

**EXPERIMENT-4**

**NAME- RAJAT PANWAR**

**SAP ID- 500069414**

**ROLL NO- R171218080**

**COURSE- B.TECH[CSE-DEVOPS]**

**SUBJECT- APPLICATION CONTAINERIZATION LAB**

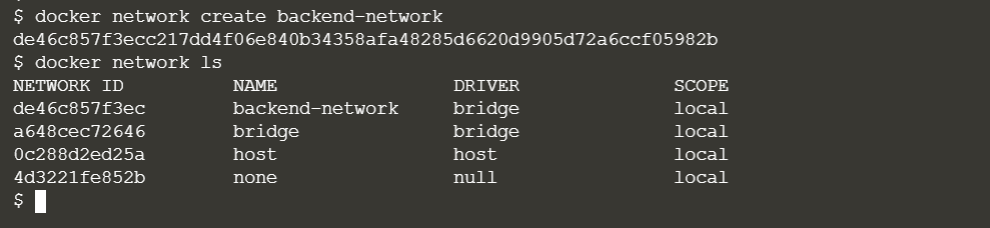
**SUBMITTED TO:-**

**MR. HITESH KUMAR SHARMA**

**Aim- Create the network and connect with the container**

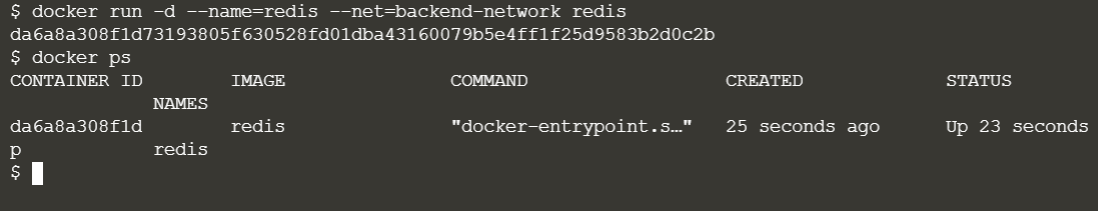
***Create Network:-***

Command:- docker network create backend-network



***Connect To Network:-***

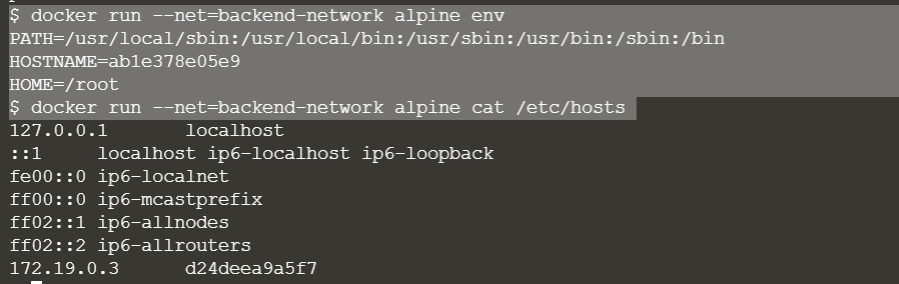
Command:- docker run -d --name=redis --net=backend-network redis



***Explore:-*** The first thing you'll notice is that Docker no longer assigns environment variables or updates the hosts file of containers. Explore using the following two commands and you'll notice it no longer mentions other containers.

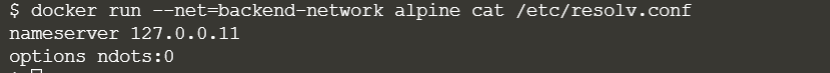
Command:- [1] $ docker run --net=backend-network alpine env

[2] $ docker run --net=backend-network alpine cat /etc/hosts

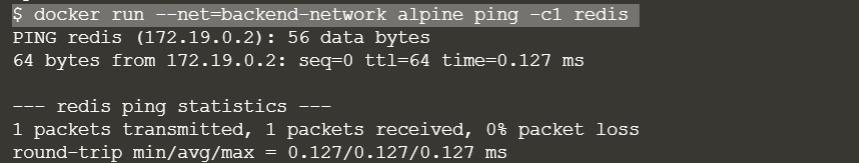


The way containers can communicate via an Embedded DNS Server in Docker.

This DNS server is assigned to all containers via the IP 127.0.0.11 and set in the *resolv.conf* file.

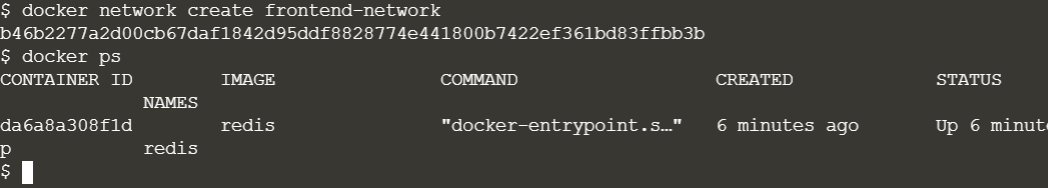


When containers access other container. The DNS server will return the IP of the container . In this case the name of redis will be *redis.backend-network*.



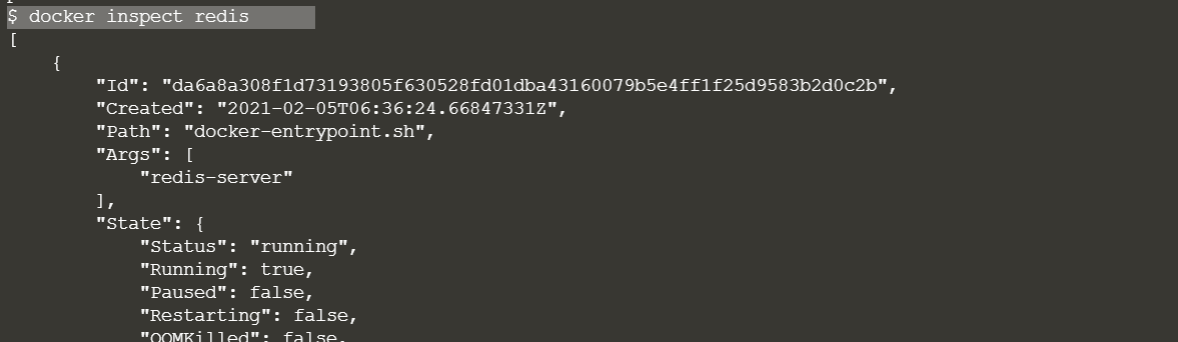
Again we create a new network with the help of this command which name is *frontend-network.*

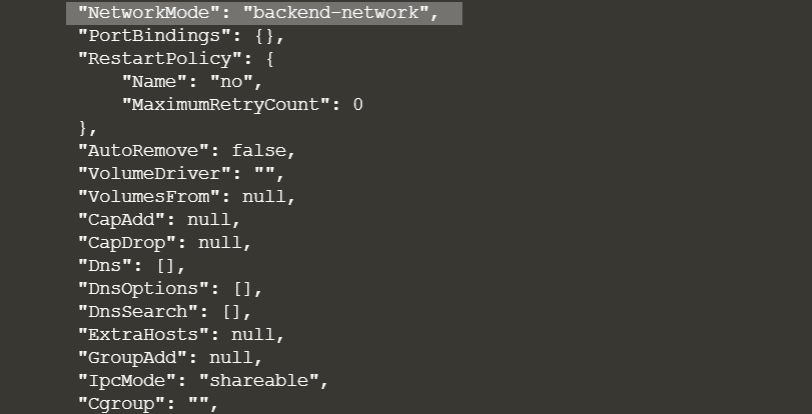
Command- $ docker network create frontend-network



* Command:- $docker inspect “container name”

Docker inspect provides detailed information on constructs controlled by Docker.



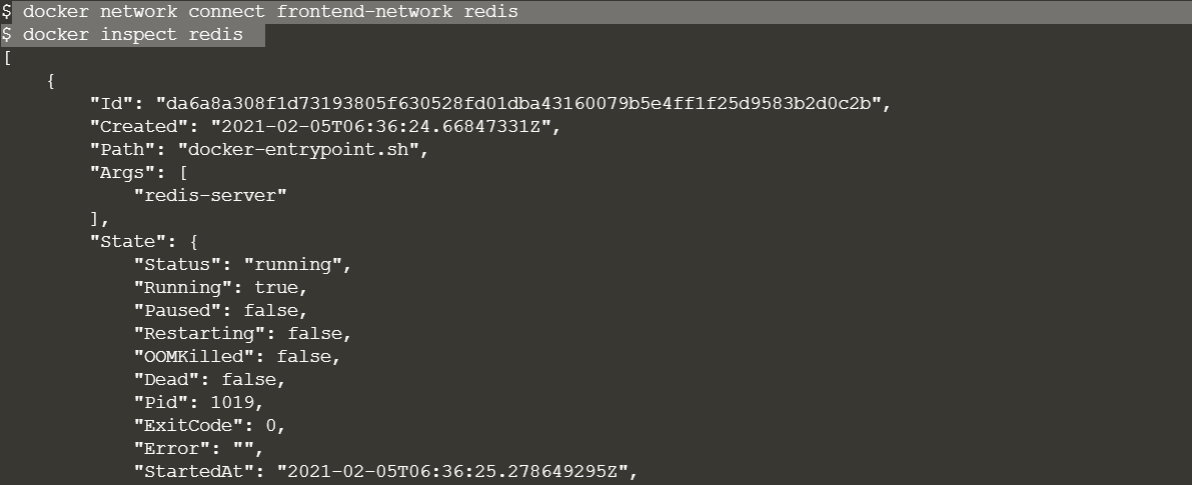


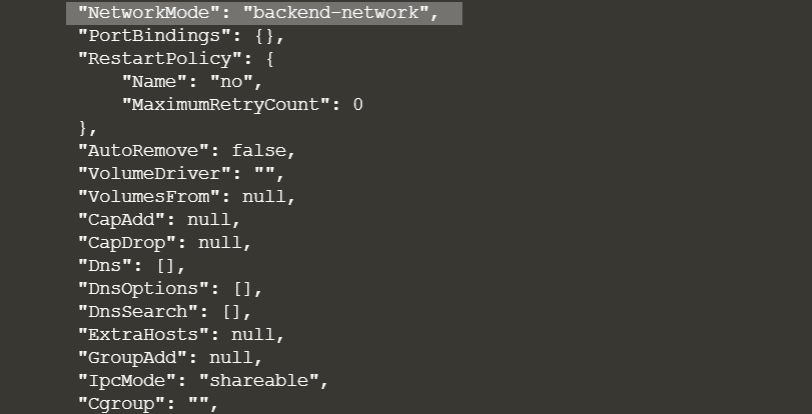
Command:- $ docker network connect frontend-network redis

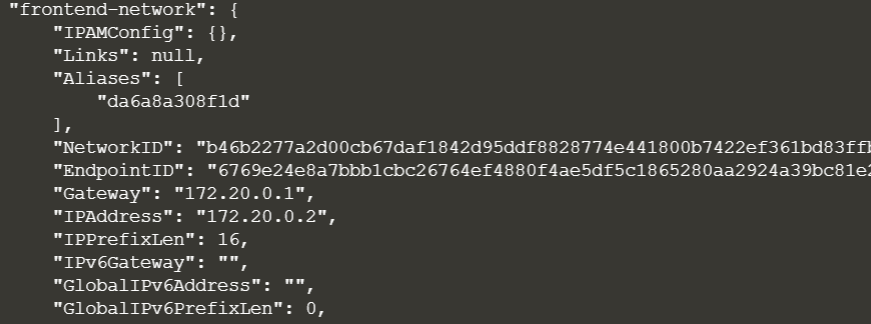
$ docker inspect redis

When using the *connect* command it is possible to attach existing containers to the network.

* Now we can see the backend and frontend network.



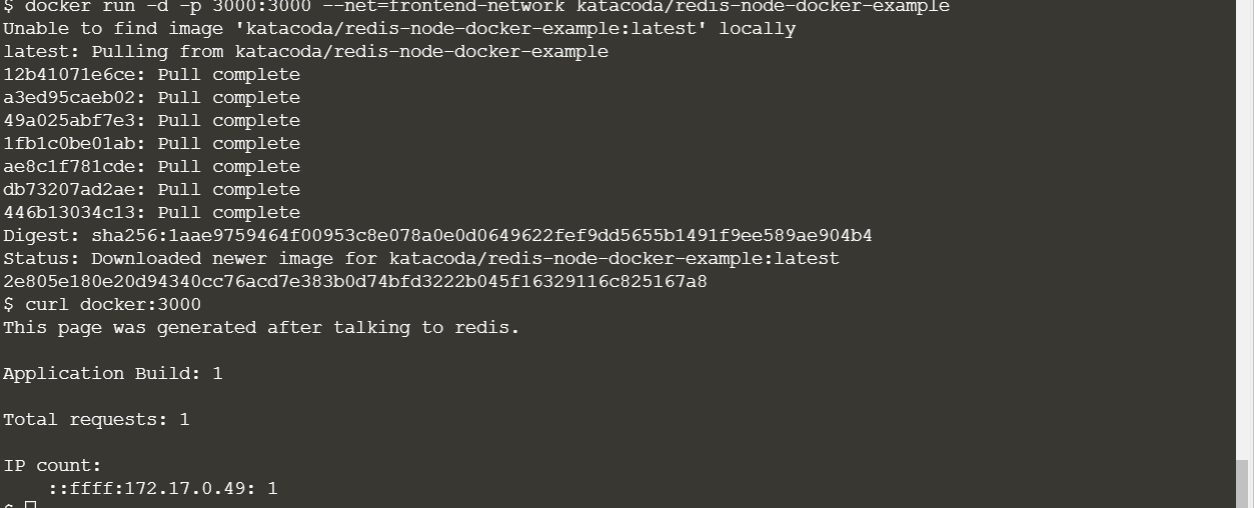




When we launch the web server, given it's attached to the same network it will be able to communicate with our Redis instance.

$ docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example

$ curl docker:3000



***Create Aliases:-***

The other approach is to provide an alias when connecting a container to a network.

***Connect container with Alias:-***

Create a new network *frontend-network2*

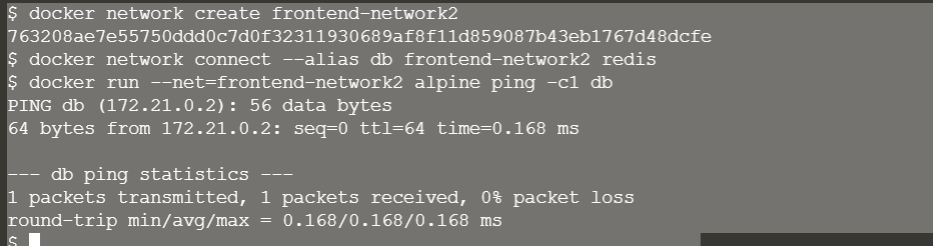
**Command:- $ docker network create frontend-network2**

**Now** The following command will connect our Redis instance to the frontend-network with the alias of *db*.

Command:- $ docker network connect --alias db frontend-network2 redis

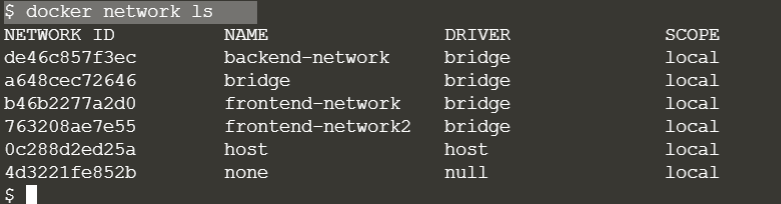
When containers attempt to access a service via the name db, they will be given the IP address of our Redis container.

$ docker run --net=frontend-network2 alpine ping -c1 db



To show the list of all the created network:-

Command:- $ docker network ls



We can see all the information about frontend-network with the help of this command.

Command:- $ docker network inspect frontend-network



***Disconnect the container:-***

The following command disconnects the redis container from the *frontend-network*

C:\Users\dell\Documents\application contai\exp4\15.PNG