Cloud Computing Assignment 2: CS GY 9223

New York University
Tandon School Of Engineering

Professor Sambit Sahu

1. Abhishek Srikumar(as17913)

2. Swetha Jagadeesan(sj4378)

Part 1: Creating a Flask and MongoDB Application

Objective: Set up a simple To-Do application using Flask as the web framework and MongoDB for data persistence.

Part 2: Containerizing the Application on Docker

Objective

To containerize the Flask and MongoDB-based To-Do application, allowing it to run in isolated environments and to be easily deployed on Kubernetes.

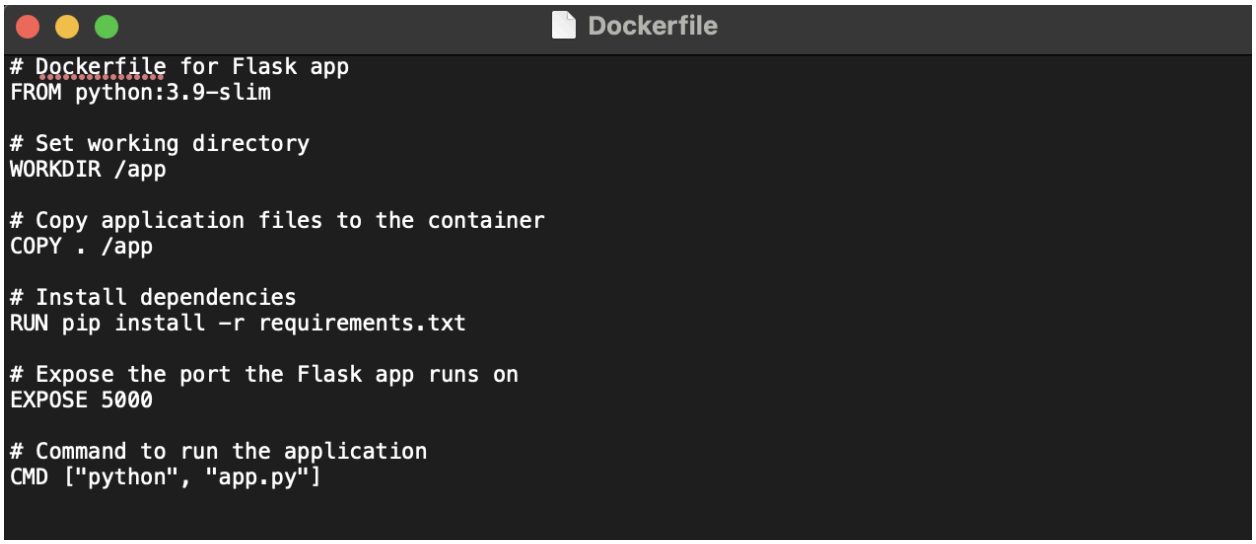
Steps

1. Docker Setup

- Confirmed Docker installation by running `docker --version` to verify it's installed.

2. Creating the Dockerfile

- The `Dockerfile` was created with instructions to install dependencies and run the Flask application.
- Verified that the `Dockerfile` has all necessary steps to install packages listed in `requirements.txt`.



3. Building the Docker Image

- Built the Docker image using the command:

docker build -t swetha0009/your-flask-app:latest .

- Successfully created the Docker image with the `latest` tag.

4. Testing with Docker Compose

- Used the `docker-compose.yml` file to start both Flask and MongoDB services:

docker-compose up

` - Verified application accessibility on `http://localhost:5001`.

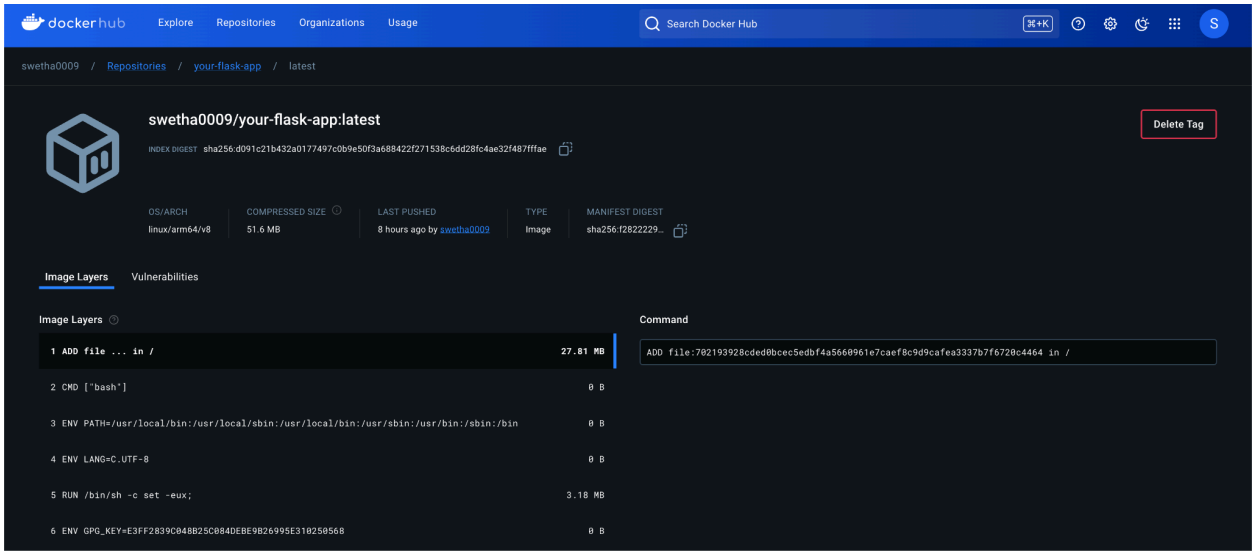
5. Pushing the Docker Image to Docker Hub

- Logged into Docker Hub and pushed the Docker image:

docker push swetha0009/your-flask-app:latest

```
swethajagadeesan@Swethas-Air Code % docker-compose up
WARN[0000] /Users/swethajagadeesan/Documents/CloudComputing/Assignment2/Cloud_Assignment_3_Files/Code/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please re
move it to avoid potential confusion
[+] Running 3/2
 ✔ Network code_default        Created                                                                                                                                    0.0s
 ✔ Container code-mongo-1      Created                                                                                                                                    0.1s
 ✔ Container code-flask-app-1  Created                                                                                                                                    0.0s
Attaching to flask-app-1, mongo-1
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.573+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify -
-sslDisabledProtocols 'none'"}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.588+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required, se
t tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.589+00:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":1,"port":27017,"dbPath":"
/data/db","architecture":"64-bit","host":"1e9a07d59a3e"}}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.589+00:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"4.4.29","gitV
ersion":"f4dda329a99811c707eb06d05ad023599f9be263","openSSLVersion":"OpenSSL 1.1.1f  31 Mar 2020","modules":[],"allocator":"tcmalloc","environment":{"distmod":"ubuntu2004","distarch":"aarc
h64","target_arch":"aarch64"}}}}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.589+00:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Ubuntu","version"
:"20.04"}}}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.589+00:00"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{"net":{"b
indIp":"*"}}}}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.591+00:00"},"s":"I",  "c":"STORAGE",  "id":22270,   "ctx":"initandlisten","msg":"Storage engine to use detected by data files","attr":{"db
path":"/data/db","storageEngine":"wiredTiger"}}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.591+00:00"},"s":"I",  "c":"STORAGE",  "id":22297,   "ctx":"initandlisten","msg":"Using the XFS filesystem is strongly recommended with the
 WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem","tags":["startupWarnings"]}
mongo-1      | {"t":{"$date":"2024-11-06T08:55:58.592+00:00"},"s":"I",  "c":"STORAGE",  "id":22315,   "ctx":"initandlisten","msg":"Opening WiredTiger","attr":{"config":"create,cache_size=1
448M,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_ti
me=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress,compact_progress],"}}
flask-app-1  |  * Serving Flask app 'app' (lazy loading)
flask-app-1  |  * Environment: production
flask-app-1  |    WARNING: This is a development server. Do not use it in a production deployment.
flask-app-1  |    Use a production WSGI server instead.
flask-app-1  |  * Debug mode: on
flask-app-1  |  * Running on all addresses.
flask-app-1  |    WARNING: This is a development server. Do not use it in a production deployment.
flask-app-1  |  * Running on http://172.19.0.3:5000/ (Press CTRL+C to quit)
flask-app-1  |  * Restarting with stat
flask-app-1  |  * Debugger is active!
flask-app-1  |  * Debugger PIN: 100-109-955
```

- The image is now available on Docker Hub, ready for Kubernetes deployment.



6. Cleanup

- Stopped and removed the Docker Compose containers after testing:
  docker-compose down

Part 3: Deploying the Application on Minikube

Objective

Deploy the containerized application on a local Kubernetes cluster using Minikube, applying Kubernetes deployment strategies, including replication control, rolling updates, and health monitoring.

Steps

1. **Minikube Setup**

- Step 1: Starting Minikube

- Started Minikube to set up a local Kubernetes cluster:

```
minikube start
```

- Configured `kubectl` to use the Minikube cluster automatically.

- Step 2: Setting Up Minikube Tunnel

- Opened a new terminal and ran:

```
minikube tunnel
```

- This command created a network route to access NodePort services from the host, allowing us to access services using Minikube's IP.

- Step 3: Getting Minikube IP

- Retrieved the Minikube IP using:

```
minikube ip
```

- The IP was used later to access the deployed Flask application on the specified NodePort.

```
minikube start

✋  Stopping node "minikube"  ...
🛑  Powering off "minikube" via SSH ...
🛑  1 node stopped.
😄  minikube v1.34.0 on Darwin 14.5 (arm64)
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.45 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

🌟  Enabled addons: default-storageclass, storage-provisioner, dashboard
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
swethajagadeesan@Swethas-Air Cloud_Assignment_3_Files % kubectl get services

NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
flask-service   NodePort    10.97.193.118    <none>        5000:30001/TCP   29m
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP          5d8h
mongo           ClusterIP   10.100.193.189   <none>        27017/TCP        22m
swethajagadeesan@Swethas-Air Cloud_Assignment_3_Files % minikube ip

192.168.49.2
swethajagadeesan@Swethas-Air Cloud_Assignment_3_Files % minikube tunnel

✅  Tunnel successfully started

📌  NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...
```

2. Applying Kubernetes Manifests

- Created and applied the Kubernetes manifests:

kubectl apply -f Deployment.yaml

kubectl apply -f Services.yaml

```
swethajagadeesan@Swethas-Air Code % kubectl apply -f Deployment.yaml
deployment.apps/flask-app created
deployment.apps/mongo-db created
swethajagadeesan@Swethas-Air Code % kubectl apply -f Services.yaml
service/flask-service created
service/mongo-service created
swethajagadeesan@Swethas-Air Code % kubectl apply -f mongodb-pvc.yaml
persistentvolumeclaim/mongodb-pvc created
swethajagadeesan@Swethas-Air Code % kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)          AGE
flask-service   LoadBalancer   10.100.143.234   a391d2182fd2744aeb193ce1859c3462-1537102876.us-east-1.elb.amazonaws.com   5000:32378/TCP   32s
kubernetes      ClusterIP      10.100.0.1       <none>                                                                    443/TCP          21m
mongo-service   ClusterIP      10.100.29.25     <none>                                                                    27017/TCP        32s
```

- Verified that the `flask-app` and `mongo-service` pods were running:

kubectl get pods

```
swethajagadeesan@Swethas-Air Cloud_Assignment_3_Files % kubectl get pods
NAME                         READY   STATUS    RESTARTS      AGE
flask-app-5b99b5c554-d45wx   1/1     Running   0             31m
flask-app-5b99b5c554-pzqw5   1/1     Running   0             31m
flask-app-5b99b5c554-qjw48   1/1     Running   0             31m
mongo-fb7c95b6f-d5fg5        1/1     Running   1 (40m ago)   75m
swethajagadeesan@Swethas-Air Cloud_Assignment_3_Files % kubectl get svc
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
flask-service   NodePort    10.99.224.183    <none>        5000:30001/TCP   29m
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP          5d9h
mongo           ClusterIP   10.100.193.189   <none>        27017/TCP        62m
mongo-service   ClusterIP   10.104.50.42     <none>        27017/TCP        32m
```

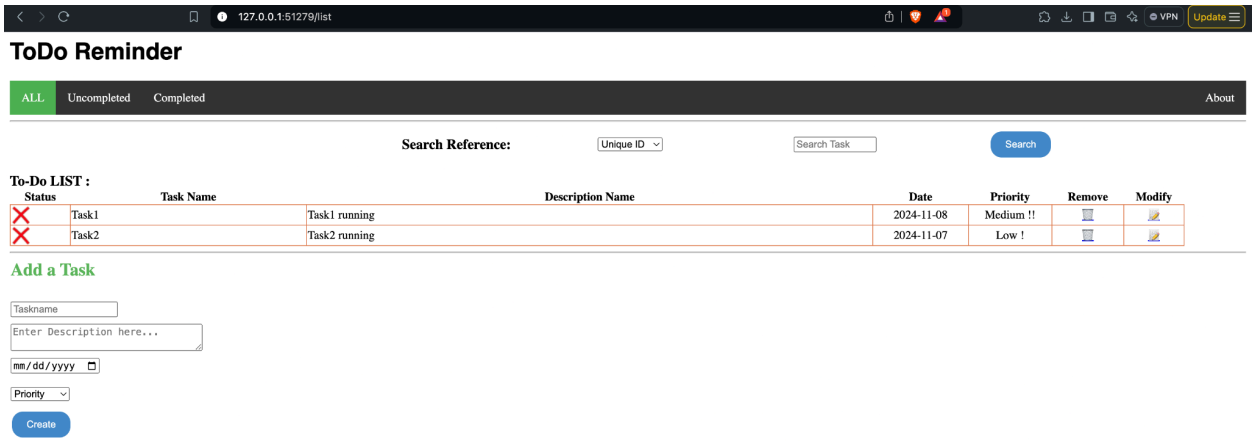
3. Accessing the Application

- Used the following command to get the URL for `flask-service`:

  minikube service flask-service --url

- Accessed the application in the browser using the provided URL (e.g., `http://127.0.0.1:51279`).

- Step 2: Interacting with the Application
  - Verified the functionality of the To-Do app by adding, viewing, and deleting tasks.
  - Confirmed that the Flask app could communicate with MongoDB, as the tasks were added and persisted correctly.



4. Cleanup

- Stopped and deleted the Minikube cluster after testing:

  minikube stop
  minikube delete

Troubleshooting

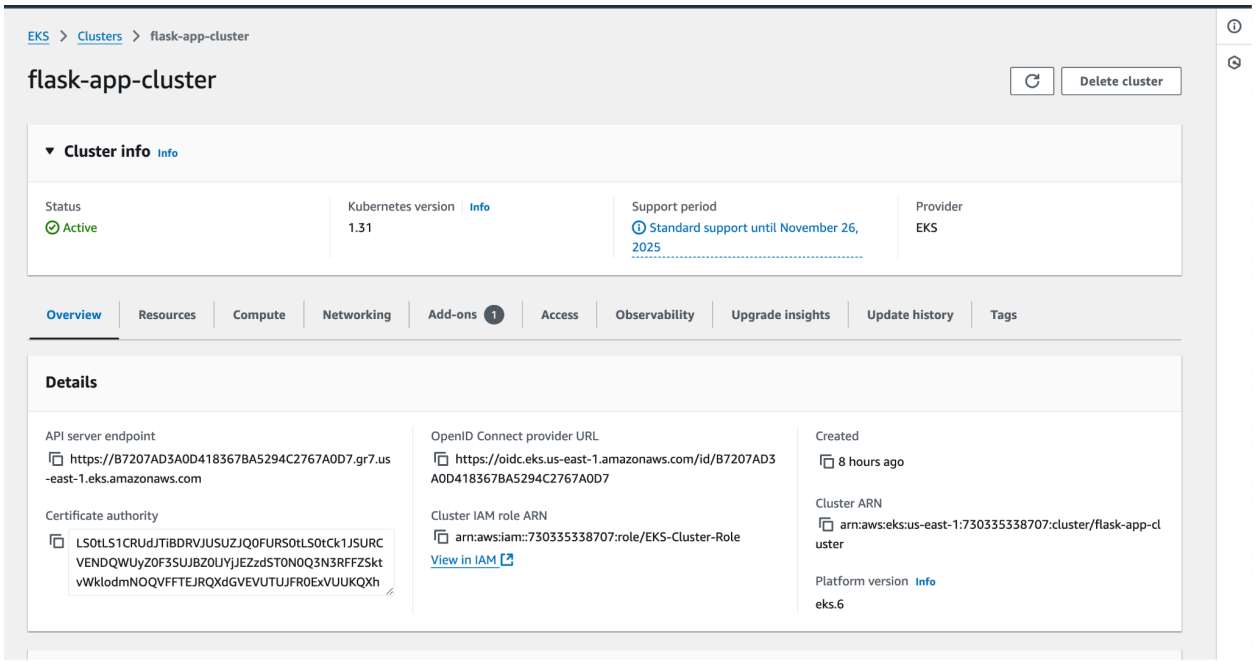
- Issue 1: MongoDB Connectivity Error
  - Problem: Encountered a `ServerSelectionTimeoutError`, indicating that the Flask app couldn't connect to MongoDB.
  - Solution: Realized that the service name in the `MONGO_URI` was incorrect. Updated the `MONGO_URI` environment variable in `flask-deployment.yaml` to `mongodb://mongo-service:27017/camp2016` (matching the MongoDB service name in `mongo-service.yaml`).

- Issue 2: Accessing the Application via Minikube IP
  - Problem: Attempted to access the app using Minikube's IP and NodePort, but received "Site can't be reached" errors.
  - Solution: Switched to using `minikube service --url flask-service` to obtain the service URL dynamically. This provided a local URL (e.g., `http://127.0.0.1:51043`), allowing successful access to the application.

Part 4: Deploying the Application on AWS EKS:

Step 1: Set Up the EKS Cluster

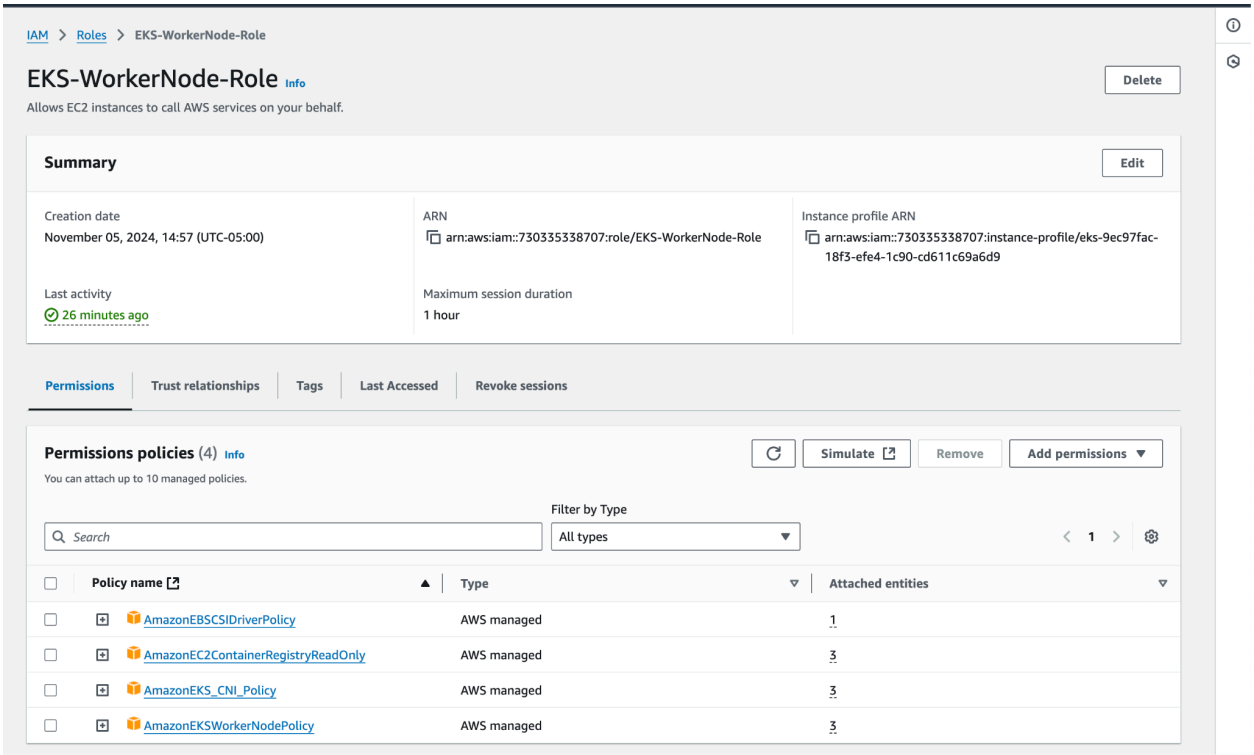
Open the AWS Management Console, navigate to EKS, and create a new cluster.



Configure the Cluster:

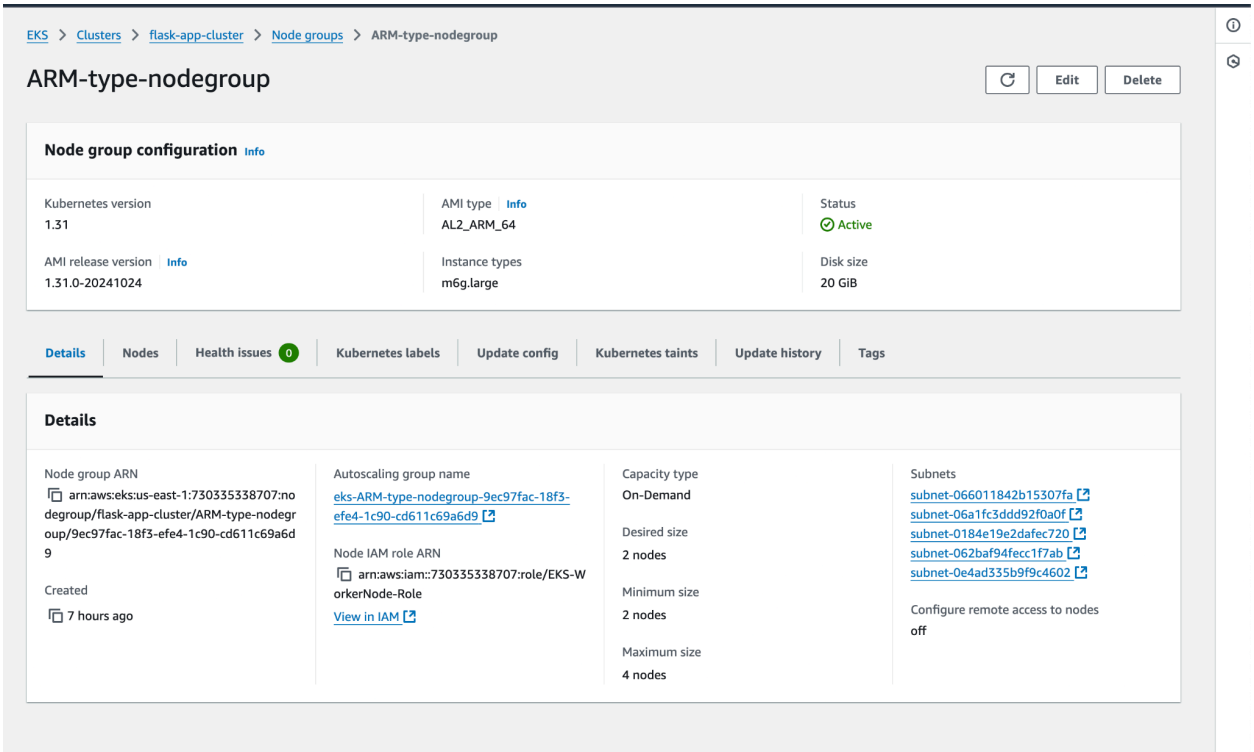
- Set a **cluster name** (e.g., flask-app-cluster).
- Select **Region** where you want to deploy the cluster.
- Set **Kubernetes version** - 1.31.
- **Networking**: Select a VPC and subnets
- **Role**: Assign an IAM role with the necessary permissions for EKS.

Create a role **EKS-WorkerNode-Role** to attach to the Cluster:



Create Node Group:

- After creating the cluster, go to the Add **node group** to add EC2 instances that will run your Kubernetes workloads.
- Configure the **node group name**, **instance types** (m6g.large), **desired capacity**, and **scaling policies**.
- Choose either AL2_ARM_64 AMI type for the architecture of the Docker image you built.
- Create the node group and wait for the nodes to become active.



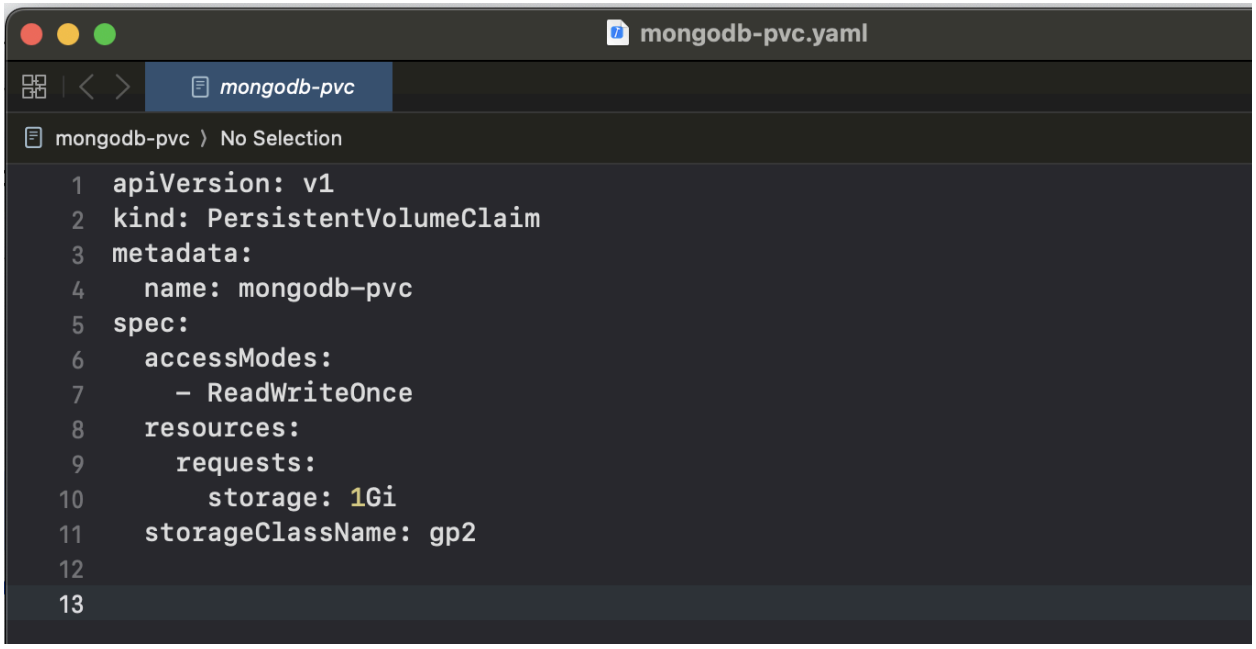
Update kubectl Configuration:

Use the following command to set up kubectl for your EKS cluster:
aws eks --region us-east-1 update-kubeconfig --name flask-app-cluster

```
swethajagadeesan@Swethas-Air Code % aws eks --region us-east-1 update-kubeconfig --name flask-app-cluster
Updated context arn:aws:eks:us-east-1:730335338707:cluster/flask-app-cluster in /Users/swethajagadeesan/.kube/config
```

2. Apply the Persistent Volume Claim (Optional but Recommended)

My mongodb-pvc.yaml:



Applying the mongo-db-pvc.yaml:

kubectl apply -f mongodb-pvc.yaml

3. Deploy the Flask Application and MongoDB with Kubernetes YAML Files

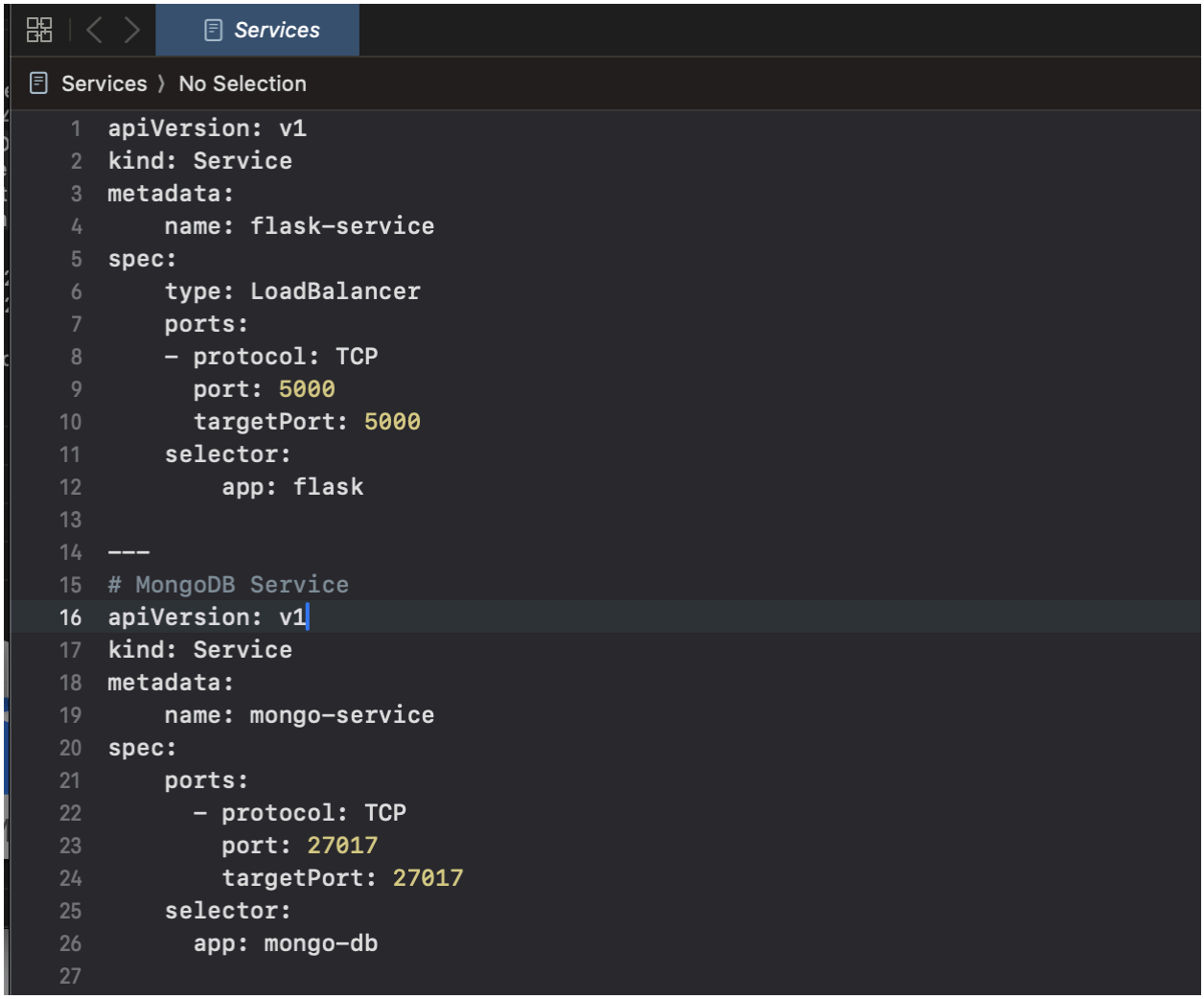
First, apply the Deployment.yaml file to create the Flask application deployment:
kubectl apply -f Deployment.yaml

Deployment

Deployment 〉 No Selection

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  selector:
    matchLabels:
      app: flask
  template:
    metadata:
      labels:
        app: flask
    spec:
      containers:
      - name: flask-container
        image: swetha0009/your-flask-app:v2  # Update with the new image version
        ports:
        - containerPort: 5000
        env:
        - name: MONGO_URI
          value: "mongodb://mongo-service:27017/camp2016"
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-db
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo-db
  template:
    metadata:
      labels:
        app: mongo-db
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
        - containerPort: 27017
        volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
      volumes:
      - name: mongodb-data
        persistentVolumeClaim:
          claimName: mongodb-pvc

```

Then, deploy the Services.yaml file to create services for Flask and MongoDB:
kubectl apply -f Services.yaml



4. Verify the Deployment

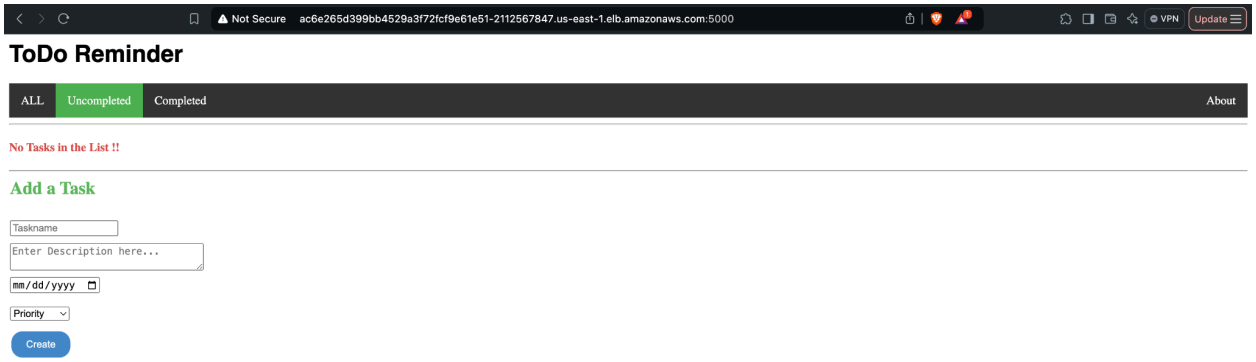
Check that the pods are running successfully:

kubectl get pods

Check that services are created, especially the flask-service which should have an external IP address assigned (LoadBalancer service):

kubectl get services

The application is running successfully on EKS-cluster.



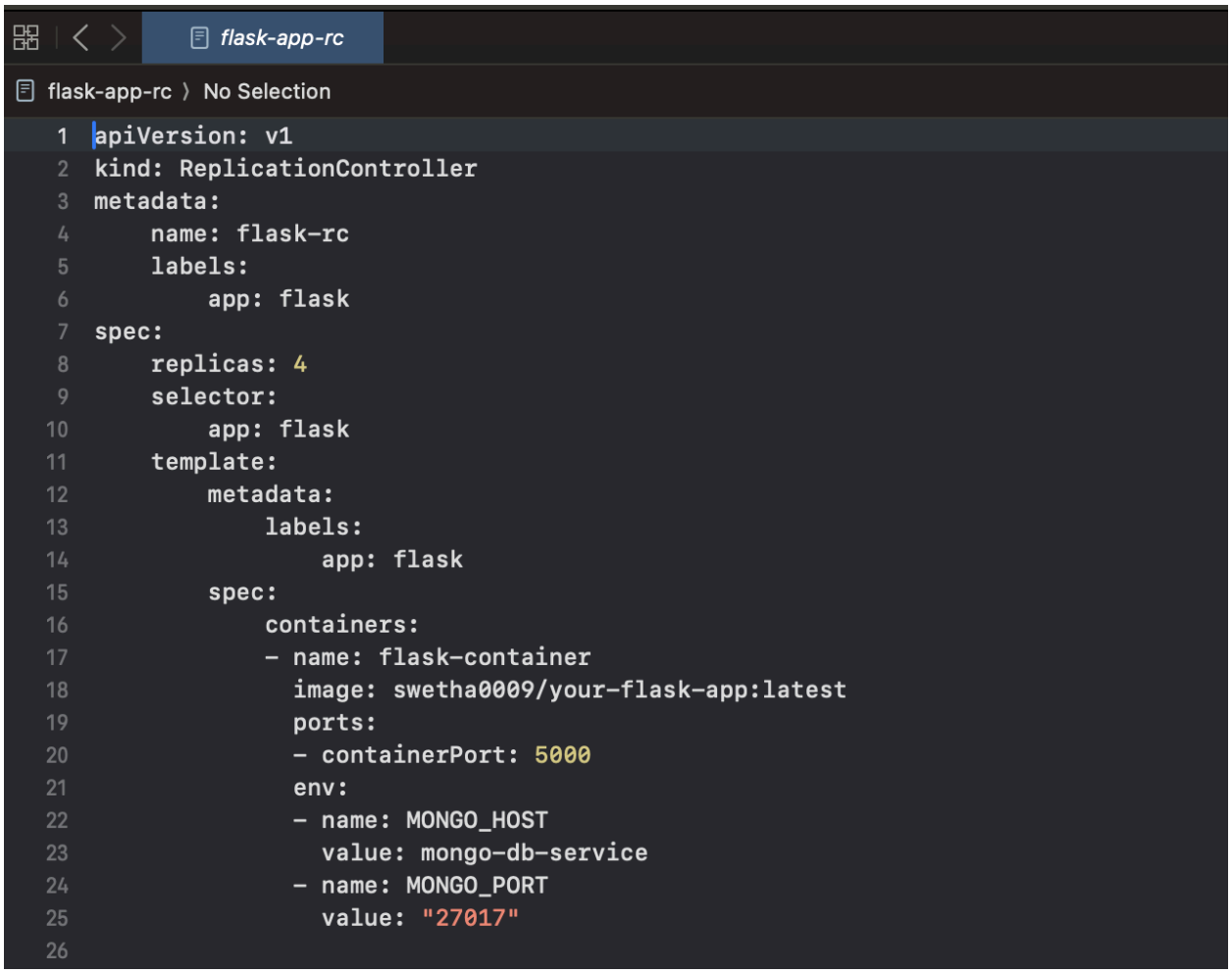
Part 5: Replication Controller Feature Documentation

A ReplicationController in Kubernetes is used to manage the replicas of our Flask application. A ReplicationController ensures that a specified number of pod replicas are running at any given time. This feature provides high availability and fault tolerance, automatically replacing failed pods or scaling the application as needed.

Steps to Create and Configure the ReplicationController:

1. Define the ReplicationController Configuration

- We created a YAML file named `**flask-app-rc.yaml**` to define the configuration for the ReplicationController.



- Explanation of Key Fields:
  - `replicas: 4`: Specifies that 4 replicas of the Flask application should be running.
  - `selector` and `labels`: Define which pods the ReplicationController should manage by matching the `app: flask` label.
  - `containers`: Specifies the container settings for the Flask application, including the Docker image (`swetha0009/your-flask-app:latest`) and the environment variables (`MONGO_HOST` and `MONGO_PORT`) for MongoDB connectivity.

2. Deploy the ReplicationController

- We applied the ReplicationController configuration using the following command:

  kubectl apply -f flask-app-rc.yaml

- To verify that the ReplicationController created the specified number of replicas, we ran:

  kubectl get pods

- The output confirmed that 4 replicas of `flask-container` were running as expected.

```
swethajagadeesan@Swethas-Air Code % kubectl apply -f flask-app-rc.yaml

replicationcontroller/flask-rc created
swethajagadeesan@Swethas-Air Code % kubectl get pods

NAME                        READY   STATUS    RESTARTS   AGE
flask-app-596b784d84-8tg2t  1/1     Running   0          6h41m
flask-app-596b784d84-fmvjr  1/1     Running   0          6h41m
flask-app-596b784d84-pbzc7  1/1     Running   0          6h41m
flask-rc-kz2v2              1/1     Running   0          65s
flask-rc-lqrrm              1/1     Running   0          65s
flask-rc-qrp4d              1/1     Running   0          65s
flask-rc-xdt2d              1/1     Running   0          65s
mongo-db-74f6f8bd65-m2gz7   1/1     Running   0          68m
```

3. Testing the ReplicationController

1. Automatic Pod Replacement - To test the ReplicationController's self-healing capability, we manually deleted one of the pods managed by the ReplicationController:

  kubectl delete pod <pod-name>

Ex: kubectl delete pod flask-rc-kz2v2

- After deleting a pod, we observed that the ReplicationController automatically created a new pod to maintain the specified replica count. Running `kubectl get pods` again showed that the total number of replicas was restored to 4.

```
swethajagadeesan@Swethas-Air Code % kubectl delete pod flask-rc-kz2v2
pod "flask-rc-kz2v2" deleted
swethajagadeesan@Swethas-Air Code % kubectl get pods

NAME                        READY   STATUS    RESTARTS   AGE
flask-app-596b784d84-8tg2t  1/1     Running   0          6h44m
flask-app-596b784d84-fmvjr  1/1     Running   0          6h44m
flask-app-596b784d84-pbzc7  1/1     Running   0          6h44m
flask-rc-jfz7n              1/1     Running   0          3s
flask-rc-lqrrm              1/1     Running   0          4m18s
flask-rc-qrp4d              1/1     Running   0          4m18s
flask-rc-xdt2d              1/1     Running   0          4m18s
mongo-db-74f6f8bd65-m2gz7   1/1     Running   0          71m
```

2. Scaling the ReplicationController - We tested the scaling capability of the ReplicationController by modifying the `replicas` count in `flask-app-rc.yaml` from 4 to 6.

- Updated `replicas` section:
  spec:
    replicas: 6

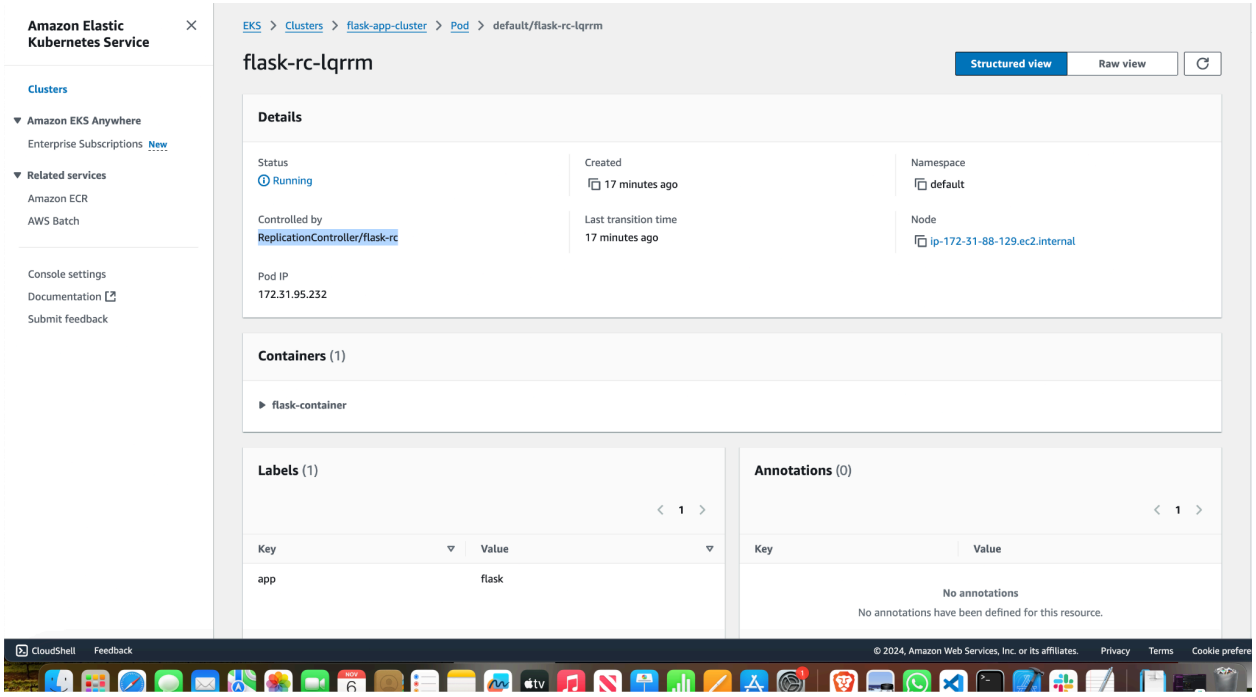
- After updating the YAML file, we reapplied the configuration:
  kubectl apply -f flask-app-rc.yaml

- Running `kubectl get pods` confirmed that the ReplicationController increased the pod count to match the new replica count of 6.

```
swethajagadeesan@Swethas-Air Code % kubectl apply -f flask-app-rc.yaml
replicationcontroller/flask-rc configured
swethajagadeesan@Swethas-Air Code % kubectl get pods

NAME                        READY   STATUS    RESTARTS   AGE
flask-app-596b784d84-8tg2t  1/1     Running   0          6h54m
flask-app-596b784d84-fmvjr  1/1     Running   0          6h54m
flask-app-596b784d84-pbzc7  1/1     Running   0          6h54m
flask-rc-74rbl              1/1     Running   0          2s
flask-rc-cxzjr              1/1     Running   0          2s
flask-rc-jfz7n              1/1     Running   0          10m
flask-rc-lqrrm              1/1     Running   0          14m
flask-rc-qrp4d              1/1     Running   0          14m
flask-rc-xdt2d              1/1     Running   0          14m
mongo-db-74f6f8bd65-m2gz7   1/1     Running   0          81m
```

4. Displaying the creation of replication controller on EKS AWS:

- Showing the creation of flask-rc which is basically the replication controller -Showing - creation of pods in AWS management console.



Observations and Key Takeaways

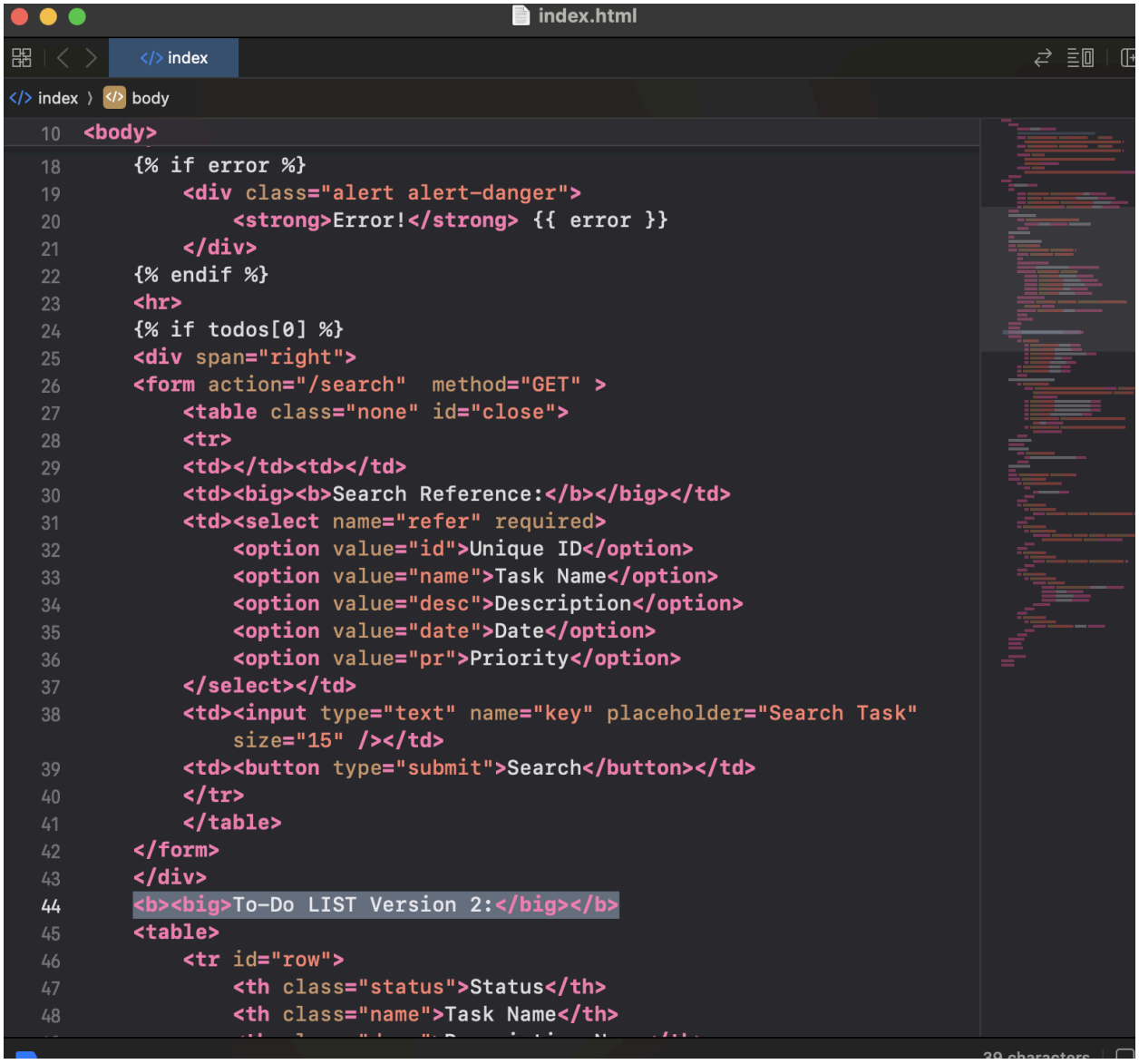
- The ReplicationController successfully maintained the specified number of replicas, even when a pod was deleted, demonstrating Kubernetes' resilience and fault tolerance.
- By changing the replica count in the YAML file, we could scale the application up or down, confirming that ReplicationController supports basic scaling.
- ReplicationController vs. ReplicaSet: Although ReplicationController performs basic replica management well, ReplicaSet and Deployments in Kubernetes are generally preferred due to additional features like rolling updates and more flexible selectors.

Part 6: Rolling Update Strategy

Objective: Implement zero-downtime deployments using a rolling update strategy.

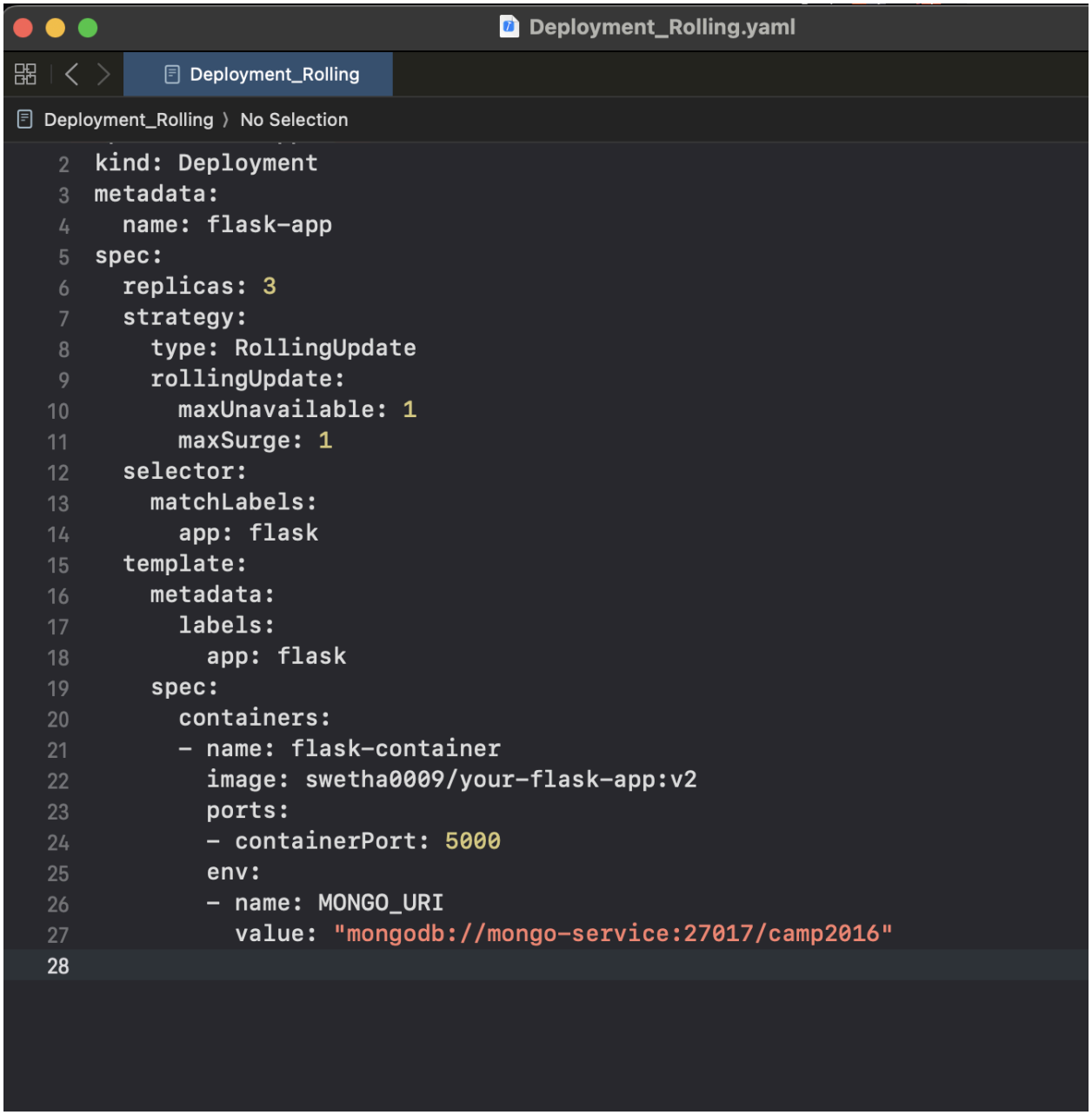
Step 1:

-Making changes to the index.html file for the rolling update strategy to be implemented.



Step 2: Setting the update strategy in Deployment_Rolling.yaml:

Having the strategy type to be RollingUpdate



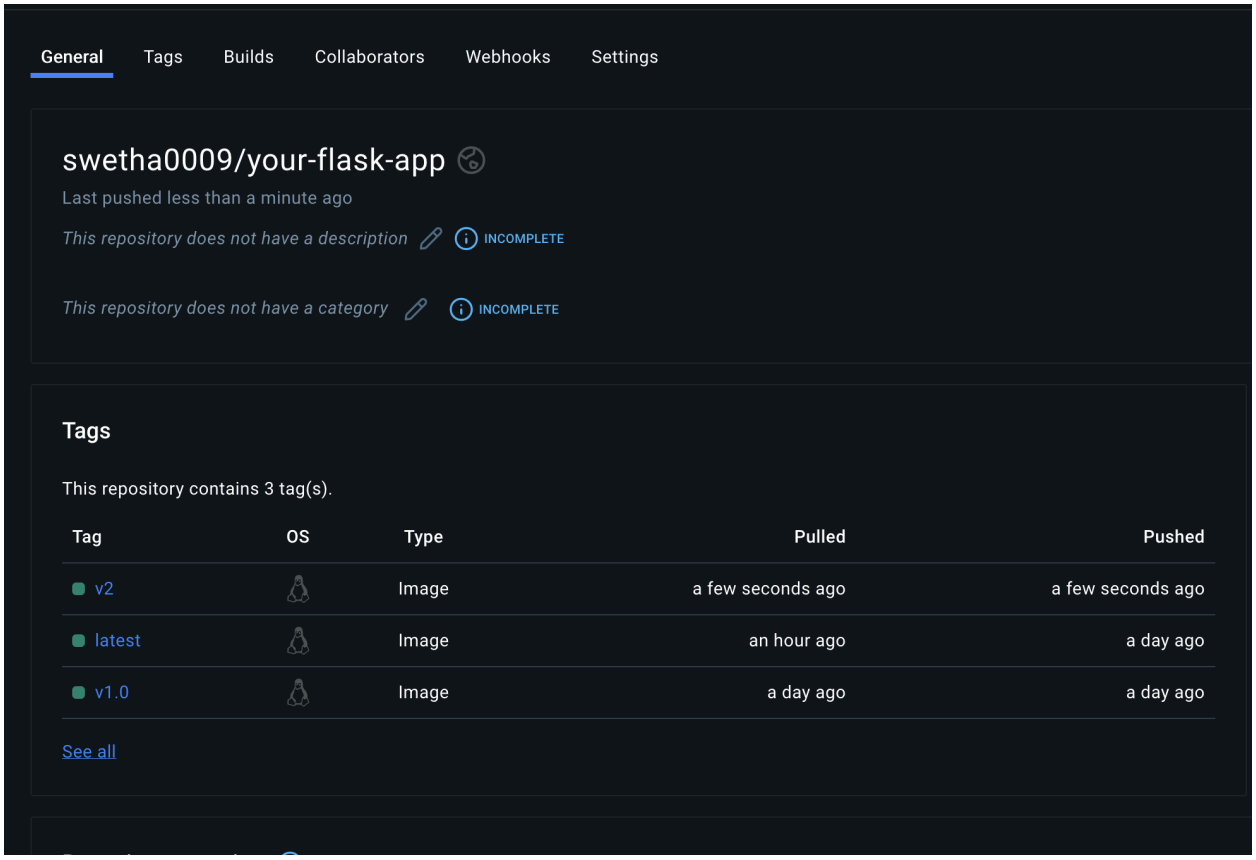
Step 3: We applied the Deployment_Rolling.yaml file and built the docker for the v2 of the application.

```
swethajagadeesan@Swethas-Air Code % kubectl apply -f Deployment_Rolling.yaml
deployment.apps/flask-app configured
swethajagadeesan@Swethas-Air Code % docker build -t swetha0009/your-flask-app:v2 .
[+] Building 4.3s (10/10) FINISHED                                                                 docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                               0.0s
 => => transferring dockerfile: 752B                                                                               0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                                 0.6s
 => [auth] library/python:pull token for registry-1.docker.io                                                      0.0s
 => [internal] load .dockerignore                                                                                  0.0s
 => => transferring context: 2B                                                                                    0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe   0.0s
 => => resolve docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe   0.0s
 => [internal] load build context                                                                                  0.0s
 => => transferring context: 6.79kB                                                                                0.0s
 => CACHED [2/4] WORKDIR /app                                                                                      0.0s
 => [3/4] COPY . /app                                                                                              0.0s
 => [4/4] RUN pip install -r requirements.txt                                                                      2.9s
 => exporting to image                                                                                             0.6s
 => => exporting layers                                                                                            0.4s
 => => exporting manifest sha256:d4576a97eba3aa8392c4d200dff789698eb79771a50f0b531107fd20be6e9e4b                  0.0s
 => => exporting config sha256:cd9aa55f9ed103ef514b9482e6167efb73cf66904840fc43953d039079609fba                    0.0s
 => => exporting attestation manifest sha256:afcc8f7677095ad8513e4b05d3e7c1d8e5a0e5e782df244731242ae1212eb553      0.0s
 => => exporting manifest list sha256:20fec1409d98818cbf227ed519b92e3ca3669d4e85b04d33014de717ffa2d050             0.0s
 => => naming to docker.io/swetha0009/your-flask-app:v2                                                            0.0s
 => => unpacking to docker.io/swetha0009/your-flask-app:v2                                                         0.1s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/80fn3i2l1b6h6qfixkwah5kpc

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
swethajagadeesan@Swethas-Air Code % docker images
REPOSITORY                         TAG                IMAGE ID       CREATED          SIZE
swetha0009/your-flask-app          v2                 20fec1409d98   10 seconds ago   242MB
swetha0009/your-flask-app          latest             2218d5ba7180   17 hours ago     242MB
code-flask-app                     latest             0e1797004a6e   24 hours ago     242MB
swetha0009/your-flask-app          v1.0               263edbe59d4b   27 hours ago     242MB
cloud_assignment_3_files-flask-app latest             8f131f8ec743   6 days ago       242MB
your-flask-app                     latest             7450dde2431d   6 days ago       241MB
swethajagadeesan/your-flask-app    latest             7450dde2431d   6 days ago       241MB
cloud_assignment_3_files-web       latest             cbc4fa84ae97   12 days ago      241MB
mongo                              latest             9342a9279a98   4 weeks ago      1.12GB
moby/buildkit                      buildx-stable-1    bc1fe18224db   8 weeks ago      298MB
gcr.io/k8s-minikube/kicbase        v0.0.45            7c93b02056db   2 months ago     1.67GB
gcr.io/k8s-minikube/kicbase        <none>             81df28859520   2 months ago     1.67GB
mongo                              4.4                52c42cbab240   8 months ago     592MB
```

Step 4: The docker v2 is then pushed into the docker hub

```
swethajagadeesan@Swethas-Air Code % docker push swetha0009/your-flask-app:v2
The push refers to repository [docker.io/swetha0009/your-flask-app]
3319b7835193: Pushed
83d624c4be2d: Layer already exists
bb113063b7ce: Layer already exists
9c0c60c92dcd: Layer already exists
ec88d2adb915: Pushed
e9825a162d70: Layer already exists
fe908d0d9cc1: Layer already exists
ac873dee40ed: Pushed
v2: digest: sha256:20fec1409d98818cbf227ed519b92e3ca3669d4e85b04d33014de717ffa2d050 size: 856
```

The version v2 of the application we just created, is successfully pushed into the docker-hub.

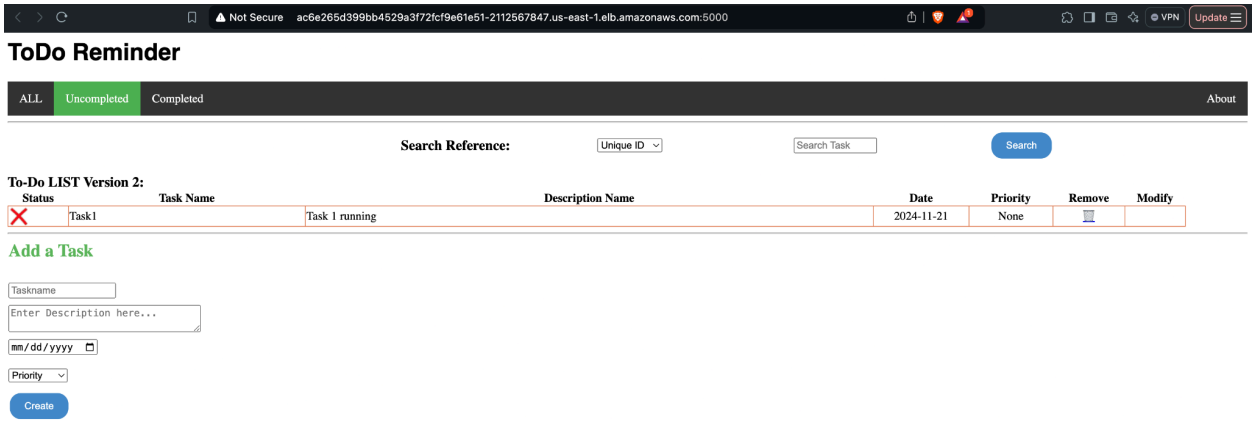


Step 5: Trigger the Rolling Update by Updating the Deployment With the New Docker Image Version:

Applying the updated deployment configuration to your cluster using ‘kubectl apply’ And then checking the rollout status

```
swethajagadeesan@Swethas-Air Code % kubectl apply -f Deployment_Rolling.yaml
deployment.apps/flask-app unchanged
swethajagadeesan@Swethas-Air Code % kubectl rollout status deployment/flask-app
deployment "flask-app" successfully rolled out
swethajagadeesan@Swethas-Air Code % kubectl describe deployment flask-app
Name:                   flask-app
Namespace:              default
CreationTimestamp:      Wed, 06 Nov 2024 19:31:00 -0500
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 3
Selector:               app=flask
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:  app=flask
  Containers:
   flask-container:
    Image:      swetha0009/your-flask-app:v2
    Port:       5000/TCP
    Host Port:  0/TCP
    Environment:
      MONGO_HOST:  mongo-db-service
      MONGO_PORT:  27017
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  flask-app-596b784d84 (0/0 replicas created), flask-app-85f645fdd5 (0/0 replicas created)
NewReplicaSet:   flask-app-68db4cc4d (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  27m    deployment-controller  Scaled up replica set flask-app-85f645fdd5 to 1
  Normal  ScalingReplicaSet  27m    deployment-controller  Scaled down replica set flask-app-596b784d84 to 2 from 3
  Normal  ScalingReplicaSet  27m    deployment-controller  Scaled up replica set flask-app-85f645fdd5 to 2 from 1
  Normal  ScalingReplicaSet  21m    deployment-controller  Scaled down replica set flask-app-596b784d84 to 1 from 2
  Normal  ScalingReplicaSet  21m    deployment-controller  Scaled up replica set flask-app-85f645fdd5 to 3 from 2
  Normal  ScalingReplicaSet  21m    deployment-controller  Scaled down replica set flask-app-596b784d84 to 0 from 1
  Normal  ScalingReplicaSet  3m30s  deployment-controller  Scaled up replica set flask-app-68db4cc4d to 1
  Normal  ScalingReplicaSet  3m30s  deployment-controller  Scaled down replica set flask-app-85f645fdd5 to 2 from 3
  Normal  ScalingReplicaSet  3m30s  deployment-controller  Scaled up replica set flask-app-68db4cc4d to 2 from 1
  Normal  ScalingReplicaSet  3m29s  deployment-controller  Scaled down replica set flask-app-85f645fdd5 to 1 from 2
  Normal  ScalingReplicaSet  3m29s  deployment-controller  Scaled up replica set flask-app-68db4cc4d to 3 from 2
  Normal  ScalingReplicaSet  3m28s  deployment-controller  Scaled down replica set flask-app-85f645fdd5 to 0 from 1
swethajagadeesan@Swethas-Air Code % kubectl get services
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)          AGE
flask-service   LoadBalancer   10.100.125.21   ac6e265d399bb4529a3f72fcf9e61e51-2112567847.us-east-1.elb.amazonaws.com   5000:30978/TCP   70m
kubernetes      ClusterIP      10.100.0.1      <none>                                                                    443/TCP          120m
mongo-service   ClusterIP      10.100.25.221   <none>                                                                    27017/TCP        70m
swethajagadeesan@Swethas-Air Code %
```

Testing the updated application: Now the v2 version of the application is running



Part 7: Health Monitoring:

Objective: Implement health checks to monitor application status and facilitate automatic recovery from failures.

Steps for Implementing Health Checks

1. **Update the Flask Application for Health Endpoints:**
- **Add a /healthz endpoint in your app.py to act as the health check route.**

app.py :

```python
@app.route("/healthz")
def healthz():
    if failure_mode:
        return "Service Unavailable", 503
    return "OK", 200

@app.route("/ready")
def ready():
    # Optionally check for readiness (e.g., database connections, etc.)
    return 'OK', 200
```

Deployment_Health.py: Added Liveness and Readiness Probes in Deployment Configuration:

Deployment_Health

Deployment_Health › No Selection

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
  selector:
    matchLabels:
      app: flask
  template:
    metadata:
      labels:
        app: flask
    spec:
      containers:
      - name: flask-container
        image: swetha0009/your-flask-app:v4
        ports:
        - containerPort: 5000
        env:
        - name: MONGO_URI
          value: "mongodb://mongo-service:27017/camp2016"
        - name: FAILURE_MODE
          value: "false"  # Set this to trigger failure mode during testing if needed
        livenessProbe:
          httpGet:
            path: /healthz
            port: 5000
          initialDelaySeconds: 10
          periodSeconds: 5
        readinessProbe:
          httpGet:
            path: /healthz
            port: 5000
          initialDelaySeconds: 5
          periodSeconds: 5
```

Added logic that will make the application/pod fail:

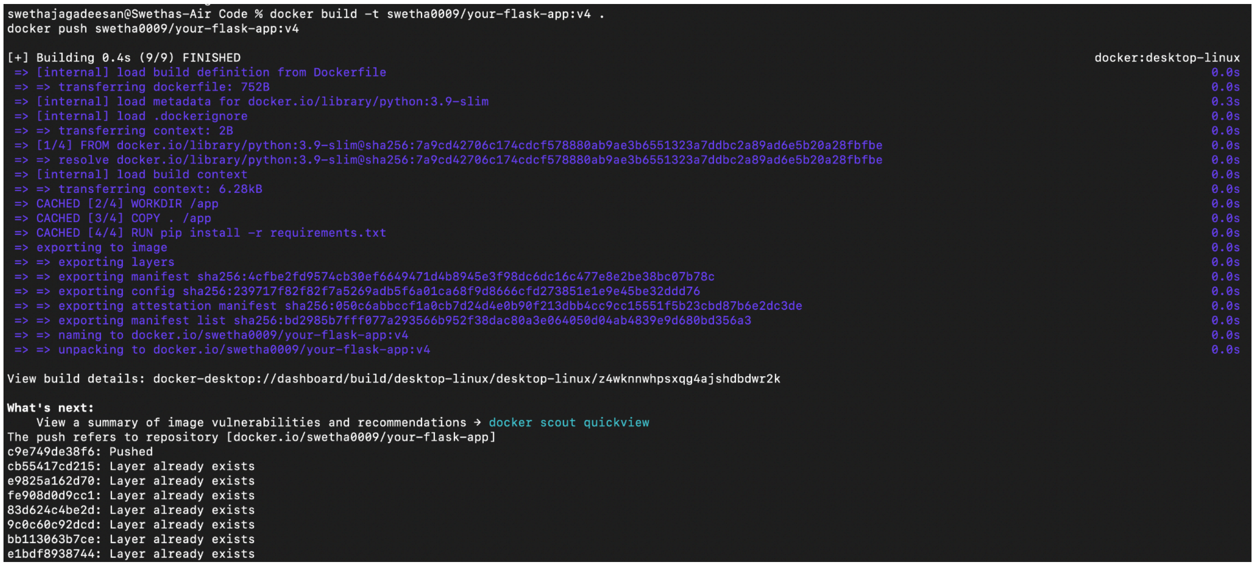
```
- name: FAILURE_MODE
  value: "true"  # Set this to trigger failure mode during testing if needed
```

app.py

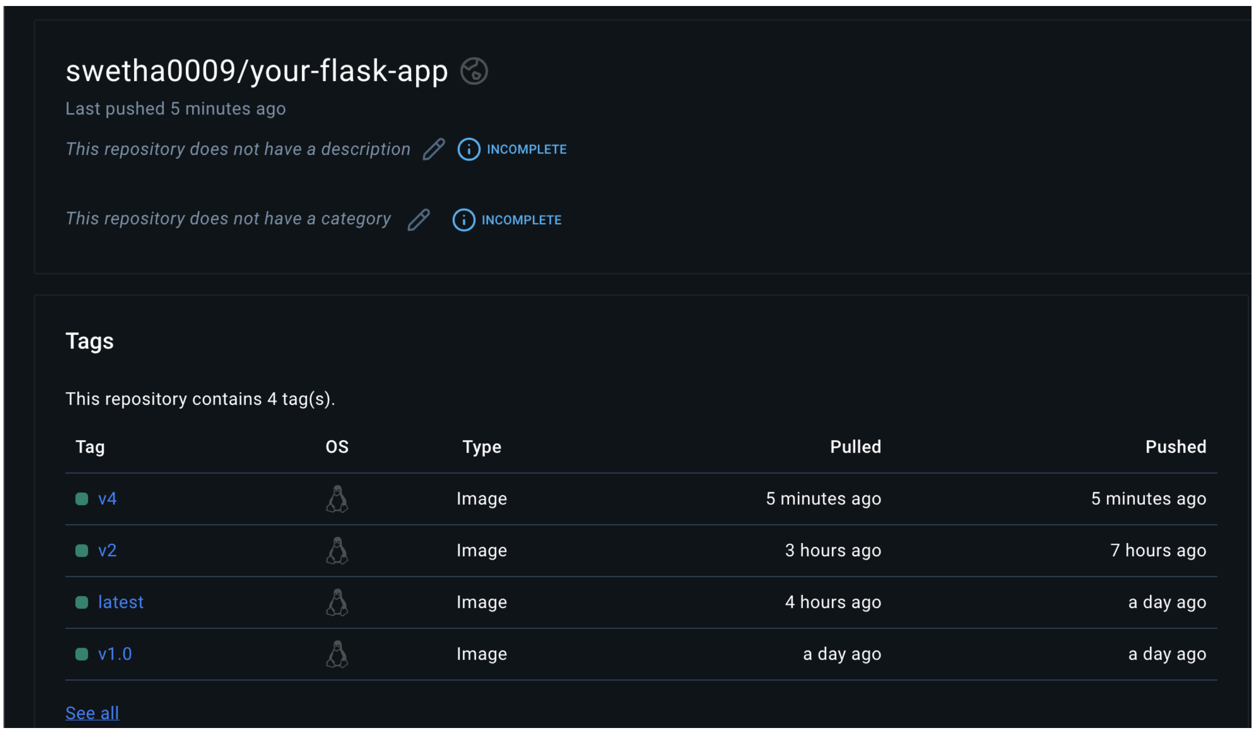
app > No Selection

```python
from flask import Flask, render_template, request, redirect, url_for
from pymongo import MongoClient
from bson.objectid import ObjectId
from bson.errors import InvalidId
import os

# Set up failure mode
failure_mode = os.environ.get('FAILURE_MODE', 'false').lower() == "true"

# MongoDB connection
mongodb_uri = os.environ.get('MONGO_URI', 'mongodb://mongo-service:27017/camp2016')
client = MongoClient(mongodb_uri)
db = client.get_default_database()
todos = db.todo

app = Flask(__name__)
title = "TODO with Flask"
heading = "ToDo Reminder"

@app.route("/healthz")
def healthz():
    if failure_mode:
        return "Service Unavailable", 503
    return "OK", 200

@app.route("/ready")
def ready():
    # Optionally check for readiness (e.g., database connections, etc.)
    return 'OK', 200

def redirect_url():
    return request.args.get('next') or \
        request.referrer or \
        url_for('index')
```

Pushing the image : (Without error)



Docker-hub: Where the v4 image is pushed:



The pods running, without any error:

```
swethajagadeesan@Swethas-Air Code % kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
flask-app-64db9ccf89-77lm7    1/1     Running   0          16s
flask-app-64db9ccf89-mqpp7    1/1     Running   0          16s
flask-app-64db9ccf89-ndc7b    1/1     Running   0          10s
flask-rc-4pswc                1/1     Running   0          4h24m
flask-rc-6whqn                1/1     Running   0          4h24m
flask-rc-nwvch                1/1     Running   0          4h24m
flask-rc-r5qt6                1/1     Running   0          4h24m
mongo-db-74f6f8bd65-88bvs     1/1     Running   0          8h
```

The pods restarting when the Deployment_Health file is applied with errors introduced:

```
swethajagadeesan@Swethas-Air Code % kubectl get pods
NAME                          READY   STATUS    RESTARTS      AGE
flask-app-55d8b786d8-svdmf    0/1     Running   2 (18s ago)   68s
flask-app-55d8b786d8-xjbs4    0/1     Running   2 (17s ago)   68s
flask-app-64db9ccf89-77lm7    1/1     Running   0             6m10s
flask-app-64db9ccf89-mqpp7    1/1     Running   0             6m10s
flask-rc-4pswc                1/1     Running   0             4h30m
flask-rc-6whqn                1/1     Running   0             4h30m
flask-rc-nwvch                1/1     Running   0             4h30m
flask-rc-r5qt6                1/1     Running   0             4h30m
mongo-db-74f6f8bd65-88bvs     1/1     Running   0             8h
swethajagadeesan@Swethas-Air Code % kubectl get pods
```

Observation:

The pods that are handled by deployment (Not by replication controller) are failing as the image pushed is causing it to do so.

But as they fail, the probe fails and so the system is trying to restart the pods.

Monitoring and Response:

- Monitored pod health using `kubectl describe pod`.

```
swethajagadeesan@Swethas-Air Code % kubectl describe pod flask-app-55d8b786d8-f7665
Name:             flask-app-55d8b786d8-f7665
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-17-88.ec2.internal/172.31.17.88
Start Time:       Thu, 07 Nov 2024 04:13:50 -0500
Labels:           app=flask
                  pod-template-hash=55d8b786d8
Annotations:      <none>
Status:           Running
IP:               172.31.22.206
IPs:
  IP:           172.31.22.206
Controlled By:  ReplicaSet/flask-app-55d8b786d8
Containers:
  flask-container:
    Container ID:   containerd://fa306bf6d011ee4fd8d610c0e02b0d9cc10b4b9f51627bfbc90f7efe17c29423
    Image:          swetha0009/your-flask-app:v4
    Image ID:       docker.io/swetha0009/your-flask-app@sha256:bd2985b7fff077a293566b952f38dac80a3e064050d04ab4839e9d680bd356a3
    Port:           5000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 07 Nov 2024 04:15:31 -0500
    Last State:     Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Thu, 07 Nov 2024 04:15:06 -0500
      Finished:     Thu, 07 Nov 2024 04:15:31 -0500
    Ready:          False
    Restart Count:  4
    Liveness:       http-get http://:5000/healthz delay=10s timeout=1s period=5s #success=1 #failure=3
    Readiness:      http-get http://:5000/healthz delay=5s timeout=1s period=5s #success=1 #failure=3
    Environment:
      MONGO_URI:     mongodb://mongo-service:27017/camp2016
      FAILURE_MODE:  true
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-td7nq (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       False
  ContainersReady             False
  PodScheduled                True
Volumes:
  kube-api-access-td7nq:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  118s                 default-scheduler  Successfully assigned default/flask-app-55d8b786d8-f7665 to ip-172-31-17-88.ec2.internal
  Normal   Pulled     117s                 kubelet            Successfully pulled image "swetha0009/your-flask-app:v4" in 115ms (115ms including waiting). Image size: 54121905 bytes.
  Normal   Killing    93s                  kubelet            Container flask-container failed liveness probe, will be restarted
  Normal   Pulling    92s (x2 over 117s)   kubelet            Pulling image "swetha0009/your-flask-app:v4"
  Normal   Created    92s (x2 over 117s)   kubelet            Created container flask-container
  Normal   Started    92s (x2 over 117s)   kubelet            Started container flask-container
  Normal   Pulled     92s                  kubelet            Successfully pulled image "swetha0009/your-flask-app:v4" in 109ms (109ms including waiting). Image size: 54121905 bytes.
  Warning  Unhealthy  68s (x10 over 108s)  kubelet            Readiness probe failed: HTTP probe failed with statuscode: 503
  Warning  Unhealthy  68s (x6 over 103s)   kubelet            Liveness probe failed: HTTP probe failed with statuscode: 503
swethajagadeesan@Swethas-Air Code %
```

Step 8: Alerting (Extra Credit)

Objective: Establish an alerting mechanism to be notified of application issues.

Step 1:

Install Prometheus: Used Helm to install Prometheus and Alertmanager in your Kubernetes cluster. Helm simplifies the installation:

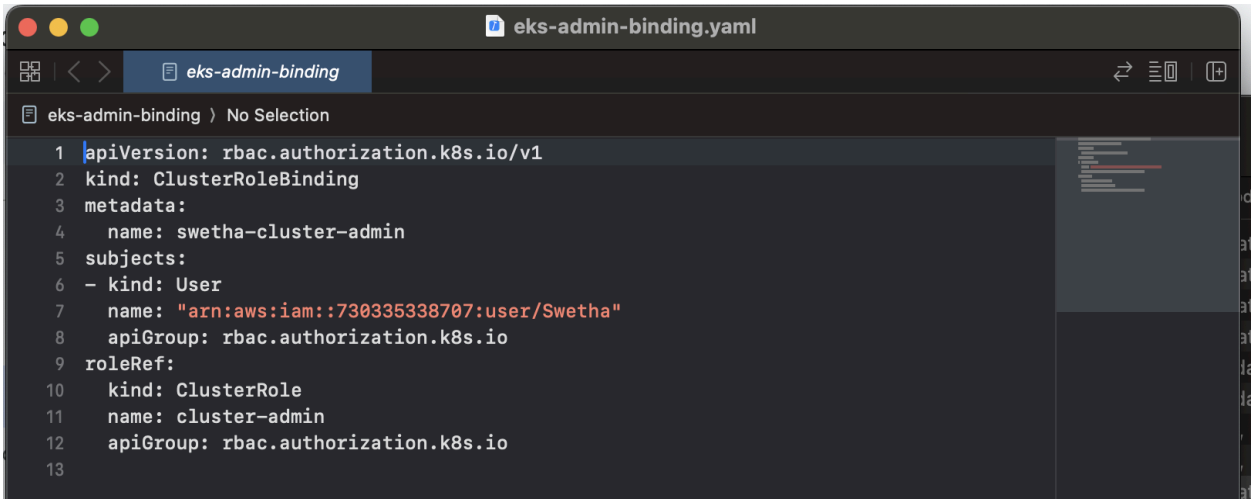
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts

helm repo update

helm install prometheus prometheus-community/kube-prometheus-stack

```
swethajagadeesan@Swethas-Air Code % nano eks-admin-binding.yaml
swethajagadeesan@Swethas-Air Code % kubectl apply -f eks-admin-binding.yaml

clusterrolebinding.rbac.authorization.k8s.io/swetha-cluster-admin created
```

eks-admin-binding.yaml:



```
swethajagadeesan@Swethas-Air Code % kubectl create namespace monitoring

namespace/monitoring created
swethajagadeesan@Swethas-Air Code % helm install prometheus prometheus-community/prometheus --namespace monitoring

NAME: prometheus
LAST DEPLOYED: Thu Nov  7 21:20:31 2024
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.monitoring.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace monitoring -l "app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace monitoring port-forward $POD_NAME 9090


The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from within your cluster:
prometheus-alertmanager.monitoring.svc.cluster.local


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace monitoring -l "app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=prometheus" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace monitoring port-forward $POD_NAME 9093
#################################################################################
######   WARNING: Pod Security Policy has been disabled by default since    #####
######            it deprecated after k8s 1.25+. use                        #####
######            (index .Values "prometheus-node-exporter" "rbac"          #####
###### .          "pspEnabled") with (index .Values                         #####
######            "prometheus-node-exporter" "rbac" "pspAnnotations")       #####
######            in case you still need it.                                #####
#################################################################################


The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-prometheus-pushgateway.monitoring.svc.cluster.local


Get the PushGateway URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace monitoring -l "app=prometheus-pushgateway,component=pushgateway" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace monitoring port-forward $POD_NAME 9091

For more information on running Prometheus, visit:
https://prometheus.io/
```

Step 2: Configure Prometheus Rules for Alerting

Create Alerting Rules: Create a PrometheusRule YAML file, like rule-file.yaml, to define custom alerting rules.

rule-file

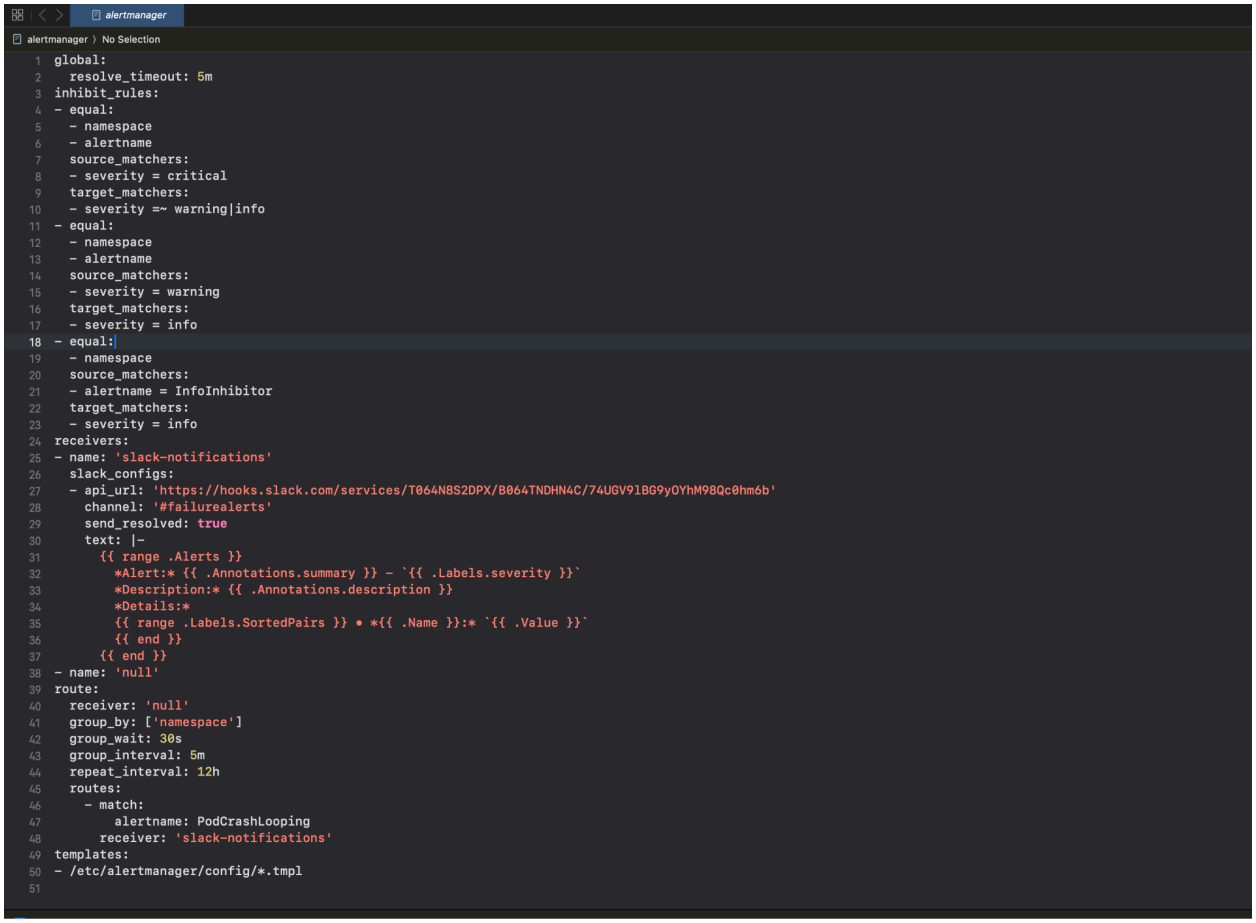
rule-file 〉 No Selection

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"monitoring.coreos.com/v1","kind":"PrometheusRule","metadata":{"annotations":{"meta.helm.sh/release-name":"my-pro","meta.helm
          .sh/release-namespace":"default","prometheus-operator-validated":"true"},"creationTimestamp":"2023-11-06T01:03:40Z","generation":2,"labels":{"app":"kube-prometheu
          .io/instance":"my-pro","app.kubernetes.io/managed-by":"Helm","app.kubernetes.io/part-of":"kube-prometheus-stack","app.kubernetes.io/version":"52.1.0","chart":"kub
          .0","heritage":"Helm","release":"my-pro"},"name":"my-pro-kube-prometheus-sta-general
          .rules","namespace":"default","resourceVersion":"1109747","uid":"9894e64d-3c15-4c2a-aede-5f8c0ae13238"},"spec":{"groups":[{"name":"example","rules":
          [{"alert":"PodCrashLooping","annotations":{"description":"Pod {{ $labels.namespace }}/{{ $labels.pod }} is crash looping.","summary":"Pod is crash
          looping"},"expr":"kube_pod_container_status_restarts_total{job=\"kube-state-metrics\"} \u003e
          5","for":"5m","labels":{"severity":"critical"}}]},{"name":"general.rules","rules":[{"alert":"TargetDown","annotations":{"description":"{{ printf \"%.4g\" $value }}
          $labels.service }} targets in {{ $labels.namespace }} namespace are down.","runbook_url":"https://runbooks.prometheus-operator.dev/runbooks/general/targetdown","su
          are unreachable."},"expr":"100 * (count(up == 0) BY (cluster, job, namespace, service) / count(up) BY (cluster, job, namespace, service)) \u003e
          10","for":"10m","labels":{"severity":"warning"}},{"alert":"Watchdog","annotations":{"description":"This is an alert meant to ensure that the entire alerting pipeli
          is always firing, therefore it should always be firing in Alertmanager\nand always fire against a receiver. There are integrations with various notification\nmecha
          when this alert is not firing. For example the\n\"DeadMansSnitch\" integration in PagerDuty.\n","runbook_url":"https://runbooks.prometheus-operator.dev/runbooks/ge
          alert that should always be firing to certify that Alertmanager is working
          properly."},"expr":"vector(1)","labels":{"severity":"none"}},{"alert":"InfoInhibitor","annotations":{"description":"This is an alert that is used to inhibit info a
          info-level alerts are sometimes very noisy, but they are relevant when combined with\nother alerts.\nThis alert fires whenever there's a severity=\"info\" alert, a
          alert with a\nseverity of 'warning' or 'critical' starts firing on the same namespace.\nThis alert should be routed to a null receiver and configured to inhibit al
          severity=\"info\".\n","runbook_url":"https://runbooks.prometheus-operator.dev/runbooks/general/infoinhibitor","summary":"Info-level alert inhibition."},"expr":"ALE
          unless on(namespace) ALERTS{alertname != \"InfoInhibitor\", severity =~ \"warning|critical\", alertstate=\"firing\"} == 1","labels":{"severity":"none"}}]}]}
    meta.helm.sh/release-name: my-pro
    meta.helm.sh/release-namespace: default
    prometheus-operator-validated: "true"
  creationTimestamp: "2023-11-06T01:03:40Z"
  generation: 3
  labels:
    app: kube-prometheus-stack
    app.kubernetes.io/instance: my-pro
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/part-of: kube-prometheus-stack
    app.kubernetes.io/version: 52.1.0
    chart: kube-prometheus-stack-52.1.0
    heritage: Helm
    release: my-pro
  name: my-pro-kube-prometheus-sta-general.rules
  namespace: default
  resourceVersion: "1457469"
  uid: 9894e64d-3c15-4c2a-aede-5f8c0ae13238
spec:
  groups:
  - name: example
    rules:
    - alert: PodCrashLooping
      annotations:
        description: Pod {{ $labels.namespace }}/{{ $labels.pod }} is crash looping.
        summary: Pod is crash looping
      expr: kube_pod_container_status_restarts_total{job="kube-state-metrics"} > 1
      for: 1m
      labels:
        severity: critical
```

```
swethajagadeesan@Swethas-Air Code % kubectl apply --server-side -f https://raw.githubusercontent.com/prometheus-operator/prometheus-operator/main/bundle.yaml

customresourcedefinition.apiextensions.k8s.io/alertmanagerconfigs.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/alertmanagers.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/podmonitors.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/probes.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/prometheusagents.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/prometheuses.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/prometheusrules.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/scrapeconfigs.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/servicemonitors.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/thanosrulers.monitoring.coreos.com serverside-applied
clusterrolebinding.rbac.authorization.k8s.io/prometheus-operator serverside-applied
clusterrole.rbac.authorization.k8s.io/prometheus-operator serverside-applied
deployment.apps/prometheus-operator serverside-applied
serviceaccount/prometheus-operator serverside-applied
service/prometheus-operator serverside-applied
swethajagadeesan@Swethas-Air Code % kubectl get crds | grep prometheusrules

prometheusrules.monitoring.coreos.com        2024-11-08T03:25:44Z
swethajagadeesan@Swethas-Air Code % kubectl apply -f rule-file.yaml

prometheusrule.monitoring.coreos.com/my-pro-kube-prometheus-sta-general.rules created
swethajagadeesan@Swethas-Air Code %
```

Step 3: Set Up Alertmanager for Notifications:

- Configured Alertmanager to forward alerts to a Slack channel using a webhook URL provided by Slack.



Step 4: Testing

The pods are trying to restart automatically after the deletion of the pods. This triggers alerts to slack.

```
swethajagadeesan@Swethas-Air Prometheus % kubectl delete pod prometheus-deployment-
pod "prometheus-deployment-75d8fdccc-q97ks" deleted
swethajagadeesan@Swethas-Air Prometheus % kubectl get pods -n default
NAME                                  READY   STATUS             RESTARTS
alertmanager-7f9c7b6c7d-dxqt9         1/1     Running            0
flask-app-5b74d58b8c-mwzwf            1/1     Running            0
flask-app-d4758d899-59sds             0/1     CrashLoopBackOff   12 (4m56s ago)
flask-app-d4758d899-jdlmj             0/1     CrashLoopBackOff   12 (4m59s ago)
flask-app-d4758d899-snhsb             0/1     CrashLoopBackOff   12 (4m50s ago)
mongo-db-74f6f8bd65-7hkk6             0/1     Pending            0
prometheus-deployment-75d8fdccc-87zlc   1/1     Running            0
```