CURRENT TRENDS IN SOFTWARE ENGINEERING

SE4010

Microservice Assignment

Bandara G.B.M.A.G.R.A.V.

IT19104218

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

TABLE OF FIGURES

PROJECT OVERVIEW

The system is an online shopping system that users can buy and sell items. Previously, the system was implemented using monolithic architecture. The team faced some difficulties with monolithic architecture in the development time. The one of the drawbacks of using 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).

GitHub Code Repository Link

https://github.com/Research-Group-CDAP/CTSE-Assignment-2

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 each 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

- Order service
- Email service

3.0 INDIVIDUAL SERVICE OVERVIEW

3.1 Order Service

Order service is responsible to create orders after the customer purchase the cart and get the order history for customer. Go programming language and Fiber framework are used to develop the order service. The workflow of the order service is described below.

The payment service make request with user authentication token to the order service once the payment process is successfully completed. When the request comes to the order service, it will make another request to the auth service to authenticate the user. If the authentication is failed, order will not place. Once the authentication success, order will create and store in the MongoDB database. Once the order creation is done, order service makes a request to the email service to send an email to the relevant customer with the order summary. All the order, auth and email services are used inter-service communication to share information between each service.

3.2 Email Service

Email service is used to send emails to the customer about order details. Go programming language is used to develop the email service. Once the order has been placed, order service send a request with order details and user information. Then email service will dispatch an email to the customer with order details.

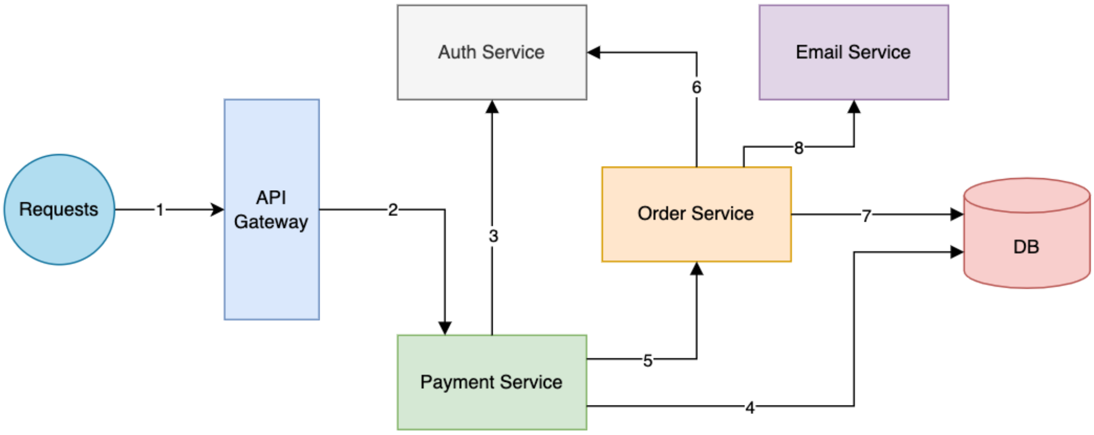


Figure 1: Inter-service communication among services

4.0 TASK 1 – DOCKERIZE APPLICATIONS

Both order service and email service applications have `.env` file that contains MongoDB connection string and cluster IP address of auth, user and email services. When Docker daemon building the Docker image, daemon does not copy the `.env` into the container image. Therefore, when run the docker container, it will throw an error saying database connection string and cluster IP address of auth, user and email services are undefined, because those credentials are not available in the container. There are two options to solve this issue.

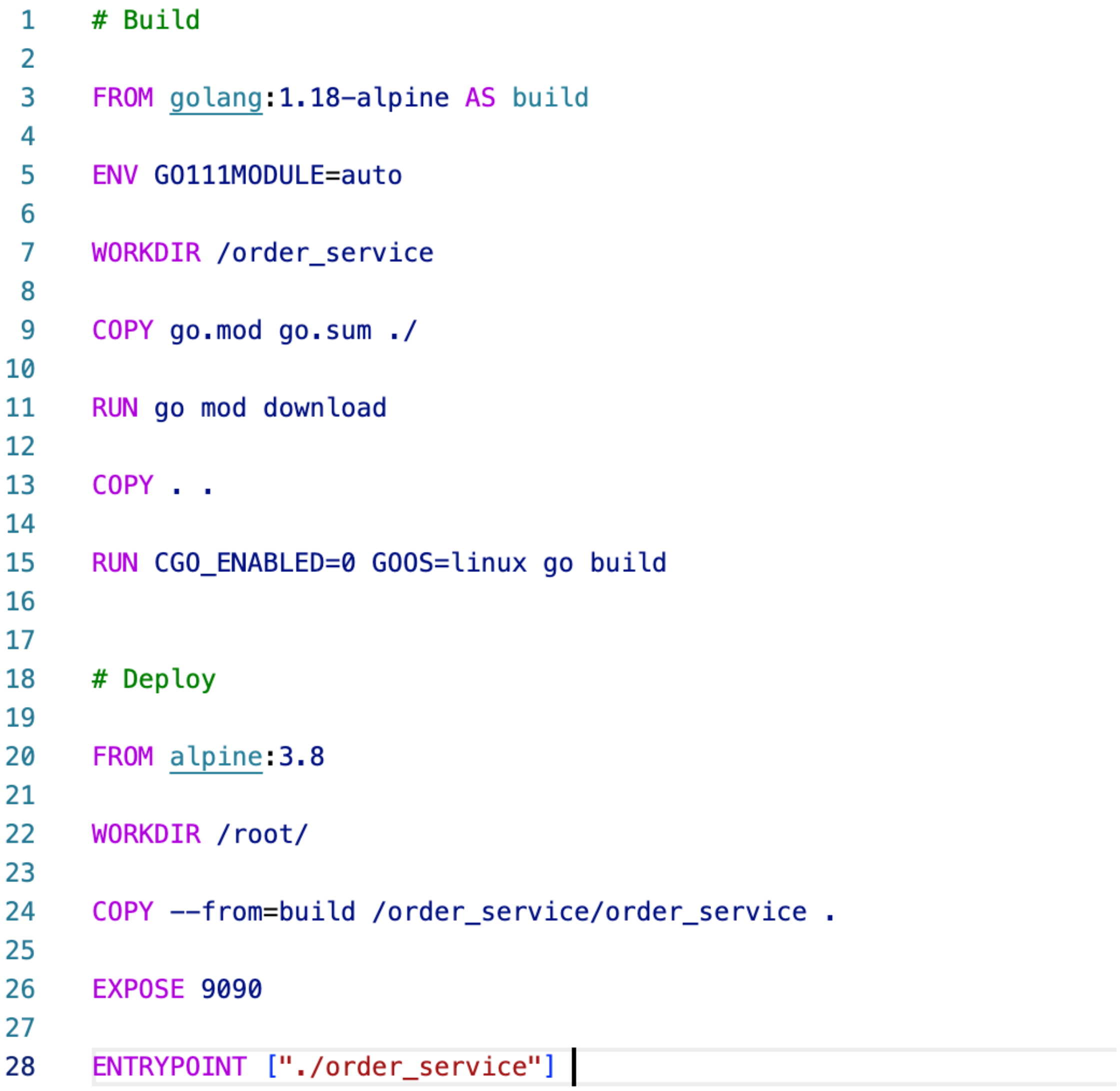
One is defining all the credentials inside Dockerfile. `ENV` syntax is used to define environmental variables inside the Dockerfile. But it is not best practice to display those credentials into public or add them as plain text.

The second option is to create a Kubernetes (k8s) secrete configuration file to store all the credentials as base64 encoded strings. Then connect the service deployment with relevant secrete configuration using the labels. When the microservice added to the k8s cluster using the deployment, the secrete configuration will automatically link with the relevant service using the label name. Therefore, inside the k8s cluster service container image can access the credentials defined in the secrete configuration just like accessing them through `.env` file. This project uses the second option because it is a best practice in k8s deployments.

4.1 Containerize Order Service

4.1.1 Dockerfile of Order Service

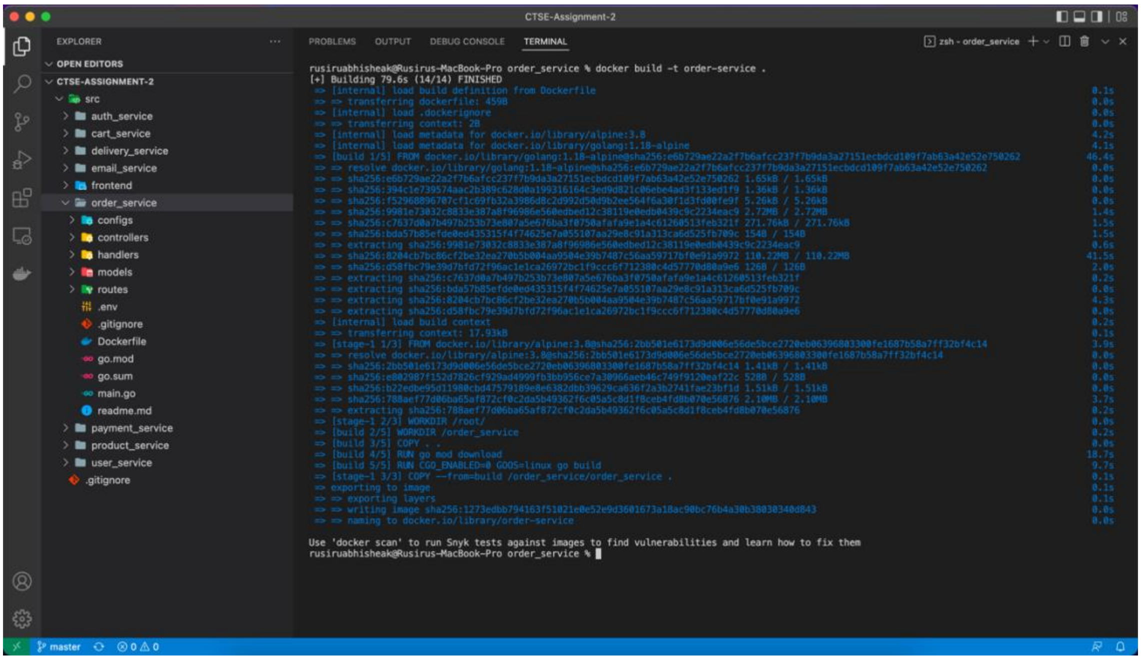
Used multistage Docker building mechanism to optimize the Docker building process and reduce the size of the image.



```
# Build

FROM golang:1.18-alpine AS build

ENV GO111MODULE=auto

WORKDIR /order_service

COPY go.mod go.sum ./

RUN go mod download

COPY . .

RUN CGO_ENABLED=0 GOOS=linux go build


# Deploy

FROM alpine:3.8

WORKDIR /root/

COPY --from=build /order_service/order_service .

EXPOSE 9090

ENTRYPOINT ["./order_service"]
```

Figure 2: Dockerfile of order service

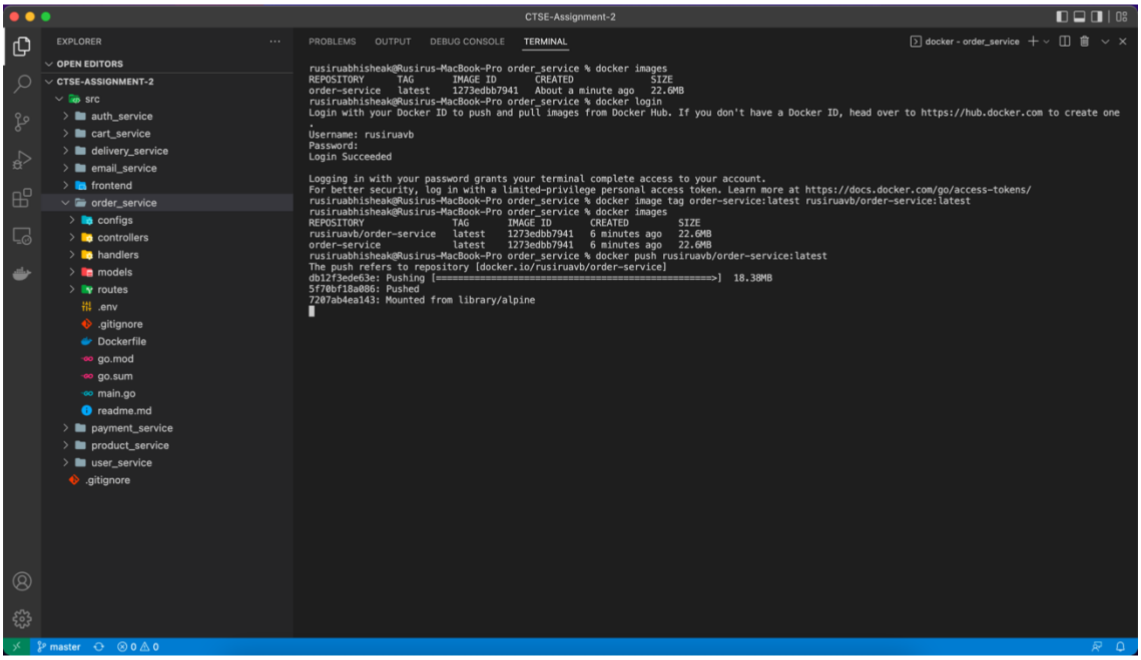
4.1.2 Order Service Container Image Building



```
rusiruabhisheak@Rusirus-MacBook-Pro order_service % docker build -t order-service .
[+] Building 79.6s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                                                        0.1s
 => => transferring dockerfile: 459B                                                                        0.0s
 => [internal] load .dockerignore                                                                           0.0s
 => => transferring context: 2B                                                                             0.0s
 => [internal] load metadata for docker.io/library/alpine:3.8                                               4.2s
 => [internal] load metadata for docker.io/library/golang:1.18-alpine                                       4.1s
 => [build 1/5] FROM docker.io/library/golang:1.18-alpine@sha256:e6b729ae22a2f7b6afcc237f7b9da3a27151ecbdcd109f7ab63a42e52e750262  46.4s
 => => resolve docker.io/library/golang:1.18-alpine@sha256:e6b729ae22a2f7b6afcc237f7b9da3a27151ecbdcd109f7ab63a42e52e750262      0.0s
 => => sha256:e6b729ae22a2f7b6afcc237f7b9da3a27151ecbdcd109f7ab63a42e52e750262 1.65kB / 1.65kB              0.0s
 => => sha256:394c1e739574aac2b389c628d0a199316164c3ed9d821c06ebe4ad3f133ed1f9 1.36kB / 1.36kB              0.0s
 => => sha256:f52968896707cf1c69fb32a3986d8c2d992d50d9b2ee564f6a30f1d3fd00fe9f 5.26kB / 5.26kB              0.0s
 => => sha256:9981e73032c8833e387a8f96986e560edbed12c38119e0edb0439c9c2234eac9 2.72MB / 2.72MB              1.4s
 => => sha256:c7637d0a7b497b253b73e807a5e676ba3f0750afafa9e1a4c61260513feb321f 271.76kB / 271.76kB          1.5s
 => => sha256:bda57b85efde0ed435315f4f74625e7a055107aa29e8c91a313ca6d525fb709c 154B / 154B                  1.5s
 => => extracting sha256:9981e73032c8833e387a8f96986e560edbed12c38119e0edb0439c9c2234eac9                   0.6s
 => => sha256:8204cb7bc86cf2be32ea270b5b004aa9504e39b7487c56aa59717bf0e91a9972 110.22MB / 110.22MB         41.5s
 => => sha256:d58fbc79e39d7bfd72f96ac1e1ca26972bc1f9ccc6f712380c4d57770d80a9e6 126B / 126B                  2.0s
 => => extracting sha256:c7637d0a7b497b253b73e807a5e676ba3f0750afafa9e1a4c61260513feb321f                   0.2s
 => => extracting sha256:bda57b85efde0ed435315f4f74625e7a055107aa29e8c91a313ca6d525fb709c                   0.0s
 => => extracting sha256:8204cb7bc86cf2be32ea270b5b004aa9504e39b7487c56aa59717bf0e91a9972                   4.3s
 => => extracting sha256:d58fbc79e39d7bfd72f96ac1e1ca26972bc1f9ccc6f712380c4d57770d80a9e6                   0.0s
 => [internal] load build context                                                                           0.2s
 => => transferring context: 17.93kB                                                                        0.1s
 => [stage-1 1/3] FROM docker.io/library/alpine:3.8@sha256:2bb501e6173d9d006e56de5bce2720eb06396803300fe1687b58a7ff32bf4c14  3.9s
 => => resolve docker.io/library/alpine:3.8@sha256:2bb501e6173d9d006e56de5bce2720eb06396803300fe1687b58a7ff32bf4c14       0.0s
 => => sha256:2bb501e6173d9d006e56de5bce2720eb06396803300fe1687b58a7ff32bf4c14 1.41kB / 1.41kB              0.0s
 => => sha256:e802987f152d7826cf929ad4999fb3bb956ce7a30966aeb46c749f9120eaf22c 528B / 528B                  0.0s
 => => sha256:b22edbe95d11980cbd47579189e8e6382dbb39629ca636f2a3b2741fae23bf1d 1.51kB / 1.51kB              0.0s
 => => sha256:788aef77d06ba65af872cf0c2da5b49362f6c05a5c8d1f8ceb4fd8b070e56876 2.10MB / 2.10MB              3.7s
 => => extracting sha256:788aef77d06ba65af872cf0c2da5b49362f6c05a5c8d1f8ceb4fd8b070e56876                   0.2s
 => [stage-1 2/3] WORKDIR /root/                                                                            0.0s
 => [build 2/5] WORKDIR /order_service                                                                      0.2s
 => [build 3/5] COPY . .                                                                                    0.0s
 => [build 4/5] RUN go mod download                                                                        18.7s
 => [build 5/5] RUN CGO_ENABLED=0 GOOS=linux go build                                                       9.7s
 => [stage-1 3/3] COPY --from=build /order_service/order_service .                                          0.1s
 => exporting to image                                                                                      0.1s
 => => exporting layers                                                                                     0.1s
 => => writing image sha256:1273edbb794163f51021e0e52e9d3601673a18ac90bc76b4a30b38030340d843                0.0s
 => => naming to docker.io/library/order-service                                                            0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
rusiruabhisheak@Rusirus-MacBook-Pro order_service % ▋
```

Figure 3: Docker image building process

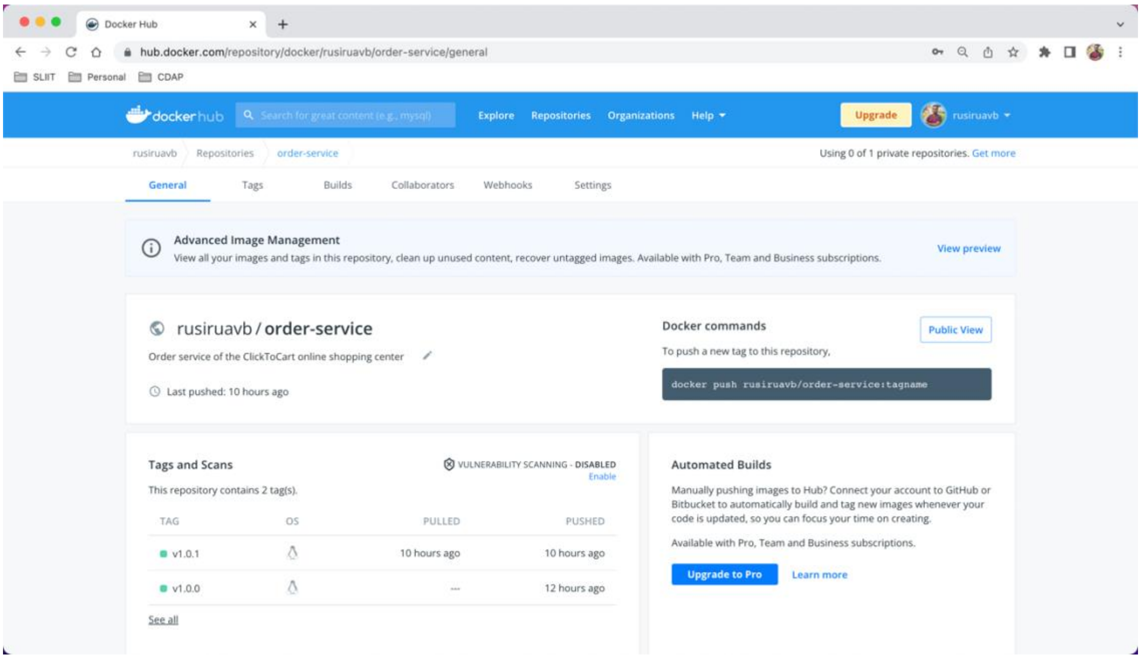
4.1.3 Push Order Docker Image to DockerHub



```
rusiruabhisheak@Rusirus-MacBook-Pro order_service % docker images
REPOSITORY      TAG       IMAGE ID       CREATED              SIZE
order-service   latest    1273edbb7941   About a minute ago   22.6MB
rusiruabhisheak@Rusirus-MacBook-Pro order_service % docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one
.
Username: rusiruavb
Password:
Login Succeeded

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/
rusiruabhisheak@Rusirus-MacBook-Pro order_service % docker image tag order-service:latest rusiruavb/order-service:latest
rusiruabhisheak@Rusirus-MacBook-Pro order_service % docker images
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
rusiruavb/order-service   latest    1273edbb7941   6 minutes ago   22.6MB
order-service             latest    1273edbb7941   6 minutes ago   22.6MB
rusiruabhisheak@Rusirus-MacBook-Pro order_service % docker push rusiruavb/order-service:latest
The push refers to repository [docker.io/rusiruavb/order-service]
db12f3ede63e: Pushing [==================================================>]  18.38MB
5f70bf18a086: Pushed
7207ab4ea143: Mounted from library/alpine
▋
```

Figure 4: Order image push to DockerHub

4.1.4 Order Service DockerHub Overview



Using 0 of 1 private repositories. Get more

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.

rusiruavb / order-service

Order service of the ClickToCart online shopping center

Last pushed: 10 hours ago

Docker commands

To push a new tag to this repository,

```
docker push rusiruavb/order-service:tagname
```

Tags and Scans

VULNERABILITY SCANNING - DISABLED

This repository contains 2 tag(s).

TAG	OS	PULLED	PUSHED
v1.0.1	🐧	10 hours ago	10 hours ago
v1.0.0	🐧	---	12 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.

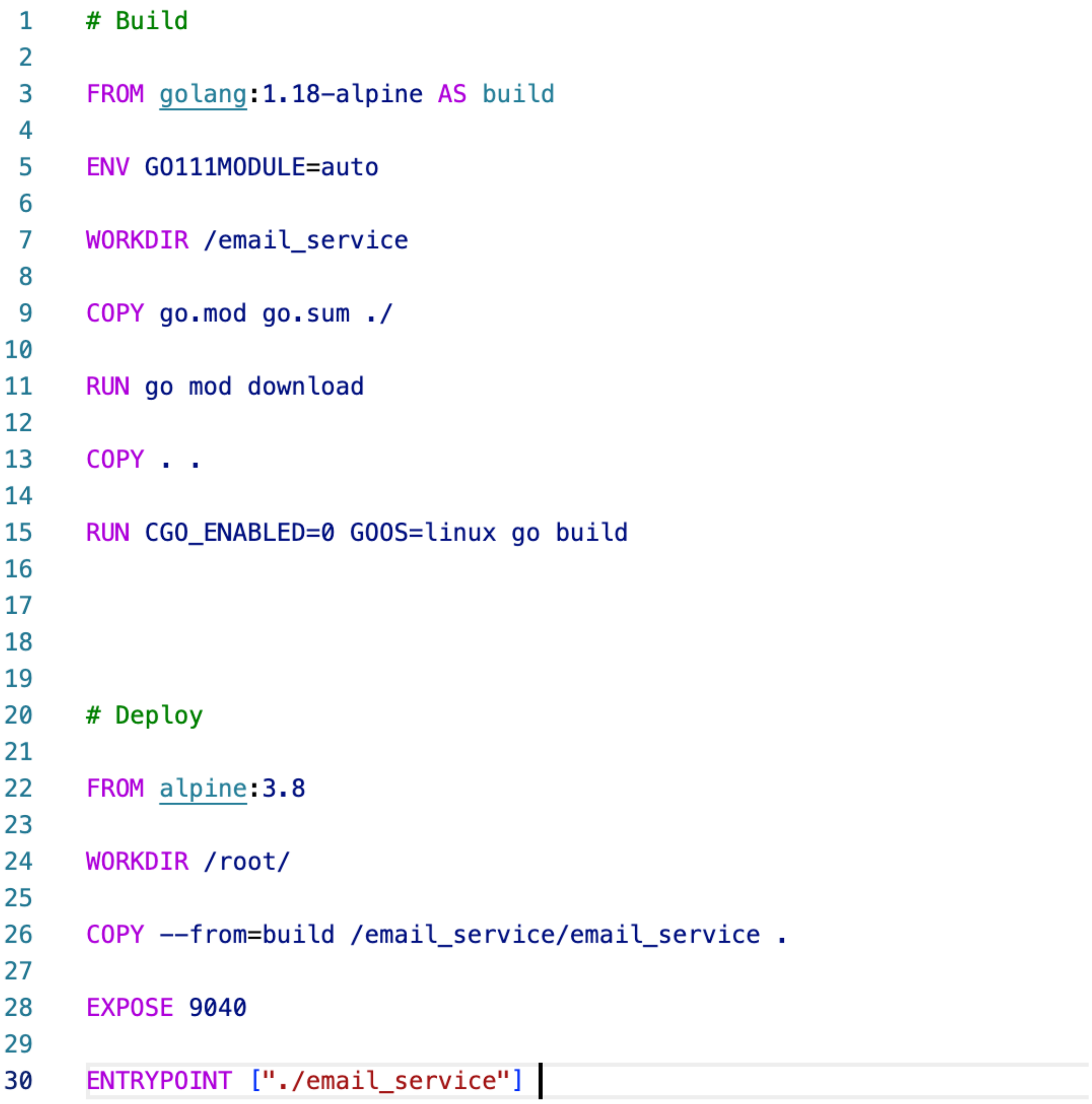
Figure 5: Order Service DockerHub Overview

Order service DockerHub link

https://hub.docker.com/repository/docker/rusiruavb/order-service

4.2 Containerize Email Service

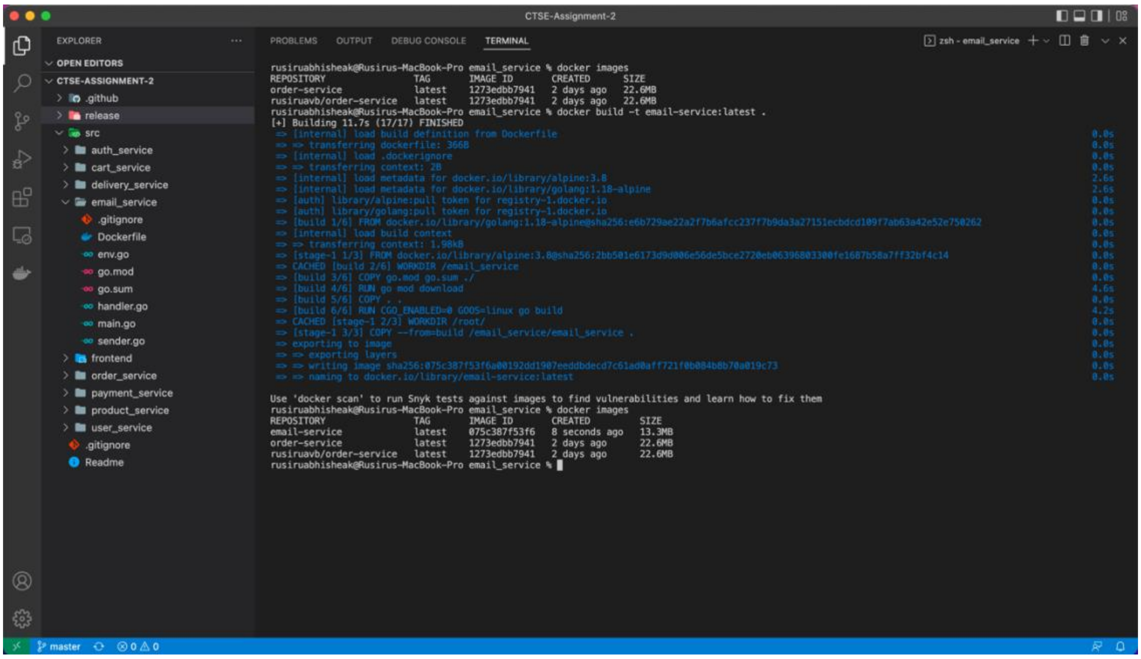
4.2.1 Dockerfile of Email Service



```dockerfile
# Build

FROM golang:1.18-alpine AS build

ENV GO111MODULE=auto

WORKDIR /email_service

COPY go.mod go.sum ./

RUN go mod download

COPY . .

RUN CGO_ENABLED=0 GOOS=linux go build




# Deploy

FROM alpine:3.8

WORKDIR /root/

COPY --from=build /email_service/email_service .

EXPOSE 9040

ENTRYPOINT ["./email_service"]
```

Figure 6: Dockerfile of Email Service

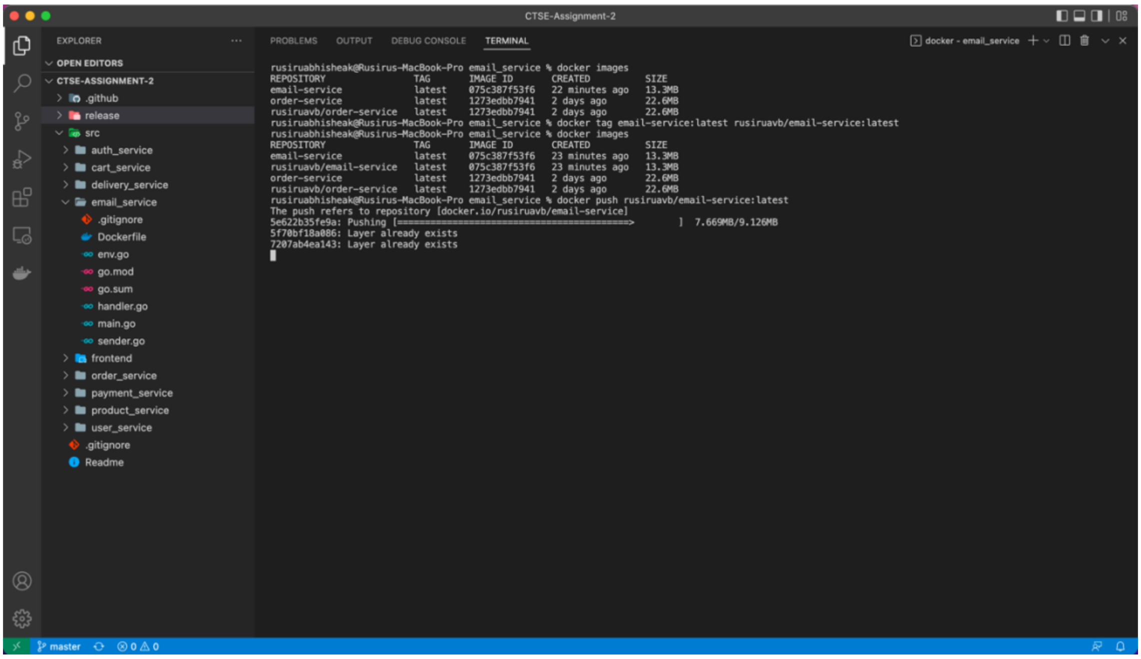
4.2.2 Email Service Container Image Building



```
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker images
REPOSITORY               TAG       IMAGE ID       CREATED       SIZE
order-service            latest    1273edbb7941   2 days ago    22.6MB
rusiruavb/order-service  latest    1273edbb7941   2 days ago    22.6MB
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker build -t email-service:latest .
[+] Building 11.7s (17/17) FINISHED
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 366B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/alpine:3.8                                                 2.6s
 => [internal] load metadata for docker.io/library/golang:1.18-alpine                                         2.6s
 => [auth] library/alpine:pull token for registry-1.docker.io                                                 0.0s
 => [auth] library/golang:pull token for registry-1.docker.io                                                 0.0s
 => [build 1/6] FROM docker.io/library/golang:1.18-alpine@sha256:e6b729ae22a2f7b6afcc237f7b9da3a27151ecbdcd109f7ab63a42e52e750262  0.0s
 => [internal] load build context                                                                             0.0s
 => => transferring context: 1.98kB                                                                           0.0s
 => [stage-1 1/3] FROM docker.io/library/alpine:3.8@sha256:2bb501e6173d9d006e56de5bce2720eb06396803300fe1687b58a7ff32bf4c14  0.0s
 => CACHED [build 2/6] WORKDIR /email_service                                                                 0.0s
 => [build 3/6] COPY go.mod go.sum ./                                                                         0.0s
 => [build 4/6] RUN go mod download                                                                           4.6s
 => [build 5/6] COPY . .                                                                                      0.0s
 => [build 6/6] RUN CGO_ENABLED=0 GOOS=linux go build                                                         4.2s
 => CACHED [stage-1 2/3] WORKDIR /root/                                                                       0.0s
 => [stage-1 3/3] COPY --from=build /email_service/email_service .                                            0.0s
 => exporting to image                                                                                        0.0s
 => => exporting layers                                                                                       0.0s
 => => writing image sha256:075c387f53f6a00192dd1907eeddbdecd7c61ad0aff721f0b084b8b70a019c73                  0.0s
 => => naming to docker.io/library/email-service:latest                                                       0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker images
REPOSITORY               TAG       IMAGE ID       CREATED          SIZE
email-service            latest    075c387f53f6   8 seconds ago    13.3MB
order-service            latest    1273edbb7941   2 days ago       22.6MB
rusiruavb/order-service  latest    1273edbb7941   2 days ago       22.6MB
rusiruabhisheak@Rusirus-MacBook-Pro email_service % █
```

Figure 7: Email service Docker image build event

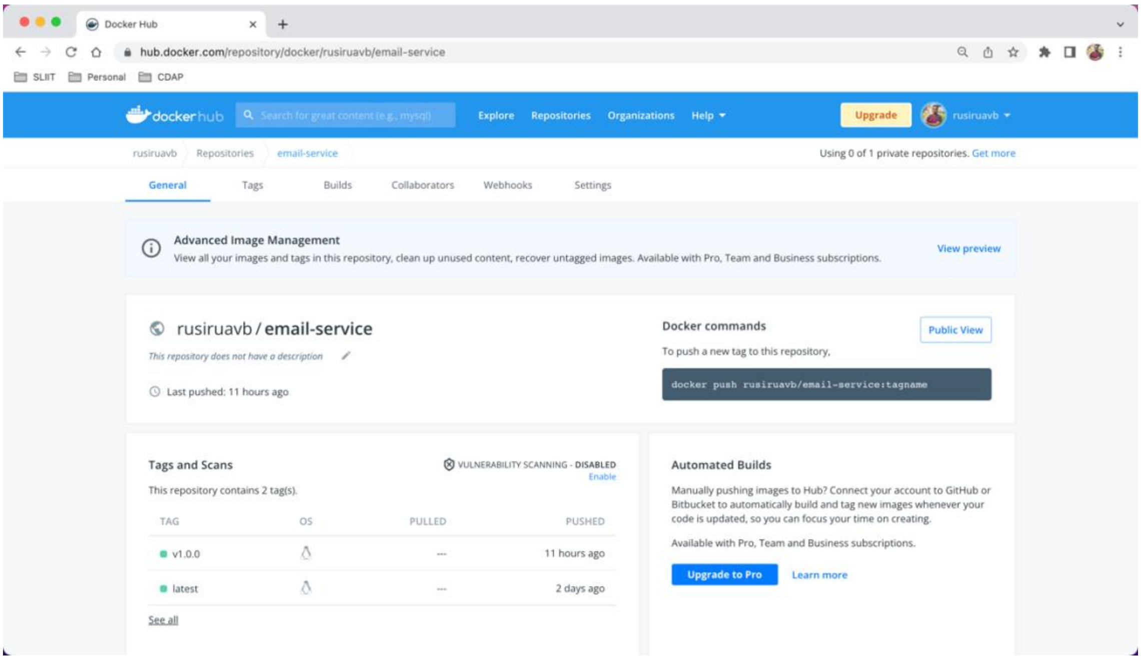
4.2.3 Push Email Docker Image to DockerHub



```
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker images
REPOSITORY               TAG       IMAGE ID       CREATED          SIZE
email-service            latest    075c387f53f6   22 minutes ago   13.3MB
order-service            latest    1273edbb7941   2 days ago       22.6MB
rusiruavb/order-service  latest    1273edbb7941   2 days ago       22.6MB
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker tag email-service:latest rusiruavb/email-service:latest
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker images
REPOSITORY               TAG       IMAGE ID       CREATED          SIZE
email-service            latest    075c387f53f6   23 minutes ago   13.3MB
rusiruavb/email-service  latest    075c387f53f6   23 minutes ago   13.3MB
order-service            latest    1273edbb7941   2 days ago       22.6MB
rusiruavb/order-service  latest    1273edbb7941   2 days ago       22.6MB
rusiruabhisheak@Rusirus-MacBook-Pro email_service % docker push rusiruavb/email-service:latest
The push refers to repository [docker.io/rusiruavb/email-service]
5e622b35fe9a: Pushing [==========================================>        ]  7.669MB/9.126MB
5f70bf18a086: Layer already exists
7207ab4ea143: Layer already exists
█
```

Figure 8: Email service Docker image pushing event

4.2.4 Email Service DockerHub Overview



Using 0 of 1 private repositories. Get more

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.

rusiruavb / email-service

This repository does not have a description

Last pushed: 11 hours ago

Docker commands

To push a new tag to this repository,

```
docker push rusiruavb/email-service:tagname
```

Tags and Scans

VULNERABILITY SCANNING - DISABLED

This repository contains 2 tag(s).

TAG	OS	PULLED	PUSHED
🟩 v1.0.0		---	11 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.

Figure 9: Email Service DockerHub Overview

Email Service DockerHub link

https://hub.docker.com/repository/docker/rusiruavb/email-service

5.0 TASK 2 - DEPLOY SERVICES TO K8S CLUSTER

Azure Kubernetes Service (AKS) used as the Kubernetes engine 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.

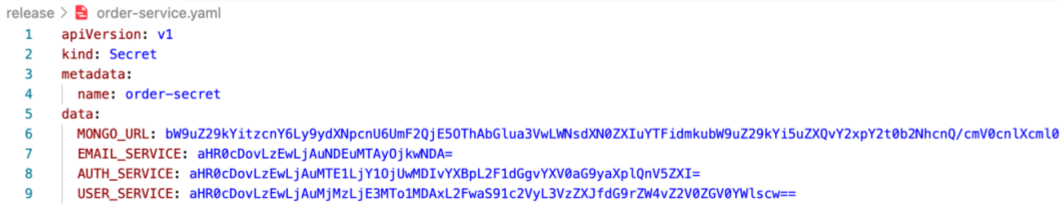
```
kubectl apply -f release/
```

5.1 Order Service k8s Config YAML Files

5.1.1 Order Service k8s Secret YAML File

Configure k8s secrete file for order service to store database URL and store the private IP addresses of email service, auth service, user services. All the secretes are encoded to base64 instead of display them as plain text on the YAML file. Following command is used to encode the plain text to base64 format. The output of the following command is added to the secrete YAML file.

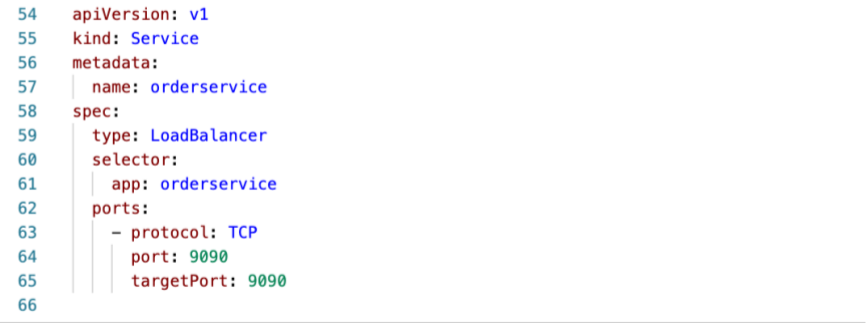
```
echo -n "http://10.30.45.245:9090" | base64 -i -
```



```yaml
apiVersion: v1
kind: Secret
metadata:
  name: order-secret
data:
  MONGO_URL: bW9uZ29kYitzcnY6Ly9ydXNpcnU6UmF2QjE5OThAbGlua3VwLWNsdXN0ZXIuYTFidmkubW9uZ29kYi5uZXQvY2xpY2t0b2NhcnQ/cmV0cnlXcml0
  EMAIL_SERVICE: aHR0cDovLzEwLjAuNDEuMTAyOjkwNDA=
  AUTH_SERVICE: aHR0cDovLzEwLjAuMTE1LjY1OjUwMDIvYXBpL2F1dGgvYXV0aG9yaXplQnV5ZXI=
  USER_SERVICE: aHR0cDovLzEwLjAuMjMzLjE3MTo1MDAxL2FwaS91c2VyL3VzZXJfdG9rZW4vZ2V0ZGV0YWlscw==
```

Figure 10: Order service k8s secret YAML file

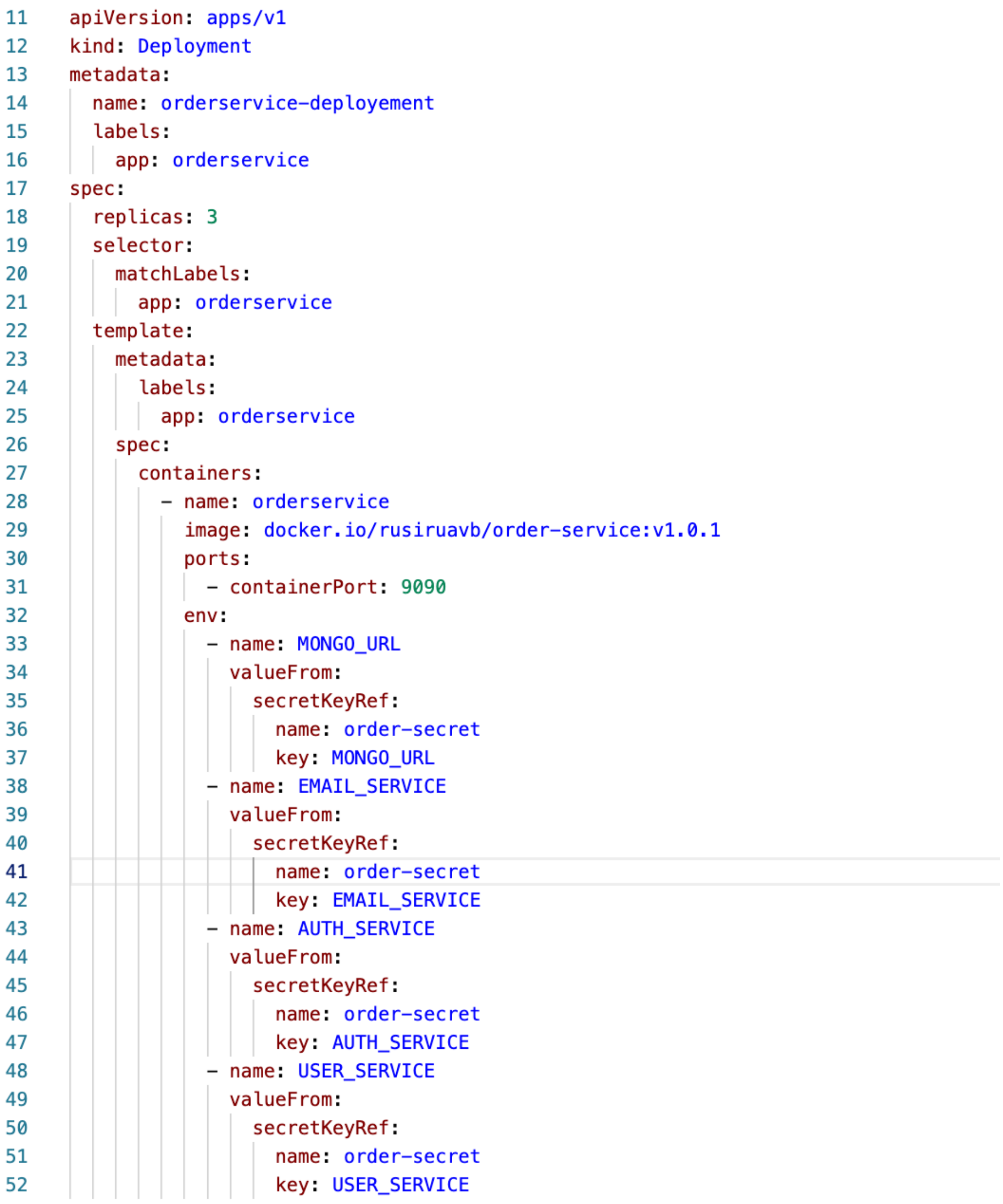
5.1.2 Order Service k8s Service YAML File



```yaml
apiVersion: v1
kind: Service
metadata:
  name: orderservice
spec:
  type: LoadBalancer
  selector:
    app: orderservice
  ports:
    - protocol: TCP
      port: 9090
      targetPort: 9090

```

Figure 11: Order service k8s configuration YAML file

5.1.3 Order Service k8s Deployment YAML File



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orderservice-deployement
  labels:
    app: orderservice
spec:
  replicas: 3
  selector:
    matchLabels:
      app: orderservice
  template:
    metadata:
      labels:
        app: orderservice
    spec:
      containers:
        - name: orderservice
          image: docker.io/rusiruavb/order-service:v1.0.1
          ports:
            - containerPort: 9090
          env:
            - name: MONGO_URL
              valueFrom:
                secretKeyRef:
                  name: order-secret
                  key: MONGO_URL
            - name: EMAIL_SERVICE
              valueFrom:
                secretKeyRef:
                  name: order-secret
                  key: EMAIL_SERVICE
            - name: AUTH_SERVICE
              valueFrom:
                secretKeyRef:
                  name: order-secret
                  key: AUTH_SERVICE
            - name: USER_SERVICE
              valueFrom:
                secretKeyRef:
                  name: order-secret
                  key: USER_SERVICE
```

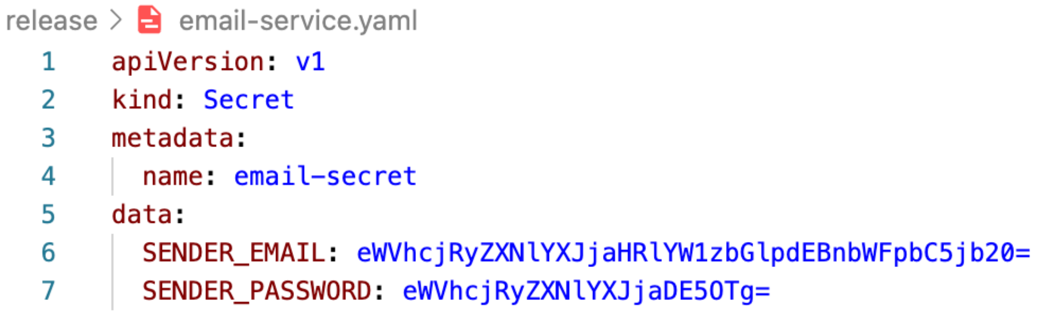
Figure 12: Order service k8s deployment YAML file

5.2 Email Service k8s Config YAML Files

5.2.1 Email Service k8s Secrete YAML File

Implement k8s secrete configuration file to store email provider’s address and password. The email provider’s address and password are sensitive information in email service application. Therefore, encode the address and password to base64 format before adding to the secrete YAML file. Following is the code to encode the plain text to base64 format.

```
echo -n "samplemail@gmail.com" | base64 -i -
```

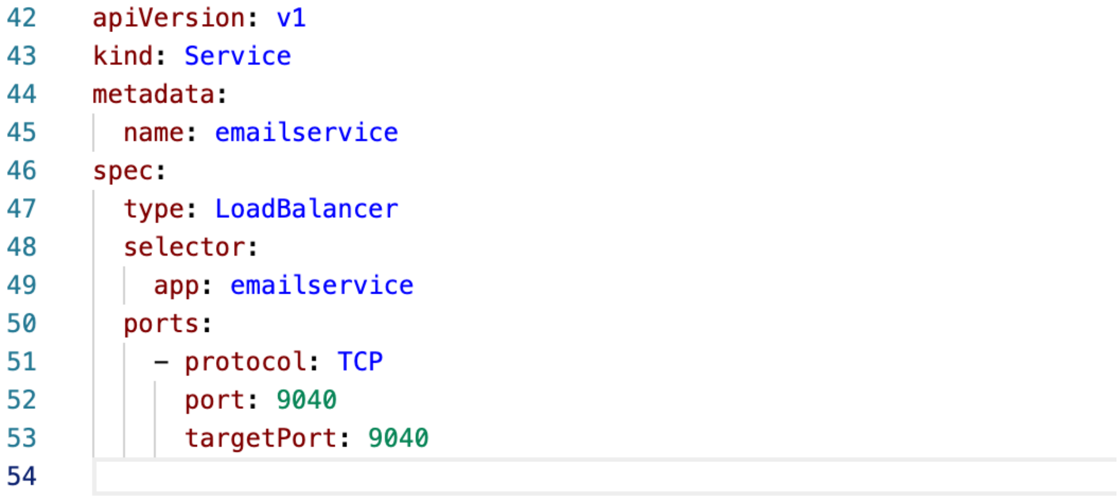


release > email-service.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: email-secret
data:
  SENDER_EMAIL: eWVhcjRyZXNlYXJjaHRlYW1zbGlpdEBnbWFpbC5jb20=
  SENDER_PASSWORD: eWVhcjRyZXNlYXJjaDE5OTg=
```

Figure 13: Email service k8s secret YAML file

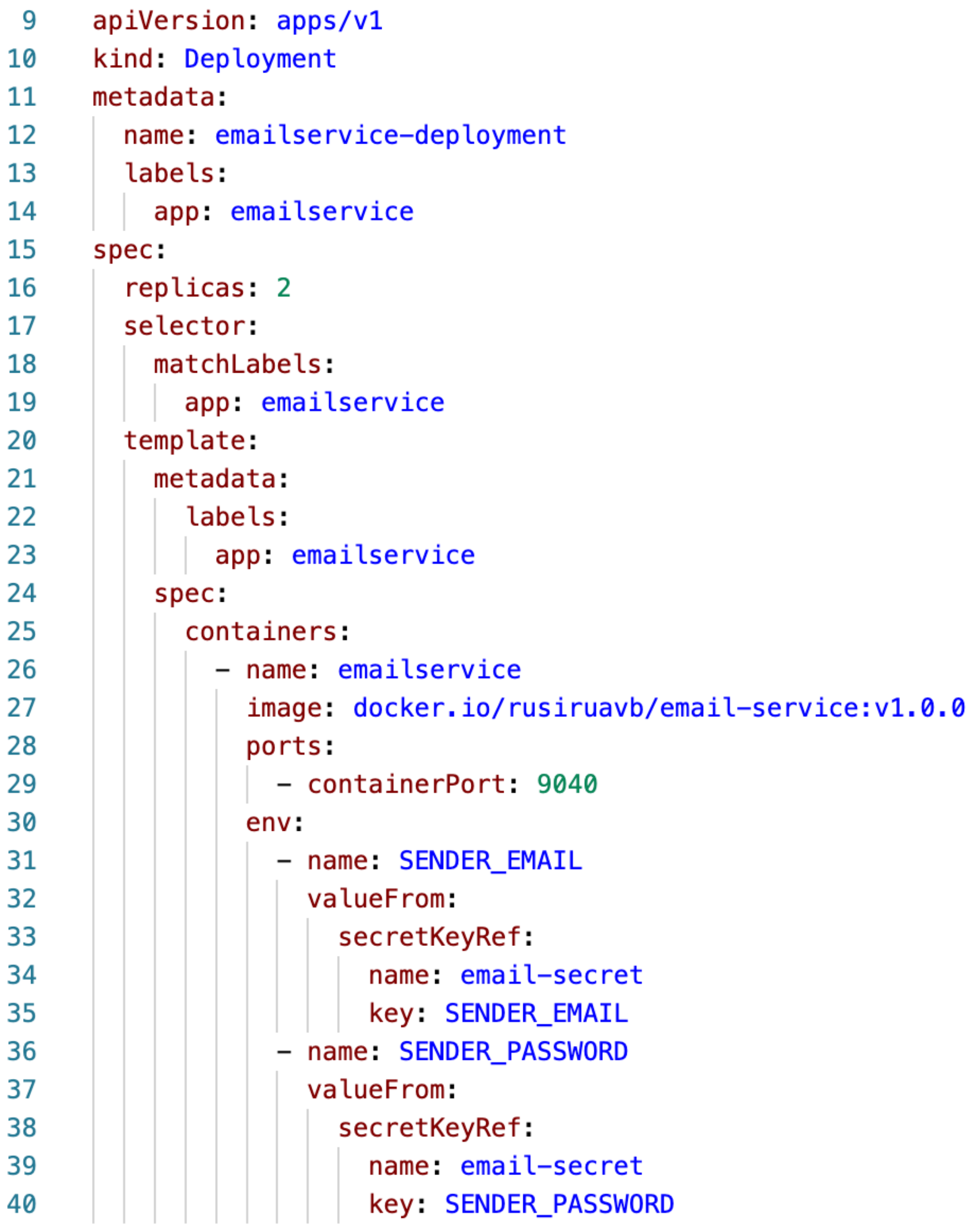
5.2.2 Email Service k8s Service YAML File



```yaml
apiVersion: v1
kind: Service
metadata:
  name: emailservice
spec:
  type: LoadBalancer
  selector:
    app: emailservice
  ports:
    - protocol: TCP
      port: 9040
      targetPort: 9040

```

Figure 14: Email service k8s configuration YAML file

5.2.3 Email Service k8s Deployment YAML File



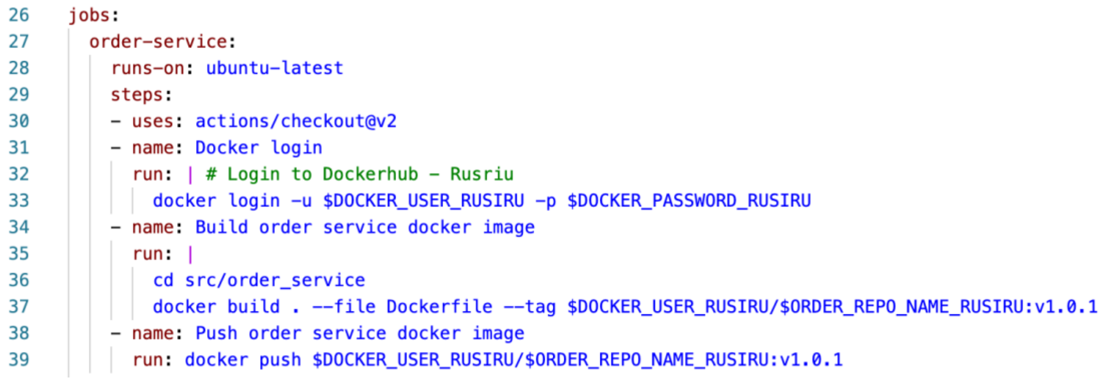
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: emailservice-deployment
  labels:
    app: emailservice
spec:
  replicas: 2
  selector:
    matchLabels:
      app: emailservice
  template:
    metadata:
      labels:
        app: emailservice
    spec:
      containers:
        - name: emailservice
          image: docker.io/rusiruavb/email-service:v1.0.0
          ports:
            - containerPort: 9040
          env:
            - name: SENDER_EMAIL
              valueFrom:
                secretKeyRef:
                  name: email-secret
                  key: SENDER_EMAIL
            - name: SENDER_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: email-secret
                  key: SENDER_PASSWORD
```

Figure 15: Email service k8s deployment YAML file

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.

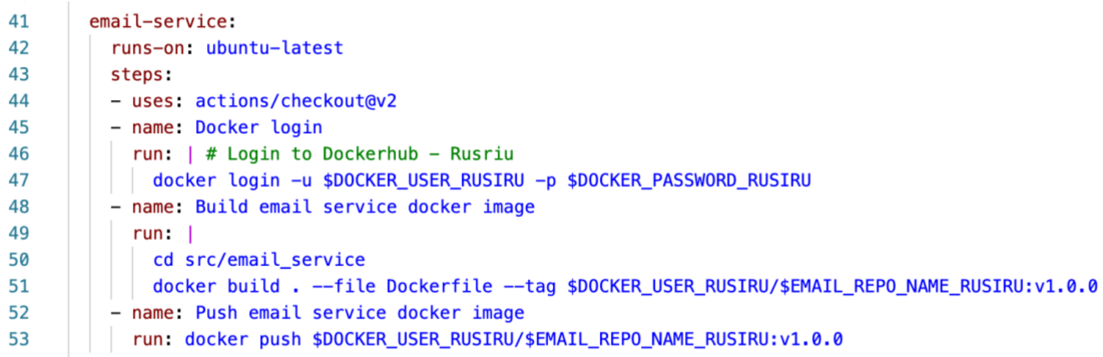
6.1 Deployment YAML Configuration of Order Service



```
jobs:
  order-service:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Docker login
      run: | # Login to Dockerhub - Rusriu
        docker login -u $DOCKER_USER_RUSIRU -p $DOCKER_PASSWORD_RUSIRU
    - name: Build order service docker image
      run: |
        cd src/order_service
        docker build . --file Dockerfile --tag $DOCKER_USER_RUSIRU/$ORDER_REPO_NAME_RUSIRU:v1.0.1
    - name: Push order service docker image
      run: docker push $DOCKER_USER_RUSIRU/$ORDER_REPO_NAME_RUSIRU:v1.0.1
```

Figure 16: Order service GitHub CI/CD pipeline

6.2 Deployment YAML Configuration of Email Service

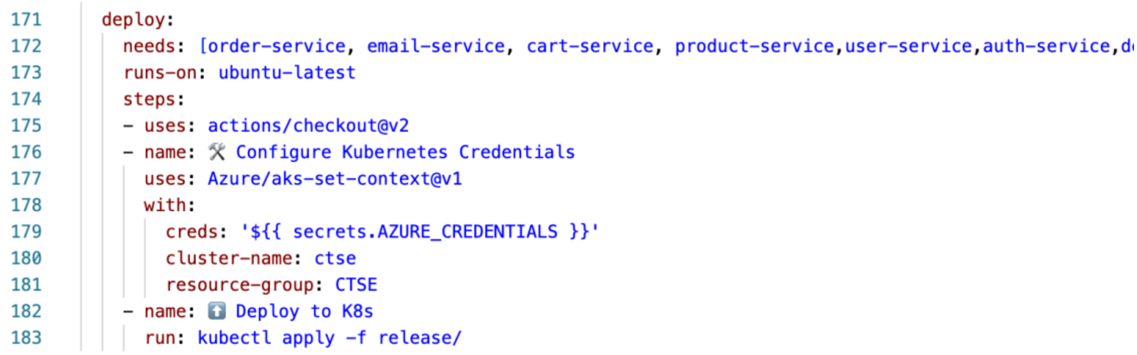


```
  email-service:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Docker login
      run: | # Login to Dockerhub - Rusriu
        docker login -u $DOCKER_USER_RUSIRU -p $DOCKER_PASSWORD_RUSIRU
    - name: Build email service docker image
      run: |
        cd src/email_service
        docker build . --file Dockerfile --tag $DOCKER_USER_RUSIRU/$EMAIL_REPO_NAME_RUSIRU:v1.0.0
    - name: Push email service docker image
      run: docker push $DOCKER_USER_RUSIRU/$EMAIL_REPO_NAME_RUSIRU:v1.0.0
```

Figure 17: Email service GitHub CI/CD pipeline

6.3 Deployment to k8s Cluster

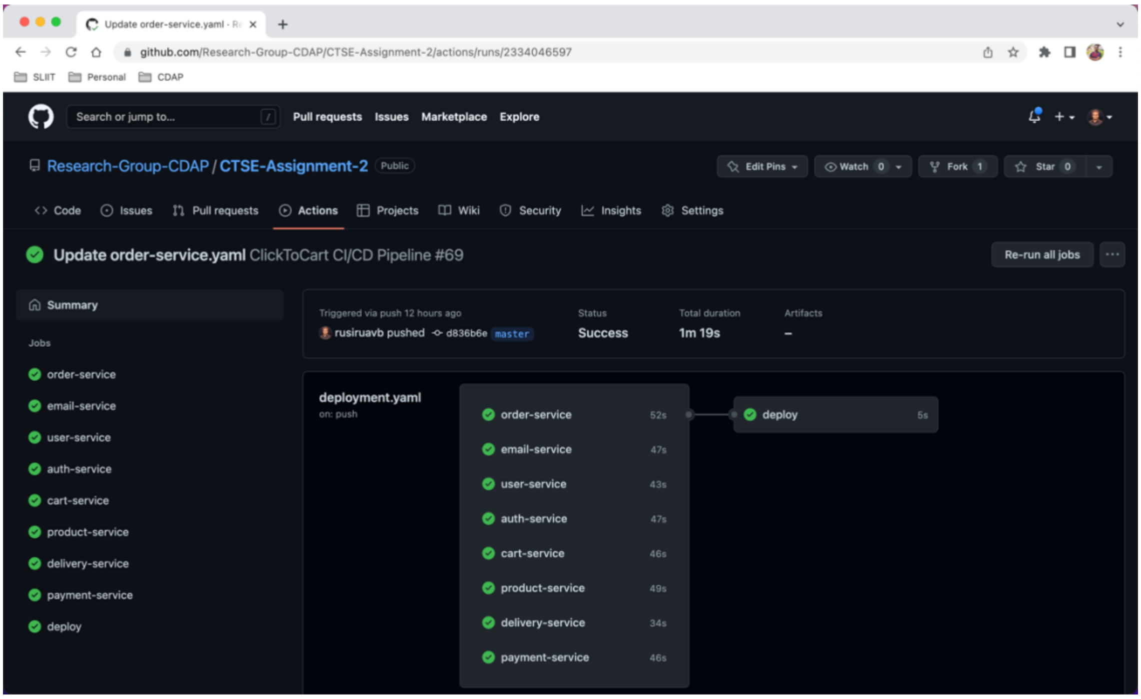
After successfully build and push the Docker images, the following deployment pipeline will start executing and eventually deploy all the microservices to the k8s cluster. The deployment pipeline wait until all the images are build and pushed.



```yaml
deploy:
  needs: [order-service, email-service, cart-service, product-service,user-service,auth-service,d
  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/
```

Figure 18: k8s deployment GitHub CI/CD pipeline

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

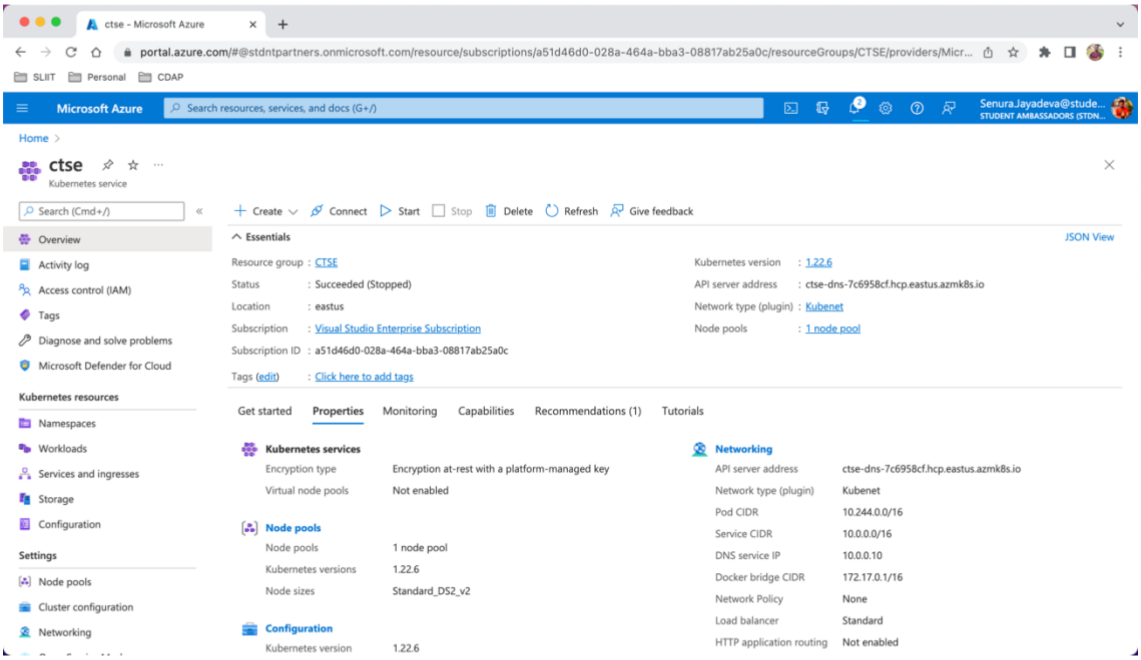
deployment.yaml
on: push

```
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
```

Figure 19: Deployment running on GitHub Actions

7.0 K8S CLUSTER INFORMATION

7.1 K8s Cluster Overview on Azure Portal



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

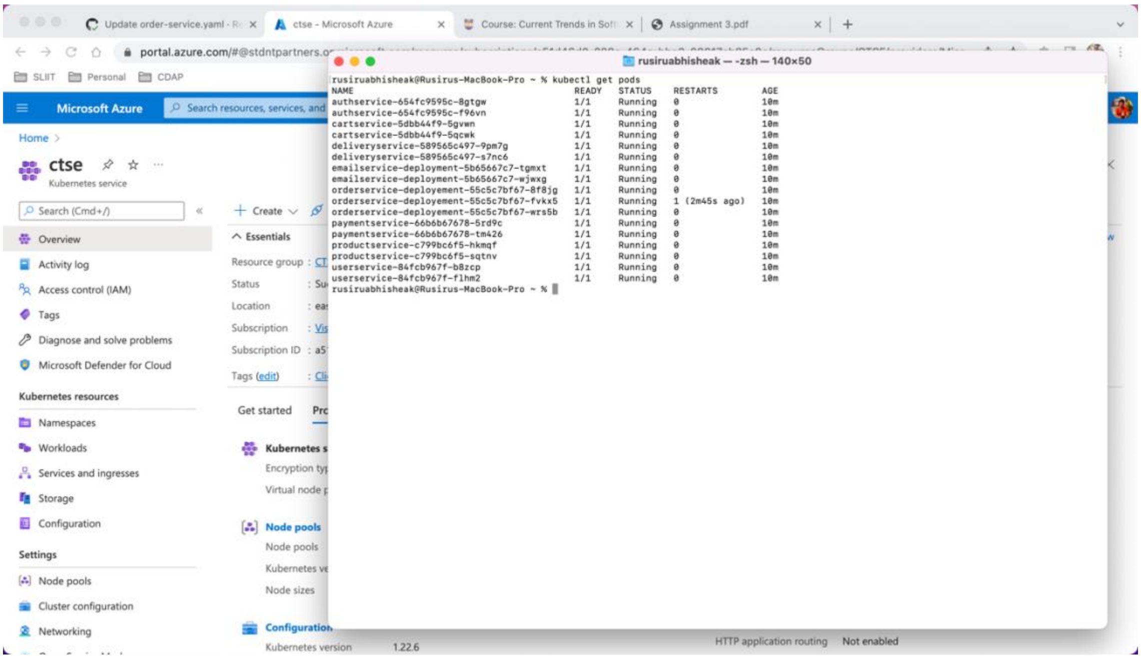
Kubernetes version	1.22.6

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

Figure 20: k8s cluster overview on Azure portal

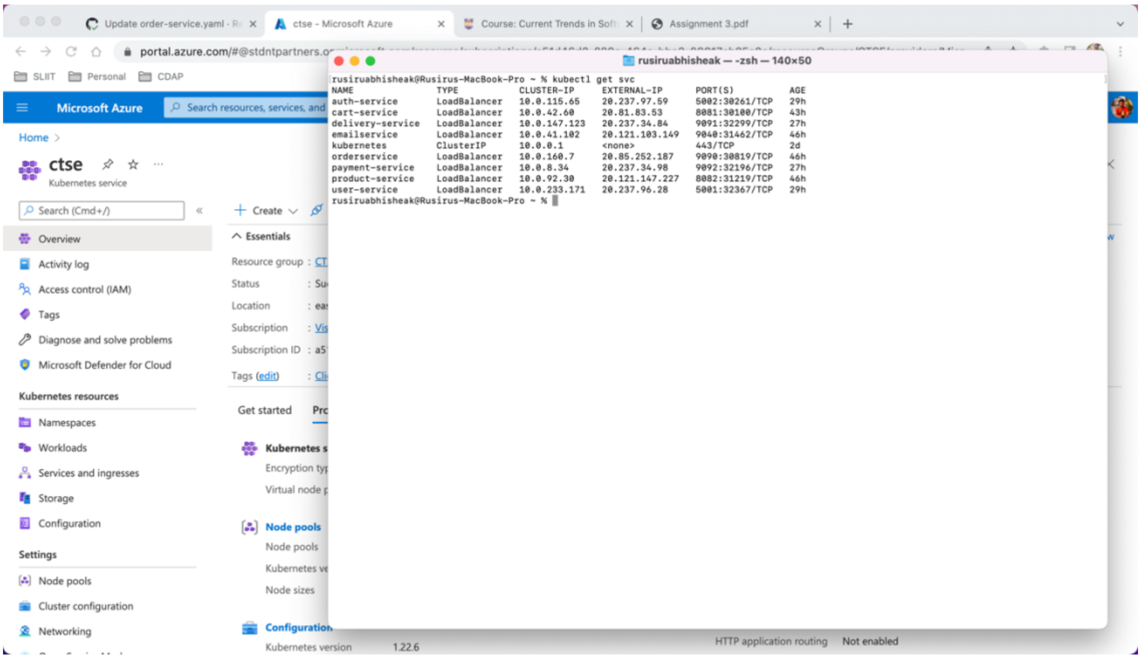
7.2 Service Pods Running on k8s Cluster


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 ~ %
```

Figure 21: Running pods on k8s cluster

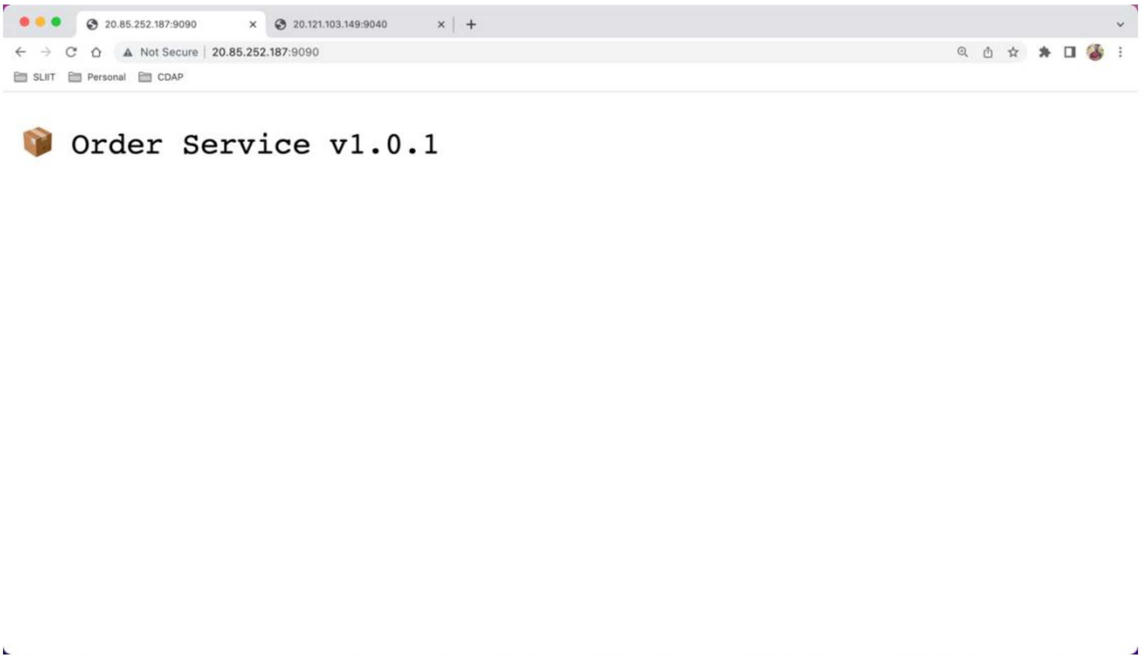
7.3 Microservices Running on k8s Cluster



```
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 ~ %
```

Figure 22: Services running on k8s cluster

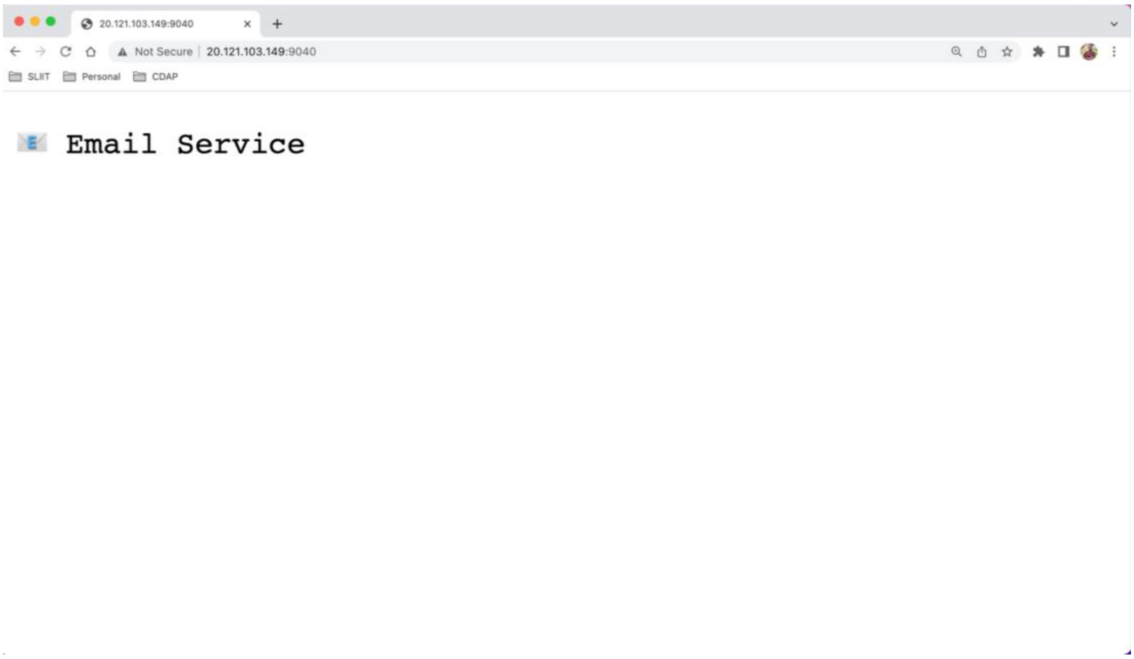
7.4 Order Service on Browser



📦 Order Service v1.0.1

Figure 23: Access order service on web browser

7.5 Email Service on Browser



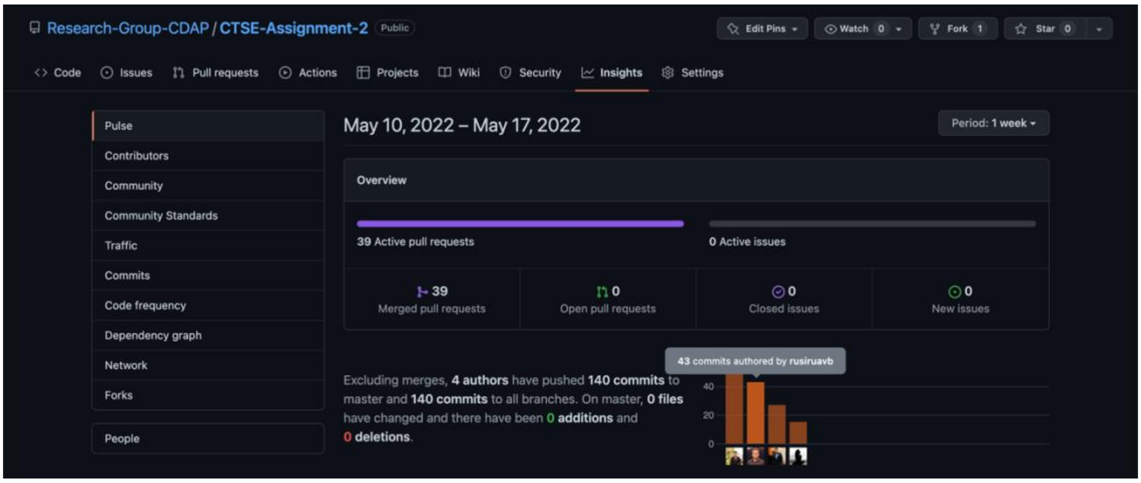
Email Service

Figure 24: Access email service on browser

8.0 GITHUB CONTRIBUTION

Git is used as the version control system and GitHub is used to collaboratively implement the microservices and create the CI/ CD pipelines. In the repository there are three branches namely **development**, **staging** and **production** (master) to maintain the code changes. Any of the member in the group cannot merge changes to the master branch directly. Therefore, if a member wants to make some changes to the code, they create a separate feature branch for the implementation. The Pull Request (PR) is created to the development branch from the feature branch. After the review process the code merge to the development branch. Then create a new PR from development branch to staging branch and finally create the PR from staging branch to master branch. In this approach makes the code management easier.

8.1 Individual Contribution to Project



May 10, 2022 – May 17, 2022

Overview

39 Active pull requests

0 Active issues

39	0	0	0
Merged pull requests	Open pull requests	Closed issues	New issues

Excluding merges, **4 authors** have pushed **140 commits** to master and **140 commits** to all branches. On master, **0 files** have changed and there have been **0 additions** and **0 deletions**.

Figure 25: Individual GitHub contribution