NETWORK

**Network:**

Group of devices interconnected with each other using physical networks (Cables,wires,Swiches..etc) Wireless networks.

**Docker Networks:**

* Whenever we are creating container out of image container will be create below 3 networks
* Each container has some IP Address
* While creating containers if we don’t explicitly mentioned network name containers are getting created in default bridge.
* Bridge(default)
* host
* none/null

If required we can create custom bridge network

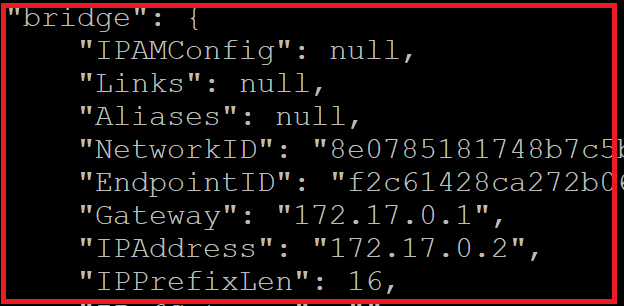
**Bridge Network (default bridge network):**

* Once micro service can communicate to another micro service if they are running same network
* Containers IPs are dynamic
* **Communication can happen using only IP Address it can’t communicate using names/hostnames**
* Since they are in same network they can communicate each other

docker network ls

docker inspect <container-name/Id>

docker inspect mypythonapp

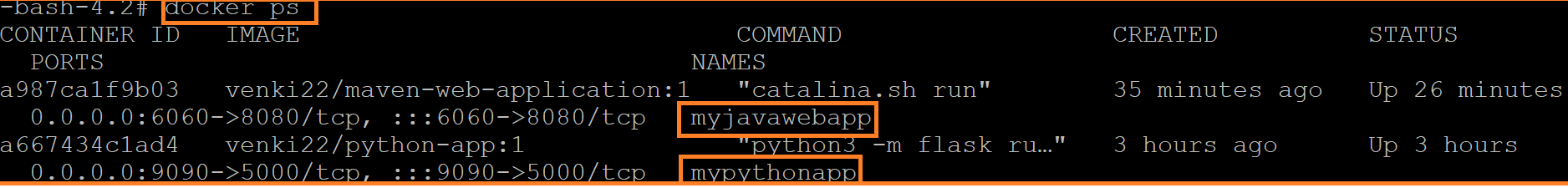


**Problem in Default bridge:**

* Communication can happened using only ***IP Address and*** *it can’t communicate using names/hostnames.*
* If same application deployed in other environment container IP will be change within bridge.
* Container IP’s are dynamic
* Within the code if the developer using container IP, in this case if any changes in IP’s of micro service developer has to re-build and re-modify the code.

**Communication to one container to other container using *default bridge network*:**

docker ps



**#To check container ip address**

docker inspect <containerName/Id>

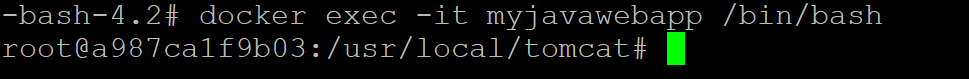
docker inspect myjavawebapp

**#Enter into inside the container**

docker exec -it <container-name> sh

docker exec -it <containerName/Id> /bin/bash

docker exec -it myjavawebapp /bin/bash

****

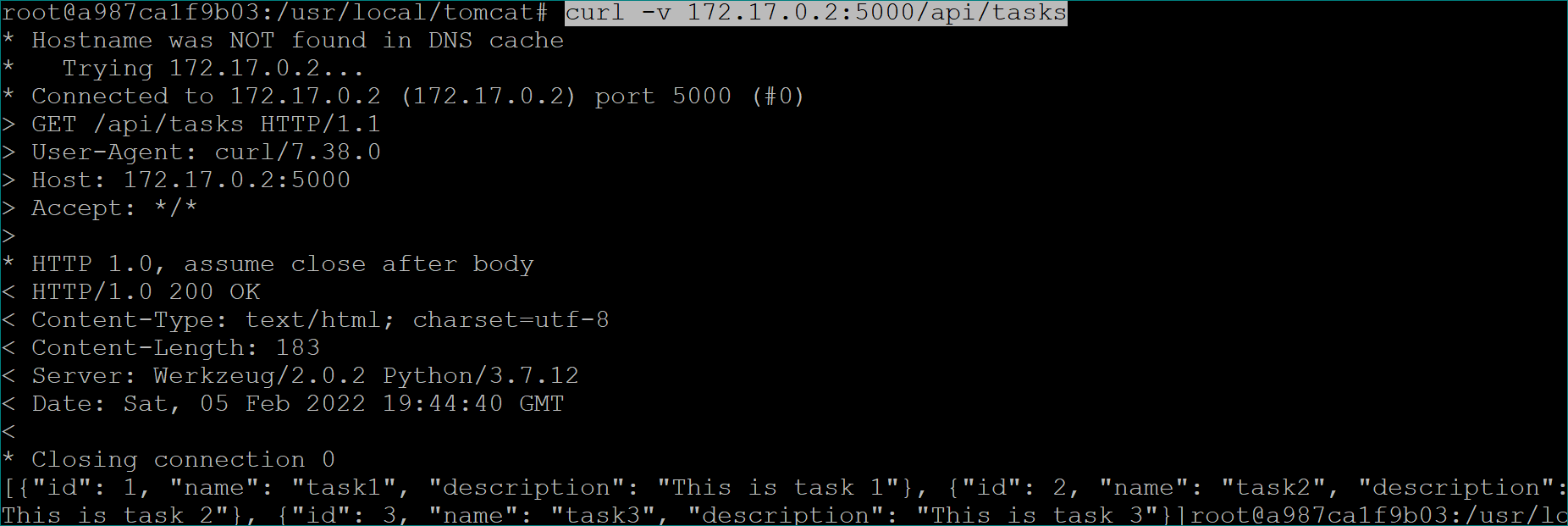
**Communicate to other container *(i.e mypythonapp)* using below command**

curl -vL telnet: //<contaienrIP>:<port>

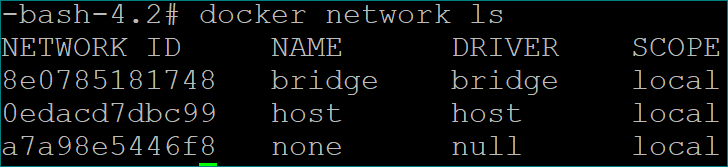
curl -vL <telnet://172.17.0.2:5000>

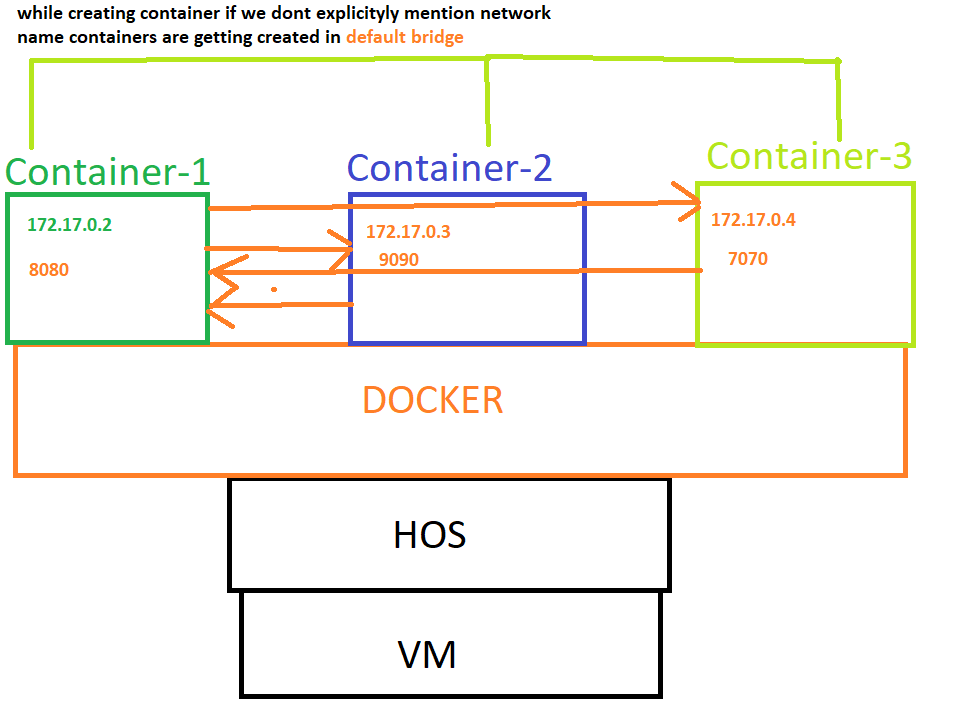
**#API call from inside container**

curl -v 172.17.0.2:5000/api/tasks

****

docker network ls



****

**Custom Bridge network:**

**#creating custom bridge network**

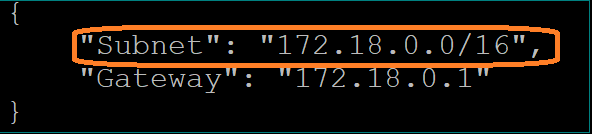
docker network create -d bridge <Custombridge-Name>

docker network create -d bridge flipkartbridge

* We can inspect the network also using below command
* We can see container network CIDR***(“Subnet”)***
* We can create containers in within CIDR range if required

docker network inspect <network-name>

docker network inspect flipkartbridge



* Within the docker port-mapping not required, port-mapping is required if we want to communicate outside the network.
* If we don’t mentioned network name container will be created in ***default bridge network***
* *Using custom-bridge network containers can communicate each other with IP address/container name as well*
* *We can’t communicate one container to another container (i.e other network)*
* *Port mapping is for external access*

docker network ls

**syntax:**

**docker run -d --name <container\_name> -p <host-port>:<container-port> --network <network\_name> <image\_name>**

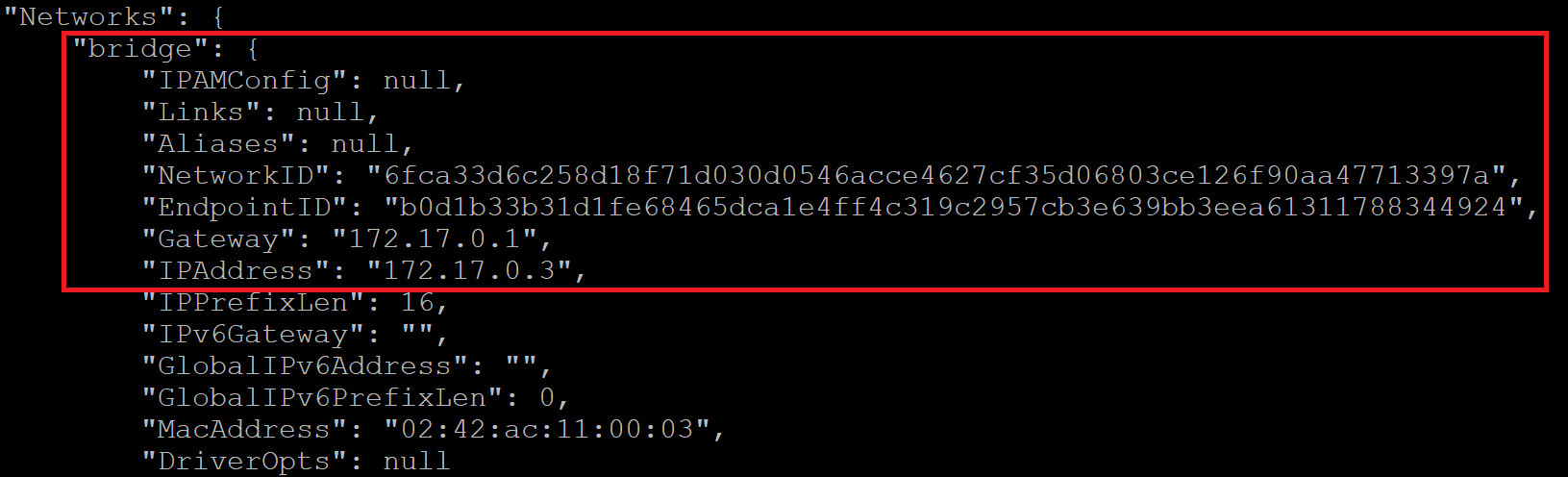
docker run -d ***--network flipkartbridge*** ***-p 3030:8080*** venki22/maven-web-application:1

**One container in multiple networks**

docker inspect <container-name>

**#here, container is having only one bridge network**

docker inspect mypythonapp

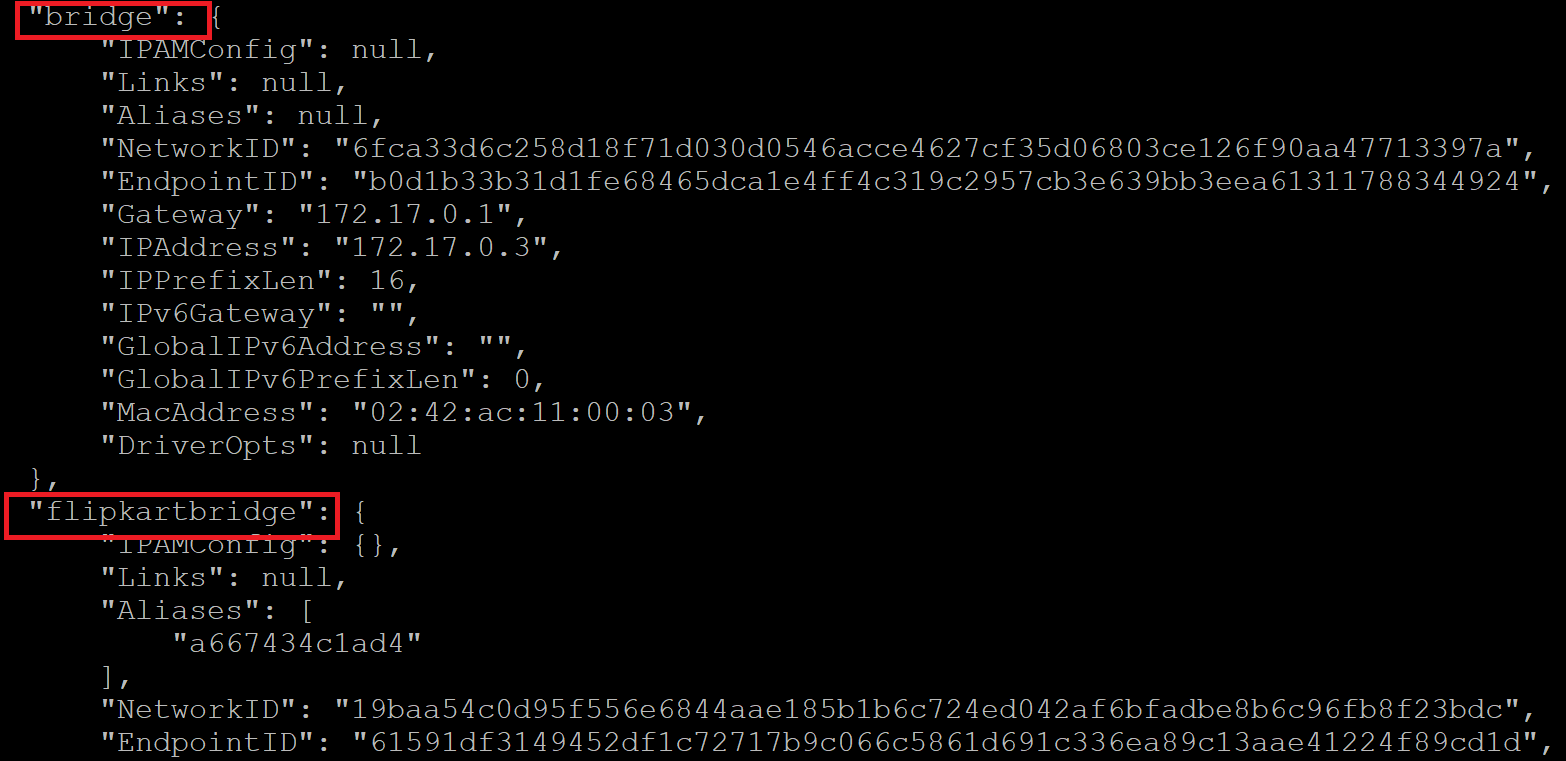


docker network connect <network-name> <container-name>

docker network disconnect ***flipkartbridge*** ***mypythonapp***

Here, *mypythonapp* container has 2 network **(i.e default bridge network and custom bridge n/w i.e *flipkartbridge)***

Docker inspect *mypythonapp*



Now, we can communicate *mypythonapp container using hostname and IP’s address as well.*

**Enter into custom bridge network container and use below commands for connectivity with hostname and IP address with other container**

*ping <IP Address>*

*ping 172.18.0.2*

*ping <Container-name>*

**We can disconnect network using below command**

docker network disconnect <network-name> <container – name>

docker network disconnect *flipkartbridge myjavawebapp*

**host network:**

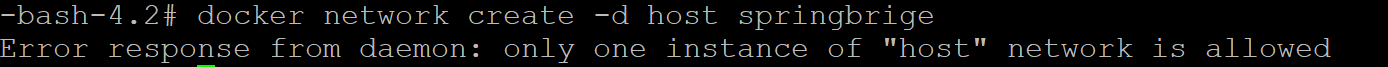
* If we create container on host ***network containers will not have IP address.***
* Containers will be created on host (Server) network.
* You no needs to do port publish/mapping in host network. Without port publishes you can access your container on host network.

**Creating host network:**

docker network create -d <driver-name> <custom-network-name>

docker network create -d host springbrige

**Note:** ***we can create only one instance of “host” network is allowed.***



We can create multiple bridge network

docker network create -d driver springbridge

docker network create -d driver flipkartbridge

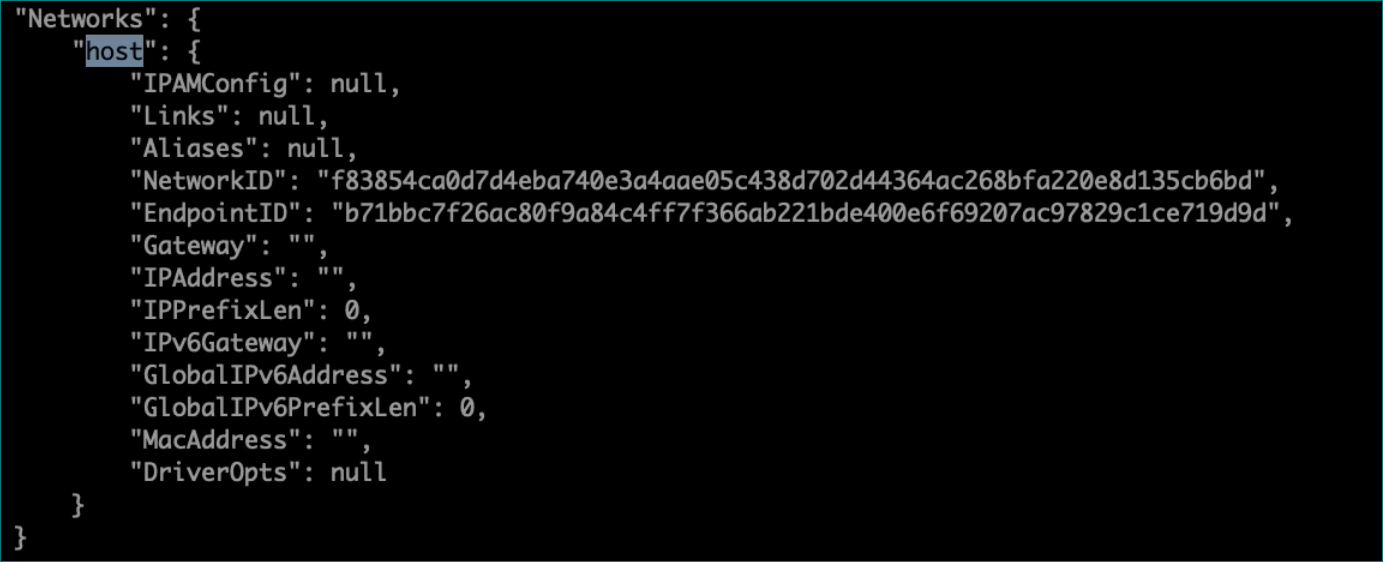
**Creating container using host network:**

docker run --name mvenapp-1 -d --network host venki22/maven-web-application:1

docker run --name mvenapp-2 -d --network host venki22/maven-web-application:1

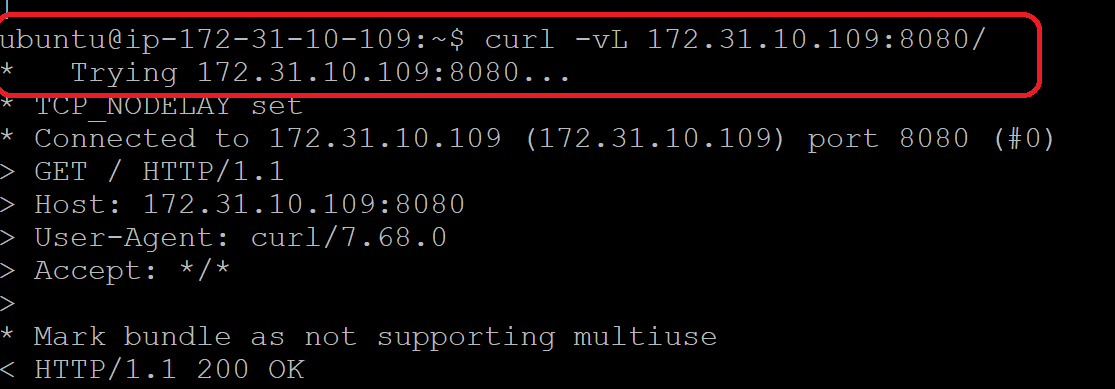
**Note:**

* Both containers are creating in host n/w we will get port conflict while accessing one application
* If we create container on host network containers will not have IP address.



We can access using server IP(i.e instance PUBLIC IP)

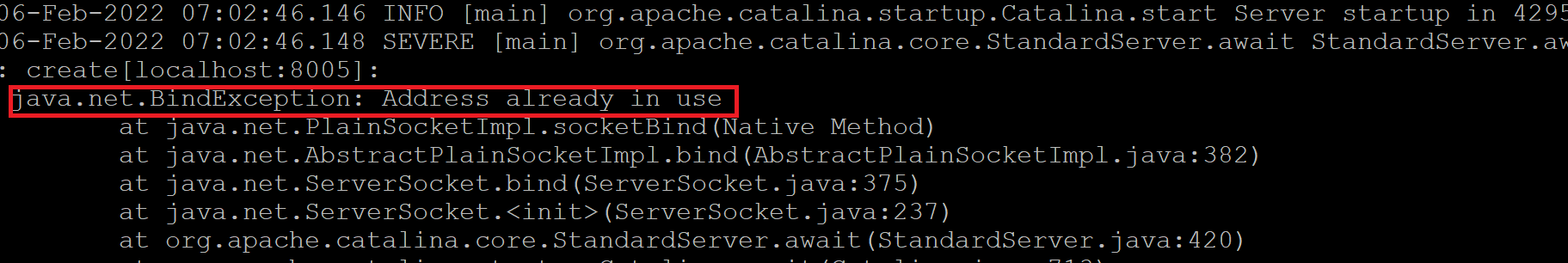
curl -vL 172.31.10.109:8080/



* Containers has directly created in host network, *we can’t create multiple container in same host network in this case other container will be Exited status while creating one container.*

docker logs <container-name>

docker logs mvenapp-2



* Go to inside running container

docker ps

#Enter into container

docker exec -it mvenapp-1 /bin/bash

#execute below command and verify response status

curl -vL 3.109.32.41:8080/maven-web-application/

* In real time, If we are using only docker we will use custom bridge
* Real time we have multiple docker servers

**None/null**

* If we create containers on ***none/null*** network containers will not have IP address. There is no network for that container so we can't access that container.
* In real time, we will go with container with container orchestration(Docker swarm or K8S or

openshift)

Multiport networking

* overlay network
* Network plugin under k8s
* CNI
* Caico
* flaneel
* weavnet

