Despite successfully deploying all microservices into running Docker containers and connecting them via MVC, the API Gateway did not work as intended. The primary issue preventing a full deployment was the API Gateway failing to properly route requests to the appropriate microservices. Although the web application loaded and displayed game pages, the microservices were not fully interacting as expected, possibly due to incorrect service discovery, routing misconfigurations in Nginx, or networking constraints between containers. The API Gateway logs indicate that it is listening on port 5000, but requests do not reach the microservices. This suggests a misconfiguration in how the API Gateway forwards traffic.

## Download Zip file of Code Base

- Download the provided zip folder of the codebase and pem file. Unzip the folder and take note of the directory. This is the directory you will work in when uploading files.

## Upload Files

- These commands upload the project directories from the local machine to the remote EC2 instance using scp (secure copy protocol). The -i BucStopProd.pem flag ensures authentication with the server.

```
scp -i BucStopProd.pem -r Team-3-BucStop-sprint_8 ec2-
user@3.145.84.12:/home/ec2-user
scp -i BucStopProd.pem -r Team-3-BucStop_APIGateway-sprint_8 ec2-
user@3.145.84.12:/home/ec2-user
scp -i BucStopProd.pem -r Team-3-BucStop_Pong-sprint_8 ec2-
user@3.145.84.12:/home/ec2-user
scp -i BucStopProd.pem -r Team-3-BucStop_Snake-sprint_8 ec2-
user@3.145.84.12:/home/ec2-user
scp -i BucStopProd.pem -r Team-3-BucStop_Tetris-sprint_8 ec2-
user@3.145.84.12:/home/ec2-user
```

## Enter EC2 Instance

- Establishes an SSH connection to the EC2 instance using the provided key for authentication.

```
ssh -i BucStopProd.pem ec2-user@3.145.84.12
```

## Navigate to Correct Folder

- Moves into the main project directory on the EC2 instance where the Docker setup is managed.

```
cd /home/ec2-user/Team-3-BucStop-sprint_8
```

## Build Docker image

- Builds the Docker image for the main web application using the specified Dockerfile.

```
docker build -t team-3-bucstop-sprint_8 -f BucStop/Dockerfile .
```

## *Repeat process for each container*

- Each command builds a separate Docker image for the respective microservice using its Dockerfile.

```
cd /home/ec2-user/Team-3-BucStop-sprint_8/APIGateway
docker build -t team-3-bucstop-sprint_8-apigateway -f Dockerfile .

cd /home/ec2-user/Team-3-BucStop-sprint_8/Pong
docker build -t team-3-bucstop-sprint_8-pong -f Dockerfile .

cd /home/ec2-user/Team-3-BucStop-sprint_8/Snake
docker build -t team-3-bucstop-sprint_8-snake -f Dockerfile .

cd /home/ec2-user/Team-3-BucStop-sprint_8/Tetris
docker build -t team-3-bucstop-sprint_8-tetris -f Dockerfile .
```

## Check running docker images:

- Lists all currently running Docker containers to confirm which services are active.

```
docker ps
```

## Remove any existing (if needed)

- Stops and removes an existing container if it’s already running, ensuring a clean start.

```
docker ps -a
docker stop team-3-bucstop-sprint_8
docker rm team-3-bucstop-sprint_8
```

## Run new containers

- Starts each microservice in a new Docker container with the correct port mappings.

```
docker run -d --name team-3-bucstop-sprint_8 -p 8000:8000 team-3-bucstop-sprint_8
docker run -d --name team-3-bucstop-sprint_8-apigateway -p 5000:5000 team-3-bucstop-sprint_8-apigateway
docker run -d --name team-3-bucstop-sprint_8-pong -p 8084:8000 team-3-bucstop-sprint_8-pong
docker run -d --name team-3-bucstop-sprint_8-snake -p 8083:8000 team-3-bucstop-sprint_8-snake
docker run -d --name team-3-bucstop-sprint_8-tetris -p 8082:8000 team-3-bucstop-sprint_8-tetris
```

## Check logs for errors

- Retrieves the last 50 log entries for the main web application to check for issues.

docker logs team-3-bucstop-sprint_8 --tail 50

### Test

- Sends requests to the web application and game services to verify functionality.

```
curl http://localhost:8000/
curl http://localhost:8000/Games
curl http://localhost:8000/Games/pong
curl http://localhost:8000/Games/snake
curl http://localhost:8000/Games/tetris
```

### The following section is if needed:

#### Restart

- Restarts the main web application container.

```
docker restart team-3-bucstop-sprint_8-bucstop
```

#### Clean rebuild

- Completely rebuilds the main web application container.

```
docker stop team-3-bucstop-sprint_8-bucstop
docker rm team-3-bucstop-sprint_8-bucstop
docker build -t team-3-bucstop-sprint_8 -f BucStop/Dockerfile .
docker run -d --name team-3-bucstop-sprint_8-bucstop -p 8000:8000 team-3-bucstop-sprint_8
```

#### If port 8000 is blocked:

- Identifies and terminates any process blocking port 8000.

```
netstat -ano | grep :8000
kill -9 <PID>
```

## Possible Next Steps:

- Investigate API Gateway Issues: The web app runs on 8000, but 5000 (API Gateway) isn't working. Check the API Gateway's networking, service discovery, and routing rules.
- Verify Internal Communication: Ensure that microservices are correctly resolving names and communicating within the Docker network.
- Validate Nginx Configuration: Double-check nginx.conf to confirm it correctly forwards traffic to the appropriate microservices.