# Report M4: Model Deployment & Orchestration

[Report M4: Model Deployment & Orchestration 1](#_Toc187262860)

[Google Cloud Platform 1](#_Toc187262861)

[GCP Initial Setup 1](#_Toc187262862)

[Kubernetes Engine 2](#_Toc187262863)

[Cluster Creation 2](#_Toc187262864)

[Deployment (using Deployment.yaml) 3](#_Toc187262865)

[Deployment.yaml 3](#_Toc187262866)

[Testing the deployed app 5](#_Toc187262867)

[Deployment with Helm Charts 6](#_Toc187262868)

[Prerequisite 6](#_Toc187262869)

[Creating the chart 6](#_Toc187262870)

[Deployment Helm Config Files 6](#_Toc187262871)

[Installing/Upgrading the release 7](#_Toc187262872)

[Helm Install 8](#_Toc187262873)

[Helm upgrade 8](#_Toc187262874)

[Installing newer version 9](#_Toc187262875)

[Link to the deployed endpoint 10](#_Toc187262876)

[Deployment yaml file: 10](#_Toc187262877)

[Helm Chart Configs 10](#_Toc187262878)

**Objective**: Deploy a machine learning model and orchestrate its operations using Kubernetes.

## Google Cloud Platform

For the M4 task, we decided to use Google Cloud Platform (GCP) Over AWS or Azure for below reasons.

* **Google Kubernetes Engine (GKE)**: Highly optimized for Kubernetes, offering seamless deployment and scaling of containerized applications.
* Transparent, predictable, and cheaper pricing model compared to AWS and Azure.

Since our task is focused on deploying the dockerized model to Kubernetes, we prefer GKE, which is highly optimized for Kubernetes.

## GCP Initial Setup

* Created a project with the name MlOpsAssignmentGroup13.
* First add the team members who will need the deployment access as principals and provide them with the required access.
  + The users must have the roles
    - Kubernetes Engine Admin – To create clusters and deploy apps.
    - Service Usage Admin – To enable the Kubernetes services.

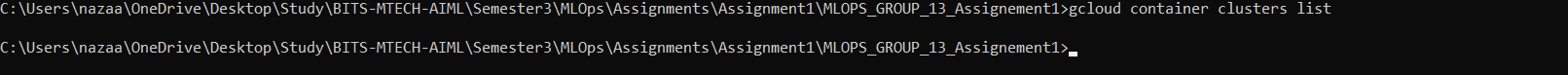
A screenshot of a computer

Description automatically generated

* Install gcloud CLI sdk if not already installed.
* Once installed, you can enable the Kubernetes services using the command **gcloud services enable container.googleapis.com** (Assuming you are already authorized and switched to the project.
  + **Login with gcloud auth login** – it will take to google sign in page.
  + **gcloud config set project mlopsassignement1group13** – Set the project you are working on with this command. The highlighted part is the project id in google cloud which was created earlier.
  + The Services can be alternatively enabled using UI also.

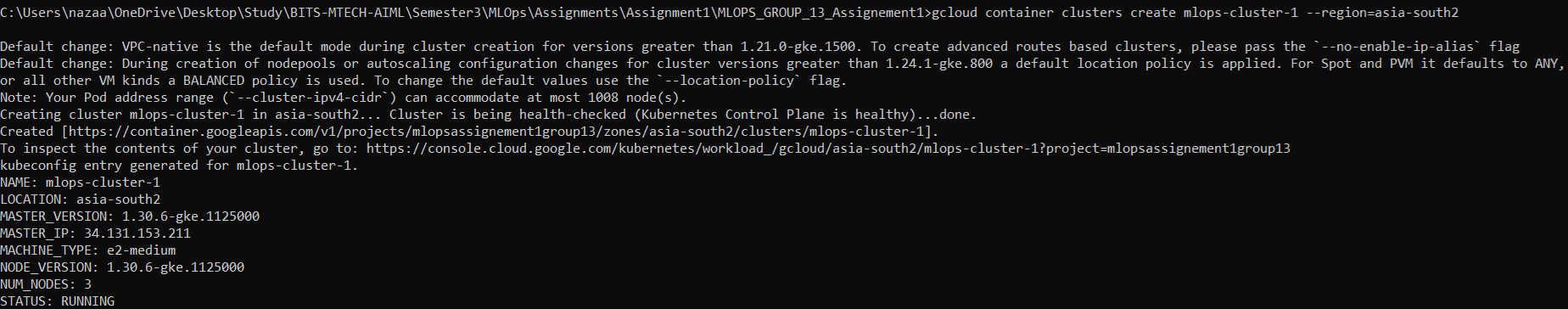
## Kubernetes Engine

### Cluster Creation

* Prior, Sign in to the Gcloud and switch to the correct project and enabled Kubernetes Services API
* You can list the clusters using the command **gcloud container clusters list** 

Initially there will be none.

* You can create a cluster using the command **gcloud container clusters create mlops-cluster-1 --region=asia-south2**
  + The region name we used is asia-south2



* Once created you can run the **gcloud container clusters list** command again and verify that cluster is created or not. It may take a few minutes to create a new cluster. Or you can view it in the GKE UI also.

A white rectangular box with colorful text

Description automatically generated

* Now you can verify that there are no workloads on the cluster in UI

A screenshot of a computer

Description automatically generated

Or using the command **kubectl get pods**

**Note:** To use kubectl commands, you need first authorize the credentials for the cluster using the command **gcloud container clusters get-credentials mlops-cluster-1 --zone asia-south2**



## Deployment (using Deployment.yaml)

Now that cluster is created, we can go ahead and deploy our image which was created in M3 task. The image is already uploaded to docker hub as **nazaarblue/mlops\_twitter\_senti:1.0.5**

A screenshot of a computer

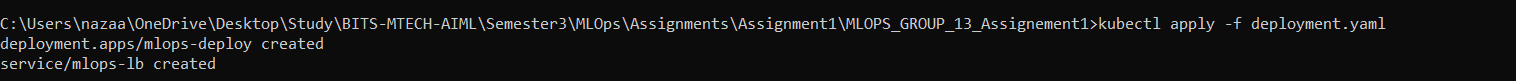
Description automatically generated

#### Deployment.yaml

|  |
| --- |
| apiVersion: apps/v1 kind: Deployment metadata:  name: mlops-deploy spec:  replicas: 1  selector:  matchLabels:  app: mlops-app  version: "v1"  template:  metadata:  labels:  app: mlops-app  version: "v1"  spec:  containers:  - name: mlops-container  image: nazaarblue/mlops\_twitter\_senti:1.0.5  imagePullPolicy: Always  ports:  - containerPort: 5001 --- apiVersion: v1 kind: Service metadata:  name: mlops-lb spec:  type: LoadBalancer  selector:  app: mlops-app  ports:  - protocol: TCP  port: 80  targetPort: 5001 |

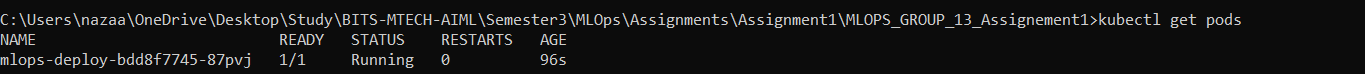
* For this activity let us just create one pod for now. This is the replicas = 1
* The image deployed in nazaarblue/mlops\_twitter\_senti:1.0.5, which is the latest version created in M3 task
* We already seen that the docker container exposes the port 5001. The load balancer port is 80.

Once the deployment file is ready, you can simply apply it easily using the command **kubectl apply -f deployment.yaml**



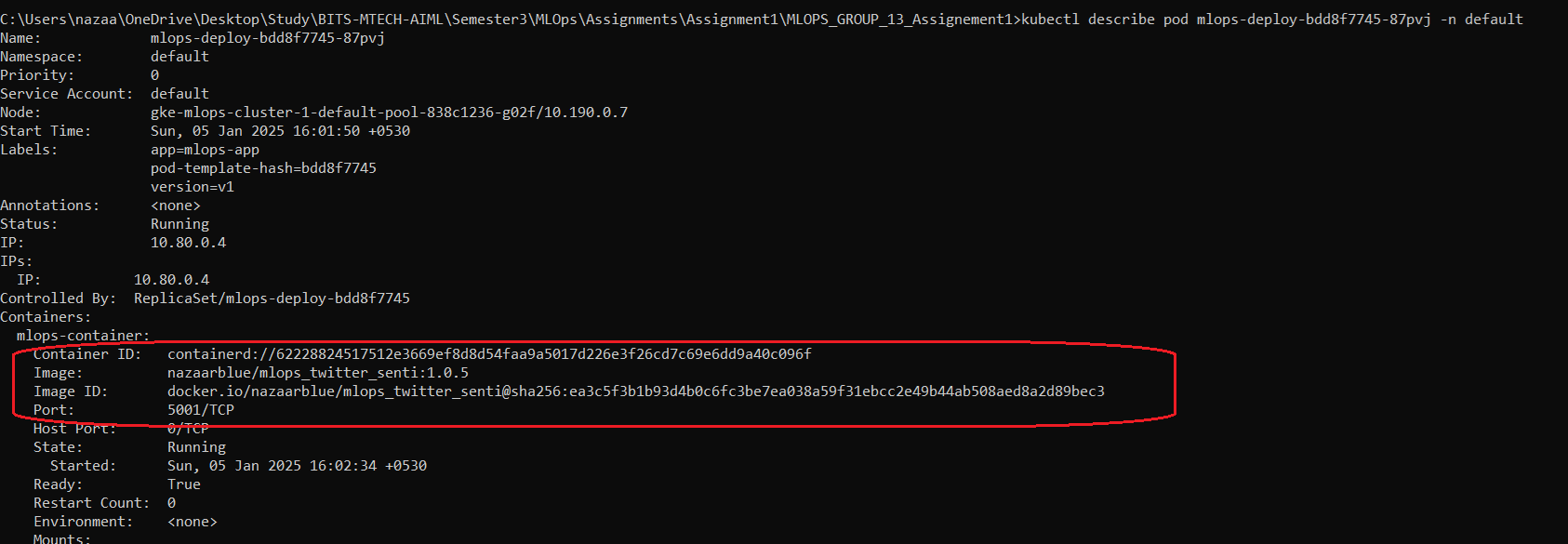
As you can see in the response the deployment is created and the service for load balancer is created.

* Now when you run the **kubectl get pods,** you can see that one pod is running.

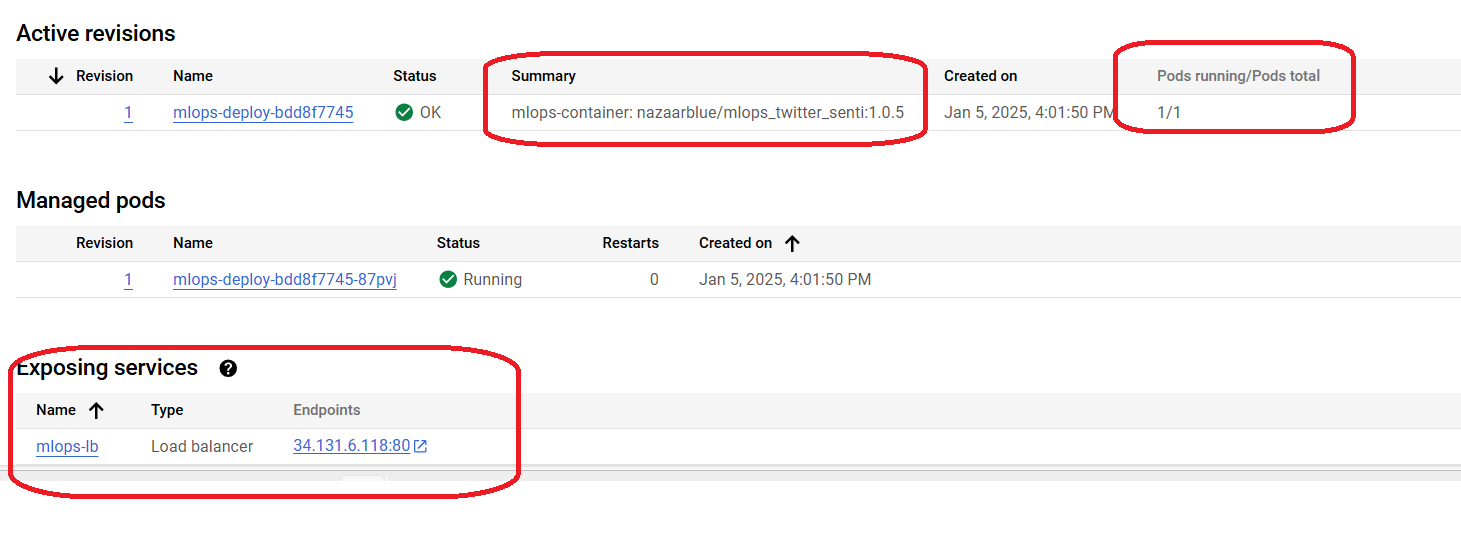


* **kubectl describe pod mlops-deploy-bdd8f7745-87pvj -n default**

Describing the pod gives more details regarding the image deployed in it



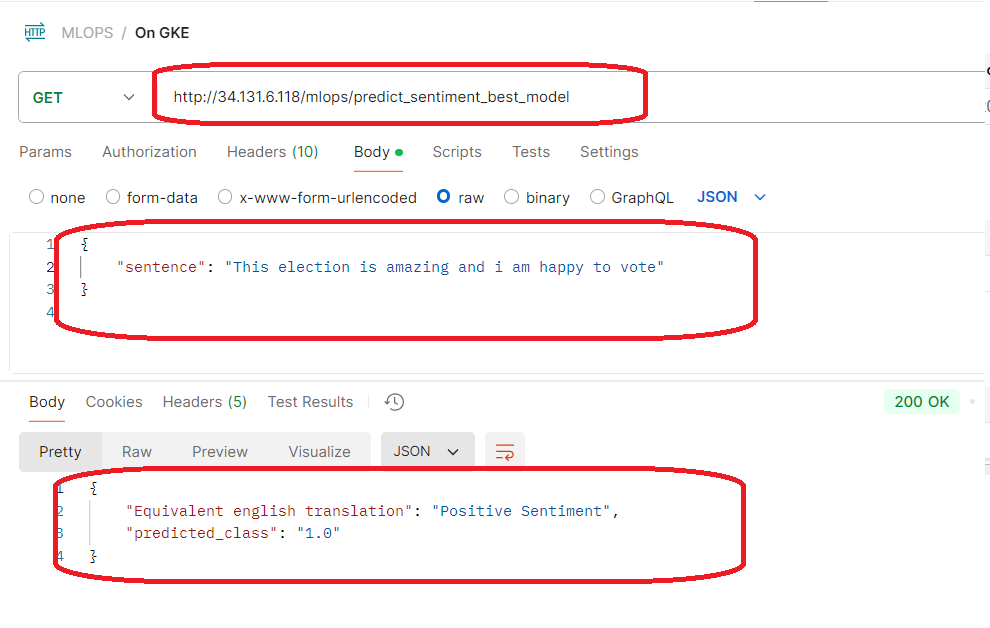
Alternatively, we can see all this information in the UI.



## Testing the deployed app

As we can see the app is currently deployed in [**http://34.131.6.118/**](http://34.131.6.118/)

We can test our model using the endpoint <http://34.131.6.118/mlops/predict_sentiment_best_model>



As we see the model is deployed on the cloud and accessible. All this is done without any issues and repeatable due to the Kubernetes and dockerization.

## Deployment with Helm Charts

Now that we have already deployed with deployment.yaml, we noticed that the details of the image, version, replicas, etc were hardcoded in the deployment.yaml. This means that we need to edit the file every time we need a change. This makes upgrading or rolling back across environments little cumbersome in production scenarios with many applications and environments. Helm charts provide a way to make deployments, rollbacks, and other orchestration easier using template configurations and dynamic values.

#### Prerequisite

Install helm using choco install kubernetes-helm. If chocolatey is not installed, install that also, to help install helm.

### Creating the chart

First you can create the chart using **helm create mlops-app.**  This command will create the chart and provide the template files, which you can edit.

A screenshot of a computer

Description automatically generated

### Deployment Helm Config Files

Helm allows to make templates of Kubernetes config manifests and use them repeatedly on different environment by changing the values.

|  |
| --- |
| **Deployment.yaml**  apiVersion: apps/v1 kind: Deployment metadata:  name: mlops-deploy spec:  replicas: {{ .Values.replicaCount }}  selector:  matchLabels:  app: {{ .Values.appName }}  version: {{ .Values.appVersion }}  template:  metadata:  labels:  app: {{ .Values.appName }}  version: {{ .Values.appVersion }}  spec:  containers:  - name: mlops-container  image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"  imagePullPolicy: Always  ports:  - containerPort: {{ .Values.containerPort }} |
| **Service.yaml**  apiVersion: v1 kind: Service metadata:  name: mlops-lb spec:  type: LoadBalancer  selector:  app: {{ .Values.appName }}  ports:  - protocol: TCP  port: 80  targetPort: {{ .Values.containerPort }} |

As shown above we can parameterize certain configs using the {{----}} format. These values can be provided dynamically when running the packages or defaulted in values.yaml file.

|  |
| --- |
| *# General Application Configuration* appName: mlops-app appVersion: v2 containerName: mlops-container replicaCount: 3  *# Image Configuration* image:  repository: nazaarblue/mlops\_twitter\_senti  tag: 1.0.1  *# Container Port Configuration* containerPort: 5001 |

As shown above we have given default values for the parameterized configs. When executing, we can override whatever the configurations we need.

### Installing/Upgrading the release

Before proceeding further we should delete the resources already existing using from our earlier deployment.

**kubectl delete deployment mlops-deploy**

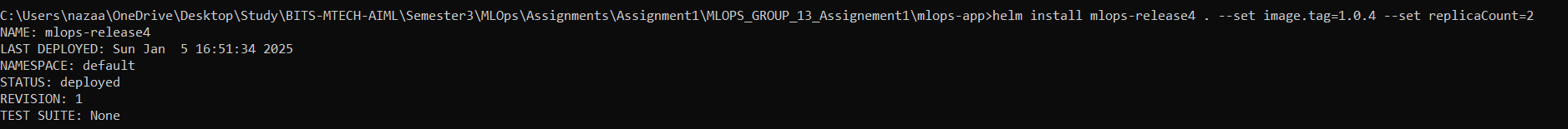
**kubectl delete service mlops-lb**

#### Helm Install

Once the chart configurations are completed, we can simply run the below command to install/upgrade.

**helm install mlops-release4 . --set image.tag=1.0.4 --set replicaCount=2**

As you can see, we are only inputting the image tag and replica count parameters. The remaining parameters would be defaulted from values.yaml. Also, I am installing an older version for now to demonstrate upgrade and rollback later.



We can see now that the 1.0.4 version is deployed in 2 pods.

A screenshot of a computer

Description automatically generated

Also we can execute the prediction service

A screenshot of a computer

Description automatically generated

#### Helm upgrade

Now to do any further change, we simply run the upgrade command with different parameters.

**helm upgrade mlops-release4 . --set image.tag=1.0.4 --set replicaCount=1**

We can immediately see that the application is running in only one pod.

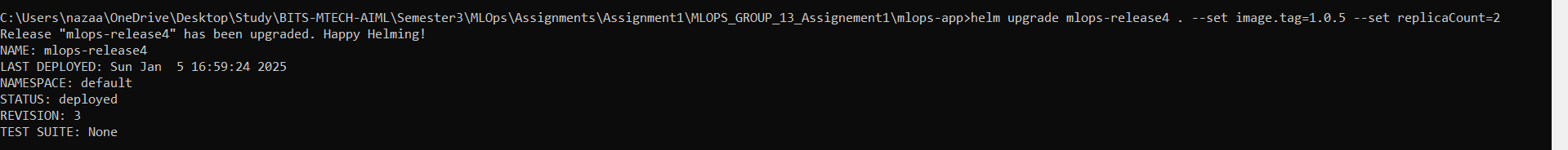
A screenshot of a computer

Description automatically generated

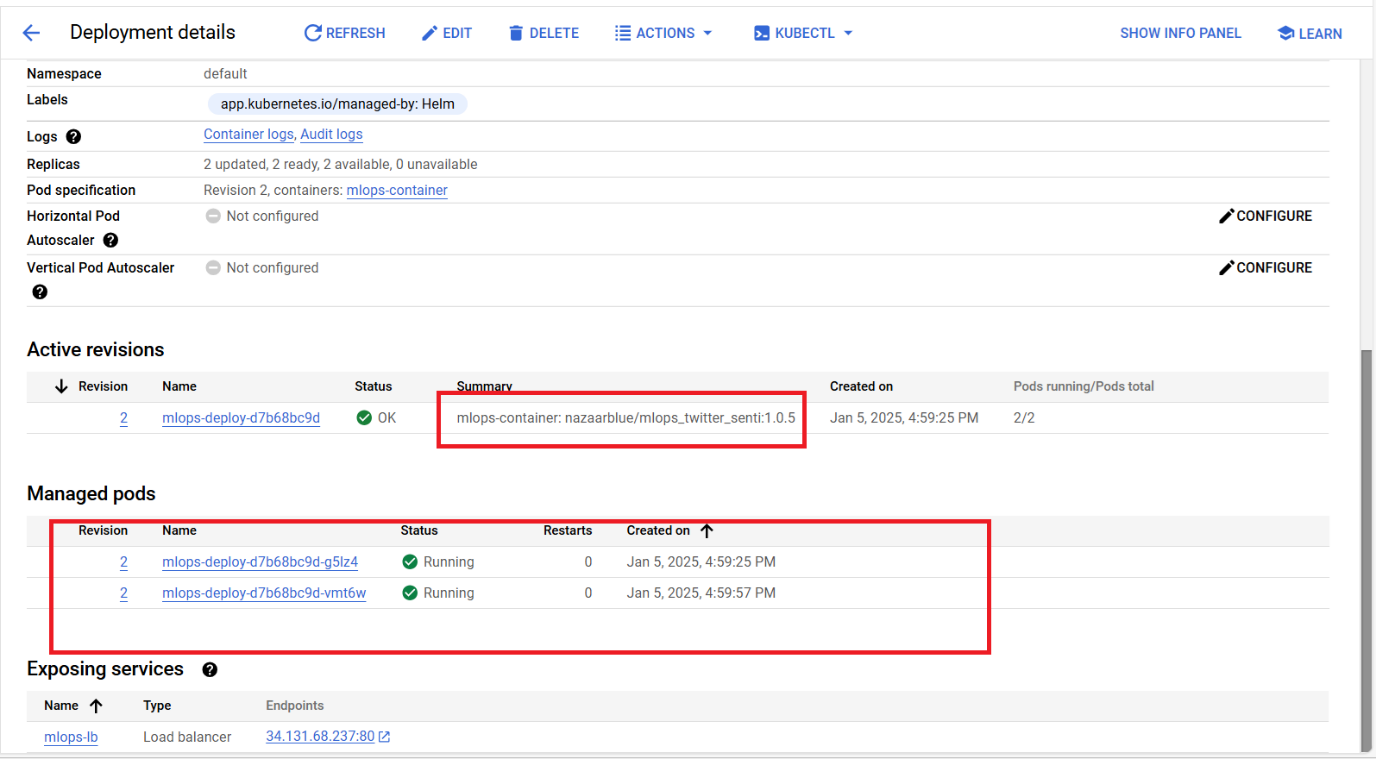
#### Installing newer version

Installing a new version, we can simply upgrade by giving the new version name.

**helm upgrade mlops-release4 . --set image.tag=1.0.5 --set replicaCount=2**

****

As seen below, the new version is deployed on 2 pods.



Also we can execute the prediction API and see it is working properly.

A screenshot of a computer

Description automatically generated

## Link to the deployed endpoint

<http://34.131.68.237/mlops/predict_sentiment_best_model>

|  |
| --- |
| Body:  {      "sentence": "This election is amazing and i am happy to vote"  } |

## Deployment yaml file:

<https://github.com/mnazaar/MLOPS_GROUP_13_Assignement1/blob/deploy/deployment.yaml>

## Helm Chart Configs

<https://github.com/mnazaar/MLOPS_GROUP_13_Assignement1/blob/deploy/mlops-app/templates/deployment.yaml>

<https://github.com/mnazaar/MLOPS_GROUP_13_Assignement1/blob/deploy/mlops-app/templates/service.yaml>

<https://github.com/mnazaar/MLOPS_GROUP_13_Assignement1/blob/deploy/mlops-app/values.yaml>

<https://github.com/mnazaar/MLOPS_GROUP_13_Assignement1/blob/deploy/mlops-app/Chart.yaml>