Container orchestration using Dockerswarm

1.Introduction

In modern software development, efficiently managing and deploying containerized applications across multiple servers is essential. Docker Swarm, a native Docker orchestration tool, provides a powerful yet simple solution for this task. It enables you to manage a cluster of Docker nodes as a single virtual system, ensuring seamless application deployment and scalability.

Docker Swarm offers key features such as load balancing, automatic scaling, and service discovery, making it an excellent choice for running microservices and distributed applications. By distributing services across multiple nodes, Docker Swarm ensures high availability and fault tolerance, automatically redistributing workloads in case of node failures.

2.Tools And Methodologies:

In this project I have used number of tools including some devops tools and the techniques are listed below.

Programming language: Python

Framework: Flask

Cloud Platform : AWS

Devops tools: Dockerswarm

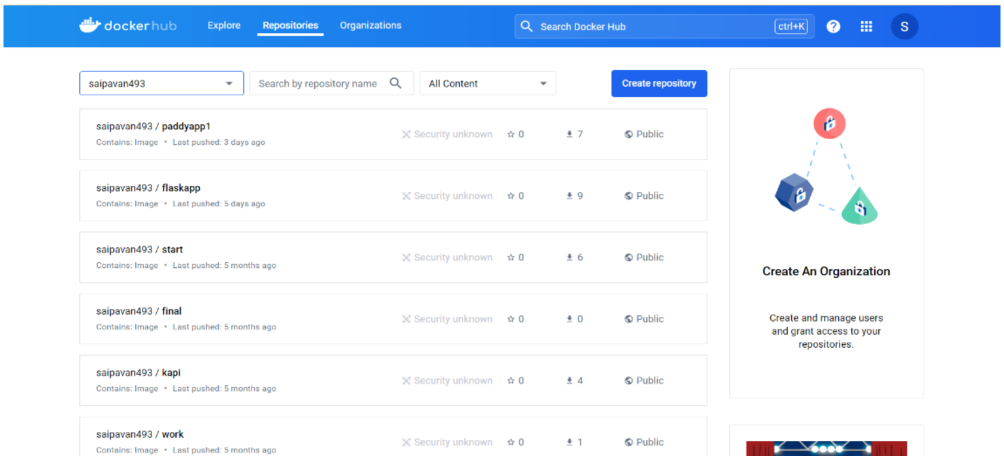
Container Repository: Dockerhub

3.Containerization:

I have used Docker to create container for my application. For local environment,I have installed docker desktop.Dockerize the application and build an image and run in a local host next push into docker hub. By following these commands.

- Build Image: docker build -t . paddyapp .
- Run image :docker run -d -p 5000:5000 paddyapp1
- Tag image: docker tag paddyapp1:latest saipavan493/paddyapp1:latest
- Login Dockerhub:docker login

- Push to Docker hub: docker push saipavan493/paddyapp1:latest



saipavan493 / paddyapp1 Contains: Image • Last pushed: 3 days ago	Security unknown	☆ 0	7	Public
saipavan493 / flaskapp Contains: Image • Last pushed: 5 days ago	Security unknown	☆ 0	9	Public
saipavan493 / start Contains: Image • Last pushed: 5 months ago	Security unknown	☆ 0	6	Public
saipavan493 / final Contains: Image • Last pushed: 5 months ago	Security unknown	☆ 0	0	Public
saipavan493 / kapi Contains: Image • Last pushed: 5 months ago	Security unknown	☆ 0	4	Public
saipavan493 / work Contains: Image • Last pushed: 5 months ago	Security unknown	☆ 0	1	Public

Create and manage users and grant access to your repositories.

Application pushed into Dockerhub

4. Container orchestration with Dockerswarm

Step 1: Create two or three Ec2 instances.one will be manger and remaining are Workers.

Step 2: Edit the security group and add the following rules.and save it

☐ -TCP port 2376 for secure Docker client communication. This port is required for Docker Machine to work. Docker Machine is used to orchestrate Docker hosts.

-TCP port 2377. This port is used for communication between the nodes of a Docker Swarm or cluster. It only needs to be opened on manager nodes.

-TCP and UDP port 7946 for communication among nodes (container network discovery).

- UDP port 4789 for overlay network traffic (container ingress networking).

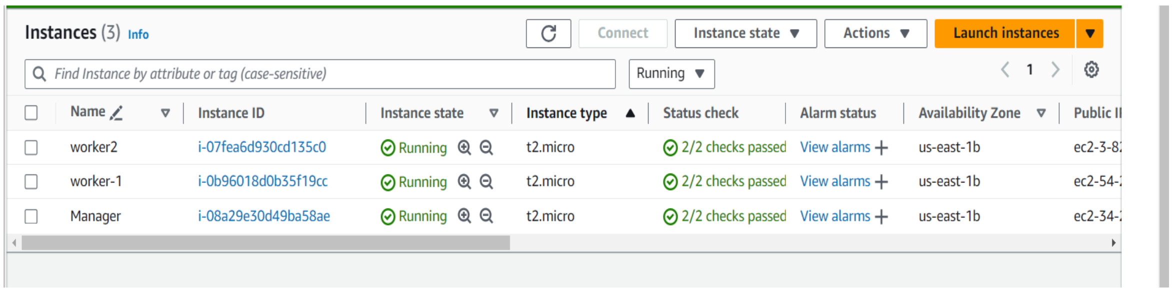
-TCP port 22 to SSH into our instances remotely

Step 3: Install docker on each machine.by using following script

```
#!/bin/bash
sudo yum update
sudo yum -y install docker
service docker start
```

usermod -a -G docker ec2-user

step 4: after sucesfully installed docker on all machines assing one instance as manager and remaining are workers.



Instances (3) Info

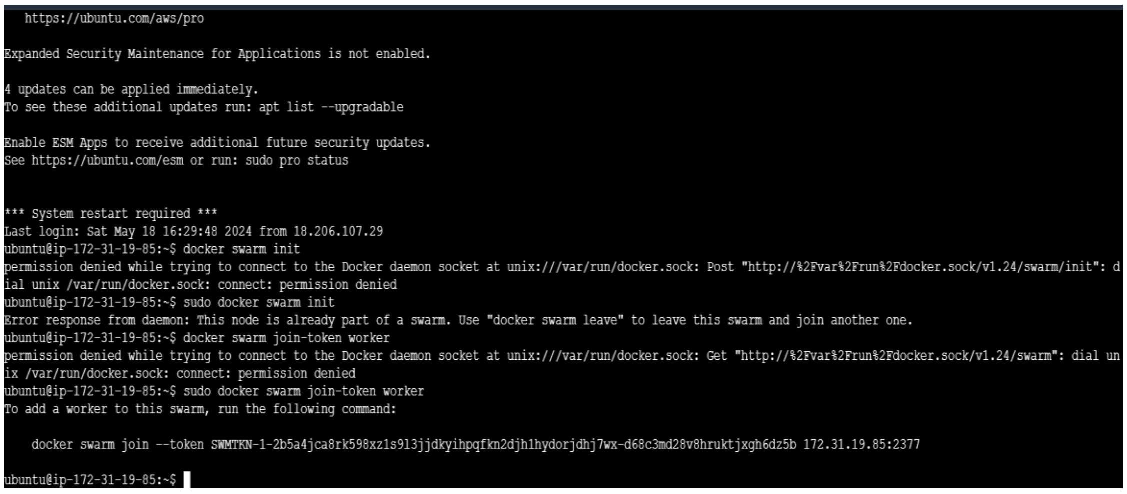
Find Instance by attribute or tag (case-sensitive) | Running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public II
worker2	i-07fea6d930cd135c0	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-3-8
worker-1	i-0b96018d0b35f19cc	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-54-
Manager	i-08a29e30d49ba58ae	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-34-

Step 5: to setup dockerswarm cluster.open the docker swarm manager node and enter the following command**.**

docker swarm init

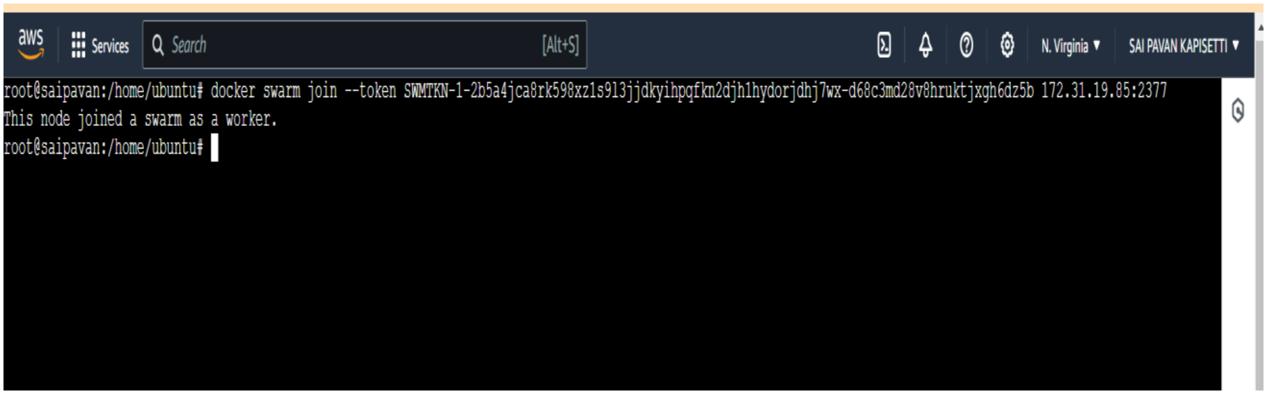
generated a token enter these token to join a cluster in worker nodes



```
   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

4 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Sat May 18 16:29:48 2024 from 18.206.107.29
ubuntu@ip-172-31-19-85:~$ docker swarm init
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/swarm/init": d
ial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-19-85:~$ sudo docker swarm init
Error response from daemon: This node is already part of a swarm. Use "docker swarm leave" to leave this swarm and join another one.
ubuntu@ip-172-31-19-85:~$ docker swarm join-token worker
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/swarm": dial un
ix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-19-85:~$ sudo docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2b5a4jca8rk598xz1s913jjdkyihpqfkn2djh1hydorjdhj7wx-d68c3md28v8hruktjxgh6dz5b 172.31.19.85:2377

ubuntu@ip-172-31-19-85:~$ 
```

step 6: open the worker nodes and join by using following command.

docker swarm join --token SWMTKN-1-2b5a4jca8rk598xz1s9l3jjdkyihpqfkn2djh1hydorjdhj7wx-d68c3md28v8hruktjxgh6dz5b 172.31.19.85:2377



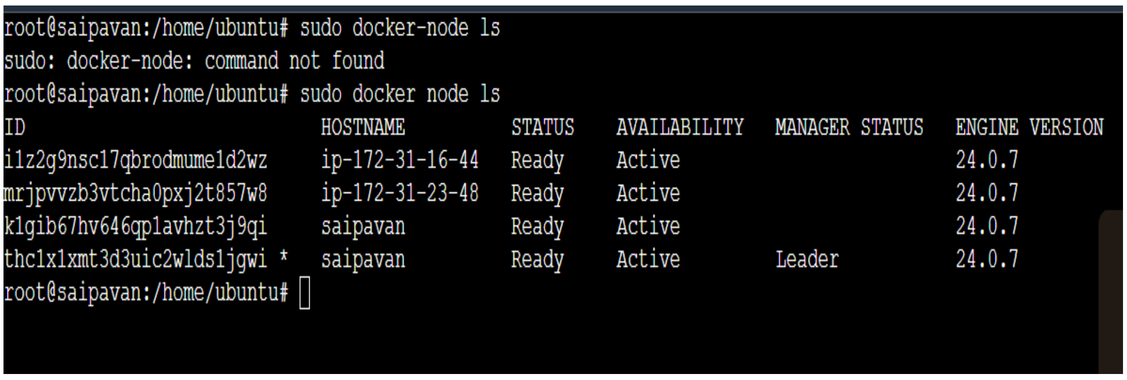
```
root@saipavan:/home/ubuntu# docker swarm join --token SWMTKN-1-2b5a4jca8rk598xz1s9l3jjdkyihpqfkn2djh1hydorjdhj7wx-d68c3md28v8hruktjxgh6dz5b 172.31.19.85:2377
This node joined a swarm as a worker.
root@saipavan:/home/ubuntu#
```

after sucessfully joined.displays message like "This node joined as a worker"

step 7: to check the status of nodes by using the following commands. Open the manager machine.

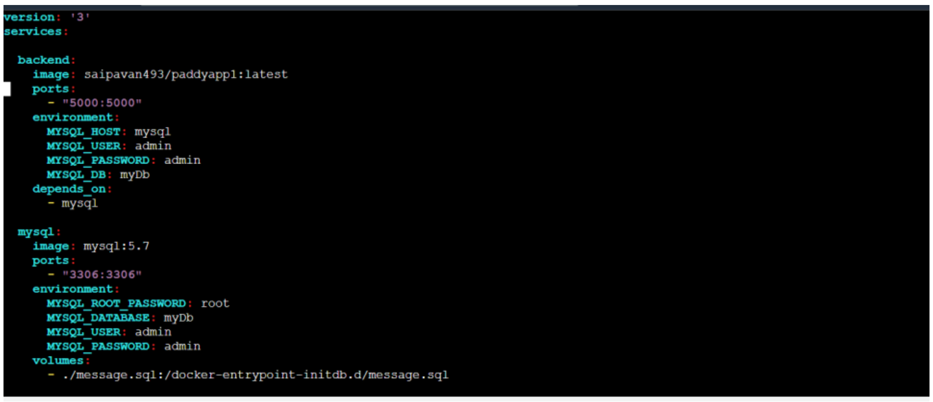
Sudo docker node ls

Displays list of all worker nodes and manger node



```
root@saipavan:/home/ubuntu# sudo docker-node ls
sudo: docker-node: command not found
root@saipavan:/home/ubuntu# sudo docker node ls
ID                            HOSTNAME         STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
i1z2g9nsc17qbrodmume1d2wz     ip-172-31-16-44  Ready     Active                          24.0.7
mrjpvvzb3vtcha0pxj2t857w8     ip-172-31-23-48  Ready     Active                          24.0.7
k1gib67hv646qp1avhzt3j9qi     saipavan         Ready     Active                          24.0.7
thc1x1xmt3d3uic2wlds1jgwi *   saipavan         Ready     Active         Leader           24.0.7
root@saipavan:/home/ubuntu#
```

Step 8: open the manager machine .and write the docker-compose.yml file with the help of vi editor.



```
version: '3'
services:

  backend:
    image: saipavan493/paddyapp1:latest
    ports:
      - "5000:5000"
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
      MYSQL_DB: myDb
    depends_on:
      - mysql

  mysql:
    image: mysql:5.7
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: myDb
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
    volumes:
      - ./message.sql:/docker-entrypoint-initdb.d/message.sql
```

Yaml file :

version: '3'

services:

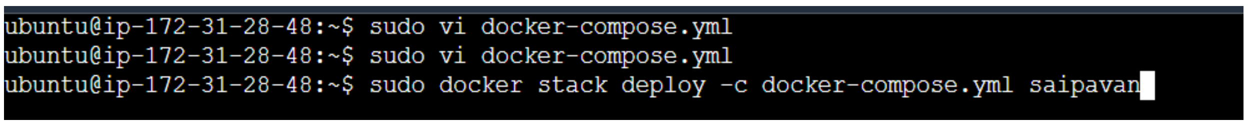
```
  backend:
    image: saipavan493/paddyapp1:latest
    ports:
      - "5000:5000"
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
      MYSQL_DB: myDb
    depends_on:
      - mysql

  mysql:
    image: mysql:5.7
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: myDb
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
    volumes:
      - ./message.sql:/docker-entrypoint-initdb.d/message.sql
      - mysql-data:/var/lib/mysql

volumes:
  mysql-data:
```

save the file with extension of yml/yaml

step 9: after saving.and execute the compose with help of the following command

docker stack deploy -c docker-compose.yml saipavan

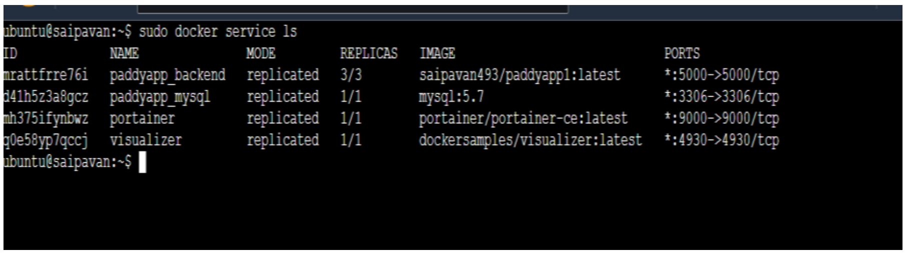
and run the following command



```
ubuntu@ip-172-31-28-48:~$ sudo vi docker-compose.yml
ubuntu@ip-172-31-28-48:~$ sudo vi docker-compose.yml
ubuntu@ip-172-31-28-48:~$ sudo docker stack deploy -c docker-compose.yml saipavan
```

step 10: to see the status use the following command

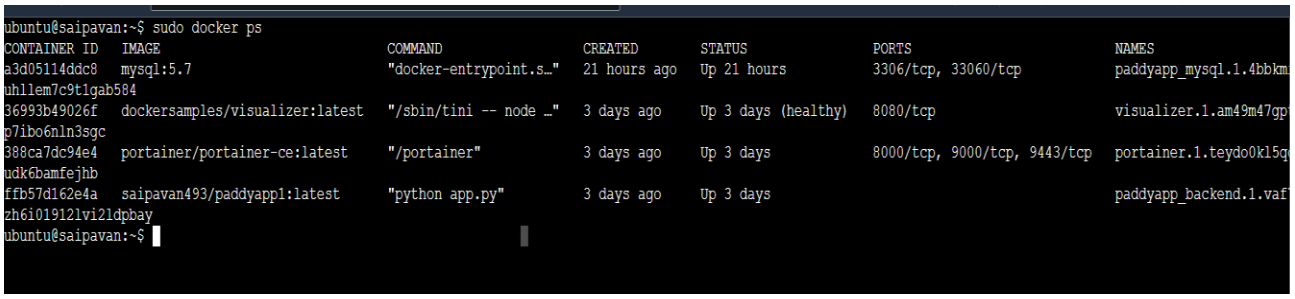
docker service ls



```
ubuntu@saipavan:~$ sudo docker service ls
ID             NAME               MODE         REPLICAS   IMAGE                                PORTS
mrattfrre76i   paddyapp_backend   replicated   3/3        saipavan493/paddyapp1:latest         *:5000->5000/tcp
d41h5z3a8gcz   paddyapp_mysql     replicated   1/1        mysql:5.7                            *:3306->3306/tcp
mh375ifynbwz   portainer          replicated   1/1        portainer/portainer-ce:latest        *:9000->9000/tcp
g0e58yp7qccj   visualizer         replicated   1/1        dockersamples/visualizer:latest      *:4930->4930/tcp
ubuntu@saipavan:~$
```

step 11: to see the list of containers using the following comand

docker ps



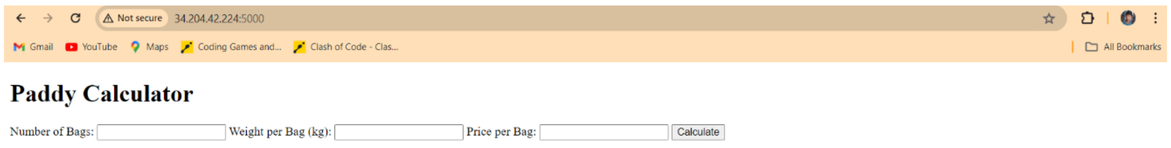
```
ubuntu@saipavan:~$ sudo docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED        STATUS                 PORTS                          NAMES
a3d05114ddc8   mysql:5.7                         "docker-entrypoint.s…"   21 hours ago   Up 21 hours            3306/tcp, 33060/tcp            paddyapp_mysql.1.4bbkm
uhllem7c9t1gab584
36993b49026f   dockersamples/visualizer:latest   "/sbin/tini -- node …"   3 days ago     Up 3 days (healthy)    8080/tcp                       visualizer.1.am49m47gp
p7ibo6nln3sgc
388ca7dc94e4   portainer/portainer-ce:latest     "/portainer"             3 days ago     Up 3 days              8000/tcp, 9000/tcp, 9443/tcp   portainer.1.teydo0kl5q
udk6bamfejhb
ffb57d162e4a   saipavan493/paddyapp1:latest      "python app.py"          3 days ago     Up 3 days                                             paddyapp_backend.1.vaf
zh6i01912lvi2ldpbay
ubuntu@saipavan:~$
```

Step 12: after sucessfully completed all steps .acess the application with instance ip along with port number in browser. Application will run in manger and worker machines.let us check one by one

Open the browser and enter the url

http://<instance-ip (manger/workers)>:portnumber

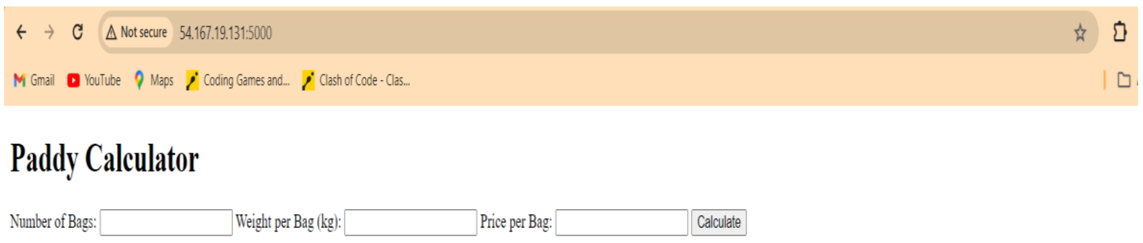
Manager:



Paddy Calculator

Number of Bags: Weight per Bag (kg): Price per Bag: Calculate

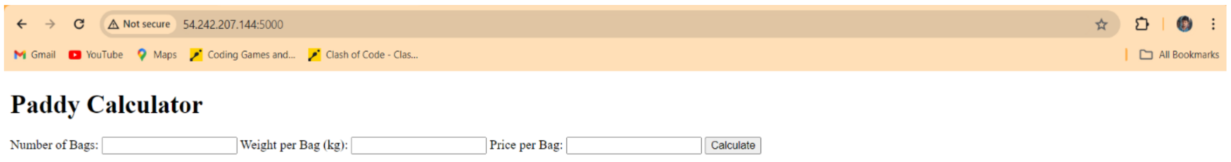
worker 1:



Paddy Calculator

Number of Bags: Weight per Bag (kg): Price per Bag: Calculate

worker 2:


Not secure 54.242.207.144:5000

Paddy Calculator

Number of Bags: Weight per Bag (kg): Price per Bag: Calculate

portainer setup:

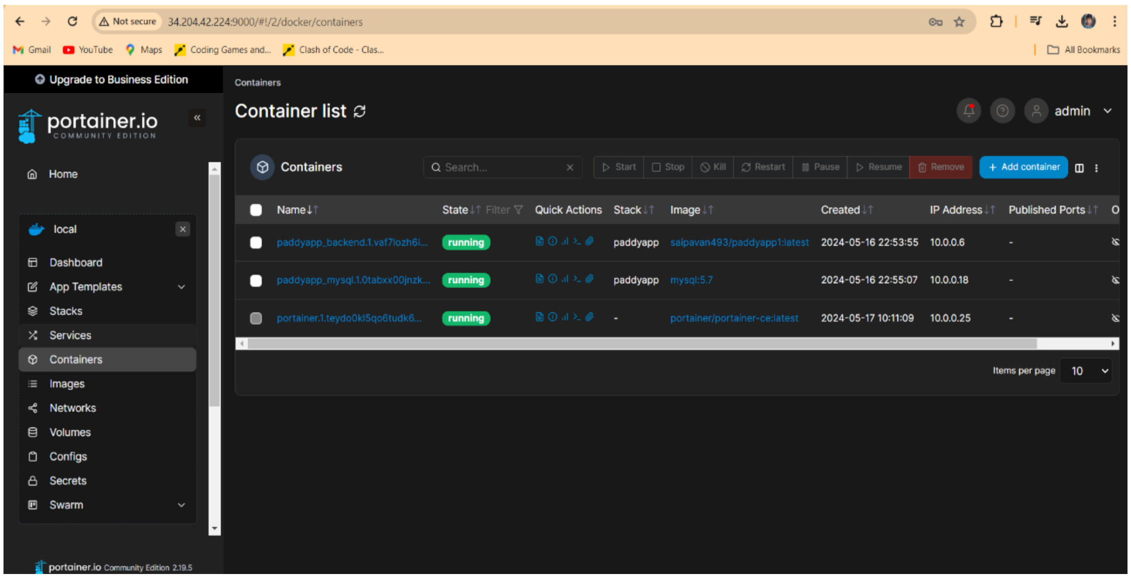
Portainer is a GUI management tool for docker.by portainer we can easily manage containers,stacks,volumes..etc

Portainer will acessible in port 9000 on manager machine.

To setup portainer in dockerswarm open the manager machine and execute the following command.

```
      docker service create \
--name portainer \
--publish 9000:9000 \
--replicas=1 \
--constraint 'node.role == manager' \
--mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
--mount type=volume,src=portainer_data,dst=/data \
portainer/portainer-ce
```

Acess portainer with http://<manager-ip>:portnumber=9000



Container list

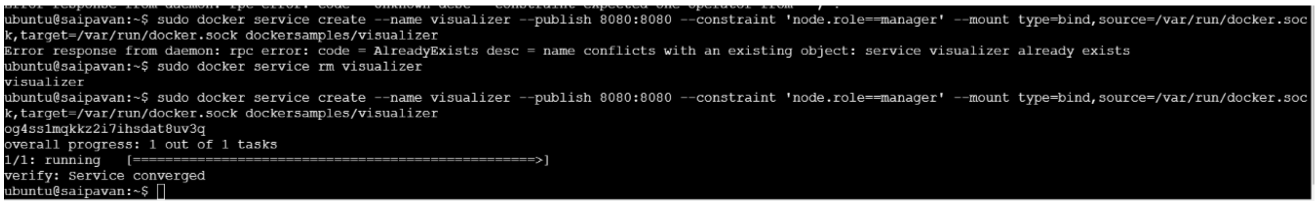
Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports
paddyapp_backend.1.vaf7lozh6i...	running		paddyapp	saipavan493/paddyapp1:latest	2024-05-16 22:53:55	10.0.0.6	-
paddyapp_mysql.1.0tabxx00jnzk...	running		paddyapp	mysql:5.7	2024-05-16 22:55:07	10.0.0.18	-
portainer.1.teydo0kl5qo6tudk6...	running		-	portainer/portainer-ce:latest	2024-05-17 10:11:09	10.0.0.25	-

In portainer dashboard we can see list of containers,volumes,stacks,services ..etc

Docker Graphic visualizer:

To set up docker graphic visualizer using the following command.

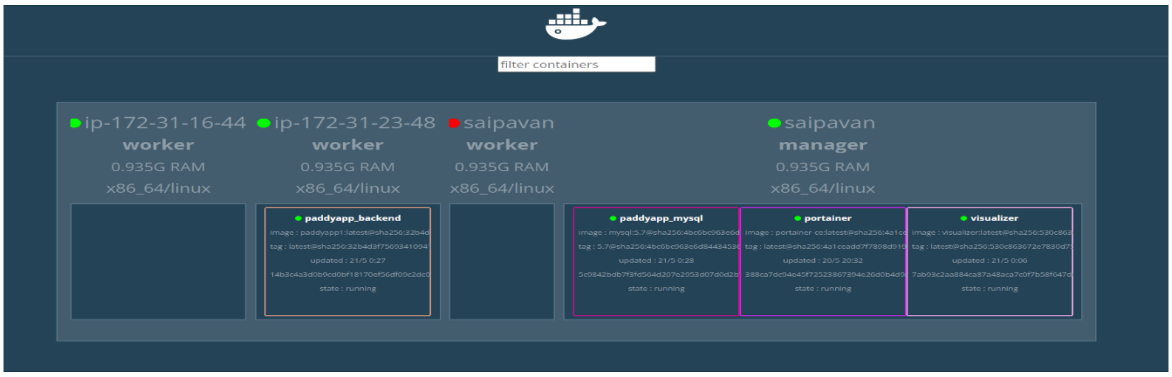
Docker service create –name visualizer –publish 8080:8080 –constraint ‘node.role==manager’ –mount type =bind,source=/var/run/docker.sock,target=/var/run/docker.sock dockersamples/visualizer



```
ubuntu@saipavan:~$ sudo docker service create --name visualizer --publish 8080:8080 --constraint 'node.role==manager' --mount type=bind,source=/var/run/docker.soc
k,target=/var/run/docker.sock dockersamples/visualizer
Error response from daemon: rpc error: code = AlreadyExists desc = name conflicts with an existing object: service visualizer already exists
ubuntu@saipavan:~$ sudo docker service rm visualizer
visualizer
ubuntu@saipavan:~$ sudo docker service create --name visualizer --publish 8080:8080 --constraint 'node.role==manager' --mount type=bind,source=/var/run/docker.soc
k,target=/var/run/docker.sock dockersamples/visualizer
og4ss1mqkkz2i7ihsdat8uv3q
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
ubuntu@saipavan:~$ 
```

After sucessfully executing the visualizer .acess the application with manager machine on port 8080.

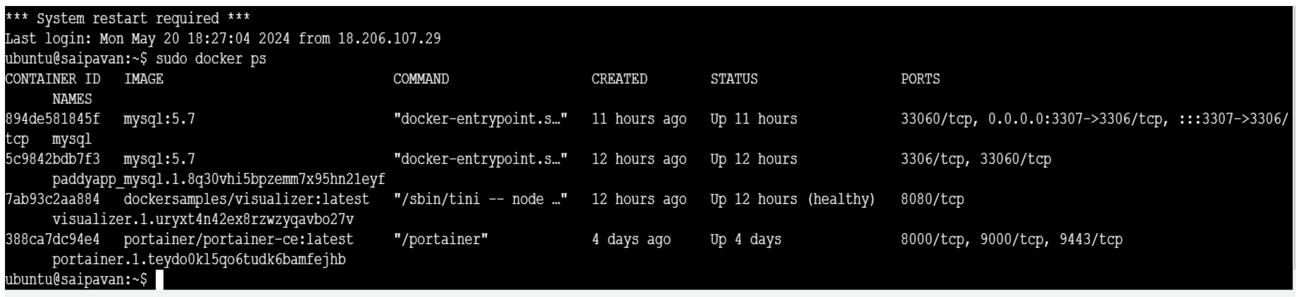
http://<managerip>:port:8080



here displays list worker and manager node containers what are running on it.

To view the list of all running containers by using the following command on manager machine

sudo docker ps



```
*** System restart required ***
Last login: Mon May 20 18:27:04 2024 from 18.206.107.29
ubuntu@saipavan:~$ sudo docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED        STATUS                   PORTS
      NAMES
894de581845f   mysql:5.7                         "docker-entrypoint.s…"   11 hours ago   Up 11 hours              33060/tcp, 0.0.0.0:3307->3306/tcp, :::3307->3306/
tcp   mysql
5c9842bdb7f3   mysql:5.7                         "docker-entrypoint.s…"   12 hours ago   Up 12 hours              3306/tcp, 33060/tcp
      paddyapp_mysql.1.8q30vhi5bpzemm7x95hn21eyf
7ab93c2aa884   dockersamples/visualizer:latest   "/sbin/tini -- node …"   12 hours ago   Up 12 hours (healthy)    8080/tcp
      visualizer.1.uryxt4n42ex8rzwzyqavbo27v
388ca7dc94e4   portainer/portainer-ce:latest     "/portainer"             4 days ago     Up 4 days                8000/tcp, 9000/tcp, 9443/tcp
      portainer.1.teydo0kl5qo6tudk6bamfejhb
ubuntu@saipavan:~$
```

Result:

The application will easily acessible in manager machines and worker machines also.