CURRENT TRENDS IN SOFTWARE ENGINEERING

SE4010

Microservice Assignment Document

A.S.V Jayadeva

IT19139036

B.Sc. (Hons) in Information Technology Specializing in Software Engineering

Department of Computer Science and Software Engineering

Sri Lanka Institute of Information Technology Sri Lanka

May 2022

TABLE OF CONTENTS

1.0 OVERVIEW

Users can buy and sell things on ClickToCart, an online shopping system. The system had previously been implemented utilizing a monolithic design. During development, the team ran upon certain issues with monolithic architecture. One of the disadvantages of monolithic design is that when it comes to application deployment, the entire application must be deployed to the server, even if the modification has no impact on other features. Other features become unavailable throughout the deployment process. The team opted to design the application using microservice architecture to address the monolithic architectural issue. In a microservice architecture, each application's functionality is separated into its own service. The main benefit of this strategy is that it allows for considerably faster application deployment than a monolith design. If the update is performed to cart service, for example, cart service will only be distributed to the cluster during deployment. As a result, no other services are affected.

GitHub Code Repository Link - https://github.com/Research-Group-CDAP/CTSE-Assignment-2

2.0 IMPLEMENTED MICROSERVICES

The ClickToCart application have 8 microservices. Following table includes the service, programming language and frameworks that used to implement the service and description about the service.

2.1 Details of Microservices

Service Name	Programming language and framework	Description
Order service	Go, Fiber web framework	Order service creates an order after the product purchase complete.
Email service	Go, Fiber web framework	Dispatch an email to the customer after the order has been placed.
Product service	Java, Sprint Boot	Provide create, update, delete and list products to customers.
Cart service	Java, Sprint Boot	Store the selected items in the MongoDB database
User service	JavaScript, Express framework	Provide manage user profile information.
Auth service	JavaScript, Express framework	Provide JSON Web Token (JWT) mechanism to authenticate the user.
Payment service	Java, Sprint Boot	Provide payment gateway to make the payments for the selected items.
Delivery service	Java, Sprint Boot	Add delivery record about the purchased products.

2.2 Individual implemented services

- Product service
- Cart service

3.0 INDIVIDUAL SERVICE OVERVIEW

3.1 Product Service

Product service is responsible to add products to the system. Java programming language and springboot framework are used to develop this service. Product service make request with Auth service to validate whether to check the user is a seller or buyer. Only seller is able to add products.

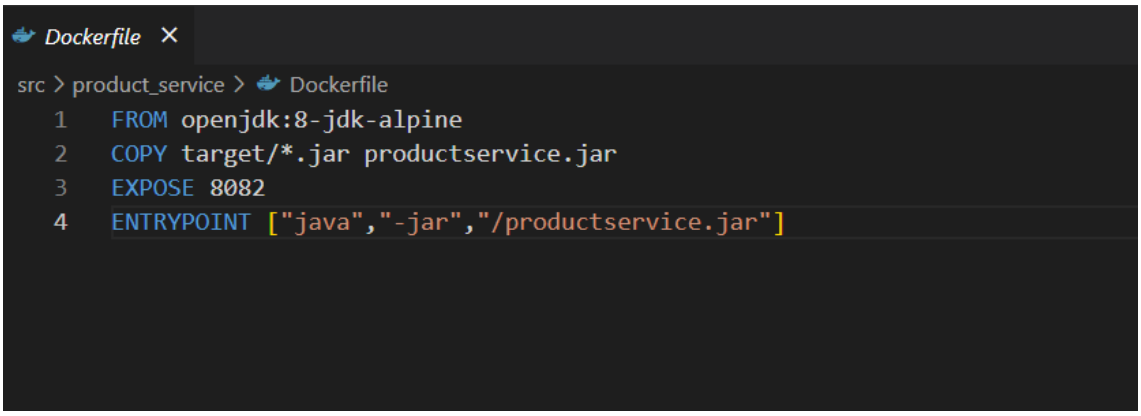
3.2 Cart Service

Product service is responsible to add products to the cart when user purchased. Java programming language and springboot framework are used to develop this service. Cart Service also calling the Auth service to validate the user role.

4.0 TASK 1 – DOCKERIZE APPLICATIONS

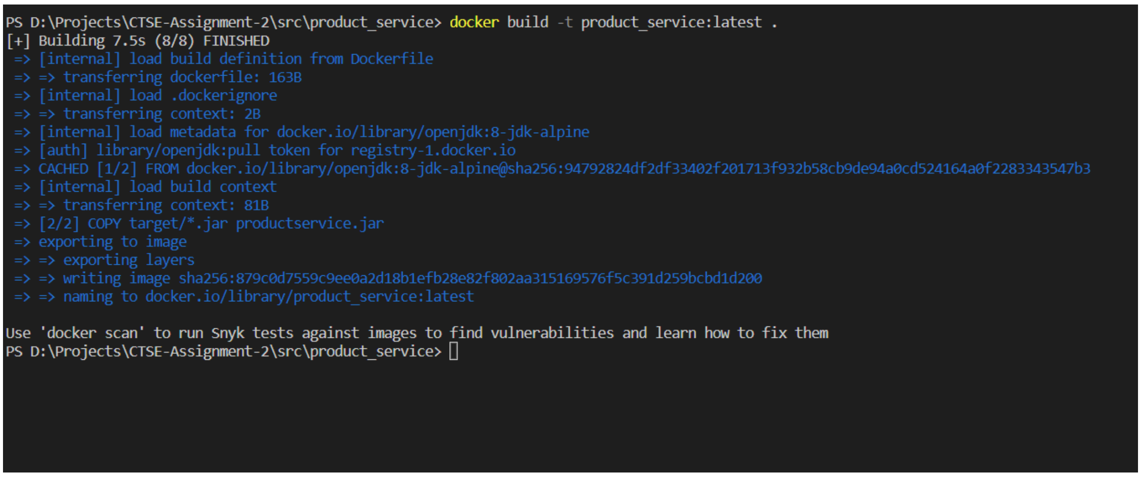
4.1 CONTAINERIZE PRODUCT SERVICE

4.1.1 Dockerfile of Product Service



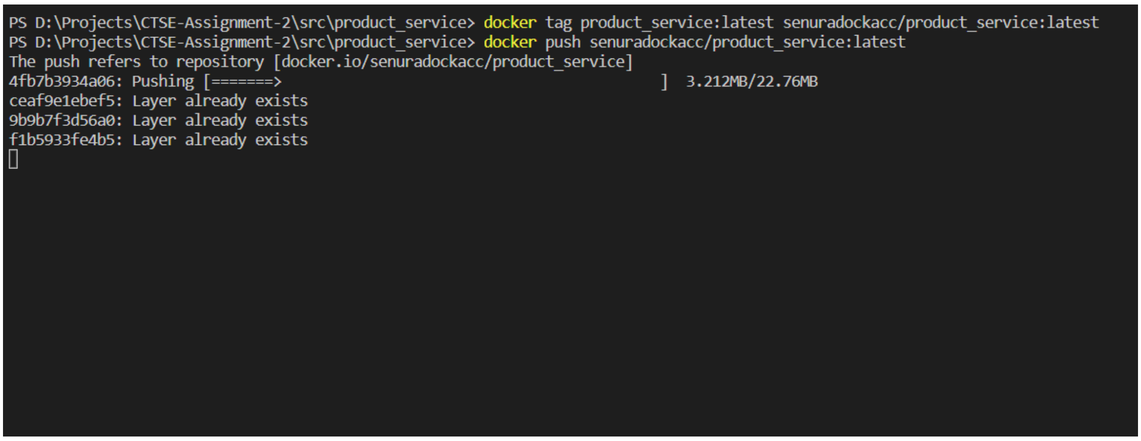
```
FROM openjdk:8-jdk-alpine
COPY target/*.jar productservice.jar
EXPOSE 8082
ENTRYPOINT ["java","-jar","/productservice.jar"]
```

4.1.2 Product Service Container Image Building



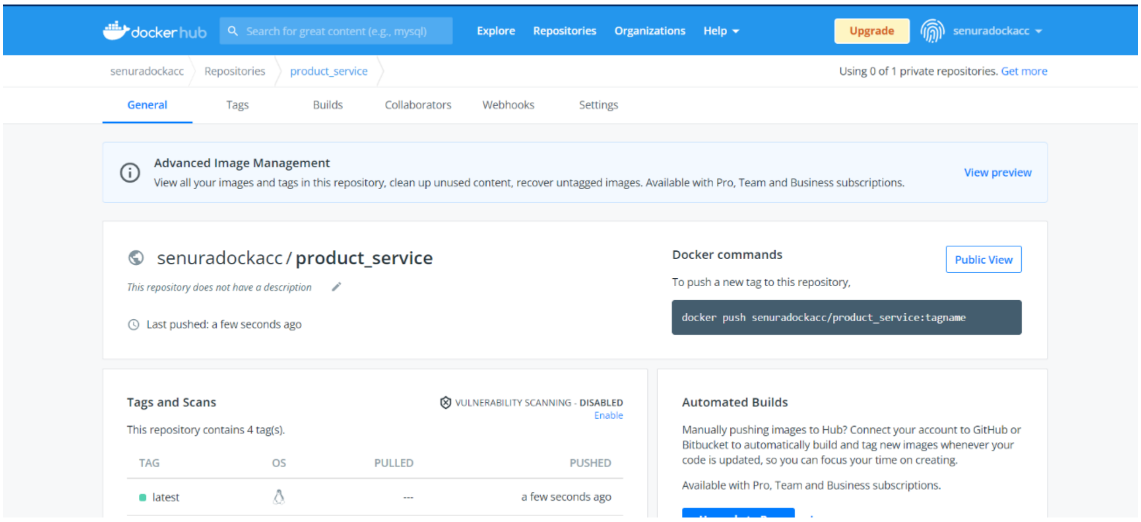
```
PS D:\Projects\CTSE-Assignment-2\src\product_service> docker build -t product_service:latest .
[+] Building 7.5s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 163B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => CACHED [1/2] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3
 => [internal] load build context
 => => transferring context: 81B
 => [2/2] COPY target/*.jar productservice.jar
 => exporting to image
 => => exporting layers
 => => writing image sha256:879c0d7559c9ee0a2d18b1efb28e82f802aa315169576f5c391d259bcbd1d200
 => => naming to docker.io/library/product_service:latest

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS D:\Projects\CTSE-Assignment-2\src\product_service> 
```

4.1.3 Push Product Docker Image to DockerHub



```
PS D:\Projects\CTSE-Assignment-2\src\product_service> docker tag product_service:latest senuradockacc/product_service:latest
PS D:\Projects\CTSE-Assignment-2\src\product_service> docker push senuradockacc/product_service:latest
The push refers to repository [docker.io/senuradockacc/product_service]
4fb7b3934a06: Pushing [=======>                                           ]  3.212MB/22.76MB
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
```

4.1.4 Product Service DockerHub Overview



Advanced Image Management

View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

senuradockacc / product_service

This repository does not have a description

Last pushed: a few seconds ago

Docker commands

To push a new tag to this repository,

```
docker push senuradockacc/product_service:tagname
```

Tags and Scans

This repository contains 4 tag(s).

TAG	OS	PULLED	PUSHED
latest	🐧	---	a few seconds ago

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

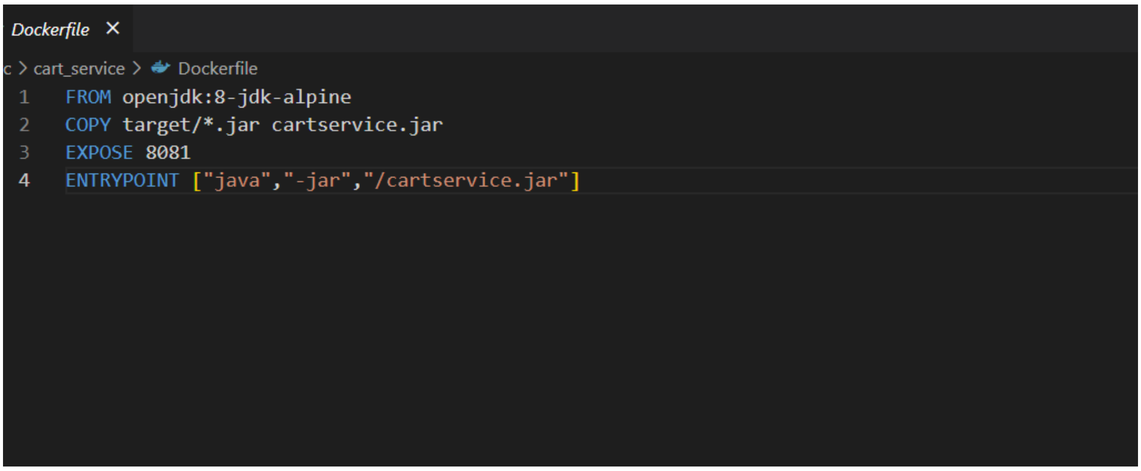
Available with Pro, Team and Business subscriptions.

Product service DockerHub link

https://hub.docker.com/repository/docker/senuradockacc/product_service

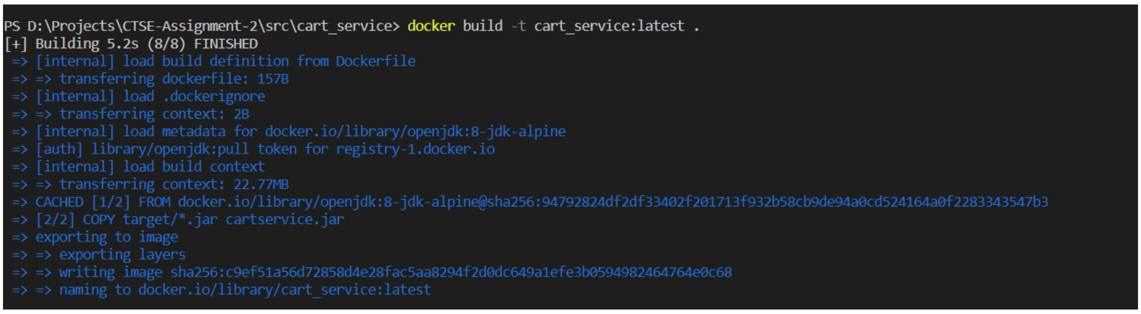
4.2 Containerize Cart Service

4.2.1 Dockerfile of Cart Service



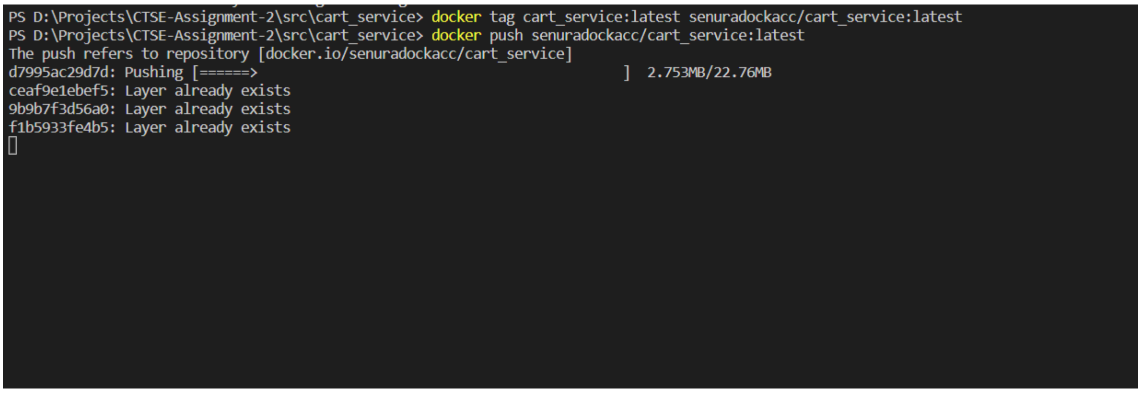
```
FROM openjdk:8-jdk-alpine
COPY target/*.jar cartservice.jar
EXPOSE 8081
ENTRYPOINT ["java","-jar","/cartservice.jar"]
```

4.2.2 Cart Service Container Image Building



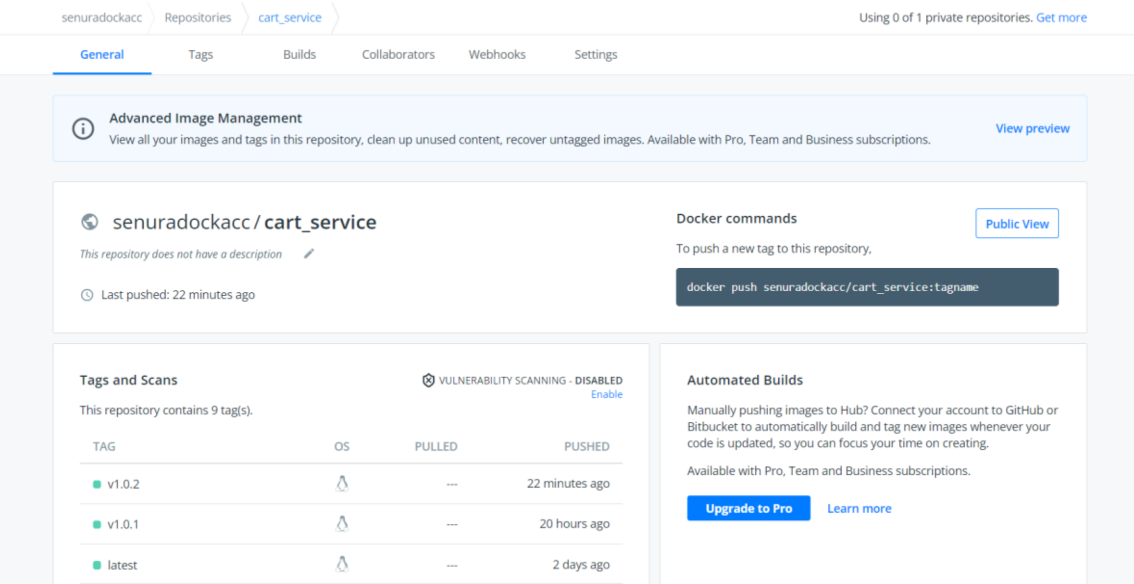
```
PS D:\Projects\CTSE-Assignment-2\src\cart_service> docker build -t cart_service:latest .
[+] Building 5.2s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 157B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => [internal] load build context
 => => transferring context: 22.77MB
 => CACHED [1/2] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3
 => [2/2] COPY target/*.jar cartservice.jar
 => exporting to image
 => => exporting layers
 => => writing image sha256:c9ef51a56d72858d4e28fac5aa8294f2d0dc649a1efe3b0594982464764e0c68
 => => naming to docker.io/library/cart_service:latest
```

4.2.3 Push Cart Docker Image to DockerHub



```
PS D:\Projects\CTSE-Assignment-2\src\cart_service> docker tag cart_service:latest senuradockacc/cart_service:latest
PS D:\Projects\CTSE-Assignment-2\src\cart_service> docker push senuradockacc/cart_service:latest
The push refers to repository [docker.io/senuradockacc/cart_service]
d7995ac29d7d: Pushing [======>                                            ]  2.753MB/22.76MB
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
```

4.2.4 Cart Service DockerHub Overview



Advanced Image Management

View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

senuradockacc / **cart_service**

This repository does not have a description

Last pushed: 22 minutes ago

Docker commands

To push a new tag to this repository,

```
docker push senuradockacc/cart_service:tagname
```

Tags and Scans

This repository contains 9 tag(s).

TAG	OS	PULLED	PUSHED
v1.0.2		---	22 minutes ago
v1.0.1		---	20 hours ago
latest		---	2 days ago

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

Cart Service DockerHub link

https://hub.docker.com/repository/docker/senuradockacc/cart_service

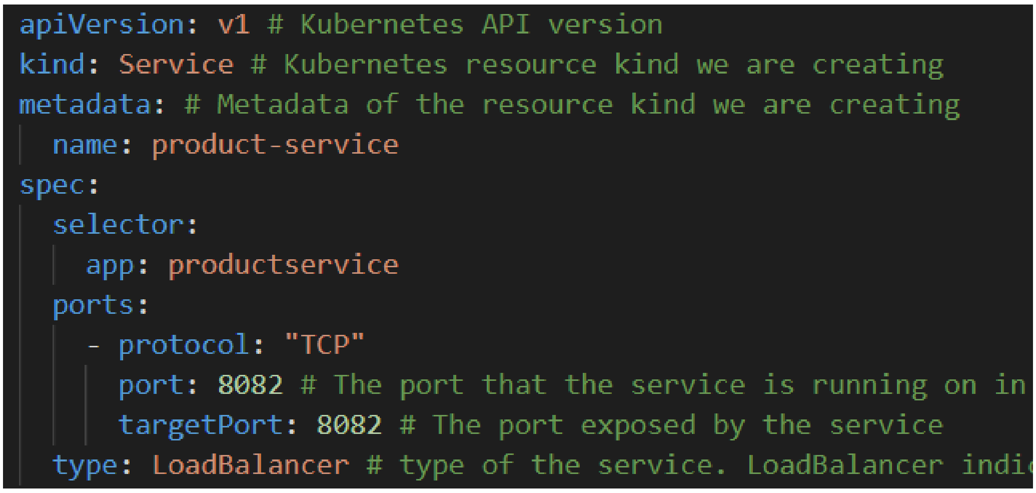
5.0 TASK 2 - DEPLOY SERVICES TO K8S CLUSTER

The cloud provider for this project was Azure Kubernetes Service (AKS). To deploy the project's microservices, a single node cluster was constructed. Then, inside the release folder, add the k8s configuration files for each microservice. As a result, we can deploy all of the microservices in the k8s cluster by performing the following command.

```
kubectl apply -f release/
```

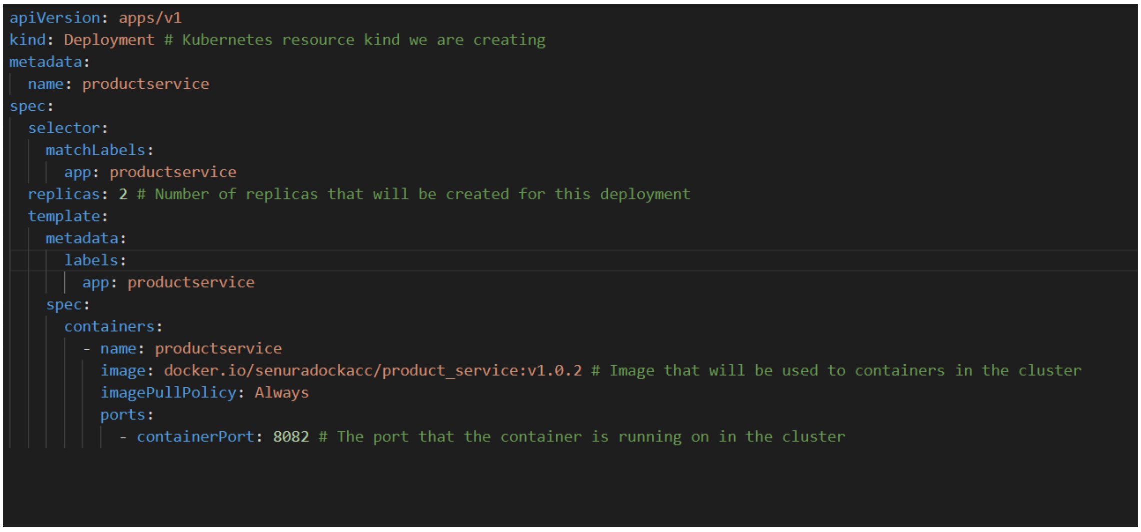
5.1 Product Service k8s Config YAML Files

5.1.1 Product Service k8s Service YAML File



```yaml
apiVersion: v1 # Kubernetes API version
kind: Service # Kubernetes resource kind we are creating
metadata: # Metadata of the resource kind we are creating
  name: product-service
spec:
  selector:
    app: productservice
  ports:
    - protocol: "TCP"
      port: 8082 # The port that the service is running on in
      targetPort: 8082 # The port exposed by the service
  type: LoadBalancer # type of the service. LoadBalancer indi
```

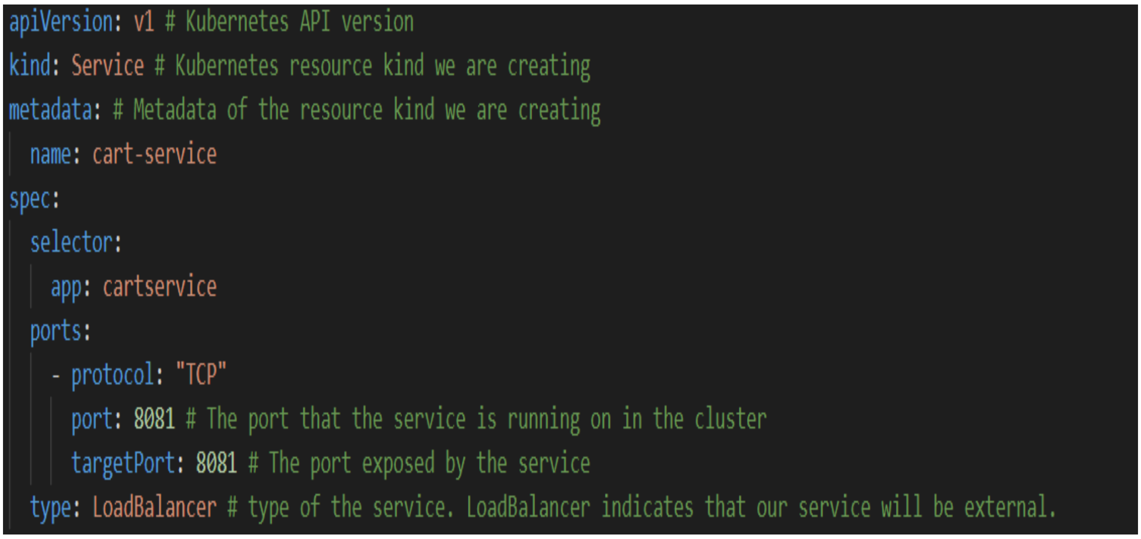
5.1.2 Product Service k8s Deployment YAML File



```yaml
apiVersion: apps/v1
kind: Deployment # Kubernetes resource kind we are creating
metadata:
  name: productservice
spec:
  selector:
    matchLabels:
      app: productservice
  replicas: 2 # Number of replicas that will be created for this deployment
  template:
    metadata:
      labels:
        app: productservice
    spec:
      containers:
        - name: productservice
          image: docker.io/senuradockacc/product_service:v1.0.2 # Image that will be used to containers in the cluster
          imagePullPolicy: Always
          ports:
            - containerPort: 8082 # The port that the container is running on in the cluster
```

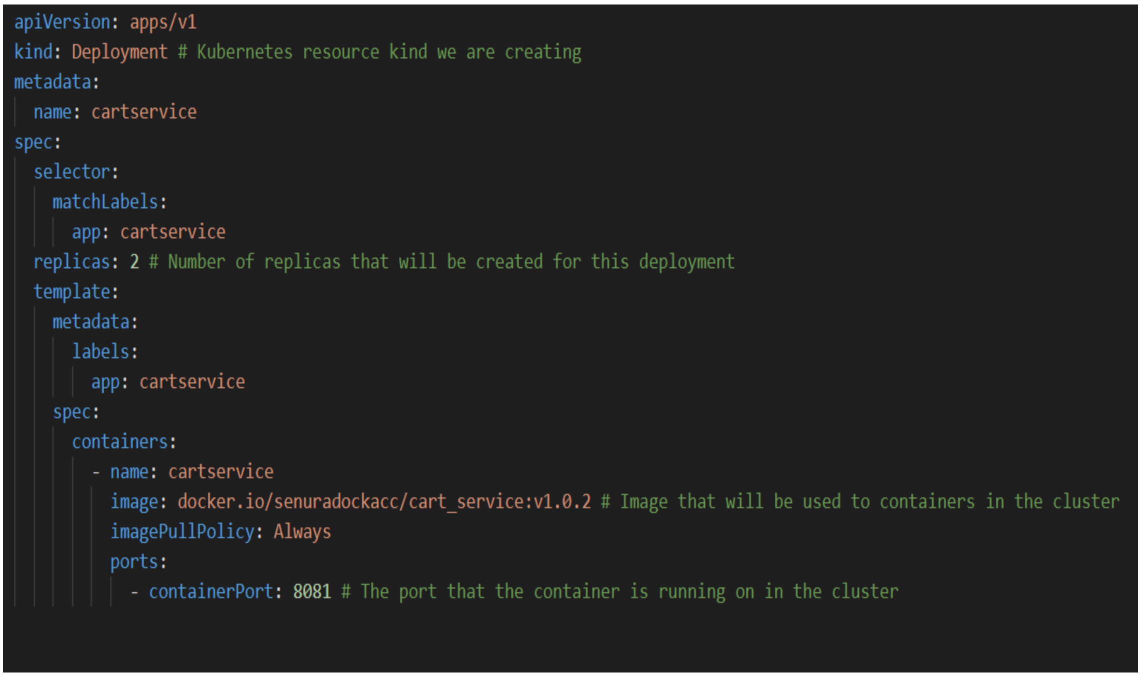
5.2 Cart Service k8s Config YAML Files

5.2.1 Cart Service k8s Service YAML File



```yaml
apiVersion: v1 # Kubernetes API version
kind: Service # Kubernetes resource kind we are creating
metadata: # Metadata of the resource kind we are creating
  name: cart-service
spec:
  selector:
    app: cartservice
  ports:
    - protocol: "TCP"
      port: 8081 # The port that the service is running on in the cluster
      targetPort: 8081 # The port exposed by the service
  type: LoadBalancer # type of the service. LoadBalancer indicates that our service will be external.
```

5.2.3 Cart Service k8s Deployment YAML File

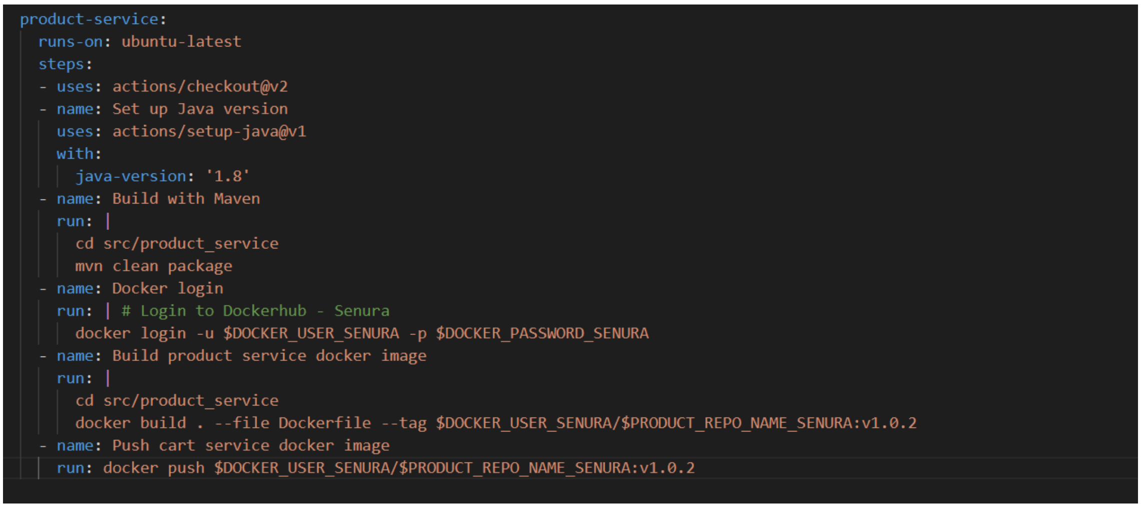


```yaml
apiVersion: apps/v1
kind: Deployment # Kubernetes resource kind we are creating
metadata:
  name: cartservice
spec:
  selector:
    matchLabels:
      app: cartservice
  replicas: 2 # Number of replicas that will be created for this deployment
  template:
    metadata:
      labels:
        app: cartservice
    spec:
      containers:
        - name: cartservice
          image: docker.io/senuradockacc/cart_service:v1.0.2 # Image that will be used to containers in the cluster
          imagePullPolicy: Always
          ports:
            - containerPort: 8081 # The port that the container is running on in the cluster
```

6.0 TASK 3 – CI/CD PIPELINE IN GITHUB ACTIONS

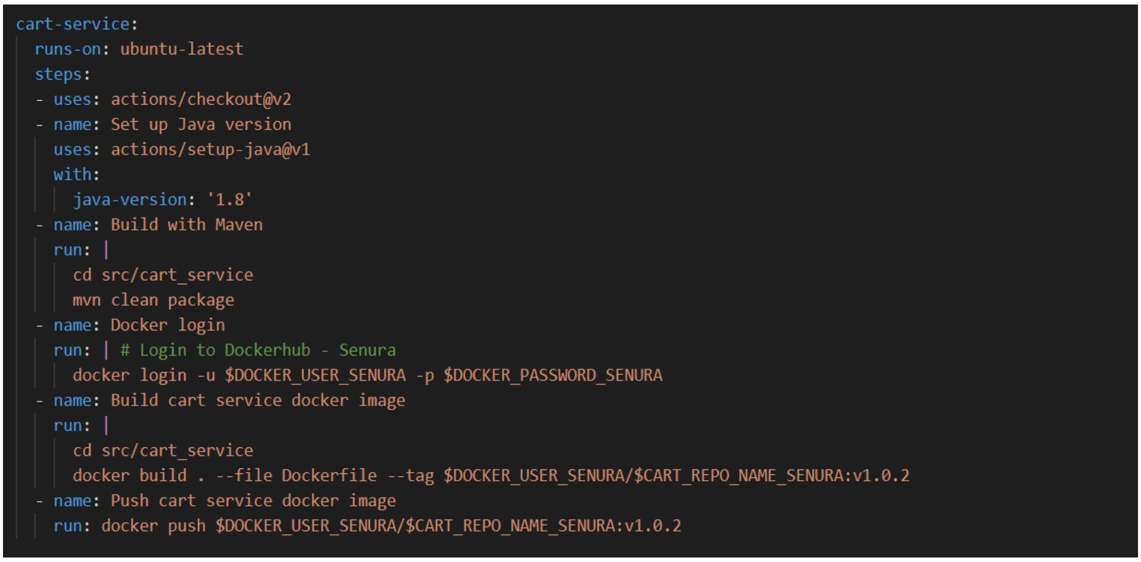
To produce the container images and publish them to the appropriate DockerHub account, this project employs a CI/CD pipeline. The deployment pipeline will deploy the new changes to the k8s cluster after the building and pushing processes for all services are completed. DockerHub credentials and k8s cluster credentials are stored in GitHub secrets. As a result, the credentials are hidden from the general public.

6.1 Deployment YAML Configuration of Product Service



```
product-service:
  runs-on: ubuntu-latest
  steps:
  - uses: actions/checkout@v2
  - name: Set up Java version
    uses: actions/setup-java@v1
    with:
      java-version: '1.8'
  - name: Build with Maven
    run: |
      cd src/product_service
      mvn clean package
  - name: Docker login
    run: | # Login to Dockerhub - Senura
      docker login -u $DOCKER_USER_SENURA -p $DOCKER_PASSWORD_SENURA
  - name: Build product service docker image
    run: |
      cd src/product_service
      docker build . --file Dockerfile --tag $DOCKER_USER_SENURA/$PRODUCT_REPO_NAME_SENURA:v1.0.2
  - name: Push cart service docker image
    run: docker push $DOCKER_USER_SENURA/$PRODUCT_REPO_NAME_SENURA:v1.0.2
```

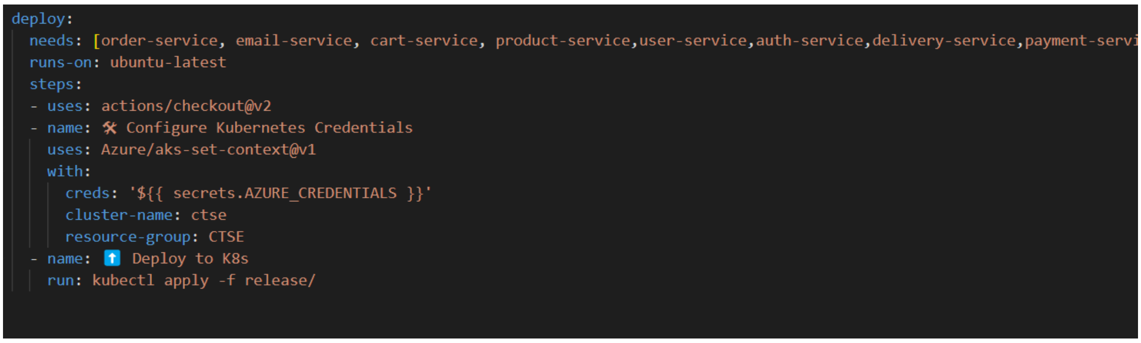
6.2 Deployment YAML Configuration of Cart Service



```
cart-service:
  runs-on: ubuntu-latest
  steps:
  - uses: actions/checkout@v2
  - name: Set up Java version
    uses: actions/setup-java@v1
    with:
      java-version: '1.8'
  - name: Build with Maven
    run: |
      cd src/cart_service
      mvn clean package
  - name: Docker login
    run: | # Login to Dockerhub - Senura
      docker login -u $DOCKER_USER_SENURA -p $DOCKER_PASSWORD_SENURA
  - name: Build cart service docker image
    run: |
      cd src/cart_service
      docker build . --file Dockerfile --tag $DOCKER_USER_SENURA/$CART_REPO_NAME_SENURA:v1.0.2
  - name: Push cart service docker image
    run: docker push $DOCKER_USER_SENURA/$CART_REPO_NAME_SENURA:v1.0.2
```

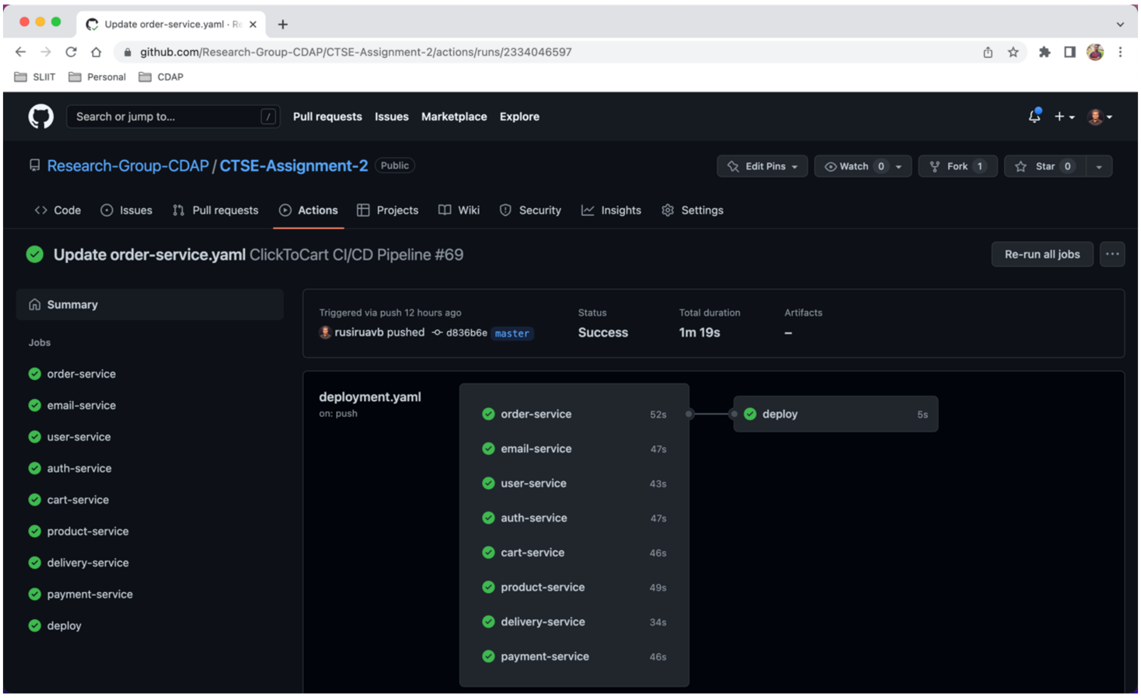
6.3 Deployment to k8s Cluster

Following the successful build and push of Docker images, the following deployment process will begin to run, finally deploying all of the microservices to the k8s cluster. The deployment pipeline will not proceed until all of the images have been created and pushed.



```
deploy:
  needs: [order-service, email-service, cart-service, product-service,user-service,auth-service,delivery-service,payment-servi
  runs-on: ubuntu-latest
  steps:
  - uses: actions/checkout@v2
  - name: 🛠 Configure Kubernetes Credentials
    uses: Azure/aks-set-context@v1
    with:
      creds: '${{ secrets.AZURE_CREDENTIALS }}'
      cluster-name: ctse
      resource-group: CTSE
  - name: ⬆️ Deploy to K8s
    run: kubectl apply -f release/
```

6.4 Pipeline Running on GitHub Actions



Update order-service.yaml ClickToCart CI/CD Pipeline #69

Triggered via push 12 hours ago
rusiruavb pushed d836b6e master

Status: Success

Total duration: 1m 19s

Artifacts: –

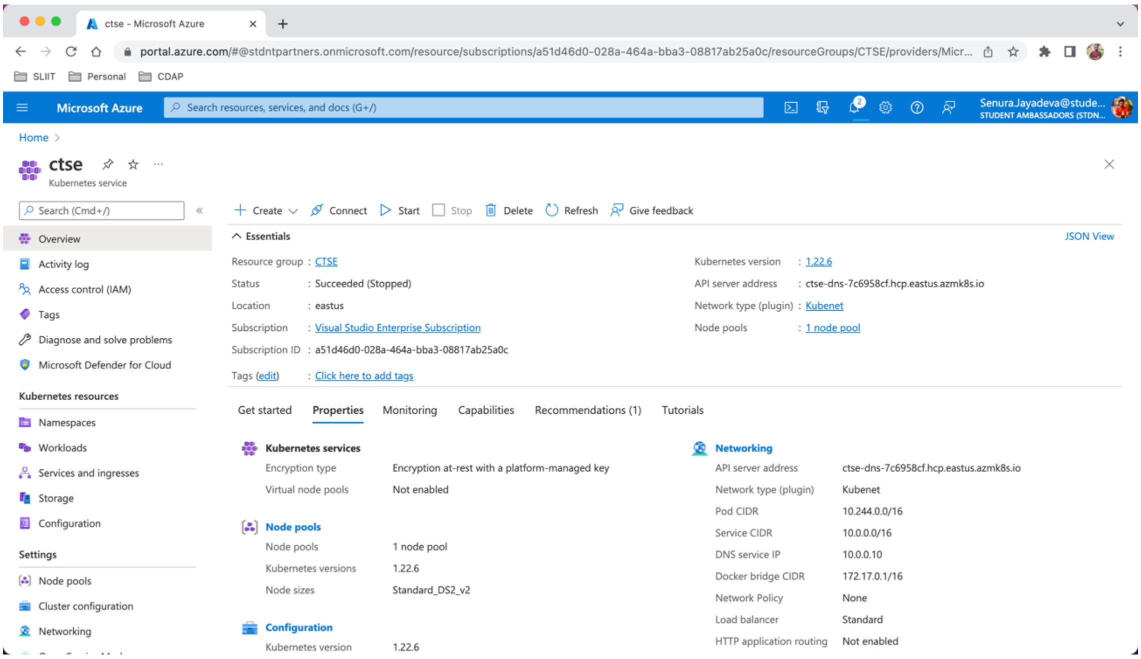
Jobs:
- order-service
- email-service
- user-service
- auth-service
- cart-service
- product-service
- delivery-service
- payment-service
- deploy

deployment.yaml
on: push

Job	Duration
order-service	52s
email-service	47s
user-service	43s
auth-service	47s
cart-service	46s
product-service	49s
delivery-service	34s
payment-service	46s
deploy	5s

7.0 K8S CLUSTER INFORMATION

7.1 K8s Cluster Overview on Azure Portal



Essentials

Resource group	CTSE
Status	Succeeded (Stopped)
Location	eastus
Subscription	Visual Studio Enterprise Subscription
Subscription ID	a51d46d0-028a-464a-bba3-08817ab25a0c
Tags (edit)	Click here to add tags
Kubernetes version	1.22.6
API server address	ctse-dns-7c6958cf.hcp.eastus.azmk8s.io
Network type (plugin)	Kubenet
Node pools	1 node pool

Kubernetes services

Encryption type	Encryption at-rest with a platform-managed key
Virtual node pools	Not enabled

Node pools

Node pools	1 node pool
Kubernetes versions	1.22.6
Node sizes	Standard_DS2_v2

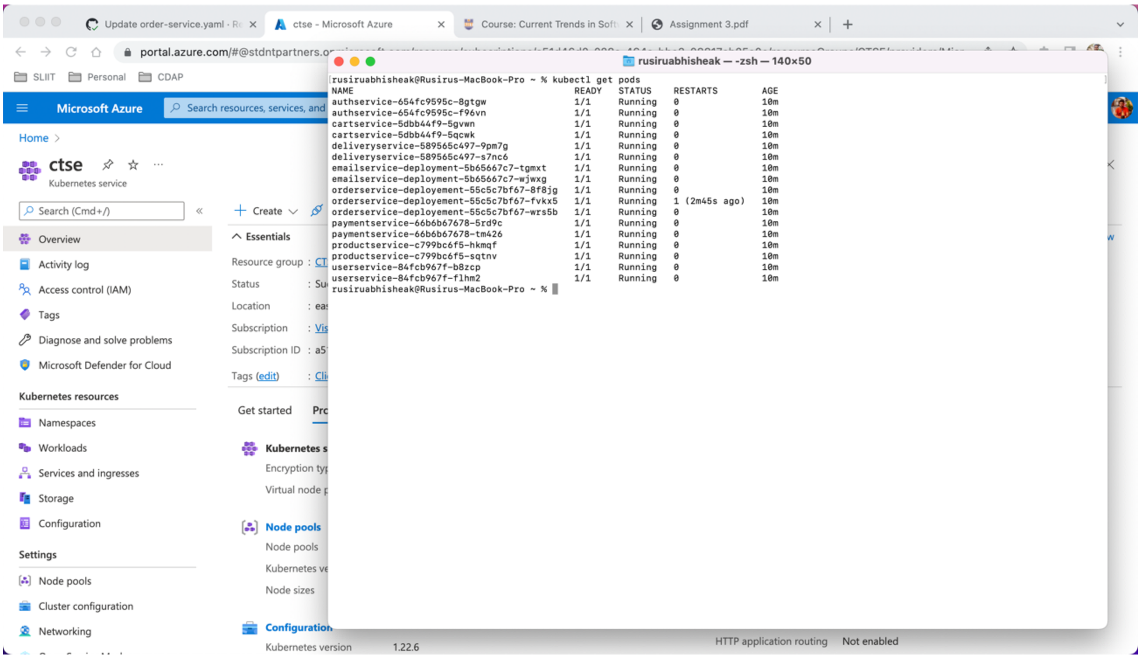
Configuration

Kubernetes version	1.22.6

Networking

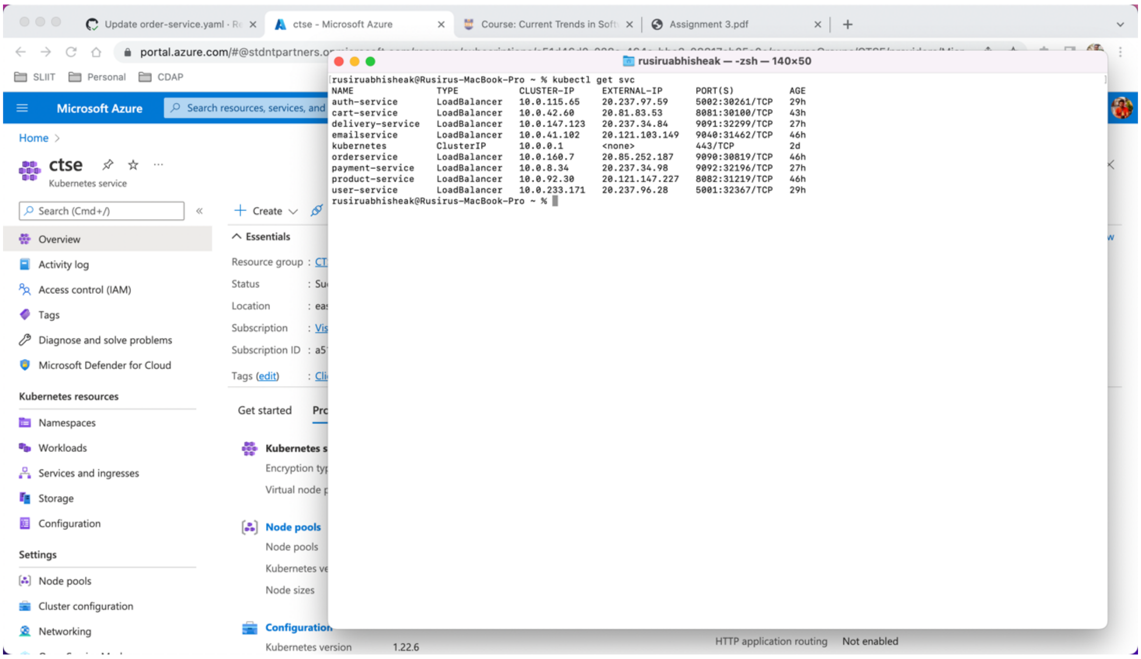
API server address	ctse-dns-7c6958cf.hcp.eastus.azmk8s.io
Network type (plugin)	Kubenet
Pod CIDR	10.244.0.0/16
Service CIDR	10.0.0.0/16
DNS service IP	10.0.0.10
Docker bridge CIDR	172.17.0.1/16
Network Policy	None
Load balancer	Standard
HTTP application routing	Not enabled

7.2 Service Pods Running on k8s Cluster


Update order-service.yaml
ctse - Microsoft Azure
Course: Current Trends in Soft
Assignment 3.pdf
rusiruabhisheak — -zsh — 140×50

```
rusiruabhisheak@Rusirus-MacBook-Pro ~ % kubectl get pods
NAME                                          READY   STATUS    RESTARTS       AGE
authservice-654fc9595c-8gtgw                  1/1     Running   0              10m
authservice-654fc9595c-f96vn                  1/1     Running   0              10m
cartservice-5dbb44f9-5gvwn                    1/1     Running   0              10m
cartservice-5dbb44f9-5qcwk                    1/1     Running   0              10m
deliveryservice-589565c497-9pm7g              1/1     Running   0              10m
deliveryservice-589565c497-s7nc6              1/1     Running   0              10m
emailservice-deployment-5b65667c7-tgmxt       1/1     Running   0              10m
emailservice-deployment-5b65667c7-wjwxg       1/1     Running   0              10m
orderservice-deployement-55c5c7bf67-8f8jg     1/1     Running   0              10m
orderservice-deployement-55c5c7bf67-fvkx5     1/1     Running   1 (2m45s ago)  10m
orderservice-deployement-55c5c7bf67-wrs5b     1/1     Running   0              10m
paymentservice-66b6b67678-5rd9c               1/1     Running   0              10m
paymentservice-66b6b67678-tm426               1/1     Running   0              10m
productservice-c799bc6f5-hkmqf                1/1     Running   0              10m
productservice-c799bc6f5-sqtnv                1/1     Running   0              10m
userservice-84fcb967f-b8zcp                   1/1     Running   0              10m
userservice-84fcb967f-flhm2                   1/1     Running   0              10m
rusiruabhisheak@Rusirus-MacBook-Pro ~ %
```

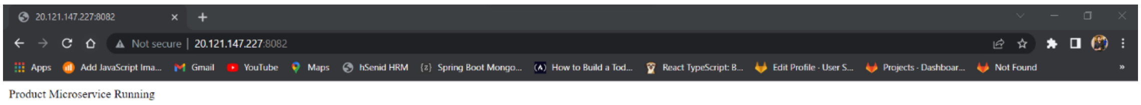
7.3 Microservices Running on k8s Cluster



rusiruabhisheak — -zsh — 140×50

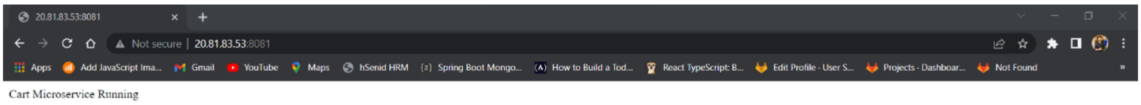
```
rusiruabhisheak@Rusirus-MacBook-Pro ~ % kubectl get svc
NAME               TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
auth-service       LoadBalancer   10.0.115.65    20.237.97.59     5002:30261/TCP   29h
cart-service       LoadBalancer   10.0.42.60     20.81.83.53      8081:30100/TCP   43h
delivery-service   LoadBalancer   10.0.147.123   20.237.34.84     9091:32299/TCP   27h
emailservice       LoadBalancer   10.0.41.102    20.121.103.149   9040:31462/TCP   46h
kubernetes         ClusterIP      10.0.0.1       <none>           443/TCP          2d
orderservice       LoadBalancer   10.0.160.7     20.85.252.187    9090:30819/TCP   46h
payment-service    LoadBalancer   10.0.8.34      20.237.34.98     9092:32196/TCP   27h
product-service    LoadBalancer   10.0.92.30     20.121.147.227   8082:31219/TCP   46h
user-service       LoadBalancer   10.0.233.171   20.237.96.28     5001:32367/TCP   29h
rusiruabhisheak@Rusirus-MacBook-Pro ~ %
```

7.4 Product Service on Browser



Product Microservice Running

7.5 Cart Service on Browser

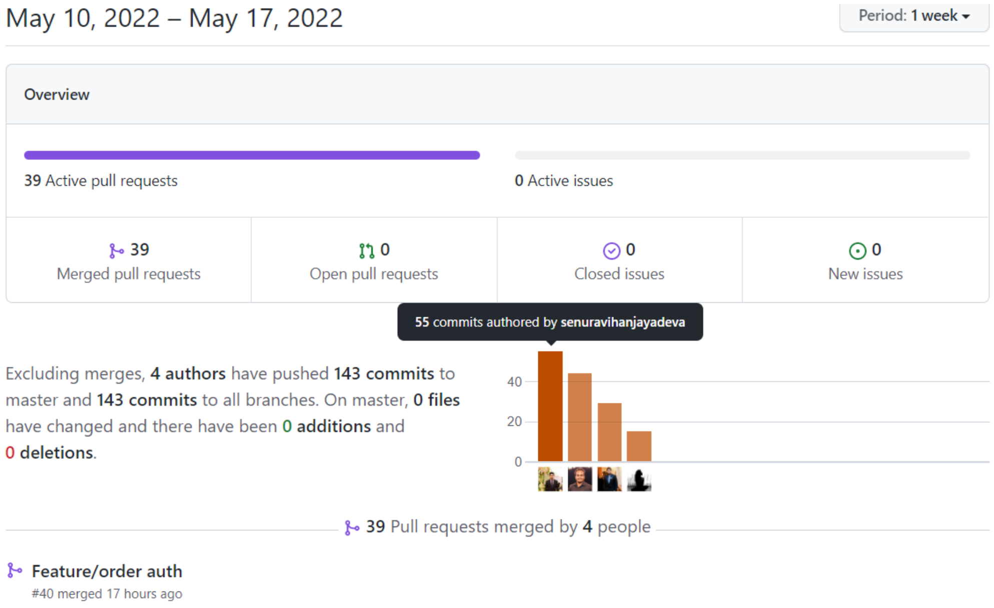


Cart Microservice Running

8.0 GITHUB CONTRIBUTION

The version control system is Git, and GitHub is used to cooperatively implement microservices and build CI/CD pipelines. To keep track of code changes, the repository has three branches: development, staging, and production (master). Any member of the group cannot directly merge modifications to the master branch. As a result, if a member wants to update the code, they should create a separate feature branch for the implementation. The feature branch creates a Pull Request (PR) to the development branch. The code is then merged into the development branch following the review process. Create a new PR from the development branch to the staging branch, and then from the staging branch to the master branch.

8.1 Individual Contribution to Project



May 10, 2022 – May 17, 2022

Period: 1 week

Overview

39 Active pull requests

0 Active issues

39 Merged pull requests

0 Open pull requests

0 Closed issues

0 New issues

55 commits authored by senuravihanjayadeva

Excluding merges, 4 authors have pushed 143 commits to master and 143 commits to all branches. On master, 0 files have changed and there have been 0 additions and 0 deletions.

39 Pull requests merged by 4 people

Feature/order auth
#40 merged 17 hours ago