**APPLICATION CONTAINERIZATION LAB**

**Experiment No. 3**

**Docker Networking**

**Submitted by:**

Devansh Markan

Roll Number: R171218036

SAP ID: 500069794

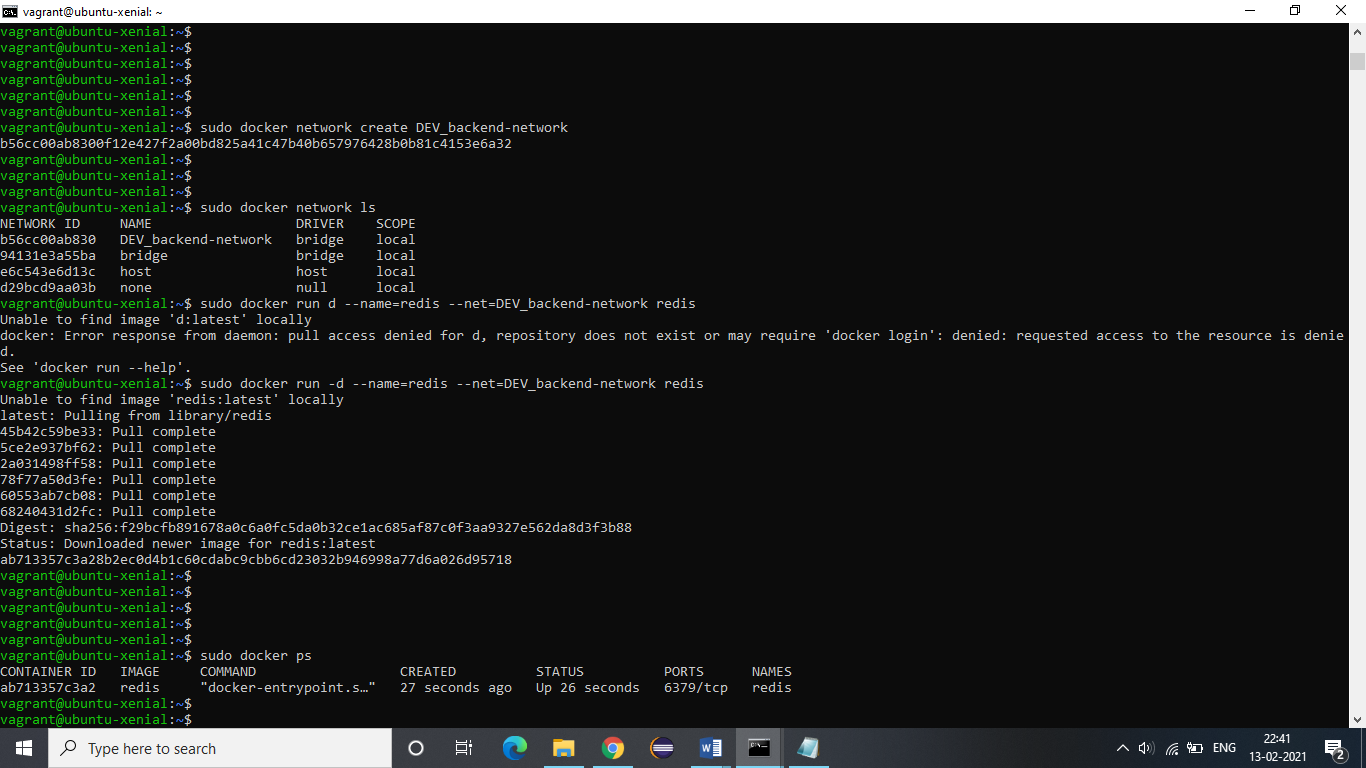
Semester: VI – Batch 1

Here, we are going to create a network in Docker. This network will allow us to attach multiple containers with one another and each container will be able to discover each other and can communicate.

**1. Create a network**

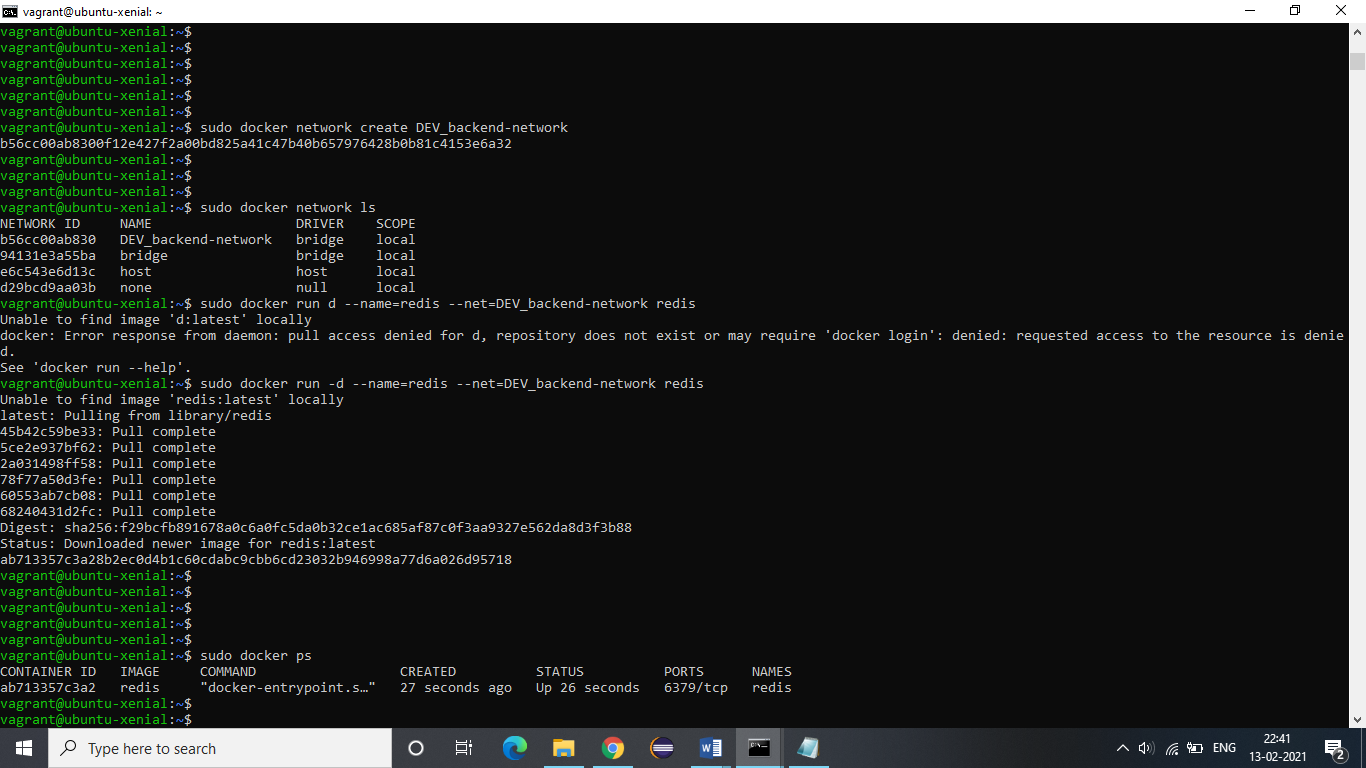
Here we will create a network with the name backend-network

**Command: docker network create backend-network**



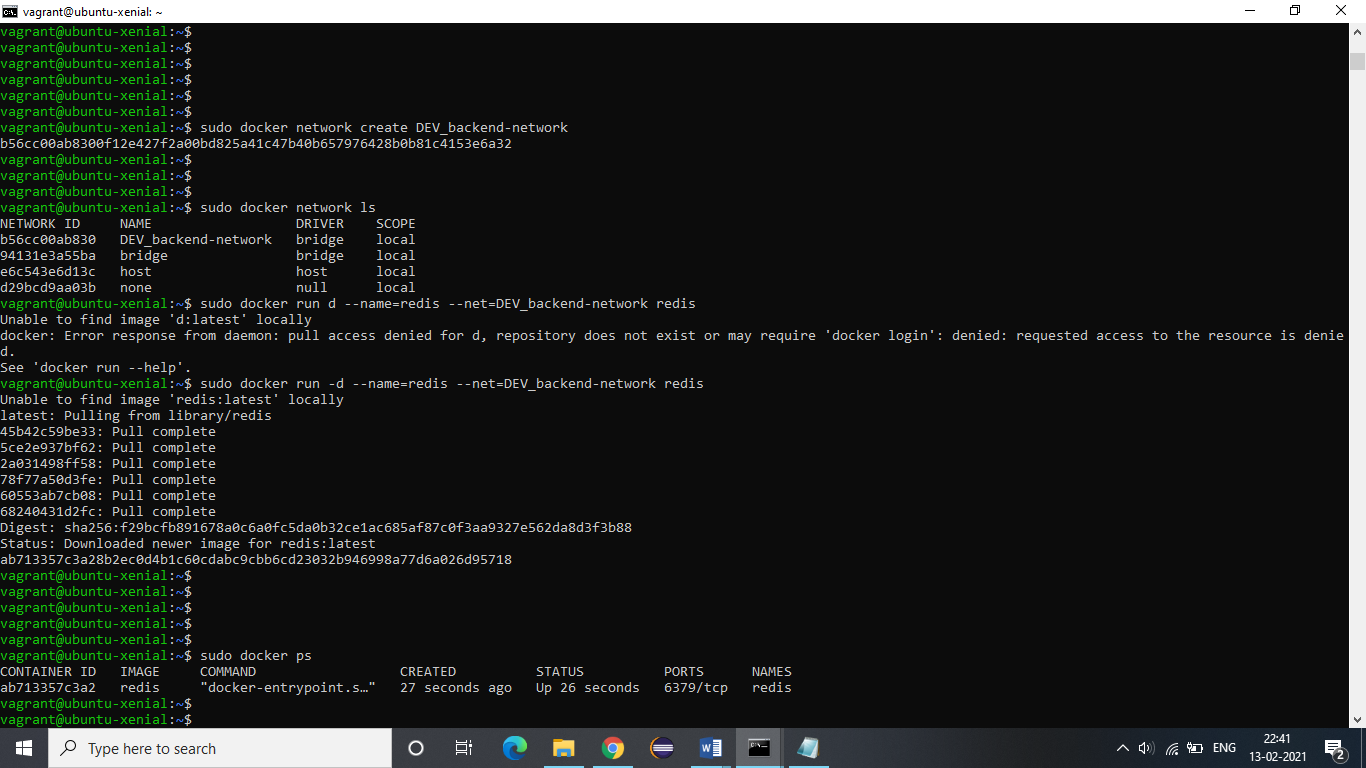
**2.** You can verify the creation of the network by listing the available networks.

**Command: docker network ls**



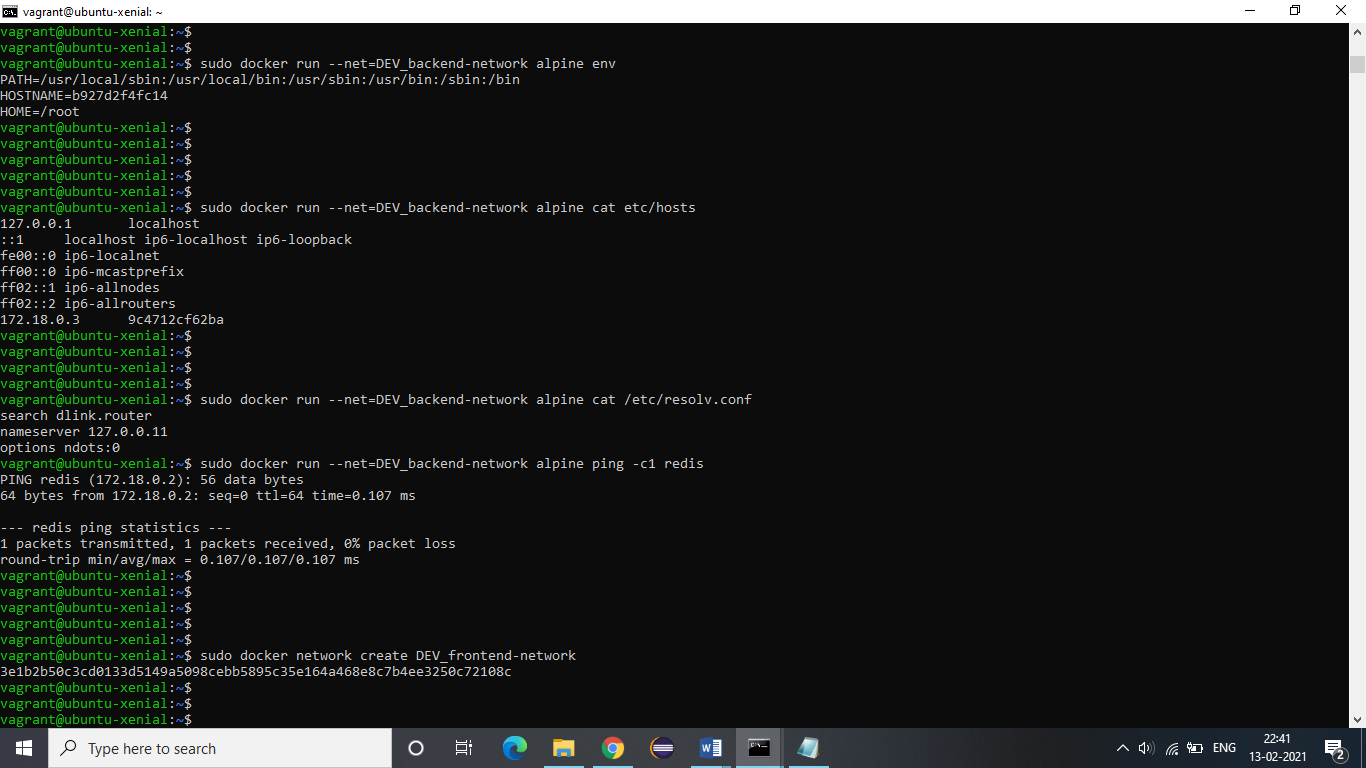
**3.** Now we will create a new container and we will assign backend-network to this container.

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



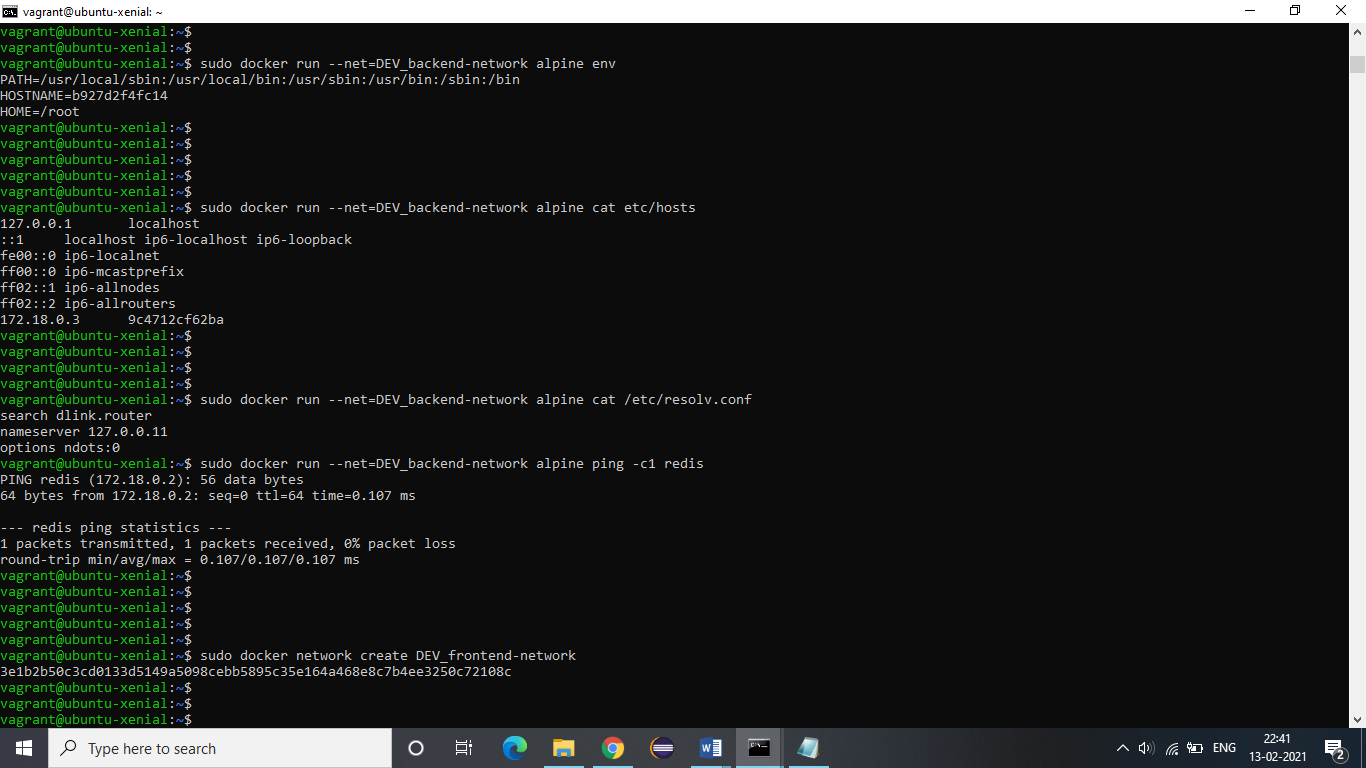
**4.** The networks in Docker do not use environment variables to discover other containers. We can verify this by creating a new container and assigning it the same network.

**Command: docker run --net=backend-network alpine env**



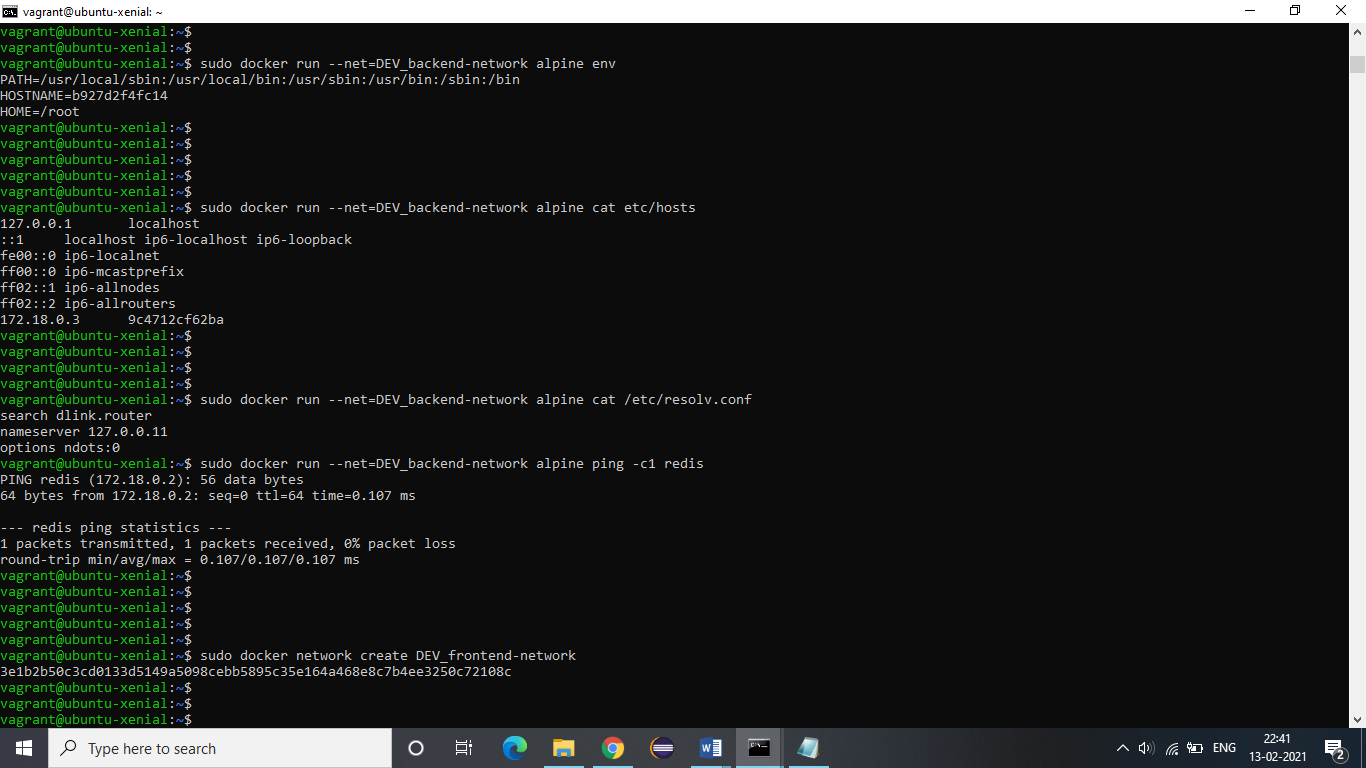
Also, Docker does not add any entries in /etc/host for the other containers. This can be verified as follows.

**Command: docker run --net=backend-network alpine cat etc/hosts**



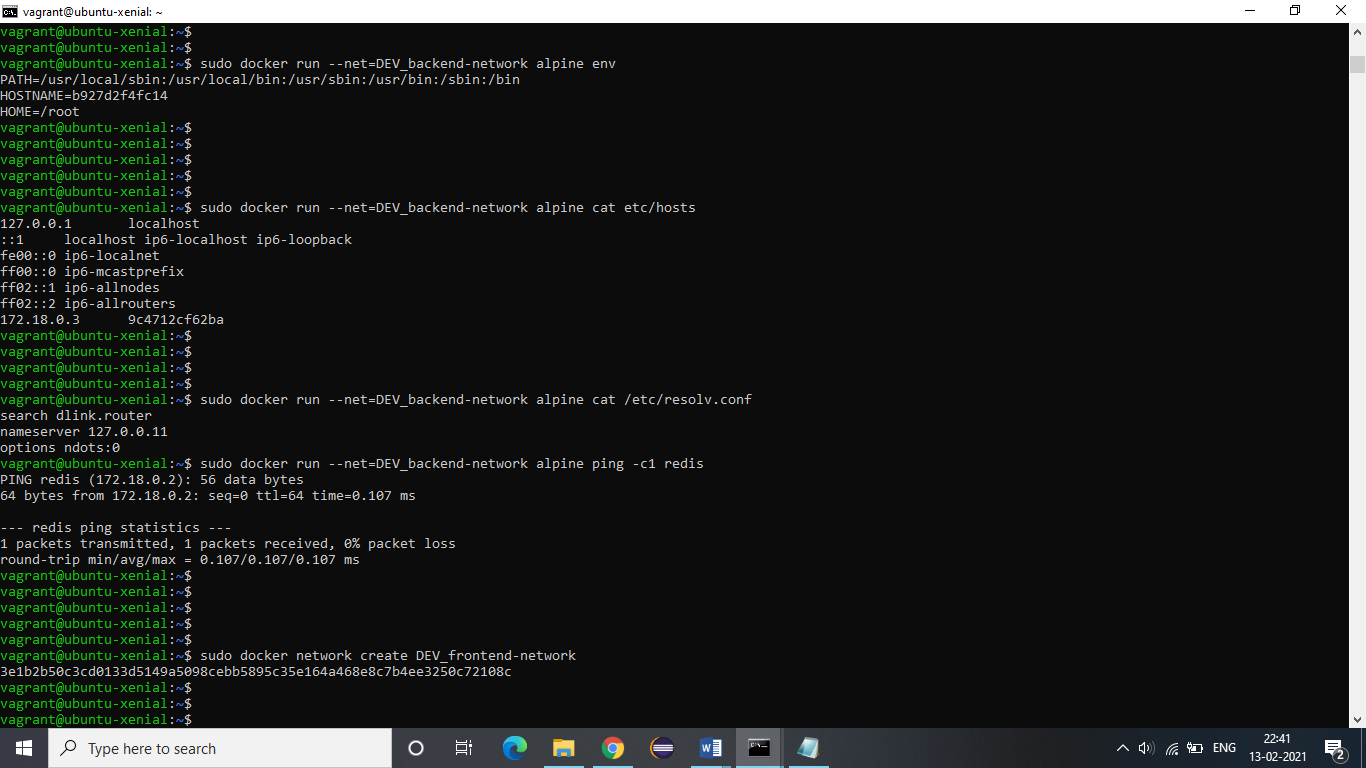
**5.** The DNS server is assigned to all containers and it is set in resolv.conf file. This can be verified as follows:

**Command: docker run --net=backend-network alpine cat /etc/resolv.conf**



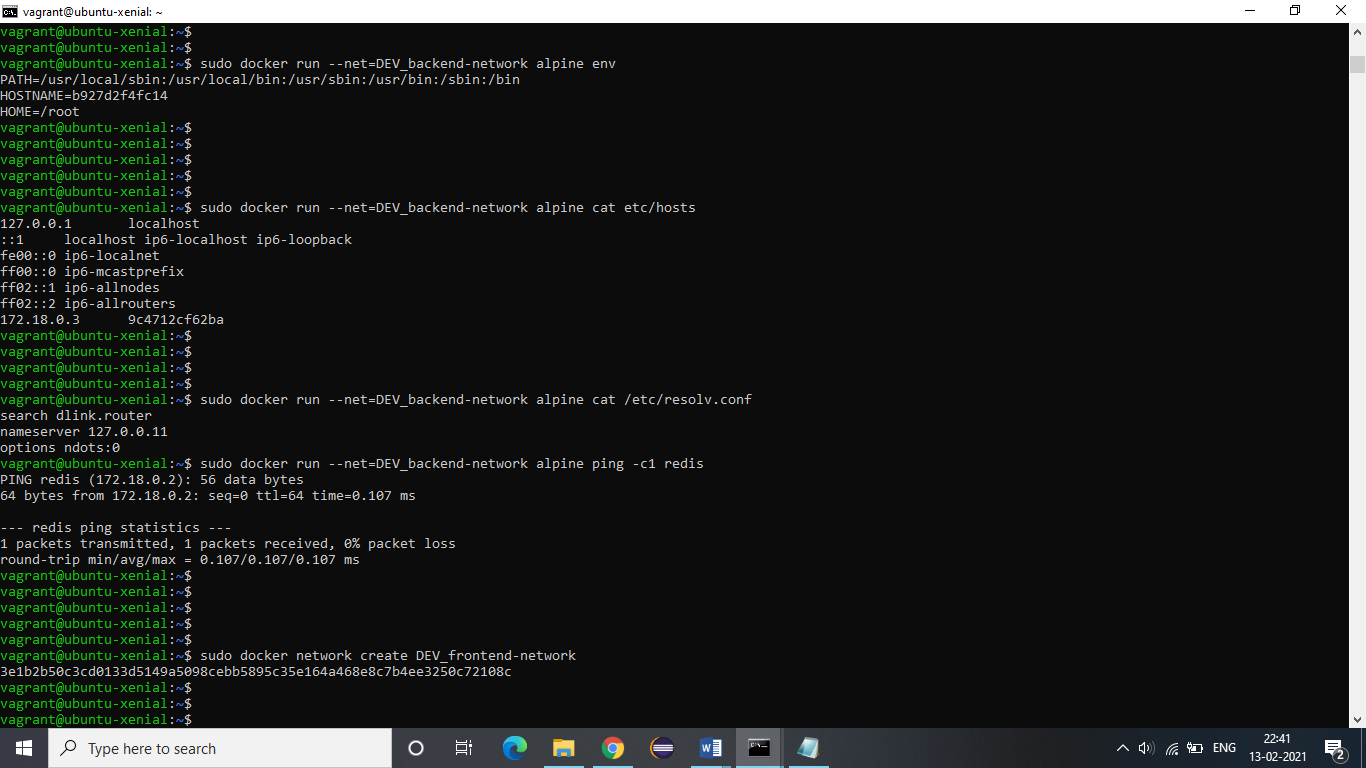
**6.** Let us try to ping the redis container which is attached to the same network.

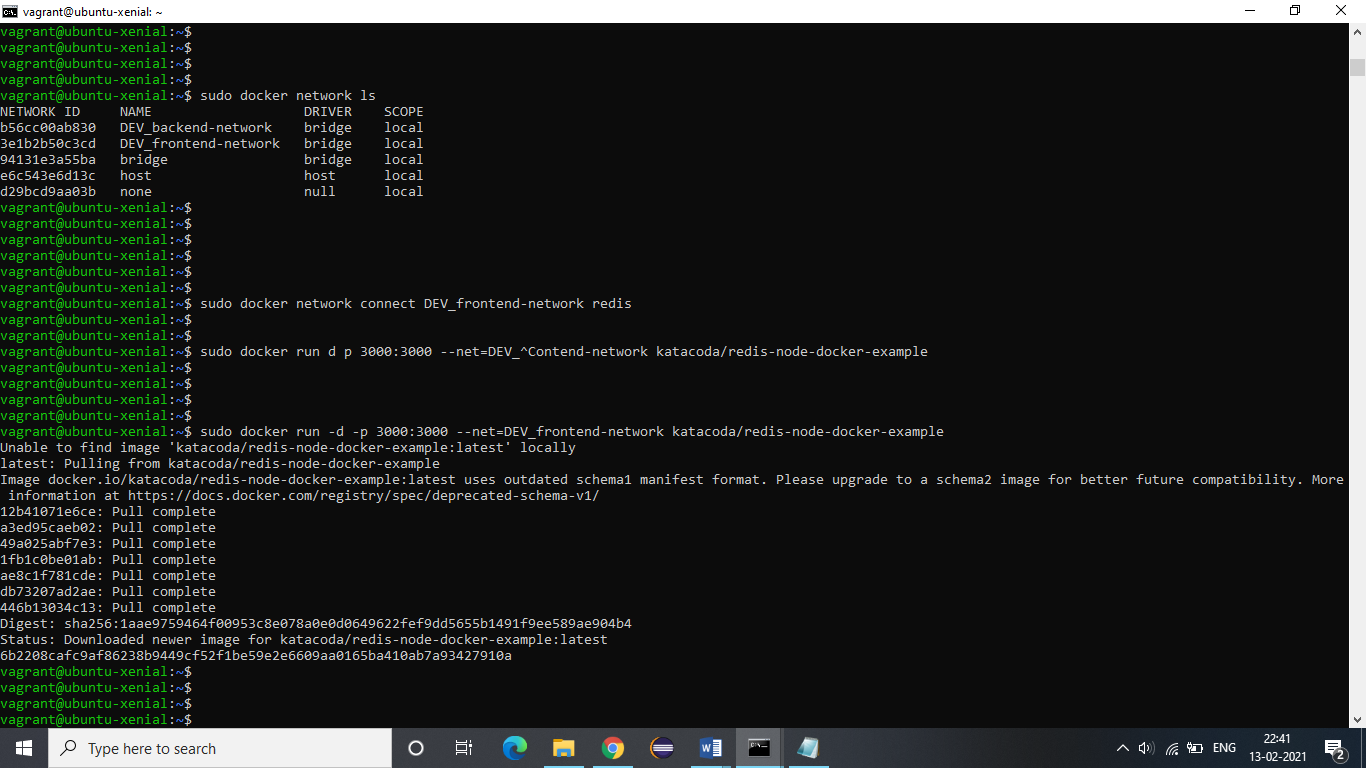
**Command: docker run --net=backend-network alpine ping -c1 redis**



**7.** We can attach a container with multiple networks. For this, let us create a new network named frontend-network.

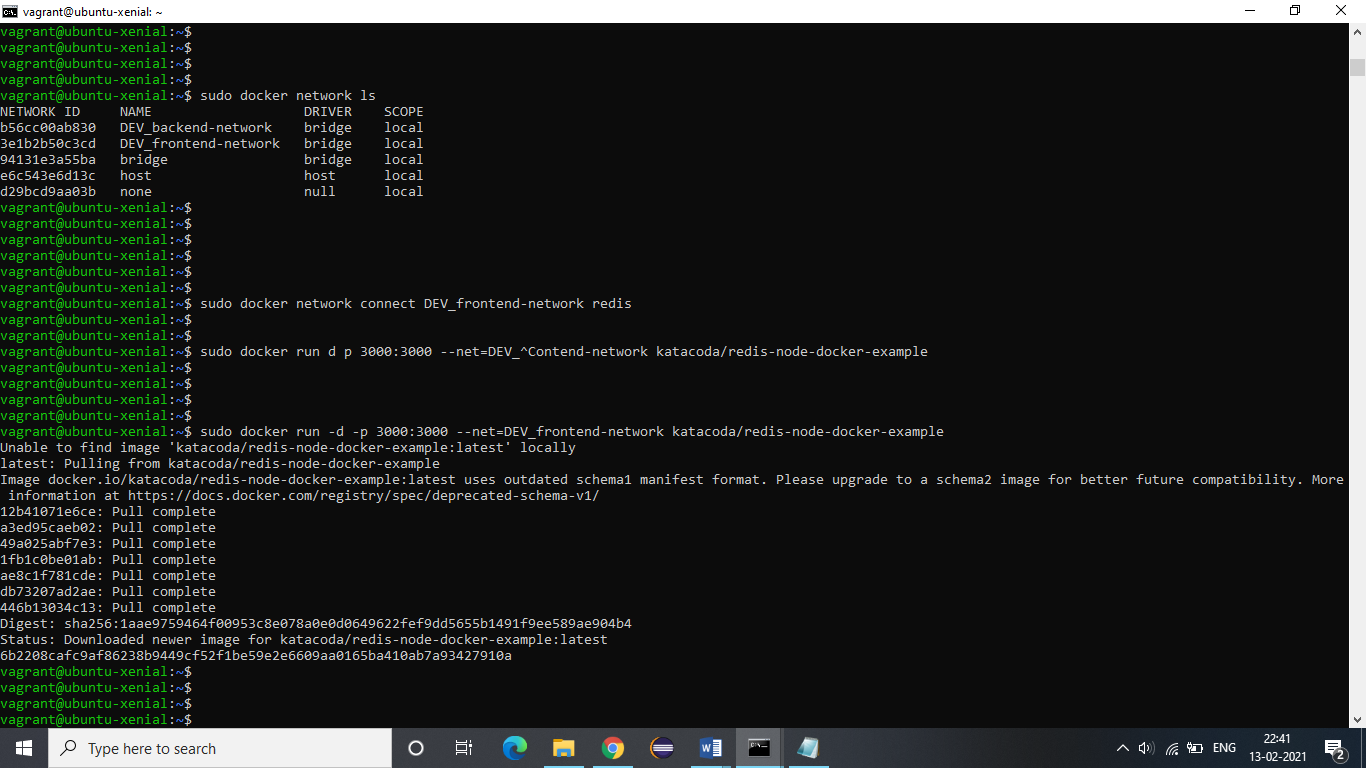
**Command: docker network create frontend-network**





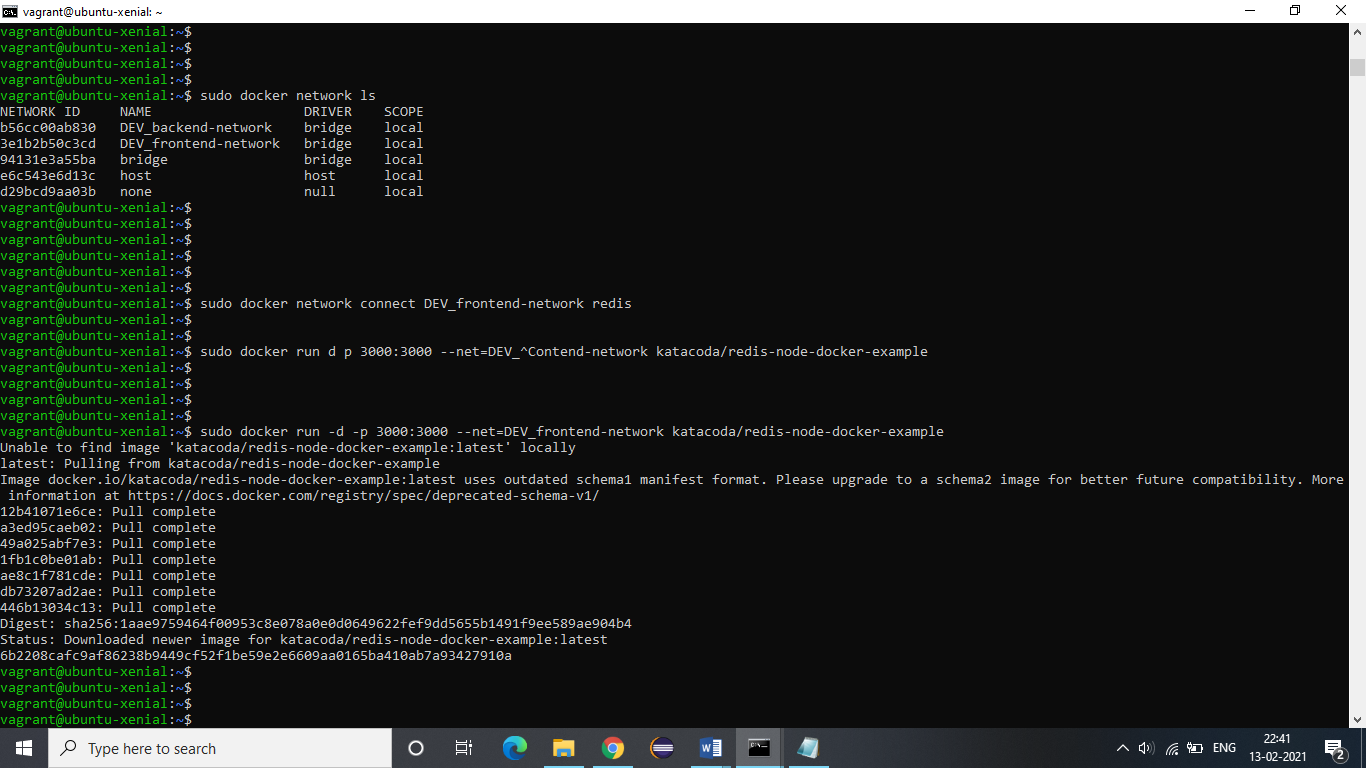
**8.** Now we will connect the previously made redis container with this frontend-network. This redis container is already connected with backend-network.

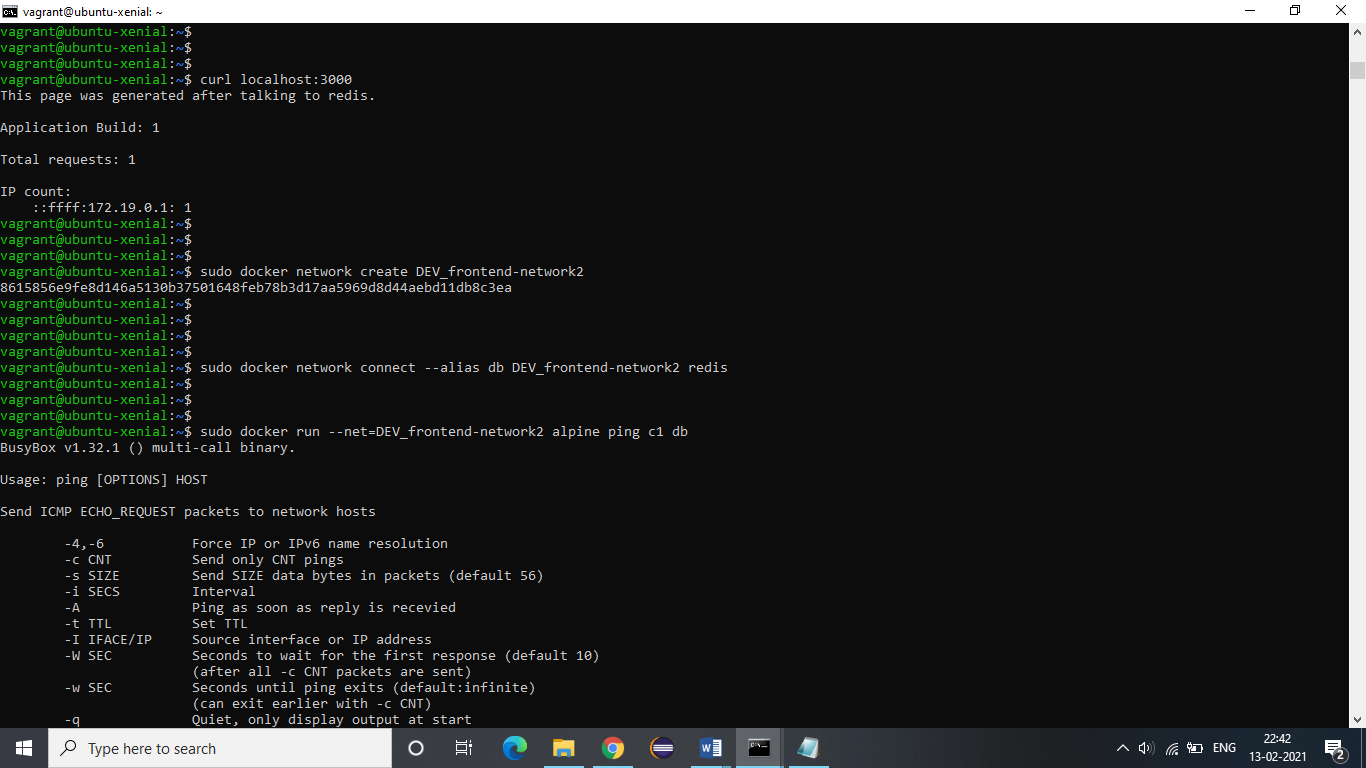
**Command: docker network connect frontend-network redis**



**9.** We will bind this customized image to port 3000.

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

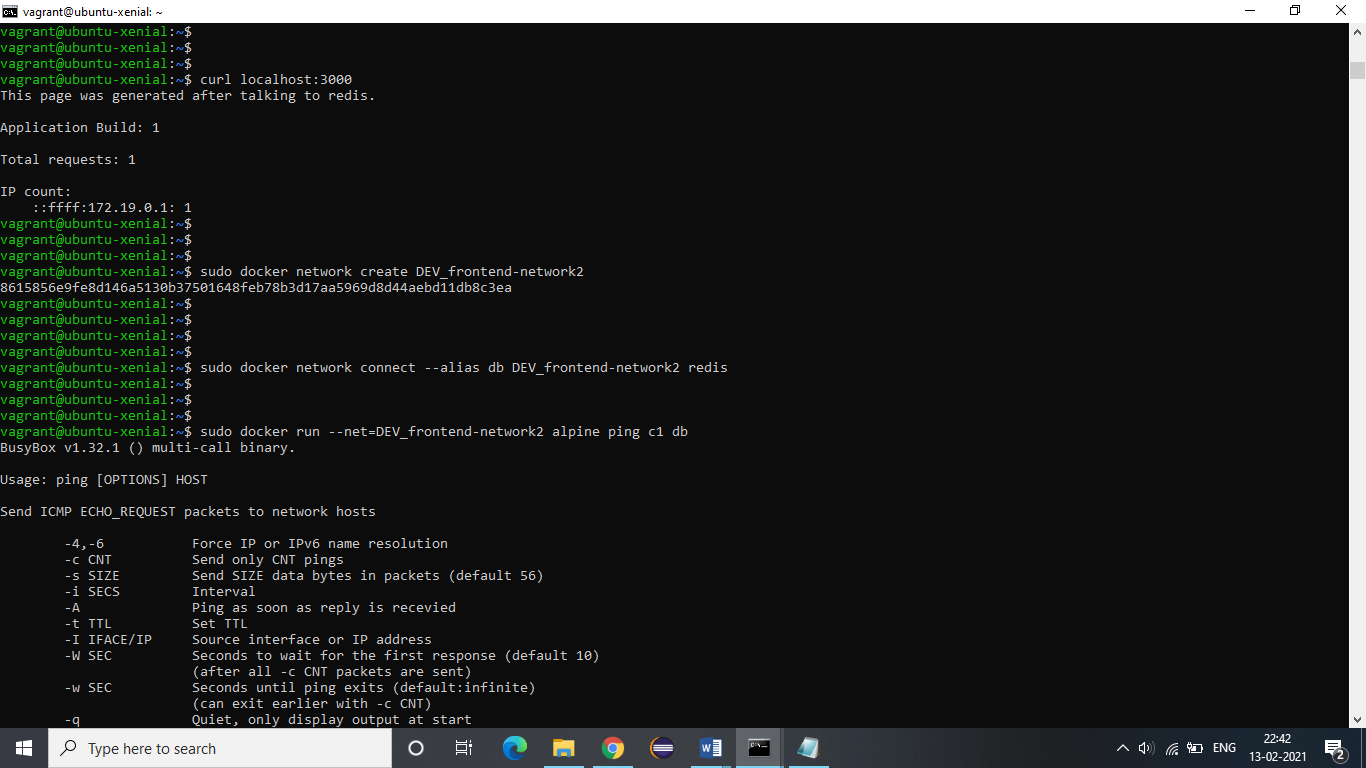




**10.** We can also create alias to a container. This will create a new entry for the container in the DNS. Let us see how this works.

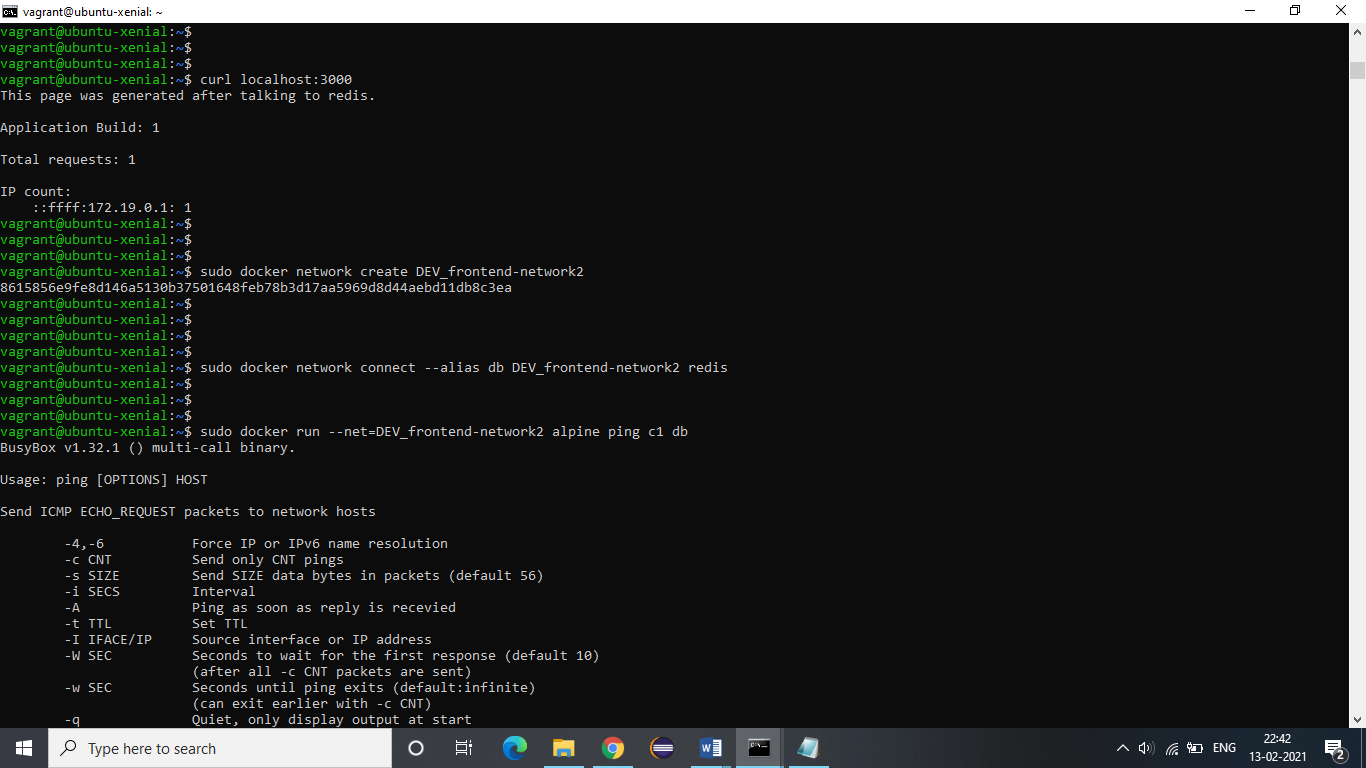
Let us first create a new network frontend-network2

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



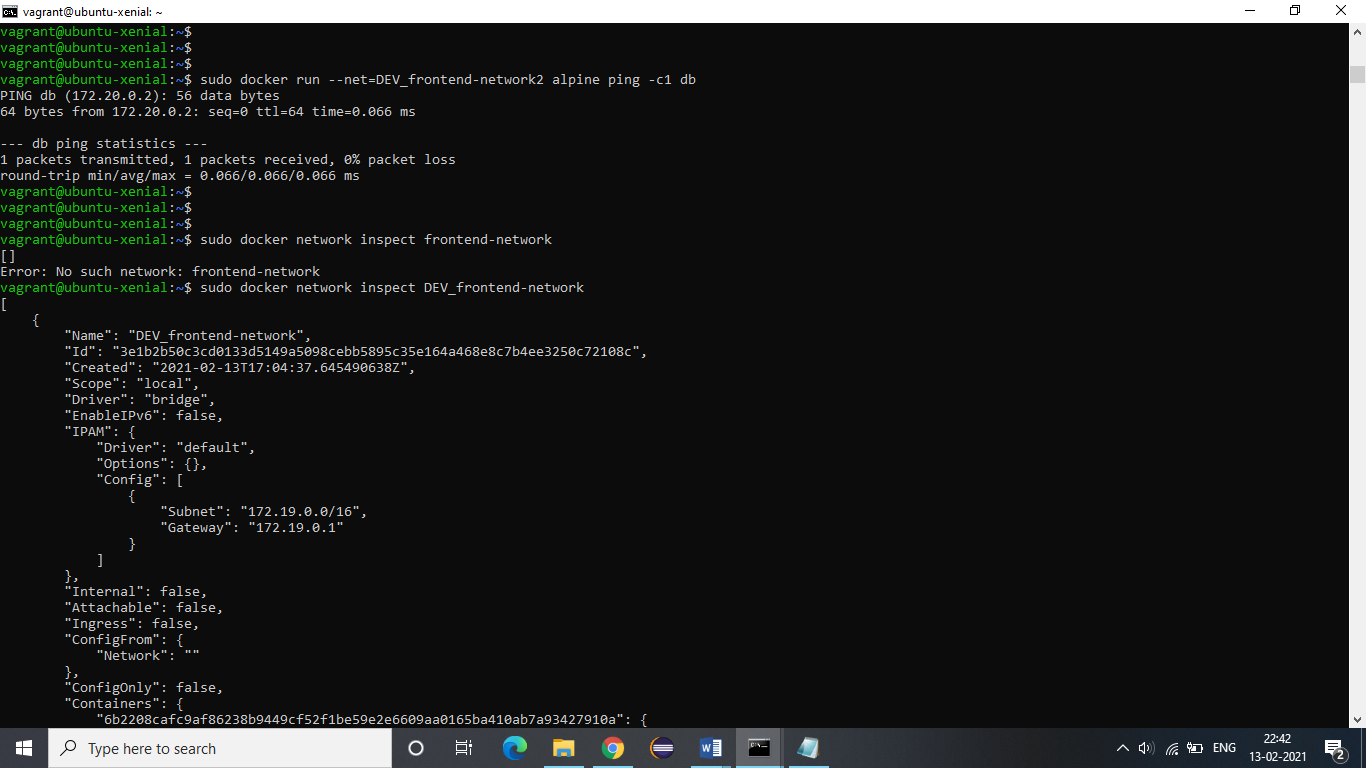
Now we will connect our redis container with this network and we will provide an alias “db”.

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



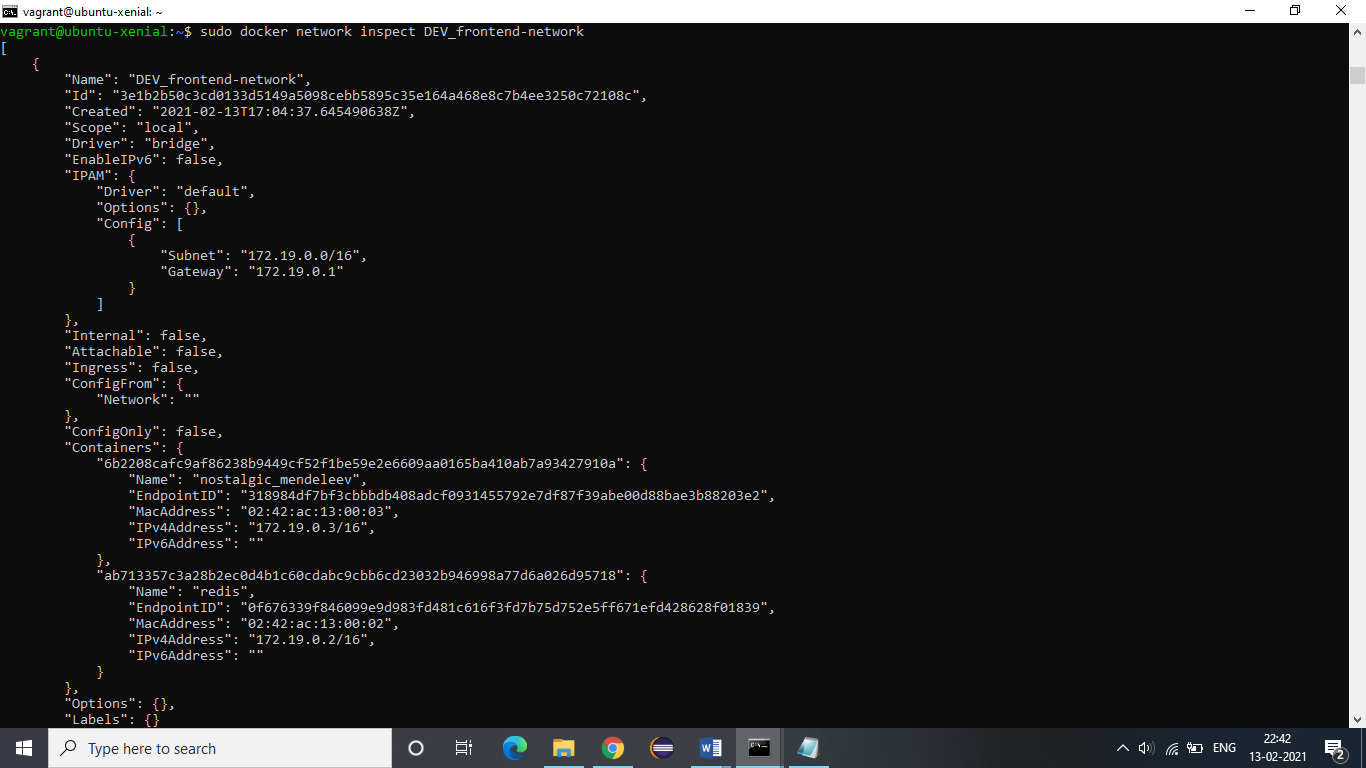
When other containers try to communicate with db, they will be provided with the ip of redis.

**Command: docker run --net=frontend-network2 alpine ping –c1 db**



**11.** We can get the details of any network by using the following command.

**Command: docker network inspect frontend-network**



**12.** We can disconnect a container from a network by disconnect command.

**Command: docker netwok disconnect frontend-network redis**

