CURRENT TRENDS IN SOFTWARE ENGINEERING

SE4010

Microservice Assignment

Palliyaguruge D.N

IT19120980

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

Contents

project Overview ... 4

1.1 Tools and Technologies ... 4

2.0 implemented microservices ... 5

2.1 Details of Microservices ... 5

2.2 Individual implemented services ... 5

3.0 individual service overview ... 6

3.1 Delivery Service ... 6

3.2 Payment Service ... 6

4.0 task 1 – Dockerize applications ... 7

4.1 Containerize Delivery Service ... 7

4.1.1 Dockerfile of Delivery Service ... 7

4.1.2 Delivery Service Container Image Building ... 8

4.1.3 Push Delivery Docker Image to DockerHub ... 9

4.1.4 Delivery Service DockerHub Overview ... 9

4.2 Containerize Payment Service ... 11

4.2.1 Dockerfile of Payment Service ... 11

4.2.2 Payment Service Container Image Building ... 11

4.2.3 Push Payment Docker Image to DockerHub ... 11

4.2.4 Payment Service DockerHub Overview ... 12

5.0 task 2 - deploy services to k8s cluster 12

5.1 Delivery Service k8s Config YAML Files 13

5.1.1 Delivery Service k8s Service YAML File 13

5.1.2 Delivery Service k8s Deployment YAML File 13

5.2 Payment Service k8s Config YAML Files 14

5.2.1 Payment Service k8s Service YAML File 14

5.2.2 Payment Service k8s Deployment YAML File 14

6.0 task 3 – ci/cd pipeline in github actions 15

6.1 Deployment YAML Configuration of Payment Service 15

6.2 Deployment YAML Configuration of Delivery Service 16

6.3 Deployment to k8s Cluster 16

6.4 Pipeline Running on GitHub Actions 17

7.0 k8s cluster information 18

7.1 K8s Cluster Overview on Azure Portal 18

7.2 Service Pods Running on k8s Cluster 18

7.3 Microservices Running on k8s Cluster 19

PROJECT OVERVIEW

This system is an e-commerce system that allow users to buy, view and sell items. Previously, the system was implemented using monolithic architecture. Because of the use of monolithic system team had to face some difficulties in the development time. One of the major drawbacks of monolithic architecture is, in the application deployment, the whole application needs to be deployed to the server even though the change is not affected to the other functionalities. This makes other functionalities not available in the deployment time. To fix the issue with monolithic architecture, the team decided to implement the application using microservice architecture.

In microservice architecture each functionality is divided into a separate service in the application. The major advantage of this approach is it makes deployment of the application much faster than the monolith architecture. For example, if the change is applied to cart service, in the deployment cart service will only get deployed to the cluster. Therefore, there is no downtime for other services.

1.1 Tools and Technologies

Multiple programming languages are used to build the microservice system. Git and GitHub are used to maintain the versions of the system and collaborate the project. Docker is used to containerize the applications. DockerHub is used to store the container images. Azure is used as cloud provider to deploy the Kubernetes cluster using Azure Kubernetes Service (AKS).

2.0 IMPLEMENTED MICROSERVICES

The application has 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

- Payment service
- Delivery service

3.0 INDIVIDUAL SERVICE OVERVIEW

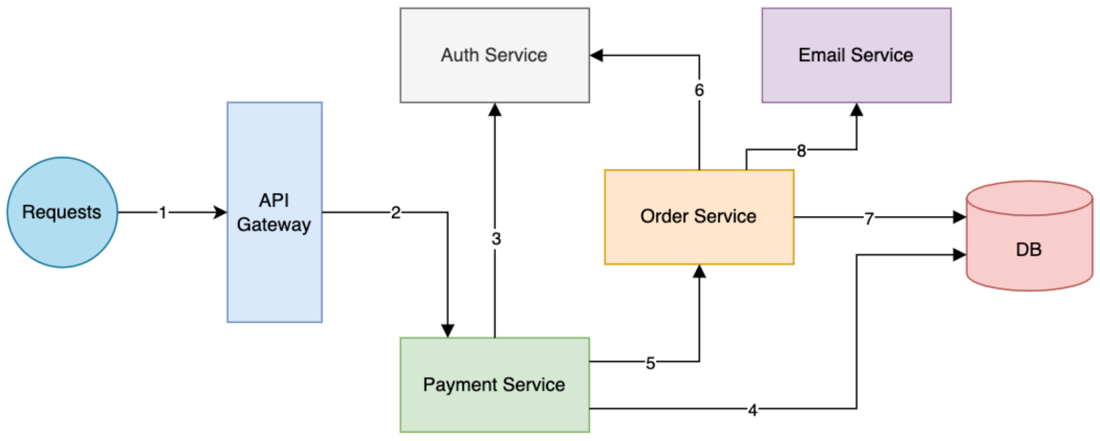
3.1 Delivery Service

Delivery service is responsible for delivering the orders that have been placed by the customers. Java programming language and spring boot framework was used as the core technology to implement the delivery service.

Once the user has made the payment the order will be placed, and the system admin can user delivery service to deliver the placed orders. Once the delivery is successful a record will be added to MongoDB database.

3.2 Payment Service

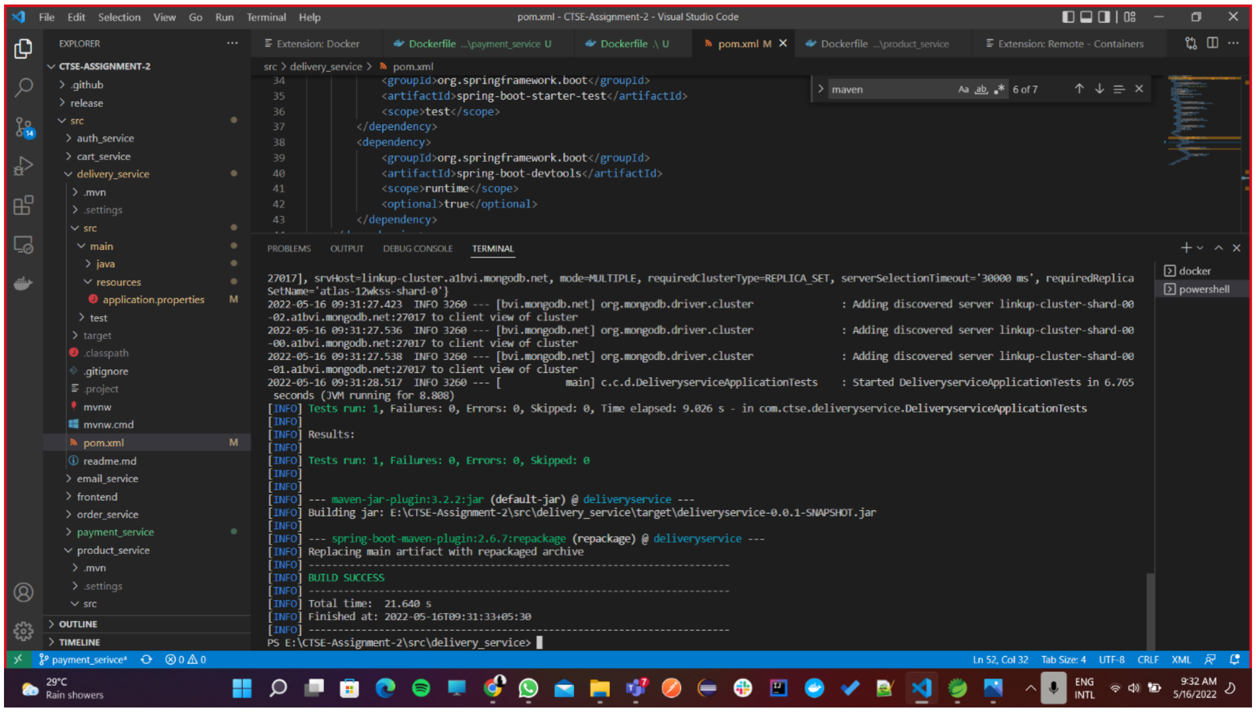
Payment service is used to make payments for the orders that have been placed. This service also implemented using Java Spring Boot. Once the payment is made and after the successful response a request will be sent to Order service and an order will be added. Successful records will be saved in the database.



4.0 TASK 1 – DOCKERIZE APPLICATIONS

Both delivery service and payment service applications have `.properties` file that contains MongoDB connection string, port number and cluster IP address of auth, user and order services. Before building the docker image we have to build a jar file of the relevant service. Since the implementation was done in Java we use the following command to build the jar.

mvn clean package



```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
```

```
27017], srvHost=linkup-cluster.a1bvi.mongodb.net, mode=MULTIPLE, requiredClusterType=REPLICA_SET, serverSelectionTimeout='30000 ms', requiredReplica
SetName='atlas-12wkss-shard-0'}
2022-05-16 09:31:27.423  INFO 3260 --- [bvi.mongodb.net] org.mongodb.driver.cluster               : Adding discovered server linkup-cluster-shard-00
-02.a1bvi.mongodb.net:27017 to client view of cluster
2022-05-16 09:31:27.536  INFO 3260 --- [bvi.mongodb.net] org.mongodb.driver.cluster               : Adding discovered server linkup-cluster-shard-00
-00.a1bvi.mongodb.net:27017 to client view of cluster
2022-05-16 09:31:27.538  INFO 3260 --- [bvi.mongodb.net] org.mongodb.driver.cluster               : Adding discovered server linkup-cluster-shard-00
-01.a1bvi.mongodb.net:27017 to client view of cluster
2022-05-16 09:31:28.517  INFO 3260 --- [           main] c.c.d.DeliveryserviceApplicationTests    : Started DeliveryserviceApplicationTests in 6.765
 seconds (JVM running for 8.808)
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 9.026 s - in com.ctse.deliveryservice.DeliveryserviceApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ deliveryservice ---
[INFO] Building jar: E:\CTSE-Assignment-2\src\delivery_service\target\deliveryservice-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.6.7:repackage (repackage) @ deliveryservice ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  21.640 s
[INFO] Finished at: 2022-05-16T09:31:33+05:30
[INFO] ------------------------------------------------------------------------
PS E:\CTSE-Assignment-2\src\delivery_service>
```

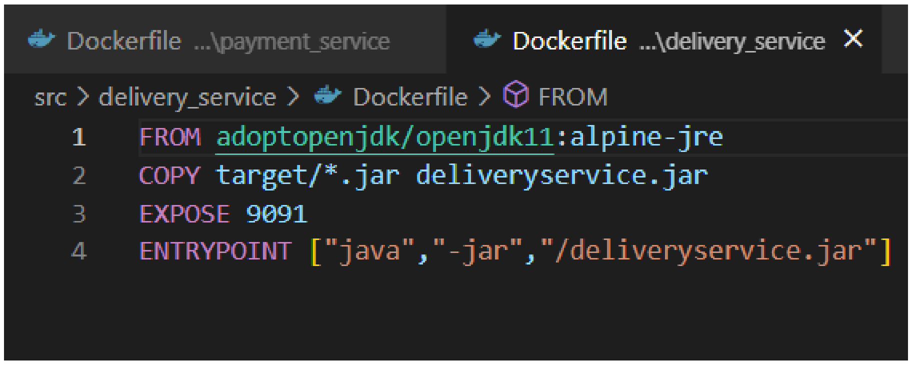
After the build is success we use relevant docker commands to build the docker image.

docker build -t deliveryservice:latest .

4.1 Containerize Delivery Service

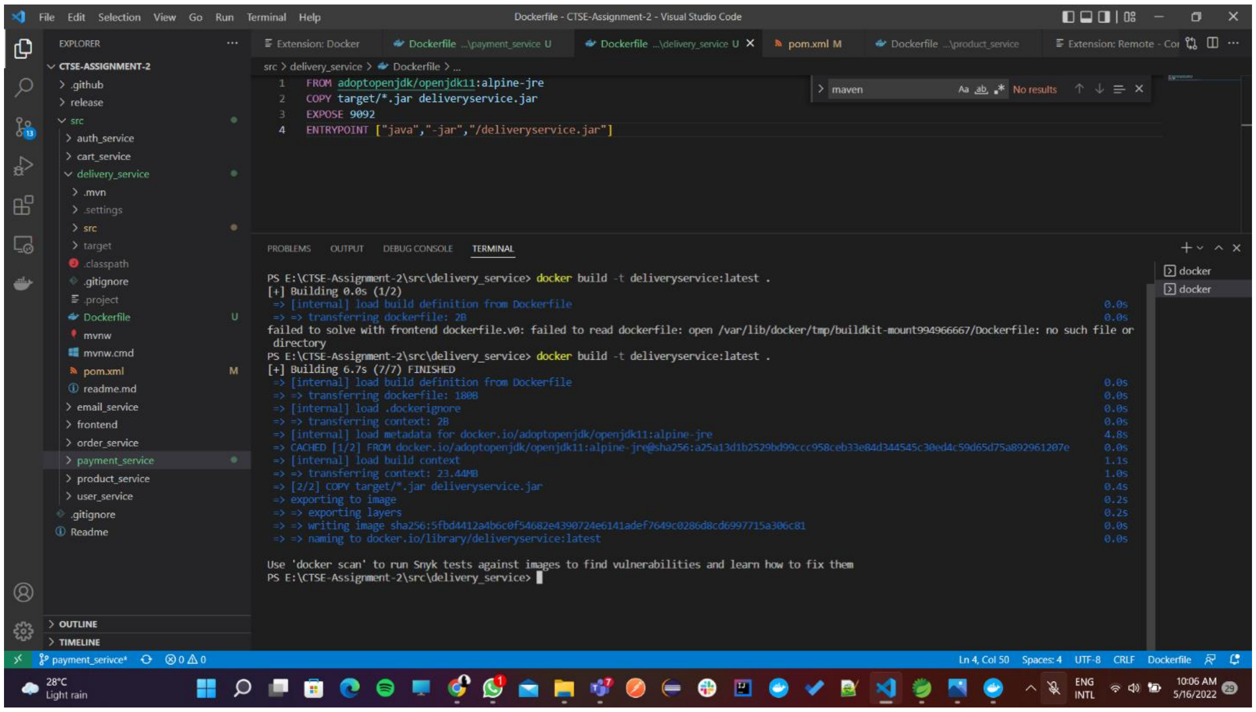
4.1.1 Dockerfile of Delivery Service

Used multistage Docker building mechanism to optimize the Docker building process and reduce the size of the image.



```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY target/*.jar deliveryservice.jar
EXPOSE 9091
ENTRYPOINT ["java","-jar","/deliveryservice.jar"]
```

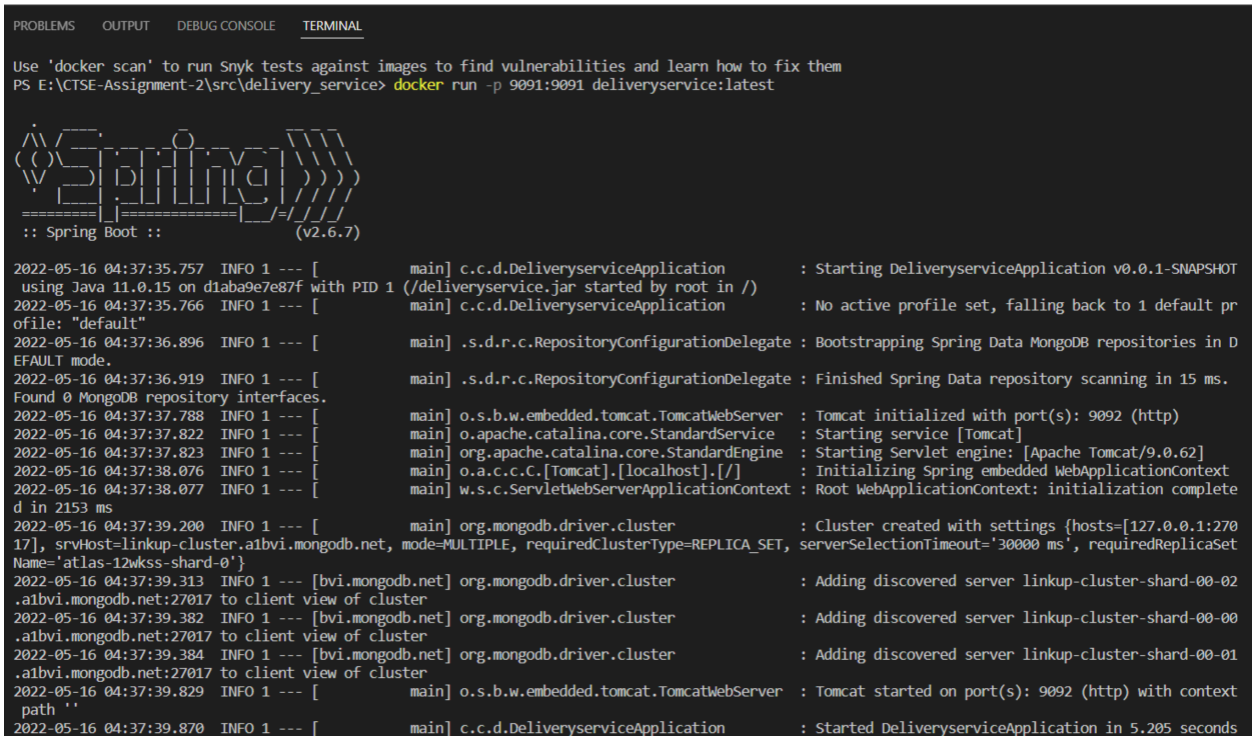
4.1.2 Delivery Service Container Image Building



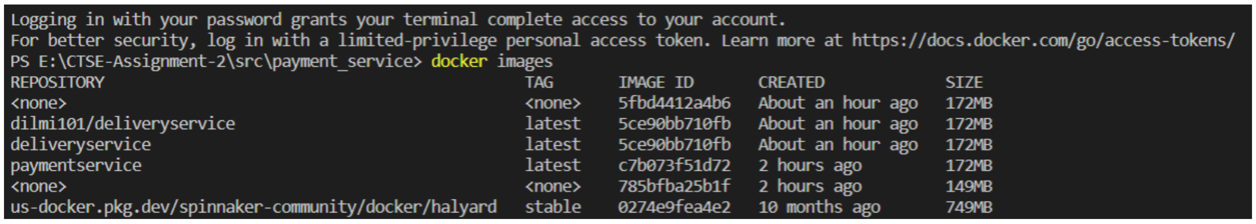
```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY target/*.jar deliveryservice.jar
EXPOSE 9092
ENTRYPOINT ["java","-jar","/deliveryservice.jar"]
```

```
PS E:\CTSE-Assignment-2\src\delivery_service> docker build -t deliveryservice:latest .
[+] Building 0.0s (1/2)
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 2B                                                                      0.0s
failed to solve with frontend dockerfile.v0: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount994966667/Dockerfile: no such file or
 directory
PS E:\CTSE-Assignment-2\src\delivery_service> docker build -t deliveryservice:latest .
[+] Building 6.7s (7/7) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 180B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/adoptopenjdk/openjdk11:alpine-jre                            4.8s
 => CACHED [1/2] FROM docker.io/adoptopenjdk/openjdk11:alpine-jre@sha256:a25a13d1b2529bd99ccc958ceb33e84d344545c30ed4c59d65d75a892961207e   0.0s
 => [internal] load build context                                                                       1.1s
 => => transferring context: 23.44MB                                                                    1.0s
 => [2/2] COPY target/*.jar deliveryservice.jar                                                         0.4s
 => exporting to image                                                                                  0.2s
 => => exporting layers                                                                                 0.2s
 => => writing image sha256:5fbd4412a4b6c0f54682e4390724e6141adef7649c0286d8cd6997715a306c81            0.0s
 => => naming to docker.io/library/deliveryservice:latest                                               0.0s

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

4.1.3 Push Delivery Docker Image to DockerHub

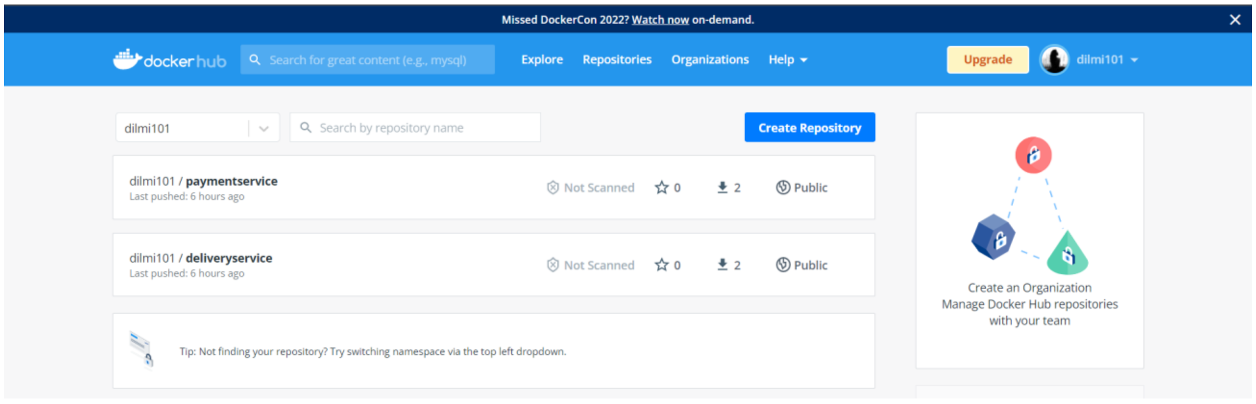


```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS E:\CTSE-Assignment-2\src\delivery_service> docker run -p 9091:9091 deliveryservice:latest

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.7)

2022-05-16 04:37:35.757  INFO 1 --- [           main] c.c.d.DeliveryserviceApplication        : Starting DeliveryserviceApplication v0.0.1-SNAPSHOT
 using Java 11.0.15 on d1aba9e7e87f with PID 1 (/deliveryservice.jar started by root in /)
2022-05-16 04:37:35.766  INFO 1 --- [           main] c.c.d.DeliveryserviceApplication        : No active profile set, falling back to 1 default pr
ofile: "default"
2022-05-16 04:37:36.896  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data MongoDB repositories in D
EFAULT mode.
2022-05-16 04:37:36.919  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 15 ms.
Found 0 MongoDB repository interfaces.
2022-05-16 04:37:37.788  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 9092 (http)
2022-05-16 04:37:37.822  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-05-16 04:37:37.823  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.62]
2022-05-16 04:37:38.076  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-05-16 04:37:38.077  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization complete
d in 2153 ms
2022-05-16 04:37:39.200  INFO 1 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[127.0.0.1:270
17], srvHost=linkup-cluster.a1bvi.mongodb.net, mode=MULTIPLE, requiredClusterType=REPLICA_SET, serverSelectionTimeout='30000 ms', requiredReplicaSet
Name='atlas-12wkss-shard-0'}
2022-05-16 04:37:39.313  INFO 1 --- [bvi.mongodb.net] org.mongodb.driver.cluster               : Adding discovered server linkup-cluster-shard-00-02
.a1bvi.mongodb.net:27017 to client view of cluster
2022-05-16 04:37:39.382  INFO 1 --- [bvi.mongodb.net] org.mongodb.driver.cluster               : Adding discovered server linkup-cluster-shard-00-00
.a1bvi.mongodb.net:27017 to client view of cluster
2022-05-16 04:37:39.384  INFO 1 --- [bvi.mongodb.net] org.mongodb.driver.cluster               : Adding discovered server linkup-cluster-shard-00-01
.a1bvi.mongodb.net:27017 to client view of cluster
2022-05-16 04:37:39.829  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9092 (http) with context
 path ''
2022-05-16 04:37:39.870  INFO 1 --- [           main] c.c.d.DeliveryserviceApplication        : Started DeliveryserviceApplication in 5.205 seconds
```



```
Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
PS E:\CTSE-Assignment-2\src\payment_service> docker images
REPOSITORY                                          TAG       IMAGE ID       CREATED           SIZE
<none>                                              <none>    5fbd4412a4b6   About an hour ago   172MB
dilmi101/deliveryservice                            latest    5ce90bb710fb   About an hour ago   172MB
deliveryservice                                     latest    5ce90bb710fb   About an hour ago   172MB
paymentservice                                      latest    c7b073f51d72   2 hours ago       172MB
<none>                                              <none>    785bfba25b1f   2 hours ago       149MB
us-docker.pkg.dev/spinnaker-community/docker/halyard   stable    0274e9fea4e2   10 months ago     749MB
```

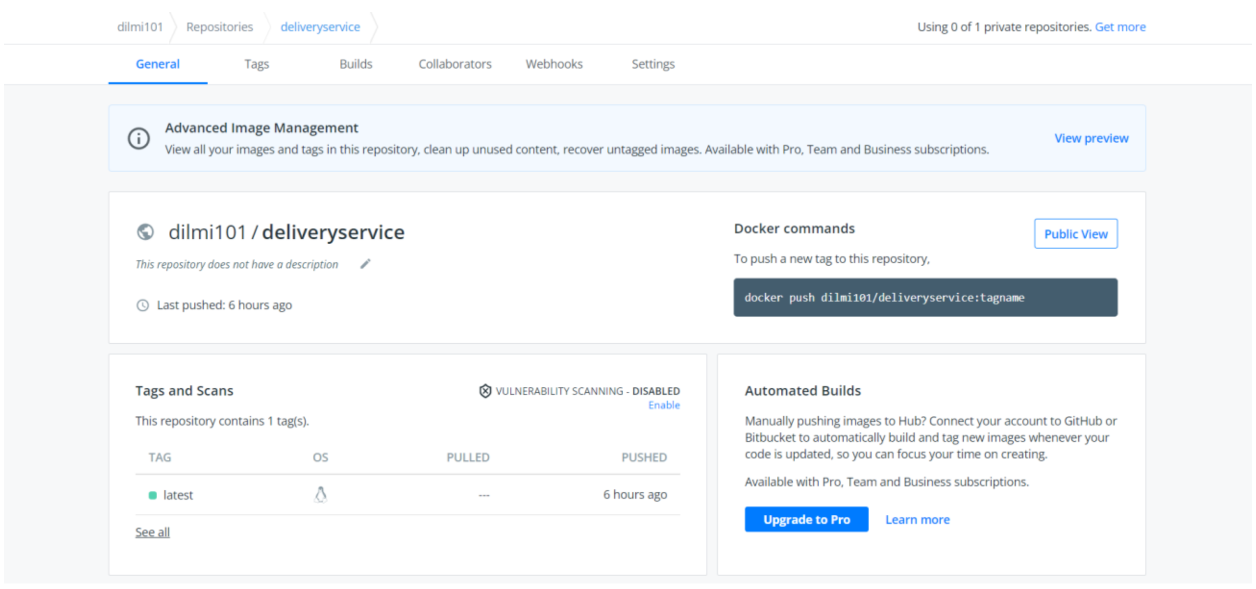
4.1.4 Delivery Service DockerHub Overview


Missed DockerCon 2022? Watch now on-demand.
docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Help
Upgrade
dilmi101
dilmi101
Search by repository name
Create Repository
Create an Organization
Manage Docker Hub repositories
with your team

dilmi101 / **paymentservice**
Last pushed: 6 hours ago
Not Scanned 0 2 Public

dilmi101 / **deliveryservice**
Last pushed: 6 hours ago
Not Scanned 0 2 Public

Tip: Not finding your repository? Try switching namespace via the top left dropdown.



dilmi101 Repositories deliveryservice

Using 0 of 1 private repositories. Get more

General Tags Builds Collaborators Webhooks Settings

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.

dilmi101 / deliveryservice

This repository does not have a description

Last pushed: 6 hours ago

Docker commands

To push a new tag to this repository,

```
docker push dilmi101/deliveryservice:tagname
```

Tags and Scans

VULNERABILITY SCANNING - DISABLED

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		---	6 hours 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.

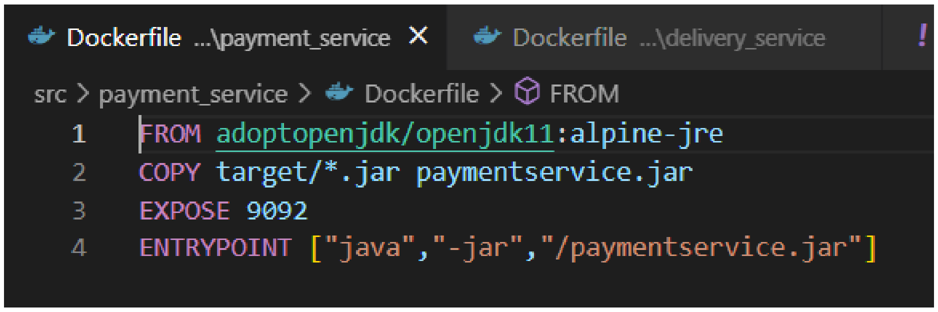
Available with Pro, Team and Business subscriptions.

Delivery service DockerHub link

https://hub.docker.com/repository/docker/dilmi101/deliveryservice

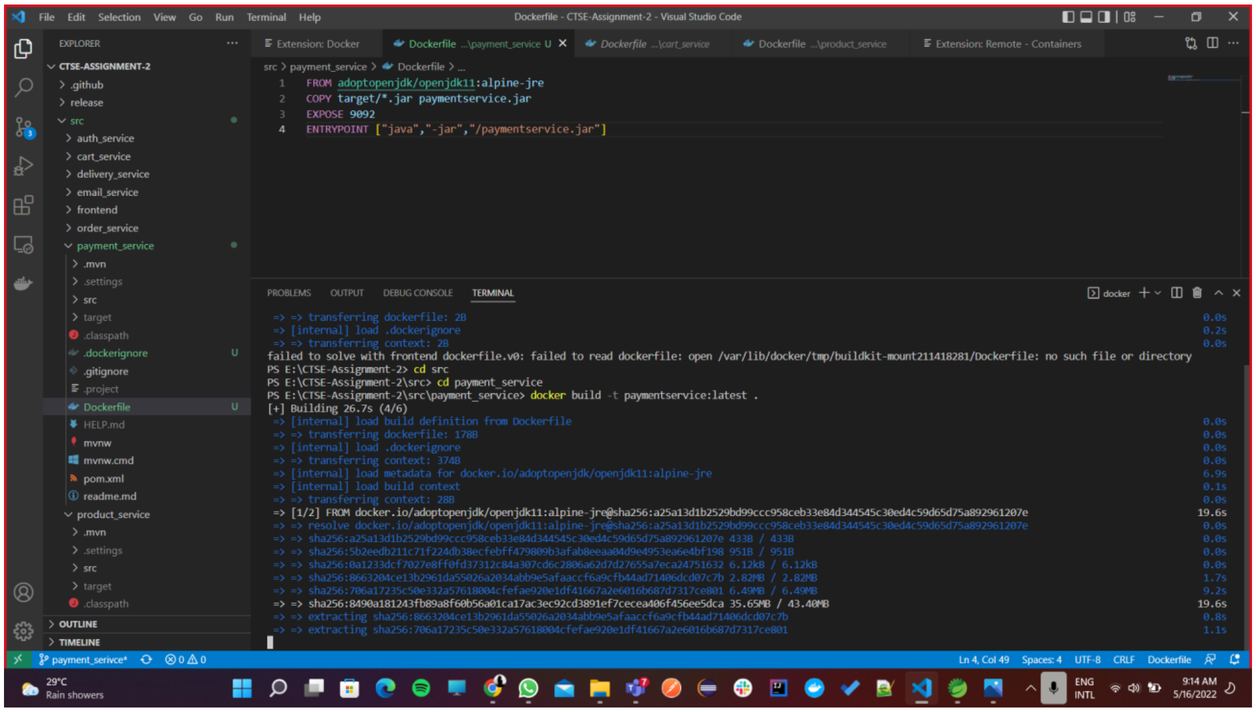
4.2 Containerize Payment Service

4.2.1 Dockerfile of Payment Service



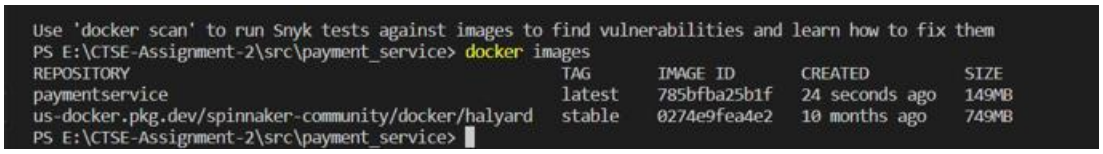
```
FROM adoptopenjdk/openjdk11:alpine-jre
COPY target/*.jar paymentservice.jar
EXPOSE 9092
ENTRYPOINT ["java","-jar","/paymentservice.jar"]
```

4.2.2 Payment Service Container Image Building



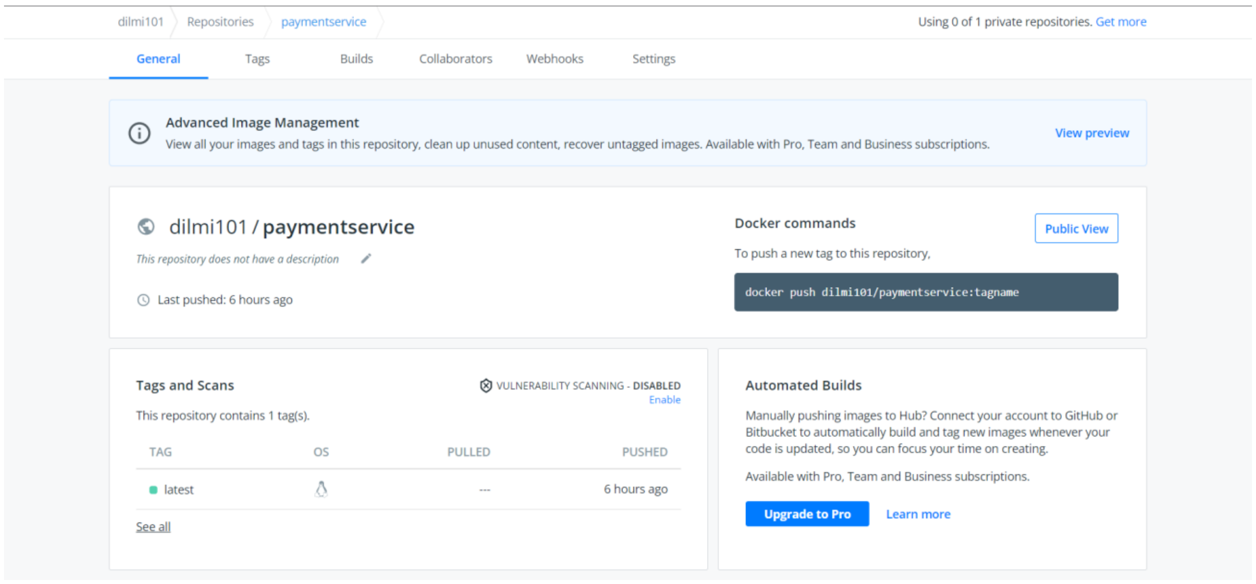
```
 => => transferring dockerfile: 2B                                                                      0.0s
 => [internal] load .dockerignore                                                                       0.2s
 => => transferring context: 2B                                                                         0.0s
failed to solve with frontend dockerfile.v0: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount211418281/Dockerfile: no such file or directory
PS E:\CTSE-Assignment-2> cd src
PS E:\CTSE-Assignment-2\src> cd payment_service
PS E:\CTSE-Assignment-2\src\payment_service> docker build -t paymentservice:latest .
[+] Building 26.7s (4/6)
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 178B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 374B                                                                       0.0s
 => [internal] load metadata for docker.io/adoptopenjdk/openjdk11:alpine-jre                            6.9s
 => [internal] load build context                                                                       0.1s
 => => transferring context: 28B                                                                        0.0s
 => [1/2] FROM docker.io/adoptopenjdk/openjdk11:alpine-jre@sha256:a25a13d1b2529bd99ccc958ceb33e84d344545c30ed4c59d65d75a892961207e   19.6s
 => => resolve docker.io/adoptopenjdk/openjdk11:alpine-jre@sha256:a25a13d1b2529bd99ccc958ceb33e84d344545c30ed4c59d65d75a892961207e   0.0s
 => => sha256:a25a13d1b2529bd99ccc958ceb33e84d344545c30ed4c59d65d75a892961207e 433B / 433B              0.0s
 => => sha256:5b2eedb211c71f224db38ecfebff479809b3afab8eeaa04d9e4953ea6e4bf198 951B / 951B              0.0s
 => => sha256:0a1233dcf7027e8ff0fd37312c84a307cd6c2806a62d7d27655a7eca24751632 6.12kB / 6.12kB          0.0s
 => => sha256:8663204ce13b2961da55026a2034abb9e5afaaccf6a9cfb44ad71406dcd07c7b 2.82MB / 2.82MB          1.7s
 => => sha256:706a17235c50e332a57618004cfefae920e1df41667a2e6016b687d7317ce801 6.49MB / 6.49MB          9.2s
 => => sha256:8490a181243fb89a8f60b56a01ca17ac3ec92cd3891ef7cecea406f456ee5dca 35.65MB / 43.40MB       19.6s
 => => extracting sha256:8663204ce13b2961da55026a2034abb9e5afaaccf6a9cfb44ad71406dcd07c7b               0.8s
 => => extracting sha256:706a17235c50e332a57618004cfefae920e1df41667a2e6016b687d7317ce801               1.1s
```

4.2.3 Push Payment Docker Image to DockerHub



```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS E:\CTSE-Assignment-2\src\payment_service> docker images
REPOSITORY                                          TAG       IMAGE ID       CREATED          SIZE
paymentservice                                      latest    785bfba25b1f   24 seconds ago   149MB
us-docker.pkg.dev/spinnaker-community/docker/halyard stable   0274e9fea4e2   10 months ago    749MB
PS E:\CTSE-Assignment-2\src\payment_service>
```

4.2.4 Payment 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.

dilmi101 / paymentservice
This repository does not have a description
Last pushed: 6 hours ago

Docker commands
To push a new tag to this repository,

```
docker push dilmi101/paymentservice:tagname
```

Tags and Scans
VULNERABILITY SCANNING - DISABLED Enable
This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		---	6 hours 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.

Payment Service DockerHub link

https://hub.docker.com/repository/docker/dilmi101/paymentservice

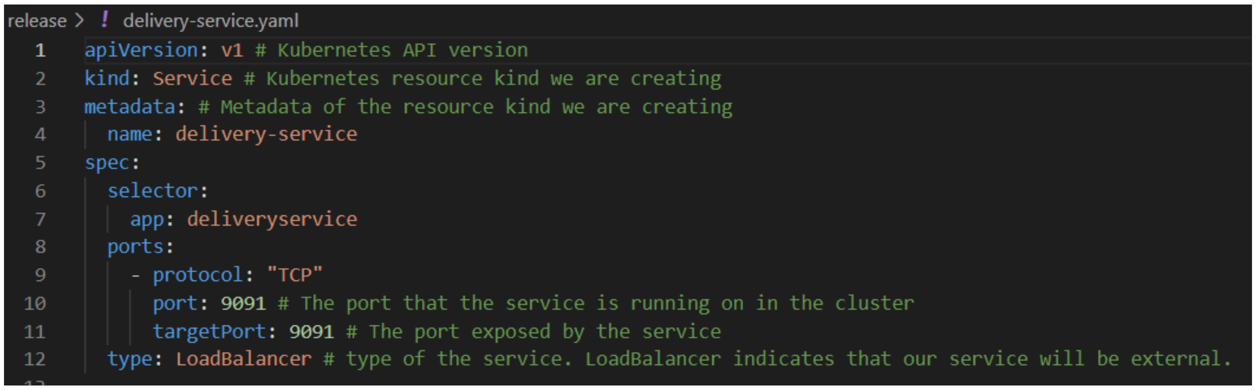
5.0 TASK 2 - DEPLOY SERVICES TO K8S CLUSTER

Azure Kubernetes Service (AKS) used as the cloud provider for this project. One node cluster has been created to deploy the microservices of the project. Then implement the k8s configuration files for each microservice inside the release folder. Therefore, we can deploy all the microservices by running following command in the k8s cluster.

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

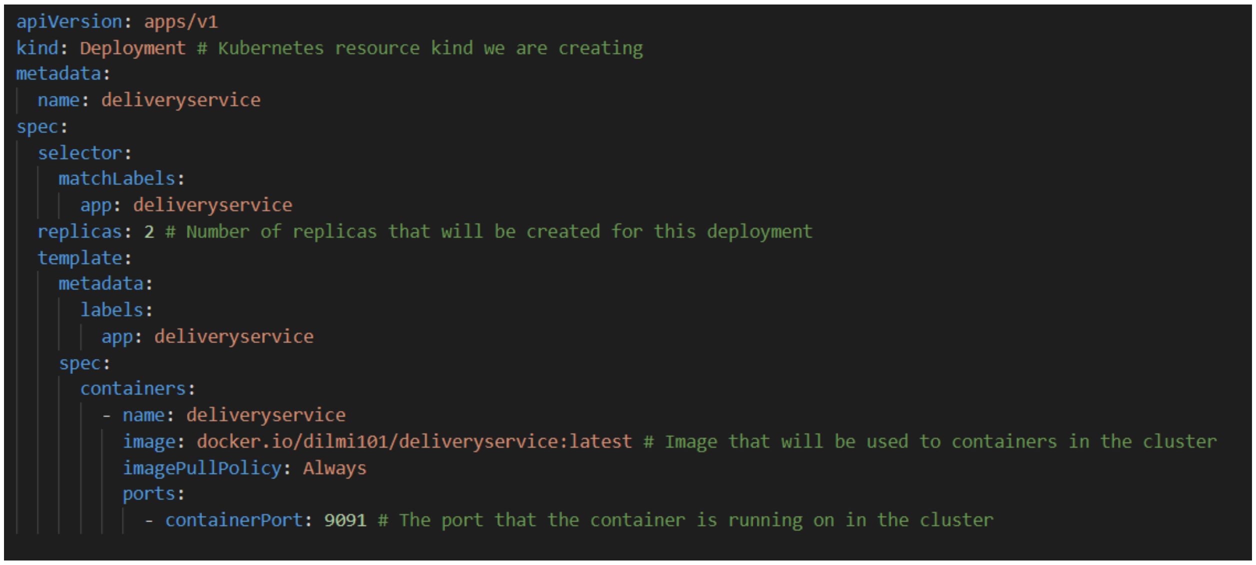
5.1 Delivery Service k8s Config YAML Files

5.1.1 Delivery 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: delivery-service
spec:
  selector:
    app: deliveryservice
  ports:
    - protocol: "TCP"
      port: 9091 # The port that the service is running on in the cluster
      targetPort: 9091 # The port exposed by the service
  type: LoadBalancer # type of the service. LoadBalancer indicates that our service will be external.
```

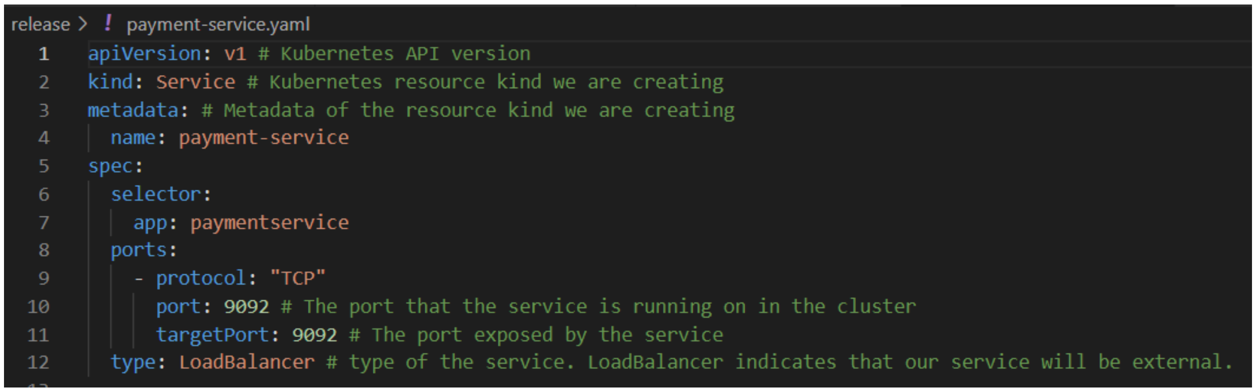
5.1.2 Delivery Service k8s Deployment YAML File



```yaml
apiVersion: apps/v1
kind: Deployment # Kubernetes resource kind we are creating
metadata:
  name: deliveryservice
spec:
  selector:
    matchLabels:
      app: deliveryservice
  replicas: 2 # Number of replicas that will be created for this deployment
  template:
    metadata:
      labels:
        app: deliveryservice
    spec:
      containers:
        - name: deliveryservice
          image: docker.io/dilmi101/deliveryservice:latest # Image that will be used to containers in the cluster
          imagePullPolicy: Always
          ports:
            - containerPort: 9091 # The port that the container is running on in the cluster
```

5.2 Payment Service k8s Config YAML Files

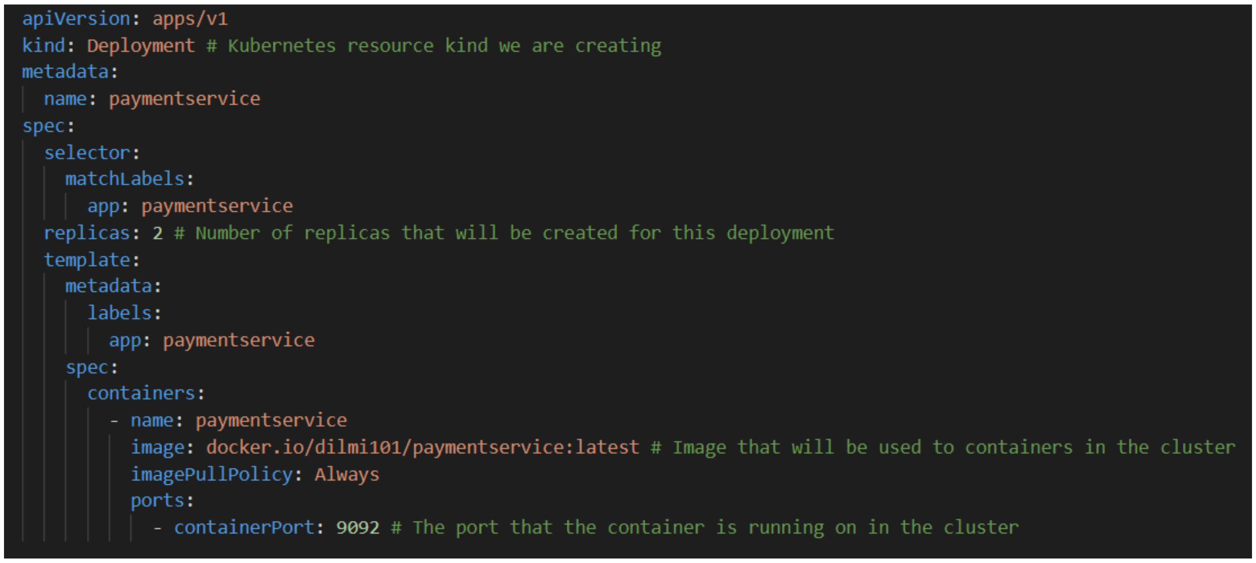
5.2.1 Payment Service k8s Service YAML File



release > ! payment-service.yaml

```yaml
apiVersion: v1 # Kubernetes API version
kind: Service # Kubernetes resource kind we are creating
metadata: # Metadata of the resource kind we are creating
  name: payment-service
spec:
  selector:
    app: paymentservice
  ports:
    - protocol: "TCP"
      port: 9092 # The port that the service is running on in the cluster
      targetPort: 9092 # The port exposed by the service
  type: LoadBalancer # type of the service. LoadBalancer indicates that our service will be external.
```

5.2.2 Payment Service k8s Deployment YAML File

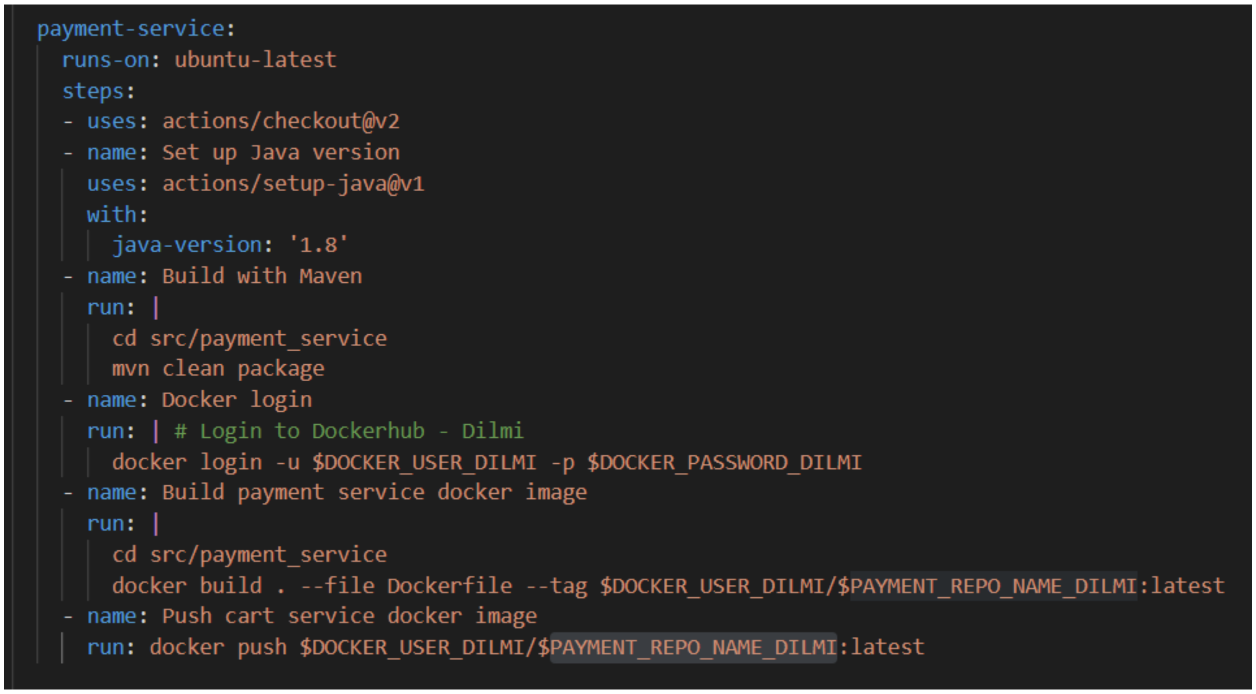


```yaml
apiVersion: apps/v1
kind: Deployment # Kubernetes resource kind we are creating
metadata:
  name: paymentservice
spec:
  selector:
    matchLabels:
      app: paymentservice
  replicas: 2 # Number of replicas that will be created for this deployment
  template:
    metadata:
      labels:
        app: paymentservice
    spec:
      containers:
        - name: paymentservice
          image: docker.io/dilmi101/paymentservice:latest # Image that will be used to containers in the cluster
          imagePullPolicy: Always
          ports:
            - containerPort: 9092 # The port that the container is running on in the cluster
```

6.0 TASK 3 – CI/CD PIPELINE IN GITHUB ACTIONS

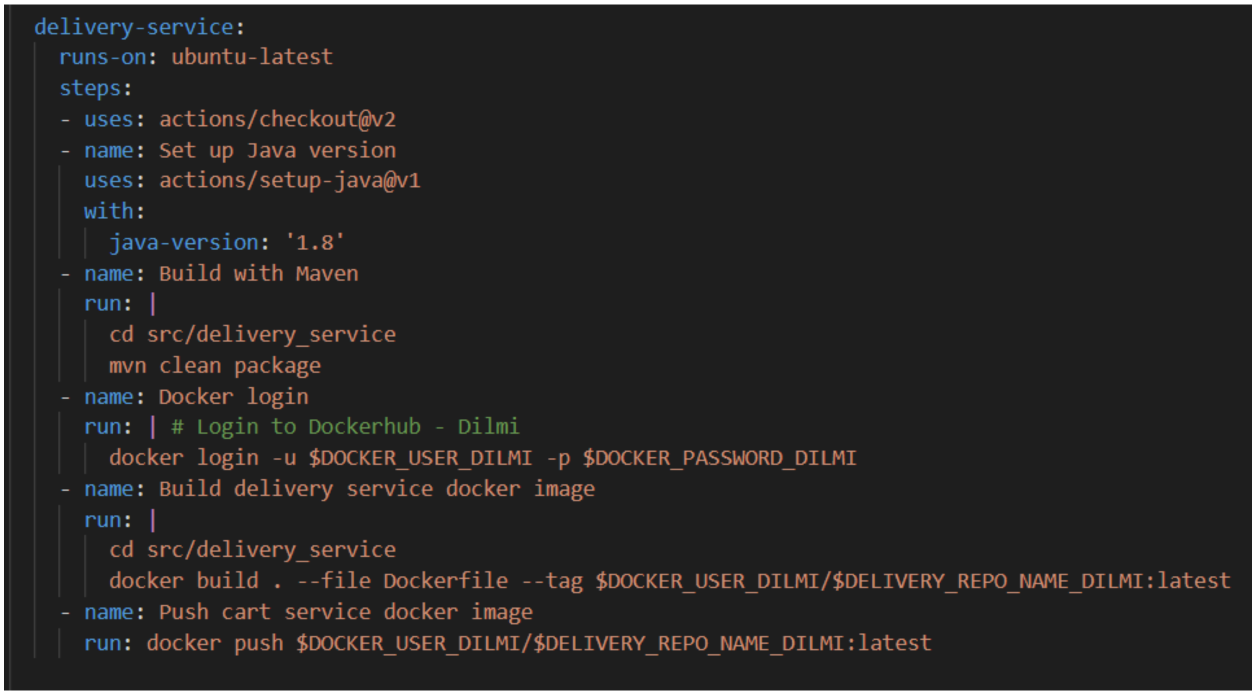
This project uses a CI/ CD pipeline to automatically build the container images and push them to the relevant DockerHub account. After the building process and pushing process is completed for all the services, the deployment pipeline will deploy the new changes to the k8s cluster. GitHub secretes are used to store DockerHub credentials and k8s cluster credentials. Therefore, the credentials are not visible the public.

6.1 Deployment YAML Configuration of Payment Service



```yaml
payment-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/payment_service
      mvn clean package
  - name: Docker login
    run: | # Login to Dockerhub - Dilmi
      docker login -u $DOCKER_USER_DILMI -p $DOCKER_PASSWORD_DILMI
  - name: Build payment service docker image
    run: |
      cd src/payment_service
      docker build . --file Dockerfile --tag $DOCKER_USER_DILMI/$PAYMENT_REPO_NAME_DILMI:latest
  - name: Push cart service docker image
    run: docker push $DOCKER_USER_DILMI/$PAYMENT_REPO_NAME_DILMI:latest
```

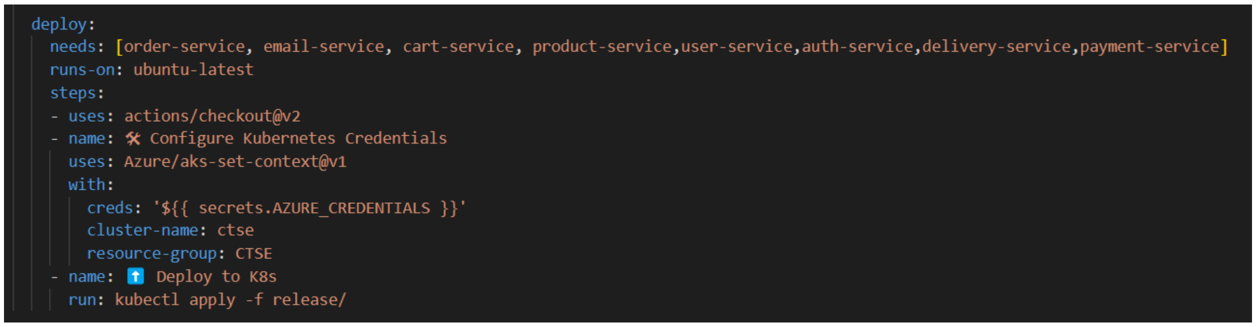
6.2 Deployment YAML Configuration of Delivery Service



```yaml
delivery-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/delivery_service
      mvn clean package
  - name: Docker login
    run: | # Login to Dockerhub - Dilmi
      docker login -u $DOCKER_USER_DILMI -p $DOCKER_PASSWORD_DILMI
  - name: Build delivery service docker image
    run: |
      cd src/delivery_service
      docker build . --file Dockerfile --tag $DOCKER_USER_DILMI/$DELIVERY_REPO_NAME_DILMI:latest
  - name: Push cart service docker image
    run: docker push $DOCKER_USER_DILMI/$DELIVERY_REPO_NAME_DILMI:latest
```

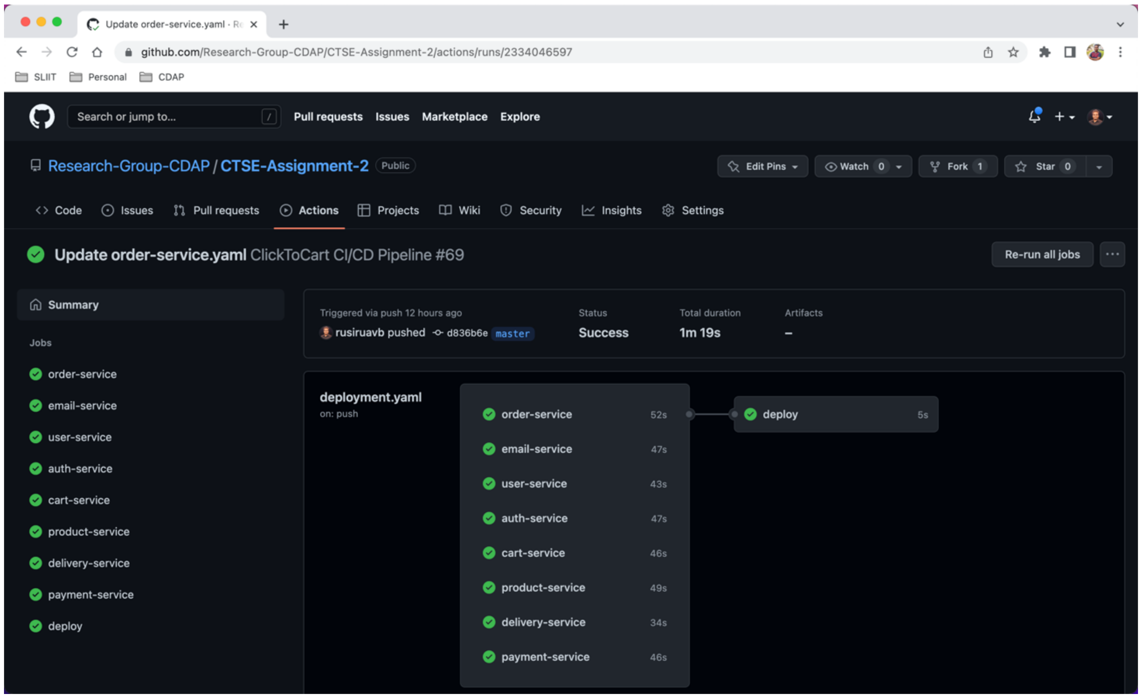
6.3 Deployment to k8s Cluster

After successfully build and push the Docker images, the following deployment pipeline will start executing and eventually deploy all the microservices to the k8s cluster. The deployment pipeline wait until all the images are build and pushed.



```yaml
deploy:
  needs: [order-service, email-service, cart-service, product-service,user-service,auth-service,delivery-service,payment-service]
  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: –

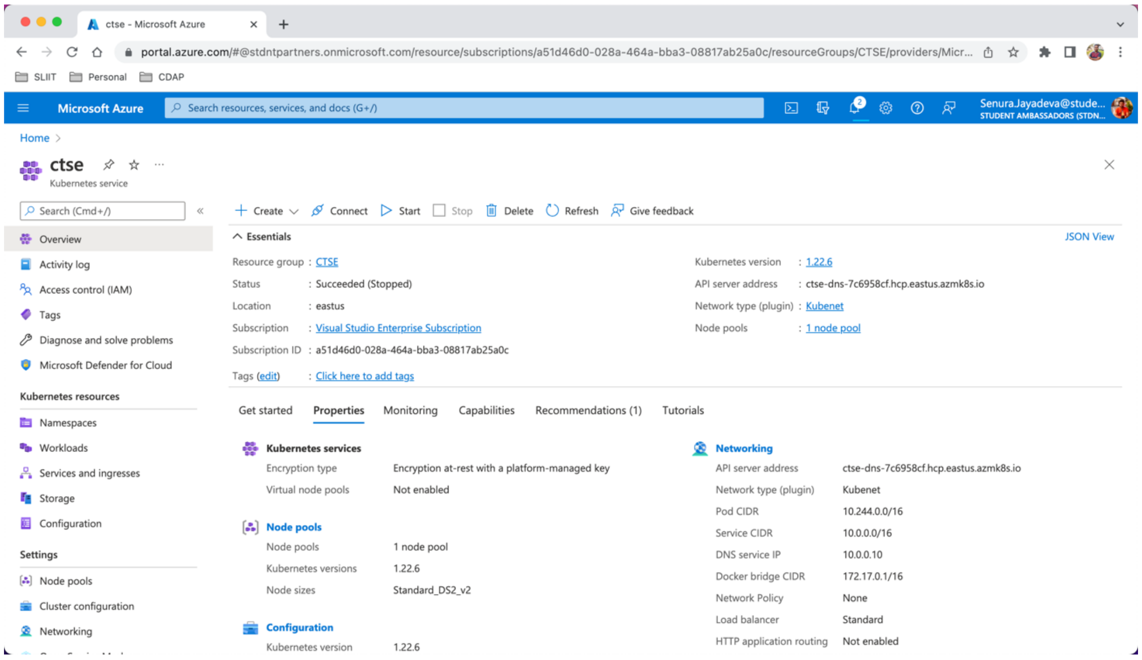
Jobs: order-service, email-service, user-service, auth-service, cart-service, product-service, delivery-service, payment-service, deploy

deployment.yaml
on: push

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



ctse
Kubernetes service

Essentials

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

Properties

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

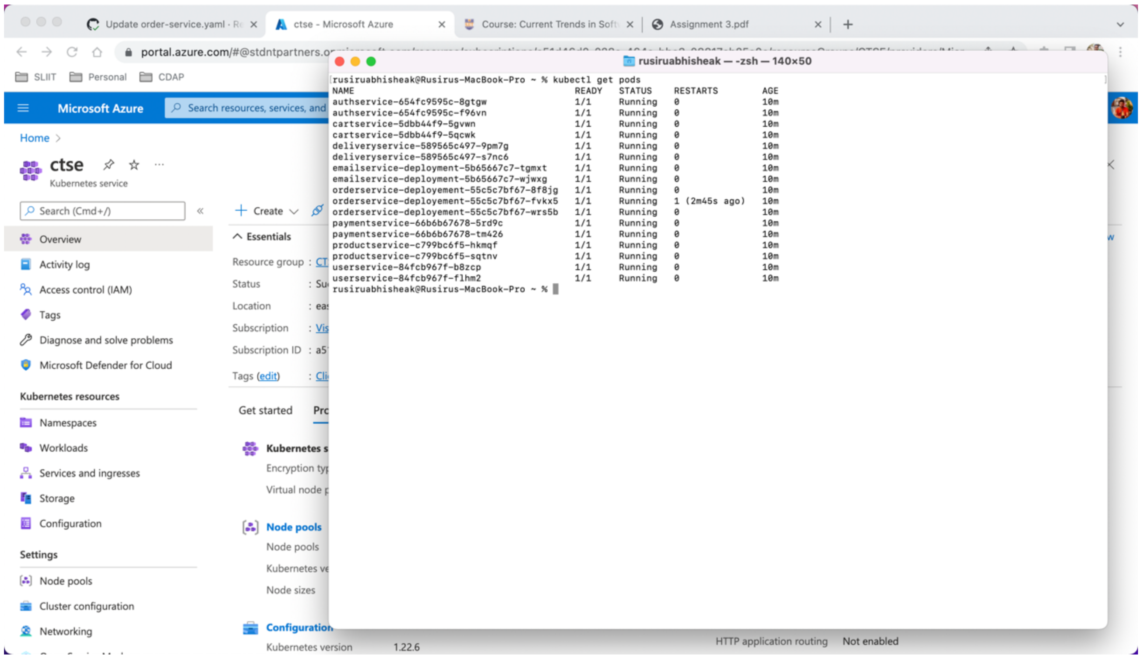
Configuration

Kubernetes version	1.22.6

Networking

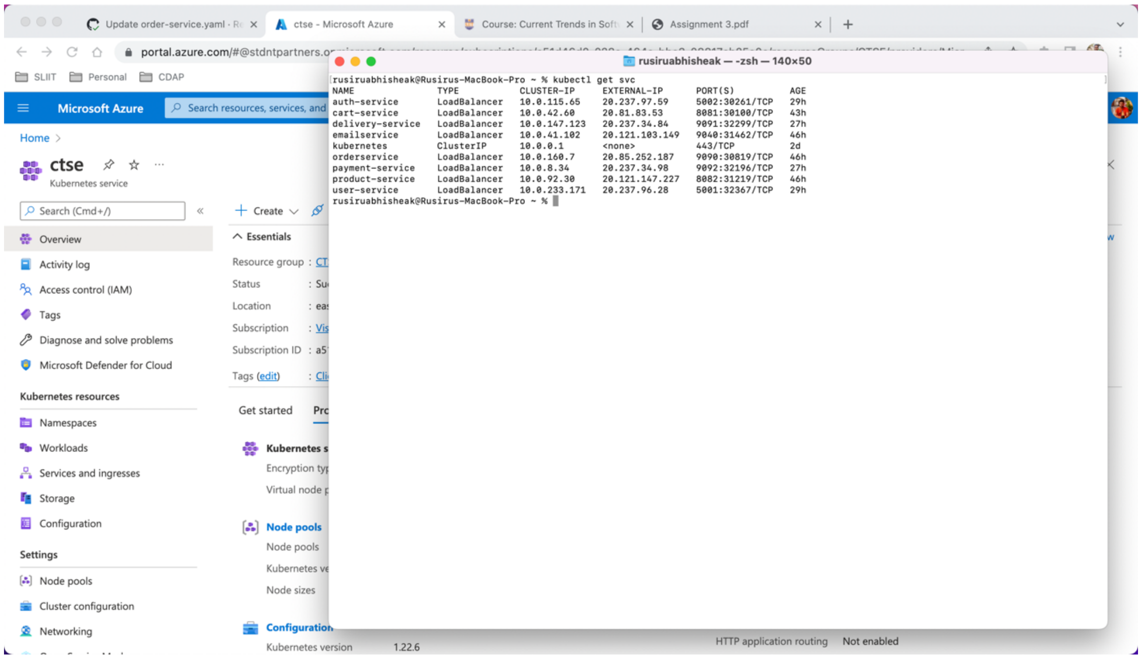
API server address	ctse-dns-7c6958cf.hcp.eastus.azmk8s.io
Network type (plugin)	Kubenet
Pod CIDR	10.244.0.0/16
Service CIDR	10.0.0.0/16
DNS service IP	10.0.0.10
Docker bridge CIDR	172.17.0.1/16
Network Policy	None
Load balancer	Standard
HTTP application routing	Not enabled

7.2 Service Pods Running on k8s Cluster



```
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@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 ~ %
```