CECS 327

Project 3: peer-to-peer

Ryan Tomas: 028210102

10/31/2025

Introduction

In this project, we are making a peer-to-peer network. This is possible with the use of Docker because each container acts like a node in the network. However, in this project, we would not be using a Docker YAML file but two Dockerfiles because there are only two Python files named bootstrap.py and node.py. I would have to build two containers for both files. There are two Dockerfiles, which are called Dockerfile and bootstrap.Dockerfile. The normal Dockerfile is for node.py and bootstrap.Dockerfile is for bootstrap.py. When all works and runs, it would print a message in the terminal which status that it received a message from another node in the network.

Docker

For this project, we are using Docker differently compared to the past projects. An example of this is that we do not make a Docker-Compose file because we are manually building and running the bootstrap and node Python file. The command we use is docker build -t container_name dockerfile and docker run -d –name python_code. The reason we do this is that we are running multiple nodes to make a peer-to-peer network. With Docker-Compose, we would not be able to make multiple and send messages from random nodes. Another thing is that I had to set the URL that the nodes are going to use to bootstrap:5000 because I’m not using my local machine, but the Docker engine to make the network possible.

Network design

The design of the network is based on a bootstrap because it acts as an initial entry point for new nodes joining the network. When a node joins, its URL is put in a registry, and it responds to the requests of new nodes. Now, for each node is made Flask API call is made which is used for sending and receiving messages for each node. Another part that the node needs is the ability to register with the network and see other peers. All these API calls are functions that I had to make. Without these calls, each node won't be able to register or see other peers in the network.

Challenges

When creating this project, there were many difficult parts that I had to revisit. One such part was troubleshooting the bootstap_url because I did not know that I had to set it in Docker to make it work. Also need to learn how Bootstrap works and apply it to this project. The last challenge was figuring out how to build and run the code because I always used Docker Compose and never two Dockerfiles in the root folder. With each challenge I faced, I learned more about how networking works on a small scale.

Conclusion

In conclusion, this project demonstrated different aspects of a peer-to-peer network using Docker. Each container functioned as an independent node capable of communicating with its peers with the help of HTTP requests. Bootstrap is very important because it serves as a central entry point that allows nodes to register and discover peers dynamically. Docker Compose helps drive the understanding of how nodes join a network and communicate with each other. The project provided valuable hands-on experience with concepts.

Video

https://drive.google.com/file/d/1LsKYlin3tdjBHwDGfBY7HyAJGu8ur7Pu/view?usp=sharing

output

sending from node9 to node9

{"status":"received"}

sending from node18 to node12

{"status":"received"}

sending from node18 to node7

{"status":"received"}

sending from node16 to node11

{"status":"received"}

sending from node9 to node5

{"status":"received"}

sending from node4 to node1

{"status":"received"}

sending from node11 to node12

{"status":"received"}

sending from node13 to node5

{"status":"received"}

sending from node20 to node14

{"status":"received"}

sending from node15 to node13

{"status":"received"}

sending from node17 to node16

{"status":"received"}

sending from node8 to node5

{"status":"received"}

sending from node6 to node8

{"status":"received"}

sending from node4 to node13

{"status":"received"}

sending from node19 to node14

{"status":"received"}

sending from node19 to node12

{"status":"received"}

sending from node19 to node10

{"status":"received"}

sending from node19 to node6

{"status":"received"}

sending from node4 to node14

{"status":"received"}