Day 19 : Docker-compose, Docker-Network, Docker-volume for DevOps Engineers

This is #90DaysofDevops challenge under the guidance of Shubham Londhe sir.

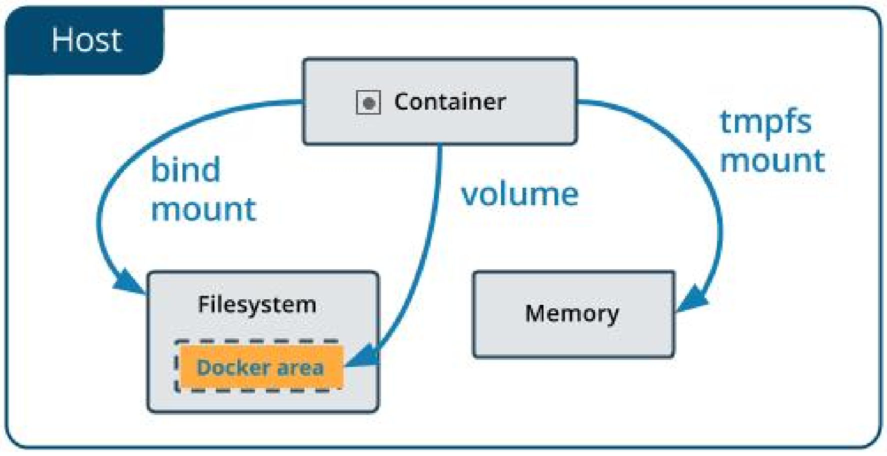
Day 19 TASK

check this for task:

https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day19/tasks.md

Docker-Volume

Docker allows you to create something called volumes. Volumes are like separate storage areas that can be accessed by containers. They allow you to store data, like a database, outside the container, so it doesn't get deleted when the container is deleted. You can also mount from the same volume and create more containers having same data.



Docker volumes are a way to persist data outside of containers. They allow data to be shared between containers and can be mounted to specific directories within a container.

Some commands used in docker volume :

- `docker volume create <name>`: Creates a new volume with the specified name.
- `docker volume ls`: Lists all volumes available on the Docker host.

- `docker volume inspect <name>`: Displays detailed information about the specified volume, including the mountpoint and driver used.
- `docker volume rm <name>`: Removes the specified volume. Note that the volume must be unmounted from all containers before it can be removed.

Docker Network

Docker allows you to create virtual spaces called networks, where you can connect multiple containers (small packages that hold all the necessary files for a specific application to run) together. This way, the containers can communicate with each other and with the host machine (the computer on which the Docker is installed).

Docker networks allow containers to communicate with each other and with external networks. By default, each container is connected to a bridge network, but Docker also provides several other types of networks, including host and overlay networks.

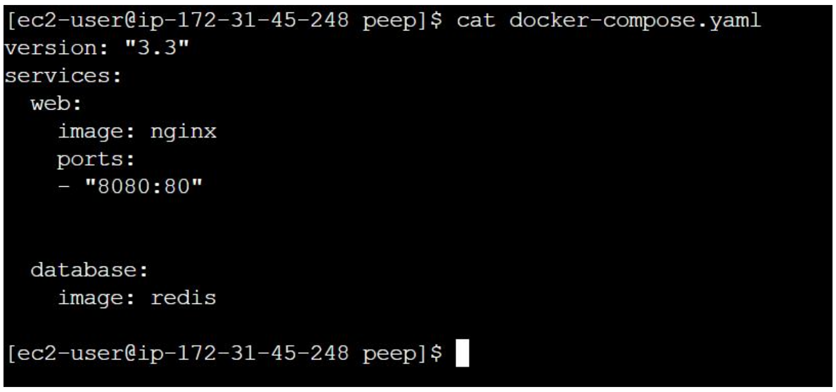
When we run a container, it has its own storage space that is only accessible by that specific container. If we want to share that storage space with other containers, we can’t do that.

Some commands used in Docker networks :

- `docker network ls`: Lists all networks available on the Docker host.
- `docker network create <name>`: Creates a new network with the specified name and driver.
- `docker network inspect <name>`: Displays detailed information about the specified network, including the containers and endpoints connected to it.
- `docker network rm <name>`: Removes the specified network. Note that the network must be disconnected from all containers before it can be removed.

Task-1

- Create a multi-container docker-compose file which will bring *UP* and bring *DOWN* containers in a single shot (Example — Create application and database container)



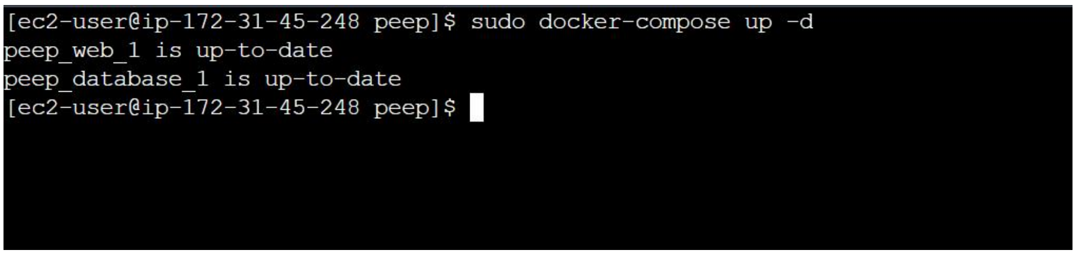
```
[ec2-user@ip-172-31-45-248 peep]$ cat docker-compose.yaml
version: "3.3"
services:
  web:
    image: nginx
    ports:
    - "8080:80"


  database:
    image: redis

[ec2-user@ip-172-31-45-248 peep]$
```

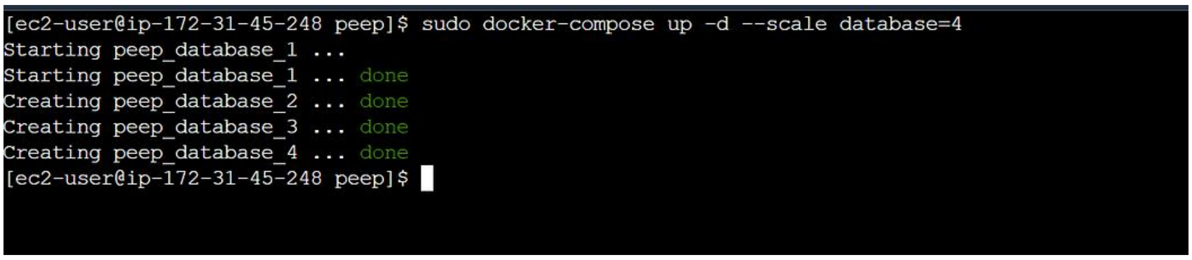
hints:

- Use the `docker-compose up` command with the `-d` flag to start a multi-container application in detached mode.



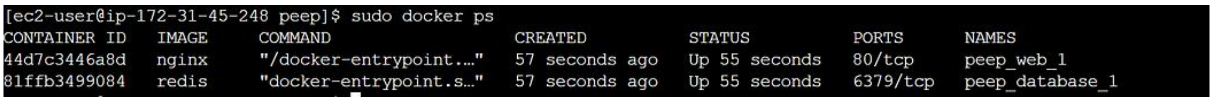
```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker-compose up -d
peep_web_1 is up-to-date
peep_database_1 is up-to-date
[ec2-user@ip-172-31-45-248 peep]$
```

- Use the `docker-compose scale` command to increase or decrease the number of replicas for a specific service. You can also add `replicas` in deployment file for *auto-scaling*.



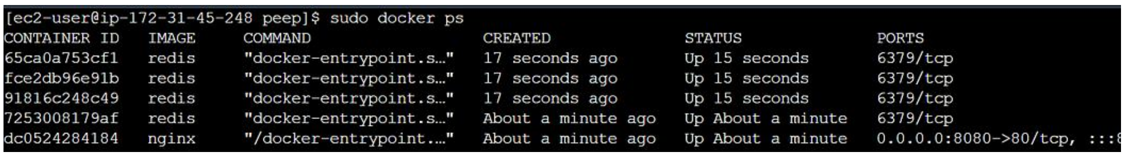
```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker-compose up -d --scale database=4
Starting peep_database_1 ...
Starting peep_database_1 ... done
Creating peep_database_2 ... done
Creating peep_database_3 ... done
Creating peep_database_4 ... done
[ec2-user@ip-172-31-45-248 peep]$
```

- Use the `docker-compose ps` command to view the status of all containers, and `docker-compose logs` to view the logs of a specific service.

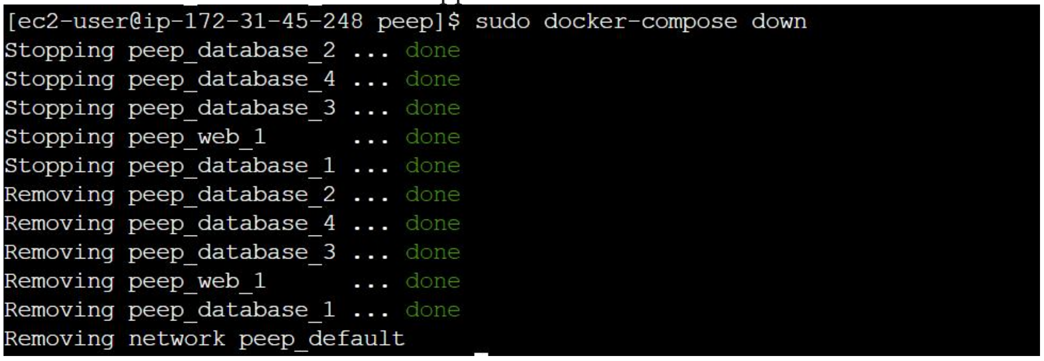


```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS      NAMES
44d7c3446a8d   nginx     "/docker-entrypoint.…"   57 seconds ago   Up 55 seconds   80/tcp     peep_web_1
81ffb3499084   redis     "docker-entrypoint.s…"   57 seconds ago   Up 55 seconds   6379/tcp   peep_database_1
```

After Scaling the database ,


```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS
65ca0a753cf1   redis     "docker-entrypoint.s…"   17 seconds ago       Up 15 seconds       6379/tcp
fce2db96e91b   redis     "docker-entrypoint.s…"   17 seconds ago       Up 15 seconds       6379/tcp
91816c248c49   redis     "docker-entrypoint.s…"   17 seconds ago       Up 15 seconds       6379/tcp
7253008179af   redis     "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp
dc0524284184   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::
```

- Use the `docker-compose down` command to stop and remove all containers, networks, and volumes associated with the application.

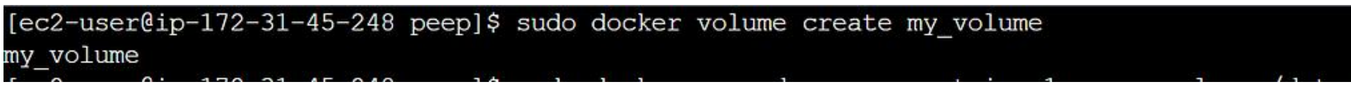

```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker-compose down
Stopping peep_database_2 ... done
Stopping peep_database_4 ... done
Stopping peep_database_3 ... done
Stopping peep_web_1      ... done
Stopping peep_database_1 ... done
Removing peep_database_2 ... done
Removing peep_database_4 ... done
Removing peep_database_3 ... done
Removing peep_web_1      ... done
Removing peep_database_1 ... done
Removing network peep_default
```

Task-2

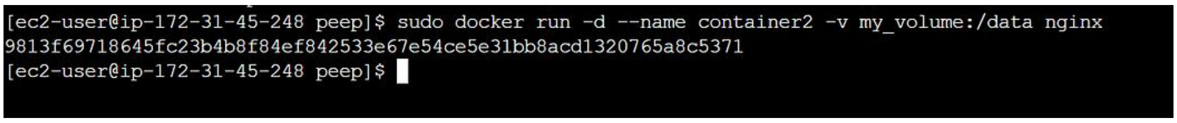
- **Learn how to use Docker Volumes and Named Volumes to share files and directories between multiple containers.**

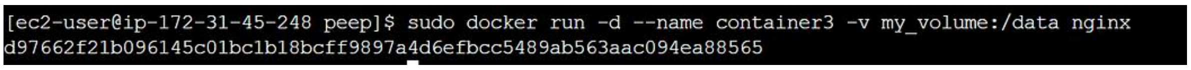
Create a Docker volume:

```
docker volume create my_volume_name
```


```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker volume create my_volume
my_volume
```

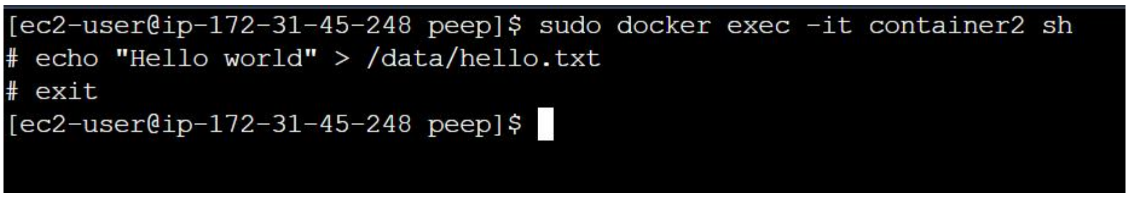
Start 2 containers and mount the volume:


```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker run -d --name container2 -v my_volume:/data nginx
9813f69718645fc23b4b8f84ef842533e67e54ce5e31bb8acd1320765a8c5371
[ec2-user@ip-172-31-45-248 peep]$
```


```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker run -d --name container3 -v my_volume:/data nginx
d97662f21b096145c01bc1b18bcff9897a4d6efbcc5489ab563aac094ea88565
```

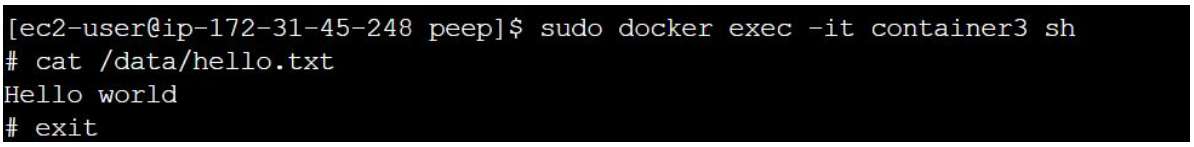
Share files between containers: Any files that are written to the `/data` directory in either container will be shared between the two containers, since they are both mounted to the same volume.

For example, you can create a file in `container1`:



```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker exec -it container2 sh
# echo "Hello world" > /data/hello.txt
# exit
[ec2-user@ip-172-31-45-248 peep]$
```

Now we can read the same file in the another container i.e. container3



```
[ec2-user@ip-172-31-45-248 peep]$ sudo docker exec -it container3 sh
# cat /data/hello.txt
Hello world
# exit
```

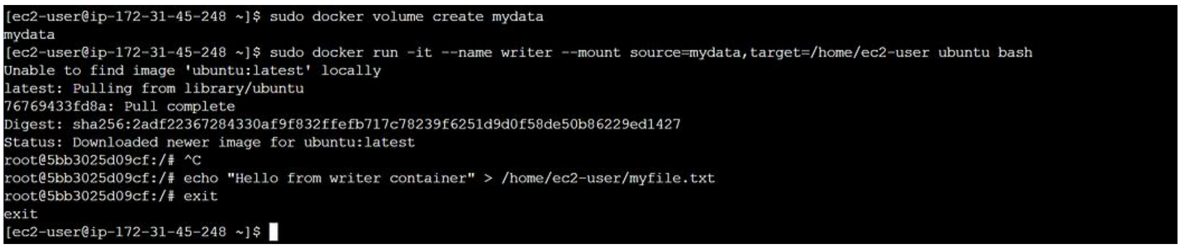
- Create two or more containers that read and write data to the same volume using the `docker run --mount` command.

Create a volume using the `docker volume create` command.

```
docker volume create mydata
```

Create a container that writes data to the volume using the `docker run` command.

```
docker run -it --name writer --mount source=mydata,target=/home/ec2-user ubuntu bash
```



```
[ec2-user@ip-172-31-45-248 ~]$ sudo docker volume create mydata
mydata
[ec2-user@ip-172-31-45-248 ~]$ sudo docker run -it --name writer --mount source=mydata,target=/home/ec2-user ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
76769433fd8a: Pull complete
Digest: sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427
Status: Downloaded newer image for ubuntu:latest
root@5bb3025d09cf:/# ^C
root@5bb3025d09cf:/# echo "Hello from writer container" > /home/ec2-user/myfile.txt
root@5bb3025d09cf:/# exit
exit
[ec2-user@ip-172-31-45-248 ~]$
```

create a container that reads data from the same volume using the `docker run` command.

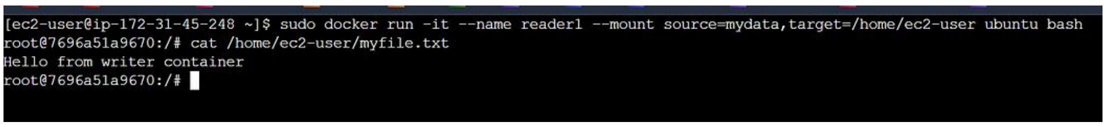
```
docker run -it --name reader --mount source=mydata,target=/home/ec2-user ubuntu bash
```

Inside the `writer` container, create a file in the /home/ec2-user directory.

```
echo "Hello from writer container" > /home/ec2-user/myfile.txt
```

Inside the `reader` container, read the file from the `/app/data` directory.

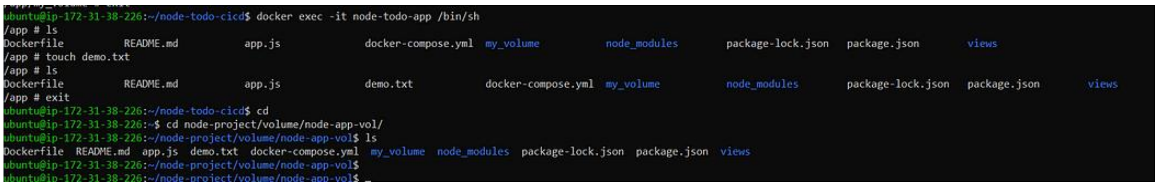
```
cat /home/ec2-user/myfile.txt
```



```
[ec2-user@ip-172-31-45-248 ~]$ sudo docker run -it --name reader1 --mount source=mydata,target=/home/ec2-user ubuntu bash
root@7696a51a9670:/# cat /home/ec2-user/myfile.txt
Hello from writer container
root@7696a51a9670:/# 
```

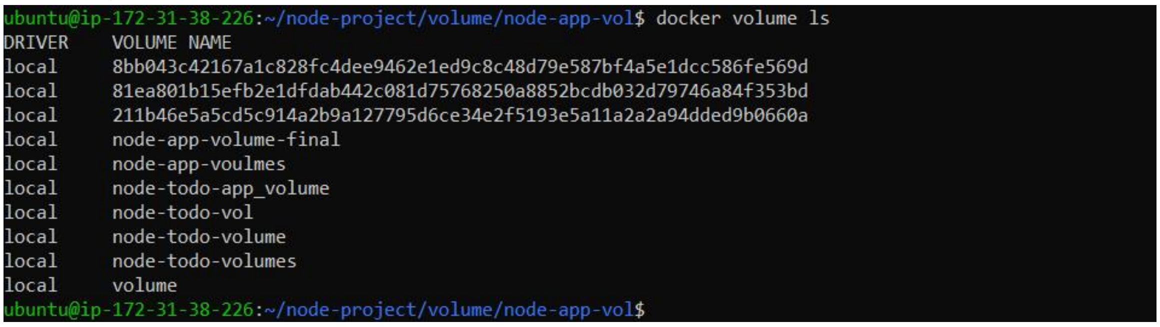
This confirms that the `writer` container has successfully written data to the `mydata` volume, and the `reader` container has successfully read the data from the same volume.

- **Verify that the data is the same in all containers by using the docker exec command to run commands inside each container.**



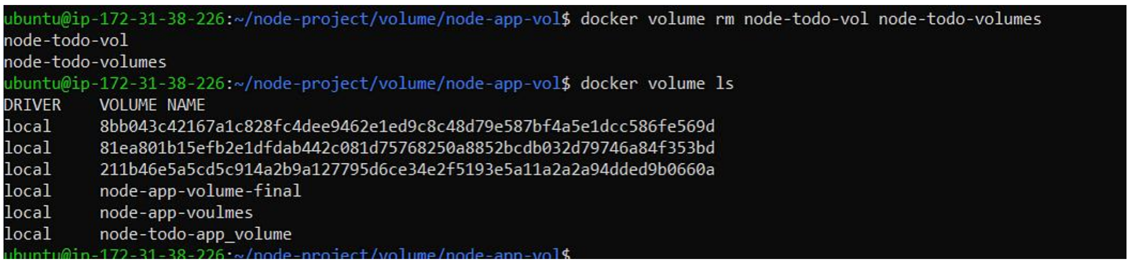
```
ubuntu@ip-172-31-38-226:~/node-todo-cicd$ docker exec -it node-todo-app /bin/sh
/app # ls
Dockerfile          README.md           app.js              docker-compose.yml  my_volume           node_modules        package-lock.json   package.json        views
/app # touch demo.txt
/app # ls
Dockerfile          README.md           app.js              demo.txt            docker-compose.yml  my_volume           node_modules        package-lock.json   package.json        views
/app # exit
ubuntu@ip-172-31-38-226:~/node-todo-cicd$ cd
ubuntu@ip-172-31-38-226:~$ cd node-project/volume/node-app-vol/
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$ ls
Dockerfile  README.md  app.js  demo.txt  docker-compose.yml  my_volume  node_modules  package-lock.json  package.json  views
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$ _
```

- **Use the docker volume ls command to list all volumes**



```
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$ docker volume ls
DRIVER    VOLUME NAME
local     8bb043c42167a1c828fc4dee9462e1ed9c8c48d79e587bf4a5e1dcc586fe569d
local     81ea801b15efb2e1dfdab442c081d75768250a8852bcdb032d79746a84f353bd
local     211b46e5a5cd5c914a2b9a127795d6ce34e2f5193e5a11a2a2a94dded9b0660a
local     node-app-volume-final
local     node-app-voulmes
local     node-todo-app_volume
local     node-todo-vol
local     node-todo-volume
local     node-todo-volumes
local     volume
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$
```

- **Use the docker volume ls command to list all volumes and docker volume rm command to remove the volume when you’re done.**



```
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$ docker volume rm node-todo-vol node-todo-volumes
node-todo-vol
node-todo-volumes
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$ docker volume ls
DRIVER    VOLUME NAME
local     8bb043c42167a1c828fc4dee9462e1ed9c8c48d79e587bf4a5e1dcc586fe569d
local     81ea801b15efb2e1dfdab442c081d75768250a8852bcdb032d79746a84f353bd
local     211b46e5a5cd5c914a2b9a127795d6ce34e2f5193e5a11a2a2a94dded9b0660a
local     node-app-volume-final
local     node-app-voulmes
local     node-todo-app_volume
ubuntu@ip-172-31-38-226:~/node-project/volume/node-app-vol$
```

If this post was helpful, please do follow and click the clap

_Thank you for reading

_Rajani