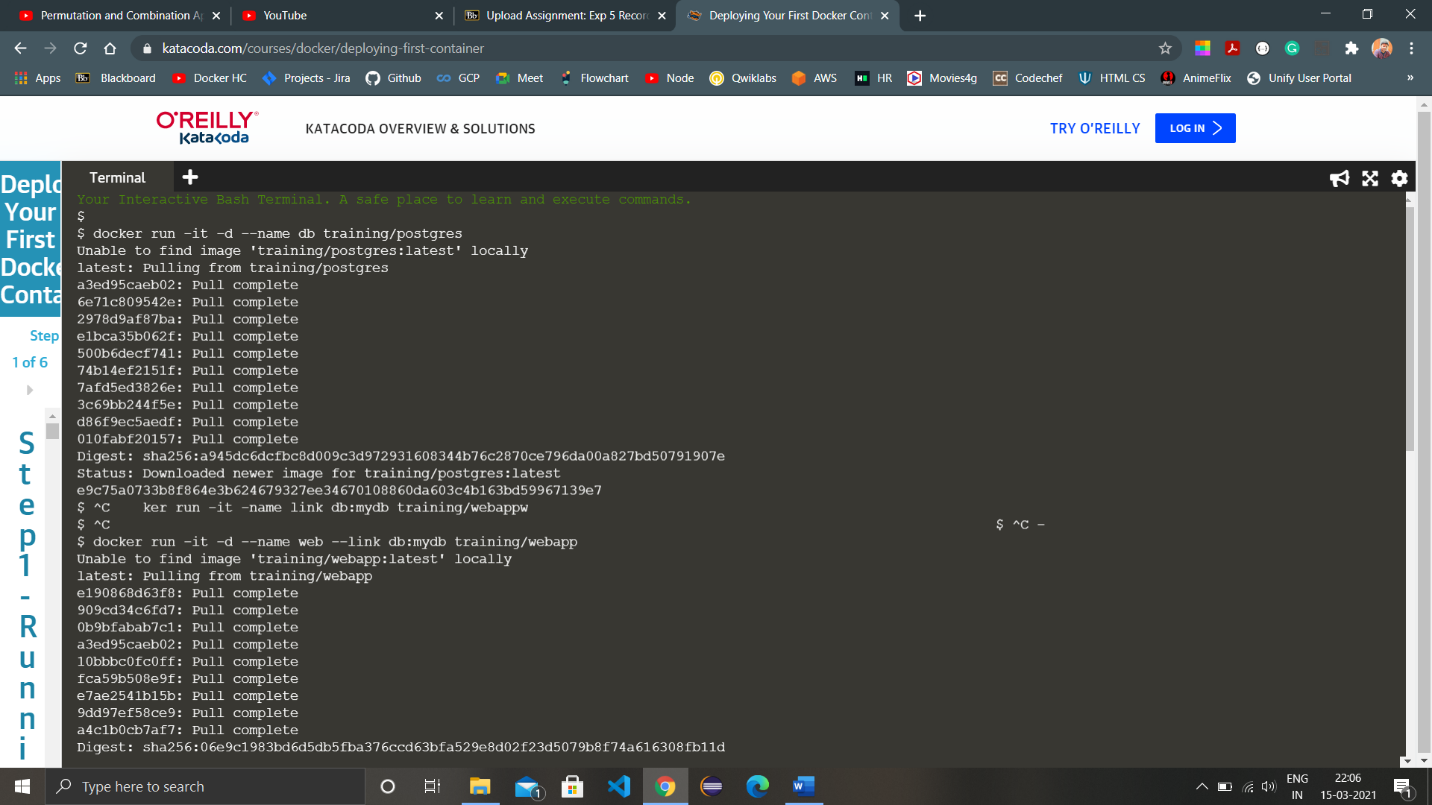
**Experiment 6:**

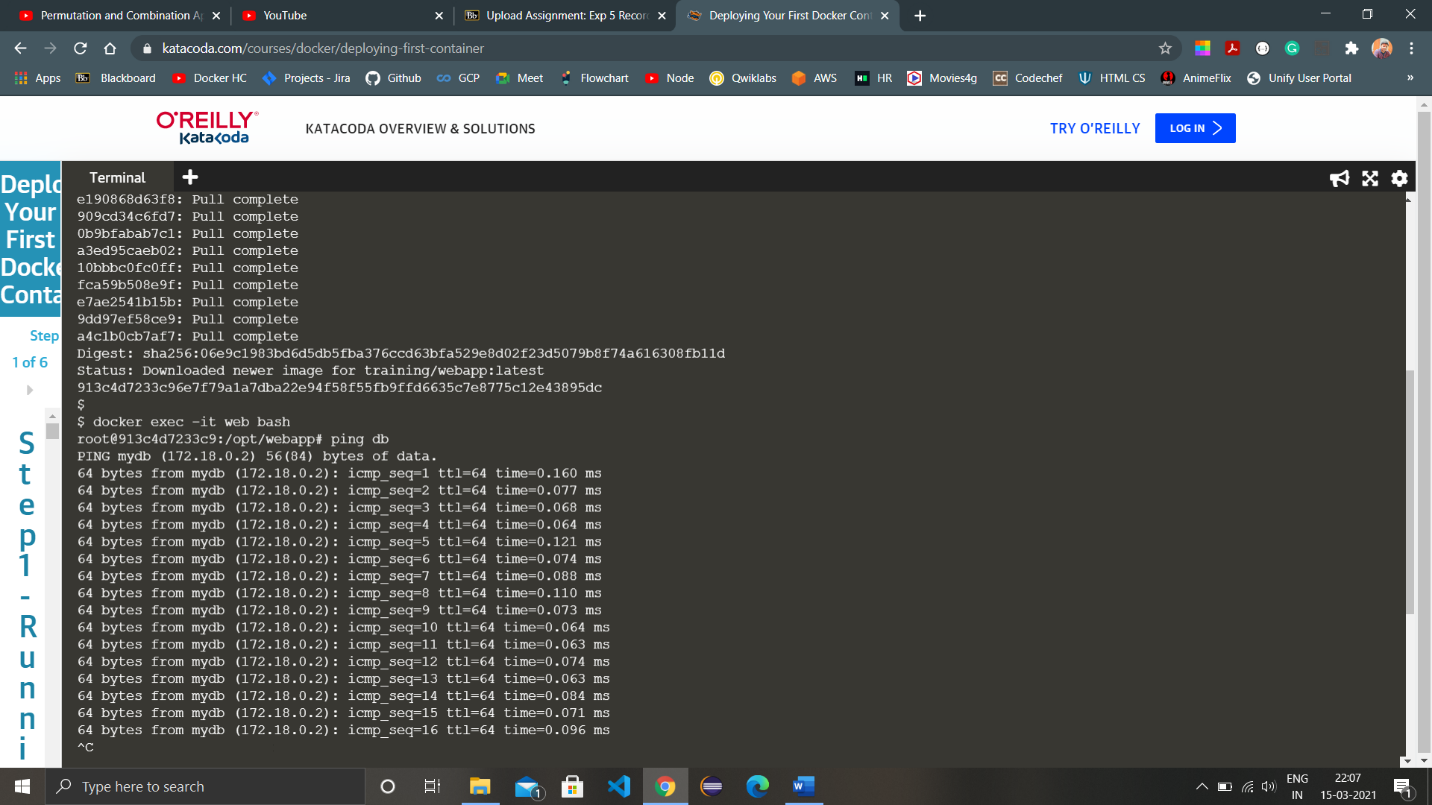
Docker Links and Swarm

**Docker Links:**

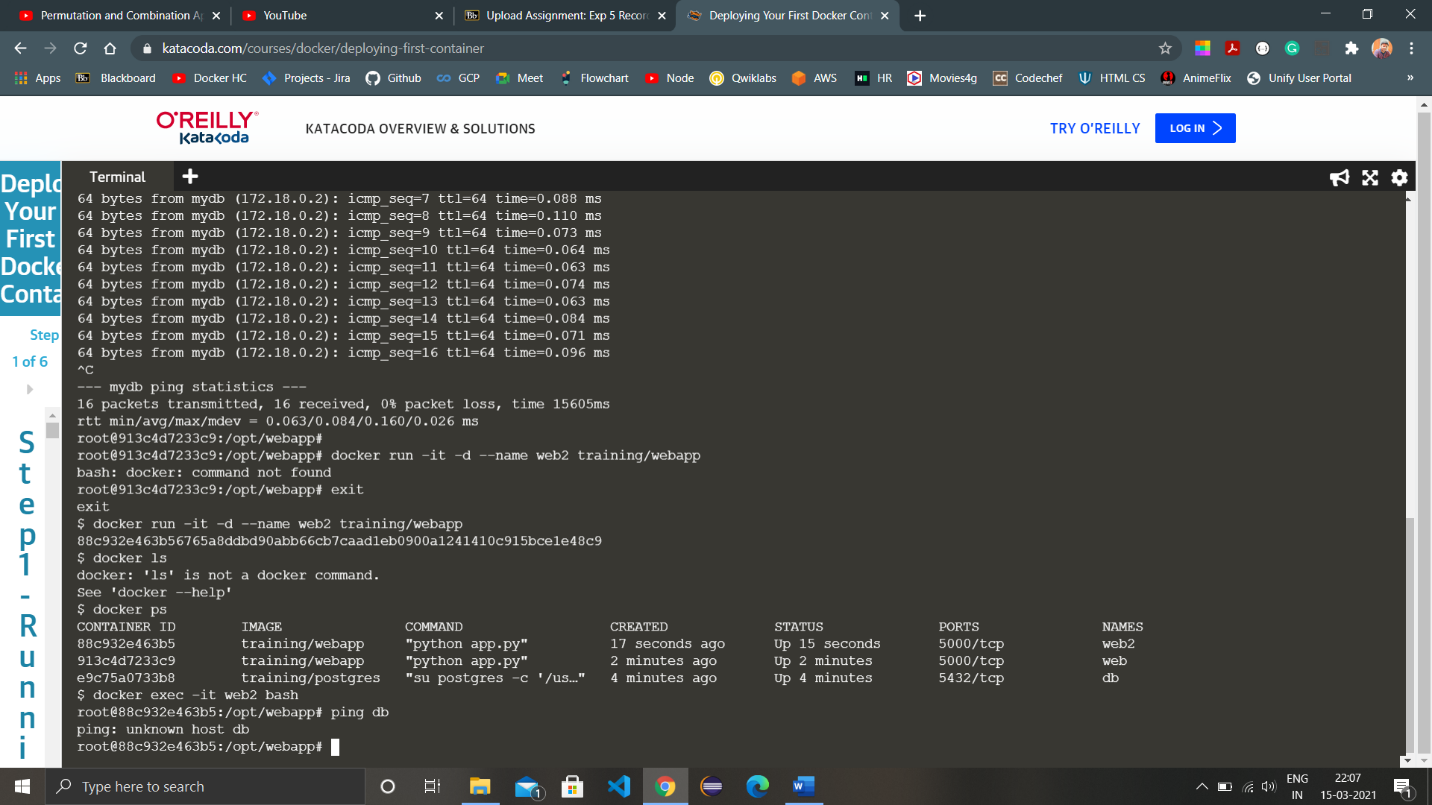
1. Open the terminal and run a postgres container by the command ‘docker run -it -d –name db training/postgres’.
2. Once the command execution is complete and the pulling of the image gets over, run another container of image webapp linking it with the db container we just created by the command ‘docker run -it -d –name web –link db:mydb training/webapp’.



1. Once the other image is pulled completely and the container is running then switch to the terminal of the container. Command for logging in or switching to container terminal is ‘docker exec -it <container-name> bash’. Here the command would be ‘docker exec -it web bash’.
2. After successfully logging in to the container terminal, ping the container with which we linked the current container and the ping command will run successfully.



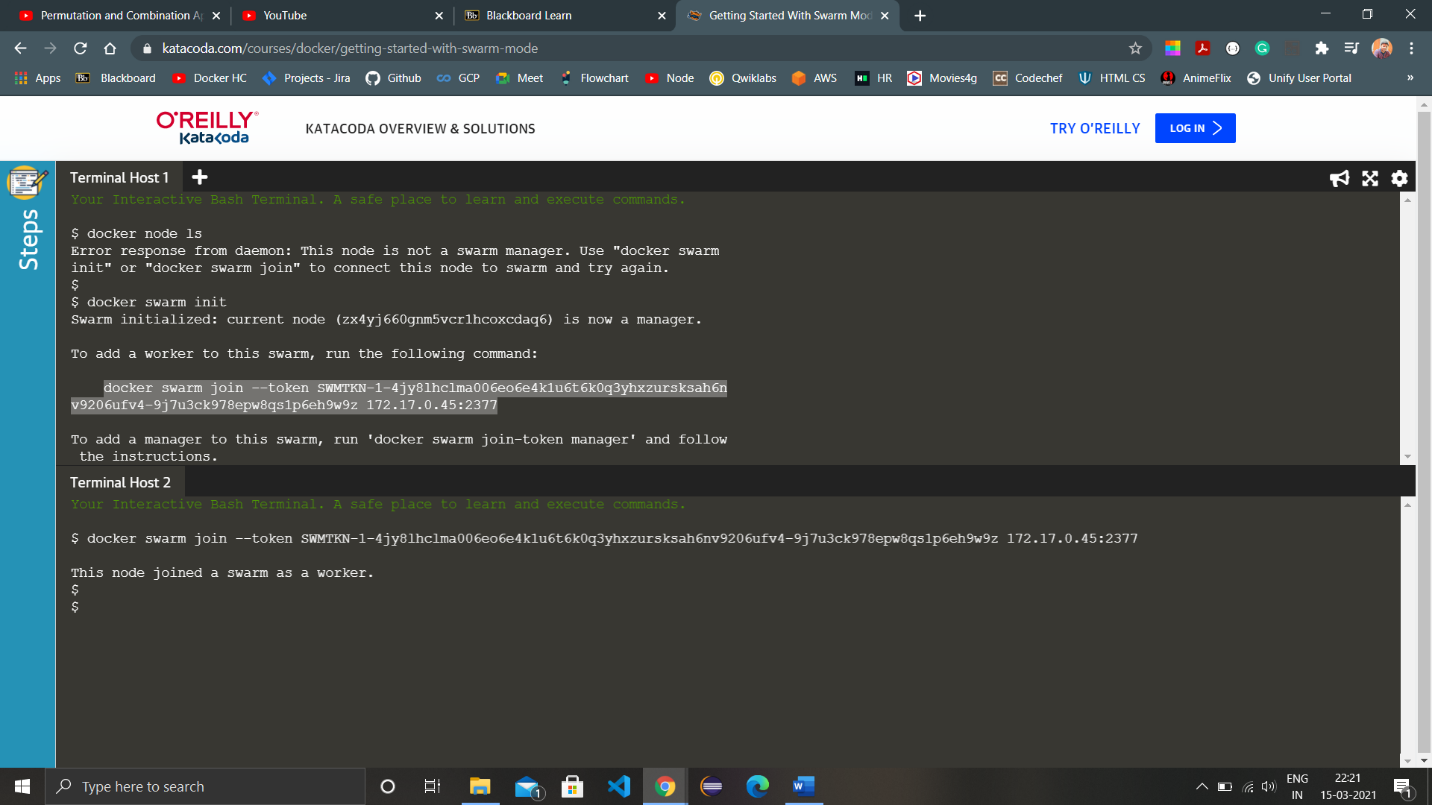
1. Now run another container without linking it to any of the previous containers and check all the containers with the command ‘docker ps’.
2. Login to the container again with which we ran the ping command once and try pinging the newly created container, the ping command will fail because the container isn’t linked with the other container.



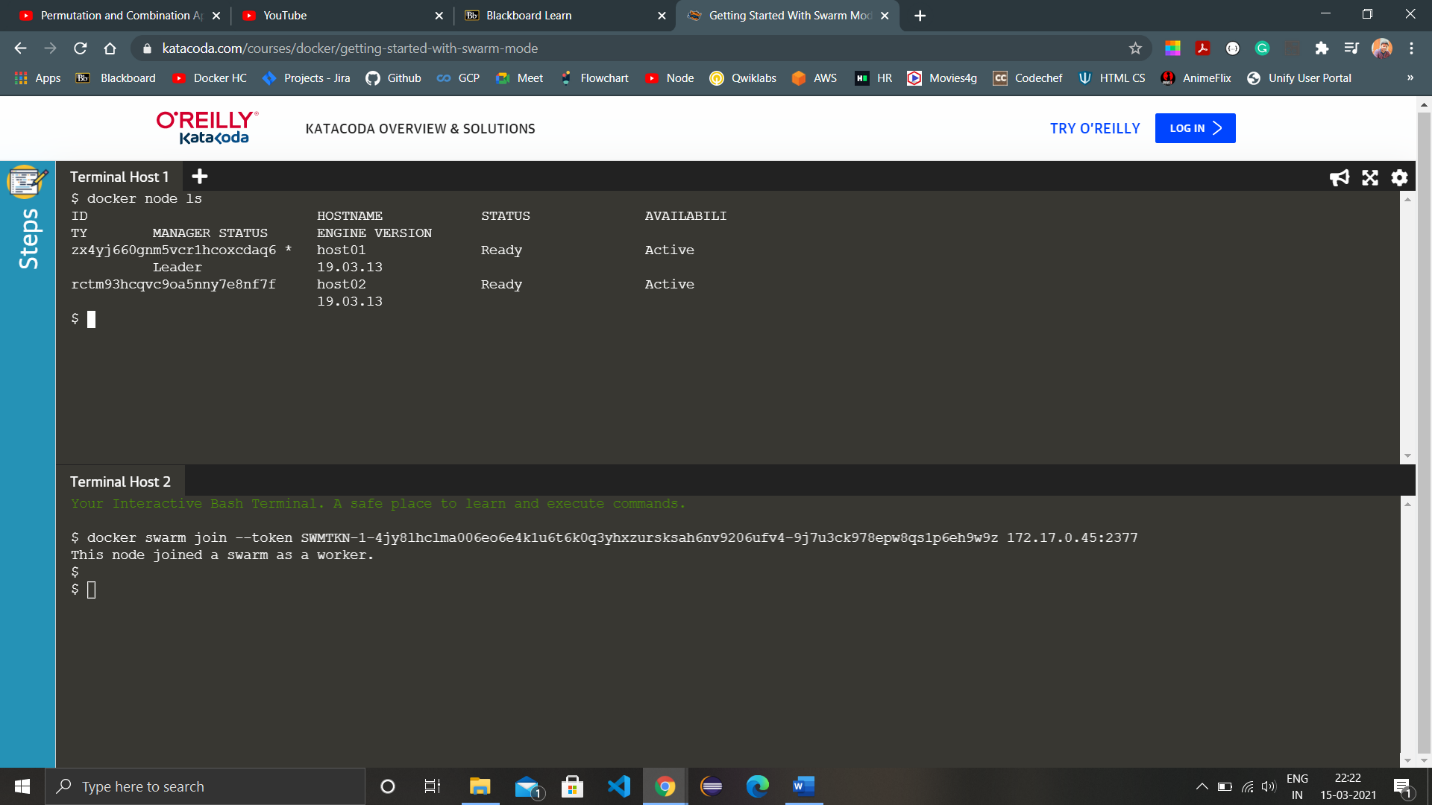
**Docker Swarm:**

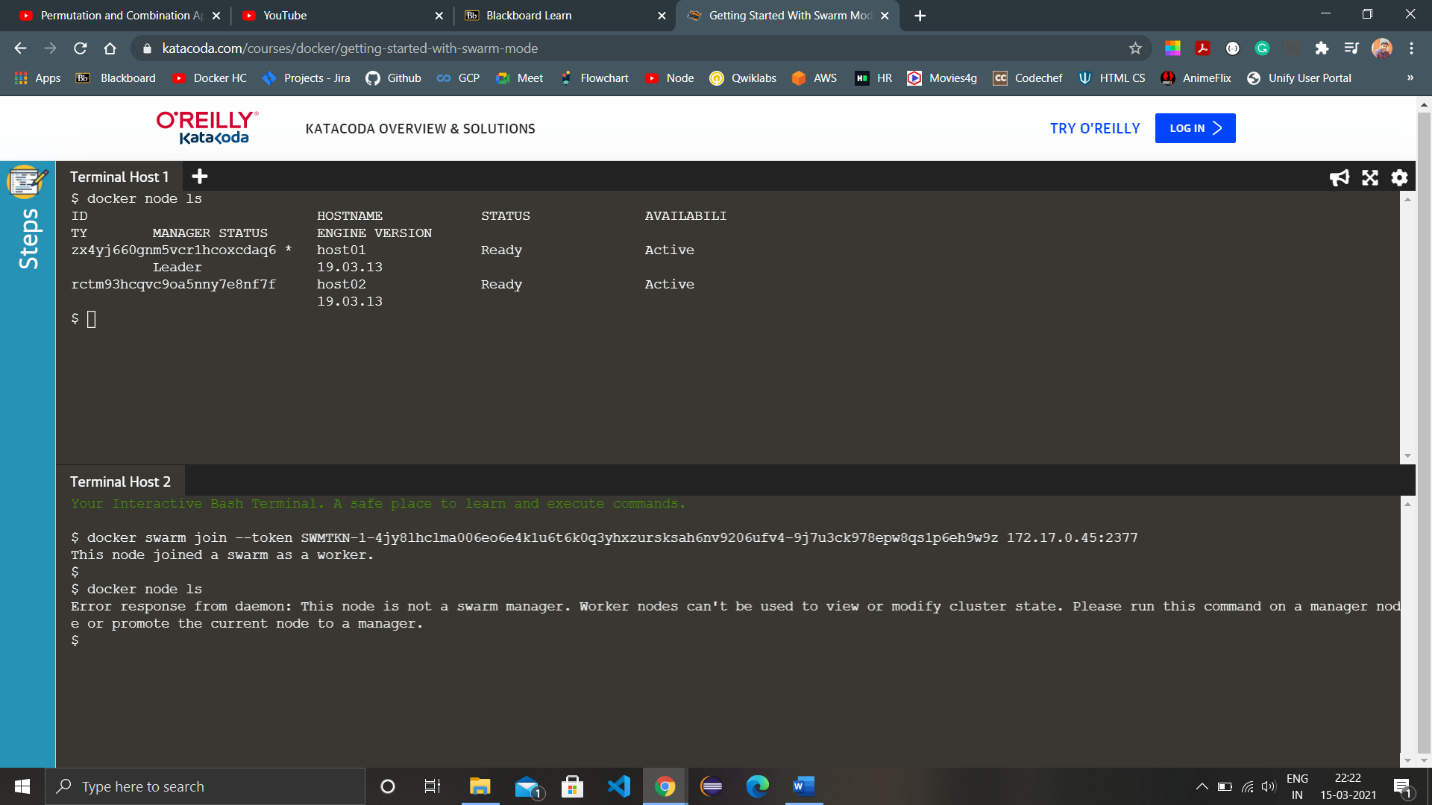
1. Open two terminals one for being the master and other for being the slave (node), check if swarm is active or not by the command ‘docker node ls’. If the swarm is active the command will list the master and the slaves but if the swarm isn’t active the command will throw an error.
2. Activate the swarm using the command ‘docker swarm init’, the output of the command will have a command having unique token and the ip address. Through this command we can connect the other terminal as a worker to the master by running the command in the terminal.

Basic syntax of the command from the output is ‘docker swarm join –token <token> <ip and port no.>’

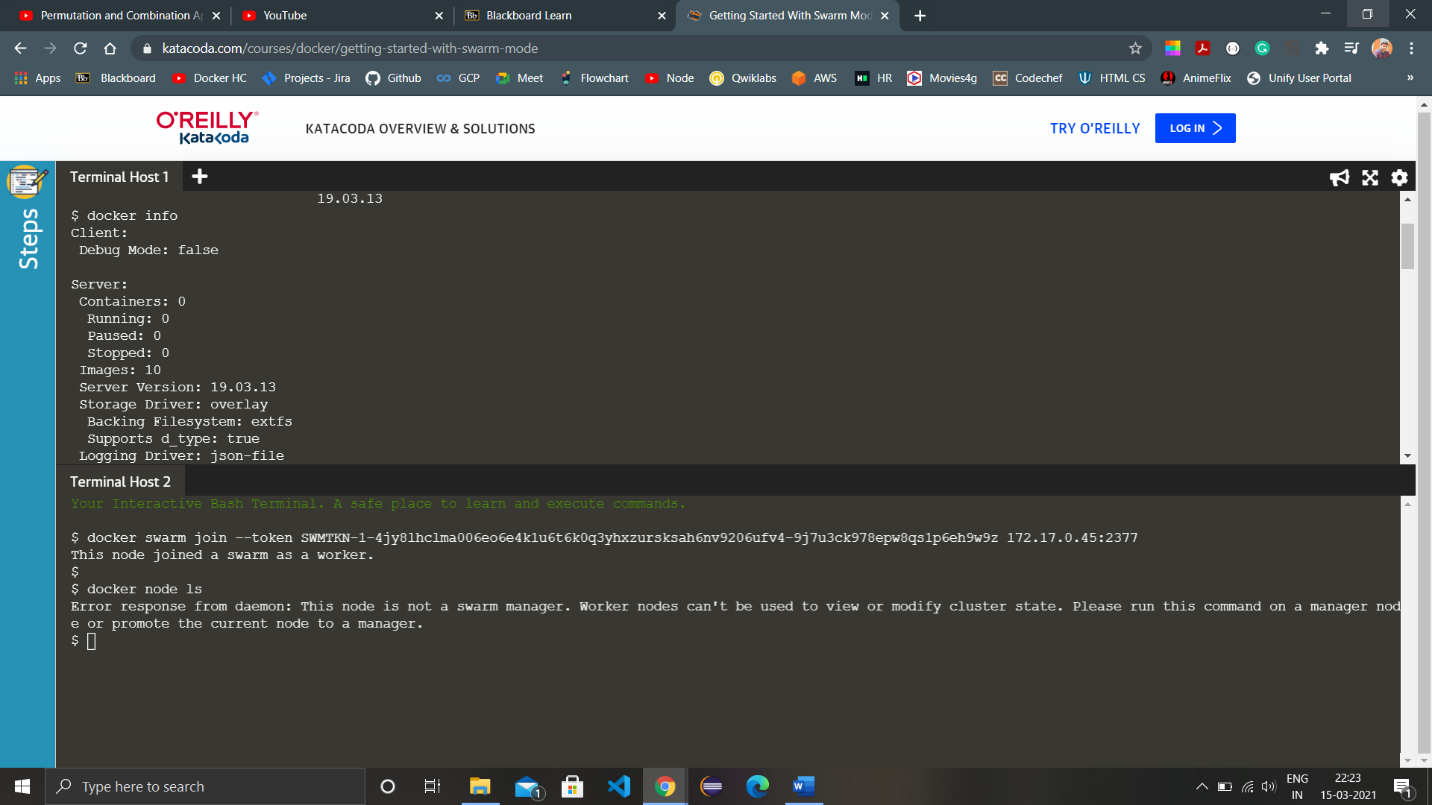


1. Again run the command ‘docker node ls’ to view all the nodes in the swarm we just activated, the nodes which are acting as worker will be written directly but the node which is acting as a manager will have Leader written with it and a ‘\*’ mark on it.
2. Only the manager can access the information regarding the number of rows and nodes in the swarm so if we try to run the command in a worker node terminal, it will fail.





1. We can check all the details of the swarm and the nodes by running the command ‘docker info’.



1. When we want to remove a node from the swarm we use the command ‘docker swarm leave --force’ if the node is a manager and ‘docker swarm leave’ if it’s a worker node. If a manager leaves the swarm, the manager properties are automatically inherited by a worker node and it becomes the new manager.

