**Task 4 Part A1**

The reason why I had selected Render as a cloud solution provider is because they have mastered the simplification of deploying Docker images. On top of that, including this service with the resources selected, it was free. Because of the service tier, the application will shut down and must spin back up after a period of non-use. This can be beneficial for customers who want to save costs on development and deployment of products.

Other services that were tried include AWS EC2, DigitalOcean Kubernetes Cluster, and Zeet, which is a image deployment manager. I was unsuccessful in trying to get AWS EC2 to build my image within the instance. Although Docker did successfully download within the instance, building and running did not work even when modifying ports and checking security parameters.

With DigitalOcean and Zeet, the solutions were too expensive and not pursued past pricing resources selected. This solution may have worked for the desired conclusion but was overall not utilized. Render was a very simple, cheap, and reliable solution which is naturally what most customers would gravitate towards in general. I am satisfied with Render and would recommend them in the future.

**Task 4 Part A2 Docker Container and Deployment Instructions**

Below are the steps used to successfully deploy the image to Render. A simplified version of the process includes creating and modifying a Dockerfile locally. After building and testing that the build works, it can be tagged and pushed to a remote DockerHub Registry. Render will use the image hosted on the Registry to spin up the application and provide a URL for it once finished. The URL I was provided was [Pizzeria Inventory Application (d424-capstone-msalzar.onrender.com)](https://d424-capstone-msalzar.onrender.com/mainscreen) Please note that if the application has not been accessed in a while, it may take time for Render to see it being accessed in order for it to be run on demand so a delay may happen.

1. Create Dockerfile
2. Edit contents to,

“

FROM maven:3.8.3-openjdk-17 AS build

COPY . .

RUN mvn clean install

FROM eclipse-temurin:17-jdk

COPY –from=build /target/demo-0.0.1-SNAPSHOT.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]

”

1. Execute, “docker build -t demo .”.
2. Execute, “docker run -p 8080:8080 demo” to test.
3. Execute, “docker tag demo mikesalzarulo/d424-capstone-msalzar:demo”.
4. Execute, “docker push mikesalzarulo/d424-capstone-msalzar:demo”.
5. Navigate to Render.com, and create new web service.
6. Connect DockerHub Registry with URL of Hub image.
7. Choose image you desire.
8. Run the web service.
9. Test that it was successful.