

Current Trends in Software Engineering

microservices project – docker, kurbernetes & CI/CD.

D. P. Kavindi Gimshani | CTSE | 17.05.2022

**IT19150826**

Malabe Batch 07.2 Group 1

TABLE OF CONTENTS

[Introduction. 2](#_Toc103689901)

[Scope of the Application. 2](#_Toc103689902)

[Use of Web services. 2](#_Toc103689903)

[Features from E-Commerce application. 2](#_Toc103689904)

[Technology for development. (Web service) 2](#_Toc103689905)

[Technology used in containerization, deploying and CI/CD pipeline. 3](#_Toc103689906)

[Process followed in the assignment. 3](#_Toc103689907)

[Individual function description. 5](#_Toc103689908)

[SOME INSIGHTS INTO THE PROJECT DEV 6](#_Toc103689909)

[PROJECT STRUCTURE 8](#_Toc103689910)

[Docker configuration 9](#_Toc103689911)

[Kubernetes configuration 12](#_Toc103689912)

[Changes after completing task 1 and task 2. 17](#_Toc103689913)

[CICD configuration 18](#_Toc103689914)

[WORKFLOW LOG FILES 25](#_Toc103689915)

[SPECIAL IMPLEMENTATIONS. 28](#_Toc103689916)

## Introduction.

### Scope of the Application.

In order to meet the user requirements of the e-commerce application, each member of our group has come up with a web service to complete the tasks according to the user requirements.

### Use of Web services.

There are two main uses of using web services in this application.

1. Backend API could be accessed through HTTP protocol.
2. Backend server could run independently. The issues of the frontend server do not affect the backend server

### Features from E-Commerce application.

* Manage a user account by using user service.
* View, add, delete, or update product details in shopping car by using product service.
* Product offer service which gives the ability for users to view and obtain latest offers being imposed on different products. View and receive the offers for the products by using product offer service.
* Insert, edit, and view the delivery details by using delivery service.

### Technology for development. (Web service)

NodeJS was used to develop the webservice and for data storage MongoDB was used by creating a cluster. VS Code was used as the tool for the development with the help of different dependencies of npm library.

A picture containing icon

Description automatically generated

Logo, company name

Description automatically generatedA picture containing text, clipart

Description automatically generated

### Technology used in containerization, deploying and CI/CD pipeline.

The project which was developed using NodeJS was added into a docker container. The running docker containers were viewed by using docker desktop. After, using Kubernetes Clusters in Google Cloud a deployment was created by using the container. The project was added to the GitHub to automate the process of building the project.

Icon

Description automatically generatedLogo

Description automatically generatedLogo, company name

Description automatically generated

### Process followed in the assignment.

1. Implement web service.
   * Backend API project for the delivery service
   * Developed using NodeJS
   * Exposed through port 3000
2. Add the application to docker container
   * Used docker desktop to add the application to docker
3. Deploy the container on Kubernetes
   * The container was deployed to Kubernetes cluster
   * Created a deployment and service
4. Automate build and deploy using GitHub
   * Used CICD automate process to docker build and Kubernetes deployment

## Individual function description.

In this assignment, I worked on creating product offer service as a NodeJS application which manages all the functionalities in managing product offers in an E-Commerce application. The service was exposed on port 3000 in local machine and the HTTP endpoints were clearly defined to use the services on frontend.

But with this assignment, implmenting a mocroservice alone is not required, the major responsibilities are:

* After the implmentation, need to create a docker image.
* The docker image should be pushed to Docker Hub and should be publicly accessible.
* Creation of a free trial account on Google cloud.
* Creating Kubernetes cluster on Google cloud platform.
* Constructing a deployment and service in Kubernetes to deploy the docker container.
* View the running pods (Max 3 replicas) and try to get the service through prot forwarding.
* Finally, create a CICD pieline in Github actions to automate the docker build process and deploying to Kubernetes.

SOME INSIGHTS INTO THE PROJECT DEV.

* First, new database was created in MongoDB cluster and configuring it to access from any machine. After the database was successfully created, the connection string was copied.
* Created a folder named ‘Backend API’ and opened it in VS Code. Using npm init inialized an emty node project.
* Installed the needed dependencies such as express, cors, body-parser, mongoose and nodemon. [Env-cmd@8.0.2](mailto:Env-cmd@8.0.2) was also installed to get control over the environment varibles (MONGODB\_URI was added as an environmental variable)
* Created a model class named ‘Product Offer’ and a Route class the HTTP endpoints were implemented with required logics.
* Created the server.JS file which is the main JS file responsible in calling the service implemented. The mongoDB\_uri was called from the enviroment variable and port 3000 was used to run the service in the local machine.
* Product-Offer Model Structure.

|  |  |  |
| --- | --- | --- |
| Column Name | Data-type | Required |
| productName | String | True |
| productPrice | Number | True |
| productDiscount | Number | True |
| offerPrice | Number | True |
| offerDiscount | Number | True |
| offerDescription | String | True |
| offerEndDate | String | True |
| offerStatus | String | True |
| productImage | String | False |
| productDescription | String | False |
| userCount | Number | True |

* HTTP Endpoint URL implemented on Product-Offer Service

|  |  |  |
| --- | --- | --- |
| HTTP Method | Short Description | Endpoint URL |
| POST | Add new product offers. | http://localhost:3000/productOffer/addProductOffer |
| GET | Get all available product offers. | http://localhost:3000/productOffer/getAllProductOffers |
| GET | Get Product offer details by offer ID. | http://localhost:3000/productOffer/getProductOfferById/:id |
| PUT | Update Product offer details | http://localhost:3000/productOffer/updateProductOfferById/:id |
| DELETE | Delete product offer detail | http://localhost:3000/productOffer/deleteProductOfferById/:id |

* Microservice Artchitecture (High-level Diagram)

Diagram

Description automatically generated

PROJECT STRUCTURE

Graphical user interface, text

Description automatically generated with medium confidence

Docker configuration.

* Created the DockerFile.

**\*\*These are not screenshots, I direclty pasted from VS Code but did not clear the formatting. This gives the same apperance of the code and commenting and components visibility.**

FROM node:latest

# Set working directory

RUN mkdir /usr/src/app

WORKDIR /usr/src

# add `/usr/src/node\_modules/.bin` to $PATH

ENV PATH /usr/src/node\_modules/.bin:$PATH

# Install and cache app dependencies

ADD package.json /usr/src/package.json

COPY . .

RUN npm install

EXPOSE 3001

# Start app

CMD ["npm", "start"]

* Authenticated in docker hub.
* Text

  Description automatically generatedIssued the docker build command to add the project to a container.
* Listed down the image using docker images ls command.

Text

Description automatically generated

* View the files in the created image

Text

Description automatically generated

* Run the docker image and check the status.

docker run -p 8080:3000 product-offer-app

* Pushed to docker hub repository.

1. Issued the docker build command.

docker build -t sanjay8330/sliit-y4-2021:v1

1. Pushed the image to the repository.

Docker push sanjay8330/sliit-y4-2021:v1

* View the image on docker hub.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Kubernetes configuration.

* Created a free account in Google cloud to use Kuberneyes services.
* Graphical user interface, text, application

  Description automatically generatedCreated a cluster in Kubernetes and added the docker container.

Calendar

Description automatically generated

* deployment.yaml File.

**\*\*These are not screenshots, I direclty pasted from VS Code but did not clear the formatting. This gives the same apperance of the code and commenting and components visibility.**

**\*\*I removed some unnecessary configuration in this yaml file below. But did not modify the original one I obtained from Kubernetes. (Not Modidifed in GitHub File)**

apiVersion: apps/v1

kind: Deployment

metadata:

  annotations:

    deployment.kubernetes.io/revision: "1"

  creationTimestamp: "2022-05-06T11:01:24Z"

  generation: 1

  labels:

    app: e-commerce

  managedFields:

  - apiVersion: apps/v1

    manager: GoogleCloudConsole

    operation: Update

    time: "2022-05-06T11:01:24Z"

  name: e-commerce

  namespace: default

  resourceVersion: "1458"

spec:

  progressDeadlineSeconds: 600

  replicas: 3

  revisionHistoryLimit: 10

  selector:

    matchLabels:

      app: e-commerce

  strategy:

    rollingUpdate:

      maxSurge: 25%

      maxUnavailable: 25%

    type: RollingUpdate

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: e-commerce

    spec:

      containers:

      - image: sanjay8330/sliit-y4-2022:v1

        imagePullPolicy: IfNotPresent

        name: sliit-y4-2022-1

        resources: {}

        terminationMessagePath: /dev/termination-log

        terminationMessagePolicy: File

      dnsPolicy: ClusterFirst

      restartPolicy: Always

      schedulerName: default-scheduler

      securityContext: {}

      terminationGracePeriodSeconds: 30

status:

  availableReplicas: 3

  conditions:

  - lastTransitionTime: "2022-05-06T11:02:01Z"

    lastUpdateTime: "2022-05-06T11:02:01Z"

    message: Deployment has minimum availability.

    reason: MinimumReplicasAvailable

    status: "True"

    type: Available

  - lastTransitionTime: "2022-05-06T11:01:24Z"

    lastUpdateTime: "2022-05-06T11:02:01Z"

    message: ReplicaSet "e-commerce-d454d5597" has successfully progressed.

    reason: NewReplicaSetAvailable

    status: "True"

    type: Progressing

  observedGeneration: 1

  readyReplicas: 3

  replicas: 3

  updatedReplicas: 3

* service.yaml File.

**\*\*These are not screenshots, I direclty pasted from VS Code but did not clear the formatting. This gives the same apperance of the code and commenting and components visibility.**

**\*\*I removed some unnecessary configuration in this yaml file below. But did not modify the original one I obtained from Kubernetes. (Not Modidifed in GitHub File)**

apiVersion: v1

kind: Service

metadata:

  annotations:

    cloud.google.com/neg: '{"ingress":true}'

  creationTimestamp: "2022-05-06T11:26:46Z"

  labels:

    app: e-commerce

  managedFields:

  name: e-commerce-service

  namespace: default

  resourceVersion: "10833"

  uid: 72abf2f4-1879-473f-b74d-802a0504fa0c

spec:

  clusterIP: 10.76.0.200

  clusterIPs:

  - 10.76.0.200

  externalTrafficPolicy: Cluster

  ipFamilies:

  - IPv4

  ipFamilyPolicy: SingleStack

  ports:

  - nodePort: 30392

    port: 80

    protocol: TCP

    targetPort: 3001

  selector:

    app: e-commerce

  sessionAffinity: None

  type: NodePort

status:

  loadBalancer: {}

* Graphical user interface, text, application, Word

  Description automatically generatedService running on Google Cloud Kubernetes Engine.
* Finally, issued kubectl get pods command to watch the running pods.

Graphical user interface, text

Description automatically generated

* Connecting to the service.

Text

Description automatically generatedCommand: gcloud container clusters get-credentials e-commerce-cluster –zone us-central1-a –project secure-air-349210 \ <printing part> && kubectl port-forward $(kubectl get pod –selector=”app=e-commerce”

Service viewed on new browser tab using port forwarding.

Graphical user interface, text, application

Description automatically generated

## Changes after completing task 1 and task 2.

* The project Structure was changed after completing task 1 and task 2.
* The project was moved inside Microservices\_E-Commerce => Services => ProductOffer\_Service
* This folder was pushed to another new Github repository and used for CICD task.
* Finally with successful completion of all the tasks this repository can be used as the final repository with the overall completion of the project.

CICD configuration.

The GitHub Repository.

A screenshot of a computer

Description automatically generated

**Configure to push Docker Image to DockerHub automatically with git push**

**STEP 01:**

First created a folder named .github and a subfolder inside named workflows in the project.

**STEP 02:**

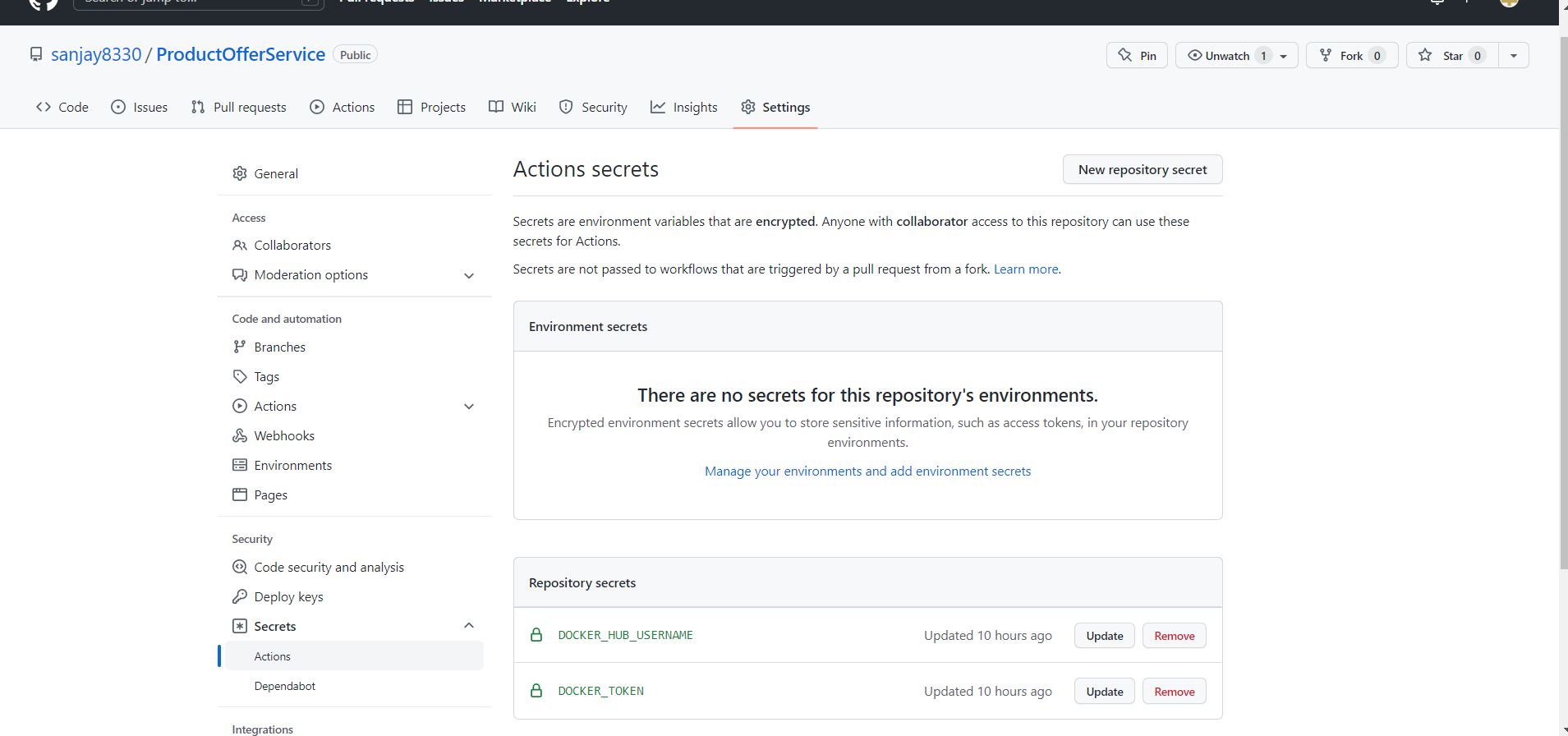
Inside the workflows folder created a file named build-deploy.yaml

**STEP 03:**

Then tried to build and deploy the changes to docker image and push it to DockerHub alone. (Didn’t make it complex by adding Google Kubernetes Engine)

**STEP 04:**

Added the tokens obtained from DockerHub to Github repository secrets.



**STEP 05:**

Build-deploy.yaml file used.

**\*\*These are not screenshots, I direclty pasted from VS Code but did not clear the formatting. This gives the same apperance of the code and commenting and components visibility.**

name: Docker Image CI

on:

  push:

    branches: [ master ]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

    - uses: actions/checkout@v3

    - name: Login to Docker Hub

      uses: docker/login-action@v1

      with:

        username: ${{ secrets.DOCKER\_HUB\_USERNAME }}

        password: ${{ secrets.DOCKER\_TOKEN }}

    - name: Build and push

      uses: docker/build-push-action@v2

      with:

        context: .

        file: ./Dockerfile

        push: true

        tags: ${{ secrets.DOCKER\_HUB\_USERNAME }}/sliit-y4-2022:${{ github.sha }}

    - name: Build the Docker image

      run: docker build . --file Dockerfile --tag sliit-y4-2022:${{ github.sha }}

**STEP 05:**

Added the file to git, committed it and finally pushed it to git.

**STEP 06:**

Observed the workflow run and finally succeeded.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Will include a complete log if in next section since the same flow runs again with the Kubernetes configurations as well.**

**Configure to push Docker Image to DockerHub with Kubernetes deployment automatically with git push**

Upon successful completion of CICD with docker build and push to DockerHub, moved to Kubernetes cluster deployment.

Referred the following Github document on how to configure github with Google Kubernetes Engine.

[Link to GitHub Document](https://docs.github.com/en/actions/deployment/deploying-to-your-cloud-provider/deploying-to-google-kubernetes-engine)

**STEP 01:**

As mentioned in the document create a service account named test-user and obtained the base64 encoded token.

Email obtained for service Account : [test-user@secure-air-349210.iam.gserviceaccount.com](mailto:test-user@secure-air-349210.iam.gserviceaccount.com)

**STEP 02:**

Added the obtained token and other secrets mentioned in the GitHub document to the repository secrets.

Graphical user interface, text, application, email

Description automatically generated

**STEP 03:**

Finally modified build-deploy.yaml file. The document had some complex scenarios, but I added the code which I was able to understand and referred on how Nilesh sir demonstrated in the lab session.

**\*\*These are not screenshots, I direclty pasted from VS Code but did not clear the formatting. This gives the same apperance of the code and commenting and components visibility.**

name: Docker Image CI

on:

  push:

    branches: [ master ]

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

    - uses: actions/checkout@v3

    - name: Login to Docker Hub

      uses: docker/login-action@v1

      with:

        username: ${{ secrets.DOCKER\_HUB\_USERNAME }}

        password: ${{ secrets.DOCKER\_TOKEN }}

    - name: Build and push

      uses: docker/build-push-action@v2

      with:

        context: .

        file: ./Dockerfile

        push: true

        tags: ${{ secrets.DOCKER\_HUB\_USERNAME }}/sliit-y4-2022:${{ github.sha }}

    - name: Build the Docker image

      run: docker build . --file Dockerfile --tag sliit-y4-2022:${{ github.sha }}

    - name: update config files in k8s

      run: sed -i.bak 's/{{tag}}/word2/g' k8s/deployment.yaml && rm k8s/\*.bak

    - uses: google-github-actions/setup-gcloud@94337306dda8180d967a56932ceb4ddcf01edae7

      with:

        service\_account\_key: ${{ secrets.KUBE\_CONFIG }}

        project\_id: ${{ secrets.PROJECT\_ID }}

    - uses: google-github-actions/get-gke-credentials@fb08709ba27618c31c09e014e1d8364b02e5042e

      with:

        cluster\_name: ${{ secrets.CLUSTER\_NAME }}

        location: ${{ secrets.CLUSTER\_LOCATION }}

        credentials: ${{ secrets.KUBE\_CONFIG }}

    - uses: actions-hub/kubectl@master

      env:

        KUBE\_CONFIG: ${{ secrets.KUBE\_CONFIG }}

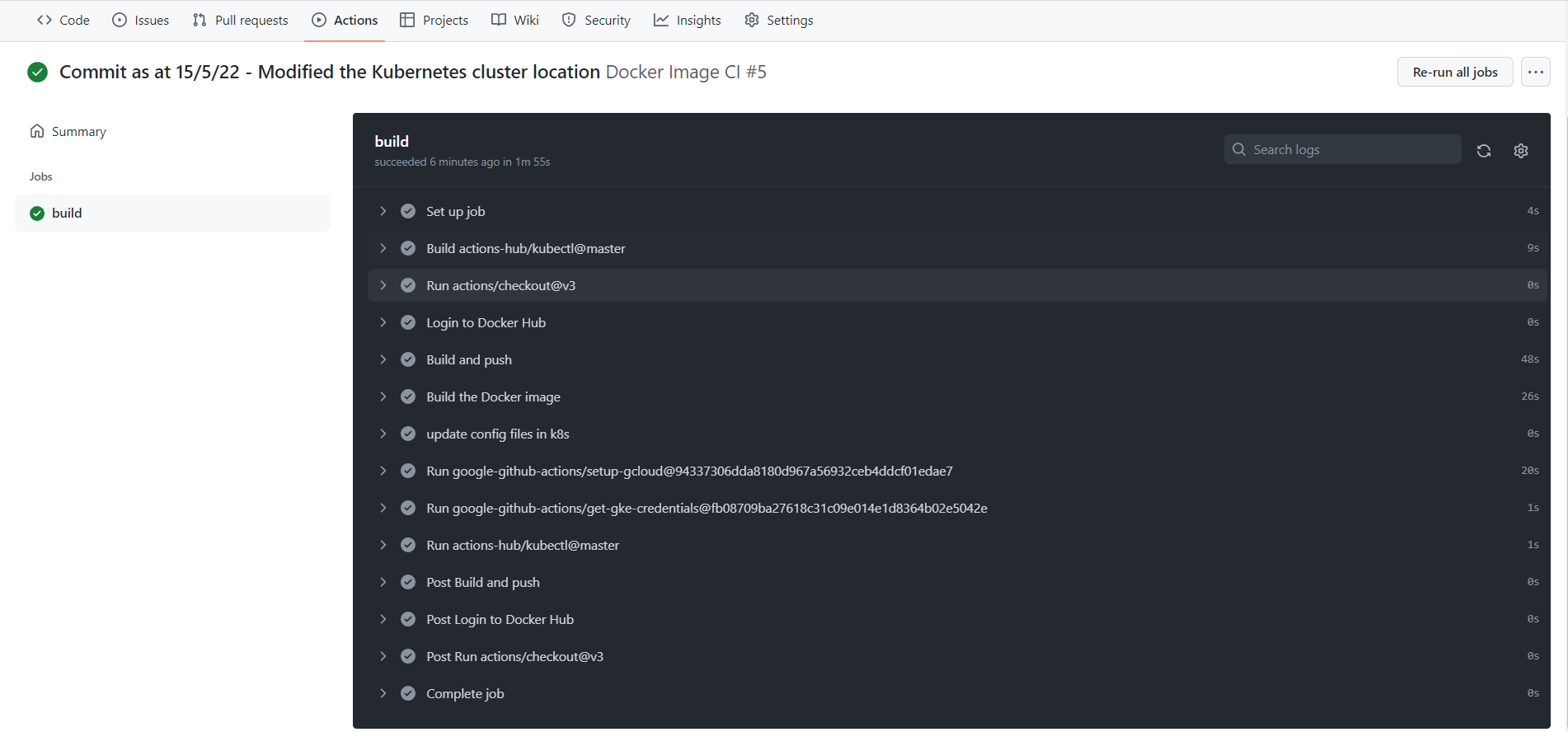
      with:

        args: get pods

**STEP 04:**

Added the file to git, committed it and finally pushed it to git.

**STEP 05**: Observe the workflow and it finally succeeded.



* **With the successful run of the workflow, it was able to see the pods running and that was the completion of task 03.**

WORKFLOW LOG FILES.

**\*\*Please note that I have not included all the logs, I have added the ones which I felt really important. If need to view the entire one please refer the GitHub Repository.**

[Link to GitHub Repository](https://github.com/sanjay8330/ProductOfferService.git)

* Login to DockerHub through Workflow.

Graphical user interface, text

Description automatically generated

* Building the DockerImage. - START

A computer screen capture

Description automatically generated with medium confidence

* Building the DockerImage. - END

Text

Description automatically generated

* Logs related to Kubernetes Configurations.

Text

Description automatically generated

* Some Post build logs for reference.

A computer screen capture

Description automatically generated with medium confidence

## SPECIAL IMPLEMENTATIONS.

* Managed to use Google Kubernetes Engine for deploying the docker image.
* Working on Google Kubernetes engine gave expose to many new features such as auto scaling and other security features provided by Kubernetes.
* Got the opportunity to fix errors that arose when creating Kubernetes cluster and deploying the docker image pushed to docker hub.
* Creating workflows on GitHub was the most challenging task which gave the opportunity to learn in depth concepts related to CICD.
* Service account creation on Google Kubernetes engine was a challenging task. Providing the necessary permission and assigning role to the created user took insights into furthermore knowledge in Kubernetes.
* Configuring build-deploy YAML file in the service project needs to be highlighted. It took effort in creating token in GitHub and adding the necessary code in automation to make it functional.
* Apart from automating the docker build and push which was a lighter task compared to the CICD task, made exploring many features provided by GitHub.
* Debugging workflows and finding the route cause for workflows to fail was a challenging task.
* Cloud shell feature available on Google Kubernetes Engine gave the opportunity to learn shell commands and get the basic understanding.