command and get the container id of one of the containers running on the swarm.

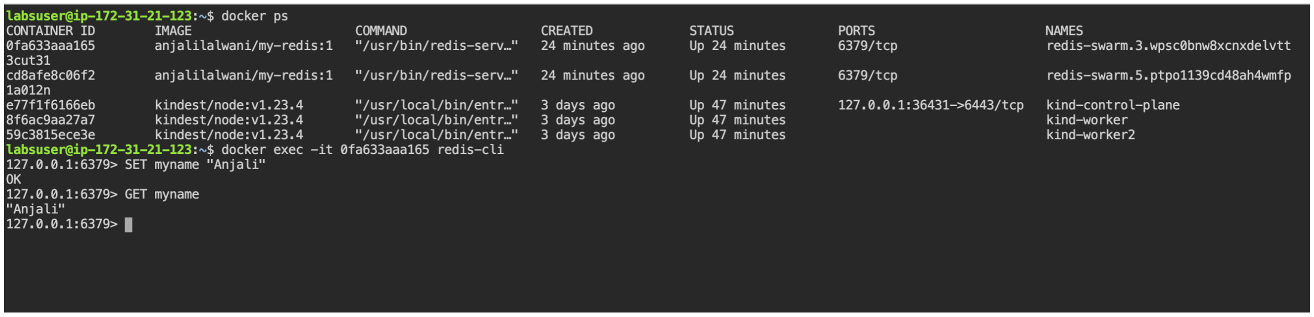
docker ps



```
labsuser@ip-172-31-21-123:~$ docker ps
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS                        NAMES
0fa633aaa165        anjalilalwani/my-redis:1   "/usr/bin/redis-serv…"   24 minutes ago      Up 24 minutes       6379/tcp                     redis-swarm.3.wpsc0bnw8xcnxdelvtt
3cut31
cd8afe8c06f2        anjalilalwani/my-redis:1   "/usr/bin/redis-serv…"   24 minutes ago      Up 24 minutes       6379/tcp                     redis-swarm.5.ptpo1139cd48ah4wmfp
1a012n
e77f1f6166eb        kindest/node:v1.23.4     "/usr/local/bin/entr…"   3 days ago          Up 47 minutes       127.0.0.1:36431->6443/tcp    kind-control-plane
8f6ac9aa27a7        kindest/node:v1.23.4     "/usr/local/bin/entr…"   3 days ago          Up 47 minutes                                    kind-worker
59c3815ece3e        kindest/node:v1.23.4     "/usr/local/bin/entr…"   3 days ago          Up 47 minutes                                    kind-worker2
labsuser@ip-172-31-21-123:~$
```

Use this container id to access the Redis cli on the container.

docker exec -it <container-id> redis-cli



```
labsuser@ip-172-31-21-123:~$ docker ps
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS                        NAMES
0fa633aaa165        anjalilalwani/my-redis:1   "/usr/bin/redis-serv…"   24 minutes ago      Up 24 minutes       6379/tcp                     redis-swarm.3.wpsc0bnw8xcnxdelvtt
3cut31
cd8afe8c06f2        anjalilalwani/my-redis:1   "/usr/bin/redis-serv…"   24 minutes ago      Up 24 minutes       6379/tcp                     redis-swarm.5.ptpo1139cd48ah4wmfp
1a012n
e77f1f6166eb        kindest/node:v1.23.4     "/usr/local/bin/entr…"   3 days ago          Up 47 minutes       127.0.0.1:36431->6443/tcp    kind-control-plane
8f6ac9aa27a7        kindest/node:v1.23.4     "/usr/local/bin/entr…"   3 days ago          Up 47 minutes                                    kind-worker
59c3815ece3e        kindest/node:v1.23.4     "/usr/local/bin/entr…"   3 days ago          Up 47 minutes                                    kind-worker2
labsuser@ip-172-31-21-123:~$ docker exec -it 0fa633aaa165 redis-cli
127.0.0.1:6379> SET myname "Anjali"
OK
127.0.0.1:6379> GET myname
"Anjali"
127.0.0.1:6379>
```

CONCLUSION:

In conclusion, this project accomplished the creation of a Redis Docker image, its deployment on a Swarm cluster, and publication on Docker Hub. The streamlined process ensures an efficient and scalable Redis solution for the organization's data storage and caching needs.

-