Cloud Computing Assignment2

All commands below based on Ubuntu 22.04 LTS and AMD64 CPU

Part 1: Creating an Application:

1. Get App from this link: https://drive.google.com/drive/u/0/folders/19AZciSY1l2neLVfJUzqRG4a_mDstnaK_
2. Install or start MongoDB

```
# All the following commands will run at Linux
sudo systemctl start mongod
```

```
#or we can start with brew
brew services start mongodb-community@6.0
```

3. Run the application

```
python3 app.py
```

Test the Todo List APP

Part 2: Containerizing the Application on Docker:

1. Create a Dockerfile

```
touch Dockerfile
```

2. Create an Dockerfile based on TA's template (https://github.com/PrateekKumar1709/Docker-Demo/blob/main/cat-gif-app/single-container/Dockerfile)

```
# Use an official Python runtime as the base image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Upgrade pip and install the required packages
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt

# Make port 5050 available to the world outside this container
EXPOSE 5050

# Define environment variable
ENV NAME="World"

# Run app.py when the container launches
CMD ["python", "app.py"]
```

3. Build Docker

```
docker build -t flask-app .
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/app$ docker images
REPOSITORY                                  TAG       IMAGE ID       CREATED          SIZE
flask-app                                   latest    8dc1078b81aa   23 minutes ago   145MB
```

5. Write a docker compose (ref: https://hub.docker.com/_/mongo, https://github.com/docker/compose)

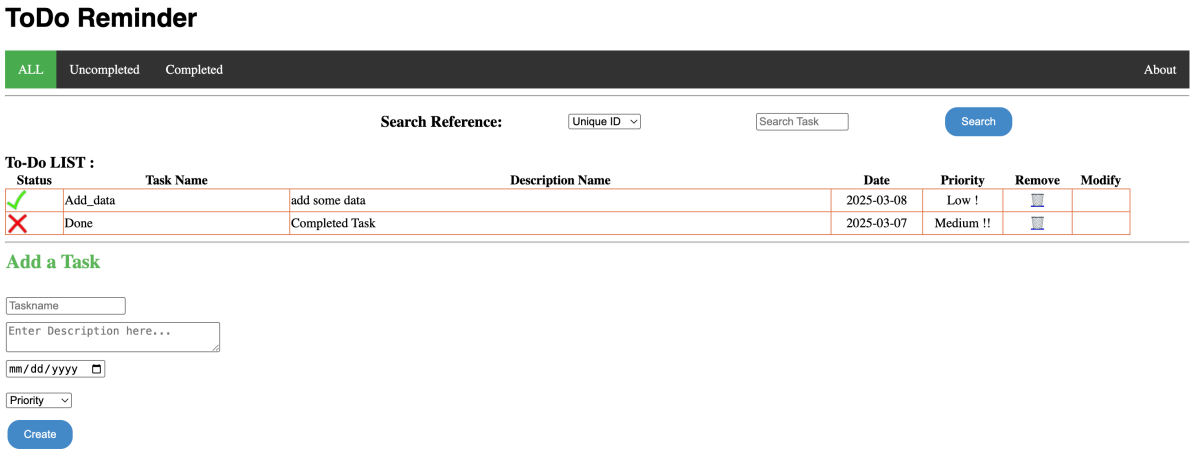
```
services:
  mongodb:
    image: "mongo:latest"
    ports:
```

```
      - "27017:27017"
    environment:
      - MONGO_INITDB_DATABASE=camp2016
    volumes:
      - mongo_data:/data/db

  flask_app:
    image: "flask-app:latest"  # Use pre-built image
    ports:
      - "5050:5050"
    depends_on:
      - mongodb
    environment:
      - MONGO_HOST=mongodb
      - MONGO_PORT=27017
volumes:
  mongo_data:
```

5. Start or stop services for the flask app and MongoDB container

```
docker compose up -d

docker compose down
```

```
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:11] "GET / HTTP/1.1" 200 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:11] "GET /static/assets/style.css HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:11] "GET /static/assets/emoji.js HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:11] "GET /static/assets/emoji.css HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:11] "GET /static/images/no.png HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:11] "GET /static/assets/twemoji.min.js HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:12] "GET /static/images/no.png HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:14] "GET /list HTTP/1.1" 200 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:14] "GET /static/assets/emoji.js HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:15] "GET /static/assets/style.css HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:15] "GET /static/images/yes.png HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:15] "GET /static/assets/emoji.css HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:15] "GET /static/assets/twemoji.min.js HTTP/1.1" 304 -
flask_app-1  | 172.80.1.1 - - [13/Mar/2025 01:31:15] "GET /static/images/no.png HTTP/1.1" 304 -
Gracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✔ Container app-flask_app-1  Stopped                                    0.3s
 ✔ Container app-mongodb-1    Stopped                                    0.2s
yuwei@yuwei-SER:~/Documents/CC-A2/app$ docker compose down
[+] Running 3/3
 ✔ Container app-flask_app-1  Removed                                    0.0s
 ✔ Container app-mongodb-1    Removed                                    0.0s
 ✔ Network app_default        Removed                                    0.2s
```



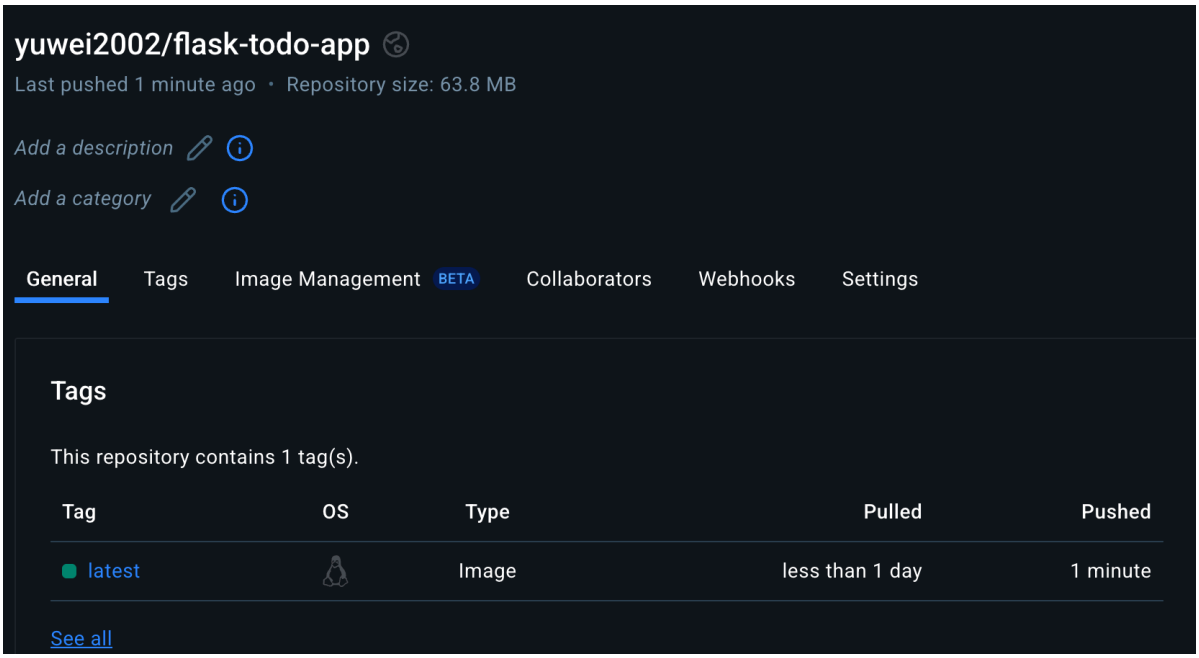
6. Tag that Docker image

```
docker tag app-flask_app yuwei2002/flask-todo-app:latest
```

7. Push the Docker image to Dockerhub

```
docker push yuwei2002/flask-todo-app:latest
```

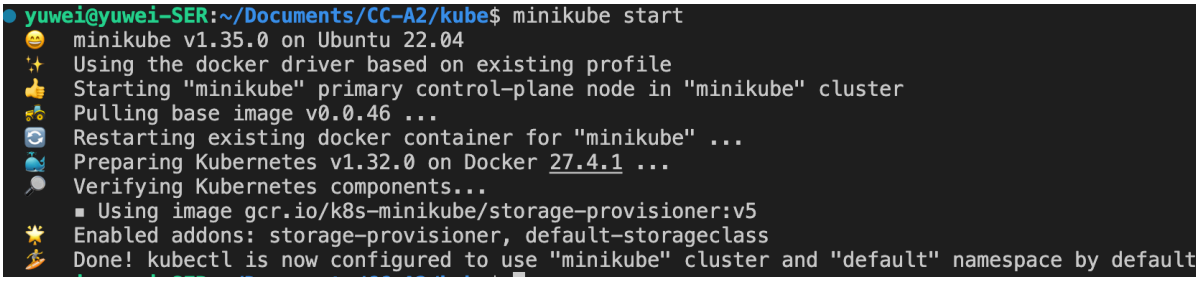
```
yuwei@yuwei-SER:~/Documents/CC-A2/app$ docker push yuwei2002/flask-todo-app:latest
The push refers to repository [docker.io/yuwei2002/flask-todo-app]
59c9ee8d6eba: Pushed
895c8499b912: Pushed
14eb7e6c0f1b: Layer already exists
70fff1cc441b: Layer already exists
62f5a620d172: Layer already exists
4a0e2c276a6a: Layer already exists
5f1ee22ffb5e: Layer already exists
latest: digest: sha256:ae7bce2b212fb56f0141cb428174a98887fff6e08b1c0d06e8cff4ae4c07daba size: 1786
```



Part 3: Deploying the Application on Minikube:

1. Start Minikube using the command-line interface:

```
minikube start
```



2. Create two pods: one for the flask app and one for the MongoDB to store data.

```
# Flask-app
# Service
apiVersion: v1
kind: Service
```

```yaml
metadata:
  name: flask-app
spec:
  selector:
    app: flask-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5050
  type: LoadBalancer
---
# Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
        - name: flask-app
          image: yuwei2002/flask-todo-app:latest
          ports:
            - containerPort: 5050
          env:
            - name: MONGO_HOST
              value: "mongodb"
            - name: MONGO_PORT
              value: "27017"
          resources:
```

```
          requests:
            cpu: "0.5"
            memory: "512Mi"
          limits:
            memory: "512Mi"
            cpu: "1"
        imagePullPolicy: Always
```

```
# MongoDB
# Persistent volume claim
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  storageClassName: standard # gp2 # AWS EBS storage
  resources:
    requests:
      storage: 1Gi
---
# Service
apiVersion: v1
kind: Service
metadata:
  name: mongodb
spec:
  selector:
    app: mongodb
  ports:
    - port: 27017
      targetPort: 27017
---
# Deployment
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb
spec:
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo:latest
          ports:
            - containerPort: 27017
          env:
            - name: MONGO_INITDB_DATABASE
              value: camp2016
          volumeMounts:
            - name: mongo-data
              mountPath: /data/db
          resources:
            requests:
              cpu: "0.3"
              memory: "512Mi"
            limits:
              memory: "512Mi"
              cpu: "0.5"
      volumes:
        - name: mongo-data
          persistentVolumeClaim:
            claimName: mongo-pvc
```

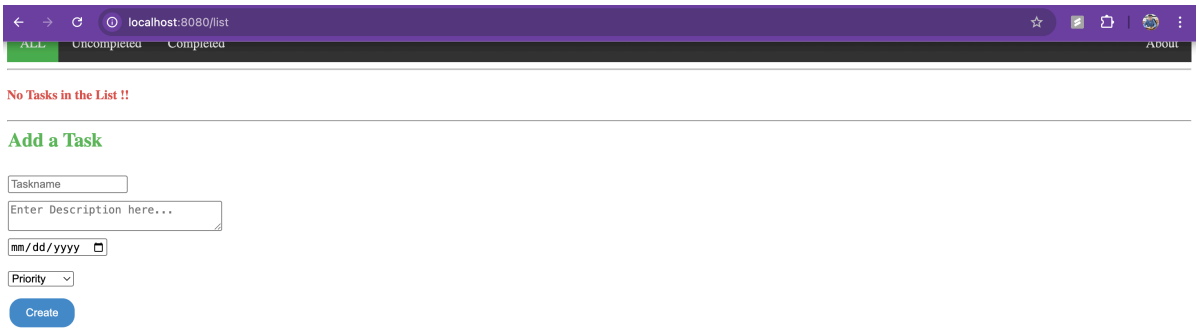
3. Apply these YAML using `kubectl`

```
kubectl apply -f app-cloud.yaml
kubectl apply -f mongo-cloud.yaml
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl apply -f app-cloud.yaml
service/flask-app unchanged
deployment.apps/flask-app created
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl apply -f mongo-cloud.yaml
persistentvolumeclaim/mongo-pvc unchanged
service/mongodb unchanged
deployment.apps/mongodb created
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
flask-app   1/1     1            1           9s
mongodb     1/1     1            1           6s
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
flask-app-85cd75cc5-c27kt    1/1     Running   0          12s
mongodb-54bcd6bb4-vcwtb      1/1     Running   0          9s
yuwei@yuwei-SER:~/Documents/CC-A2/kube$
```

4. Test the application

```
# Port Forwarding
kubectl port-forward service/flask-app 8080:80 -n default
```



5. Cleanup and stop

```
kubectl delete deployments --all
minikube stop
```

Part 4: Deploying the Application on AWS EKS:

Ref: https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html

1. Install AWS CLI

```
# Linux
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.z
unzip awscliv2.zip
sudo ./aws/install

#MacOS
brew install weaveworks/tap/eksctl
brew install awscli
```

2. Create credential file for cluster

```
aws configure
# Check identity
aws sts get-caller-identity
```

3. Setup eksctl

```
# for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`
ARCH=amd64
PLATFORM=$(uname -s)_$ARCH

curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$

# (Optional) Verify checksum
curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_che

tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz

sudo mv /tmp/eksctl /usr/local/bin
```

4. Create an AWS EKS cluster with an IAM using eksctl

```
# Create cluster
eksctl create cluster --name my-cluster --region us-east-2

# created IAM Open ID Connect provider
eksctl utils associate-iam-oidc-provider --region=us-east-2 --cluster=my-cluste
# Create IAM
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDri
    --approve
```

5. Install `aws-ebs-csi-driver` since we using AWS EBS storage

```
eksctl create addon --name aws-ebs-csi-driver --cluster my-cluster --region us
```

6. Configure the Kubernetes CLI (kubectl) to connect to the EKS cluster

```
Configure the Kubernetes CLI (kubectl) to connect to the EKS cluster
```

Check connection

```
kubectl get nodes -o wide
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get nodes -o wide
NAME                                          STATUS   ROLES    AGE     VERSION               INTERNAL-IP      EXTERNAL-IP
  OS-IMAGE         KERNEL-VERSION                  CONTAINER-RUNTIME
ip-192-168-3-45.us-east-2.compute.internal    Ready    <none>   3d23h   v1.30.9-eks-5d632ec   192.168.3.45     3.145.189.5
  Amazon Linux 2   5.10.234-225.895.amzn2.x86_64   containerd://1.7.25
ip-192-168-48-154.us-east-2.compute.internal  Ready    <none>   3d23h   v1.30.9-eks-5d632ec   192.168.48.154   3.137.142.241
  Amazon Linux 2   5.10.234-225.895.amzn2.x86_64   containerd://1.7.25
```

7. Deployment (based on previous YAML in Part 3)

   Step 1: Change the `storageClassName` in mongo-cloud.yaml from **standard** to **gp2** since we want to use the AWS EBS storage

   Step 2: Create a namespace for management

```
kubectl create namespace eks-todo-list-app
```

Step 3: Apply yaml

```
kubectl apply -f mongo-cloud.yaml -n eks-todo-list-app
kubectl apply -f app-cloud.yaml -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl apply -f mongo-cloud.yaml -n eks-todo-list-app
persistentvolumeclaim/mongo-pvc created
service/mongodb created
deployment.apps/mongodb created
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl apply -f app-cloud.yaml -n eks-todo-list-app
service/flask-app created
deployment.apps/flask-app created
```

Check If successful:

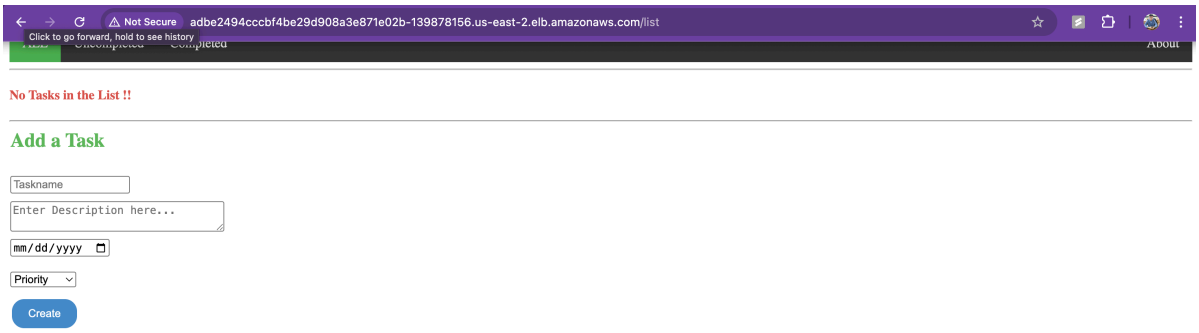
```
kubectl get pods -n eks-todo-list-app
kubectl get all -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods -n eks-todo-list-app
NAME                          READY   STATUS    RESTARTS   AGE
flask-app-755bc58598-77b2c    1/1     Running   0          34s
mongodb-6f84848985-6hf85      1/1     Running   0          38s
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get all -n eks-todo-list-app
NAME                                  READY   STATUS    RESTARTS   AGE
pod/flask-app-755bc58598-77b2c        1/1     Running   0          38s
pod/mongodb-6f84848985-6hf85          1/1     Running   0          42s

NAME                 TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PO
RT(S)        AGE
service/flask-app    LoadBalancer   10.100.105.37   adbe2494cccbf4be29d908a3e871e02b-139878156.us-east-2.elb.amazonaws.com   80
:32566/TCP   38s
service/mongodb      ClusterIP      10.100.196.43   <none>                                                                     27
017/TCP      43s

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/flask-app     1/1     1            1           38s
deployment.apps/mongodb       1/1     1            1           43s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/flask-app-755bc58598     1         1         1       38s
replicaset.apps/mongodb-6f84848985       1         1         1       43s
yuwei@yuwei-SER:~/Documents/CC-A2/kube$
```

8. Test the application

```
kubectl get service flask-app -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get service flask-app -n eks-todo-list-app
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)
     AGE
flask-app   LoadBalancer   10.100.105.37   adbe2494cccbf4be29d908a3e871e02b-139878156.us-east-2.elb.amazonaws.com   80:32566/T
CP   2m33s
```



Part 5: Replication controller feature:

1. Create an app-cloud.yaml to use ReplicationController instead of Deployment

```
# ReplicationController
apiVersion: v1
kind: ReplicationController
metadata:
  name: flask-app-rc
  namespace: eks-todo-list-app
spec:
  replicas: 5
  selector:
    app: flask-app-rc
  template:
    metadata:
      labels:
        app: flask-app-rc
    spec:
      containers:
        - name: flask-app
          image: yuwei2002/flask-todo-app:latest
          ports:
            - containerPort: 5050
          env:
            - name: MONGO_HOST
              value: "mongodb"
            - name: MONGO_PORT
```

```
            value: "27017"
          resources:
            requests:
              cpu: "0.3"
              memory: "256Mi"
            limits:
              cpu: "0.5"
              memory: "512Mi"
```

2. Create the replication controller :

```
kubectl apply -f app-rc.yaml -n eks-todo-list-app
```

verify that the specified number of replicas are created and running

```
kubectl get rc -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl apply -f app-rc.yaml -n eks-todo-list-app
replicationcontroller/flask-app-rc created
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get rc -n eks-todo-list-app
NAME           DESIRED   CURRENT   READY   AGE
flask-app-rc   5         5         3       13s
```

3. Test the replication controller:

   Step 1: Get pods name

```
kubectl get pods -l app=flask-app-rc -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods -l app=flask-app-rc -n eks-todo-li
st-app
NAME                READY   STATUS    RESTARTS   AGE
flask-app-rc-8bkwm  1/1     Running   0          12s
flask-app-rc-cdjhn  1/1     Running   0          12s
flask-app-rc-dcptc  1/1     Running   0          12s
flask-app-rc-l678n  1/1     Running   0          12s
flask-app-rc-mqldv  1/1     Running   0          12s
```

   Step 2: Delete a pod (flask-app-rc-52tzx)

```
kubectl delete pod flask-app-rc-8bkwm -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl delete pod flask-app-rc-8bkwm -n eks-todo-list-app
pod "flask-app-rc-8bkwm" deleted
```

Step 3: Watch if a new pod auto running:

```
kubectl get pods -l app=flask-app-rc -n eks-todo-list-app -w
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods -l app=flask-app-rc -n eks-todo-list-app -w
NAME                 READY   STATUS    RESTARTS   AGE
flask-app-rc-cdjhn   1/1     Running   0          65s
flask-app-rc-dcptc   1/1     Running   0          65s
flask-app-rc-jlmtf   1/1     Running   0          24s
flask-app-rc-l678n   1/1     Running   0          65s
flask-app-rc-mqldv   1/1     Running   0          65s
```

4. Update number of replicas and verify

   Step 1: Change `replicas` from **5** to **3**

   Step 2: Apply YAML:

```
kubectl apply -f app-rc.yaml -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl apply -f app-rc.yaml -n eks-todo-list-app
replicationcontroller/flask-app-rc configured
```

   Step 3: Verify:

```
kubectl get rc -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get rc -n eks-todo-list-app
NAME           DESIRED   CURRENT   READY   AGE
flask-app-rc   3         3         3       3m4s
```

Part 6: Rolling update strategy:

1. Since replication controller do not suppport update strategy, we need back to Deployment

```
# Flask-app
# Service
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-app
spec:
  selector:
    app: flask-app-rc
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5050
  type: LoadBalancer
---
# Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app-deployment
  namespace: eks-todo-list-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: flask-app-rc
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: flask-app-rc
    spec:
      containers:
        - name: flask-app
          image: yuwei2002/flask-todo-app:v2 # latest
```

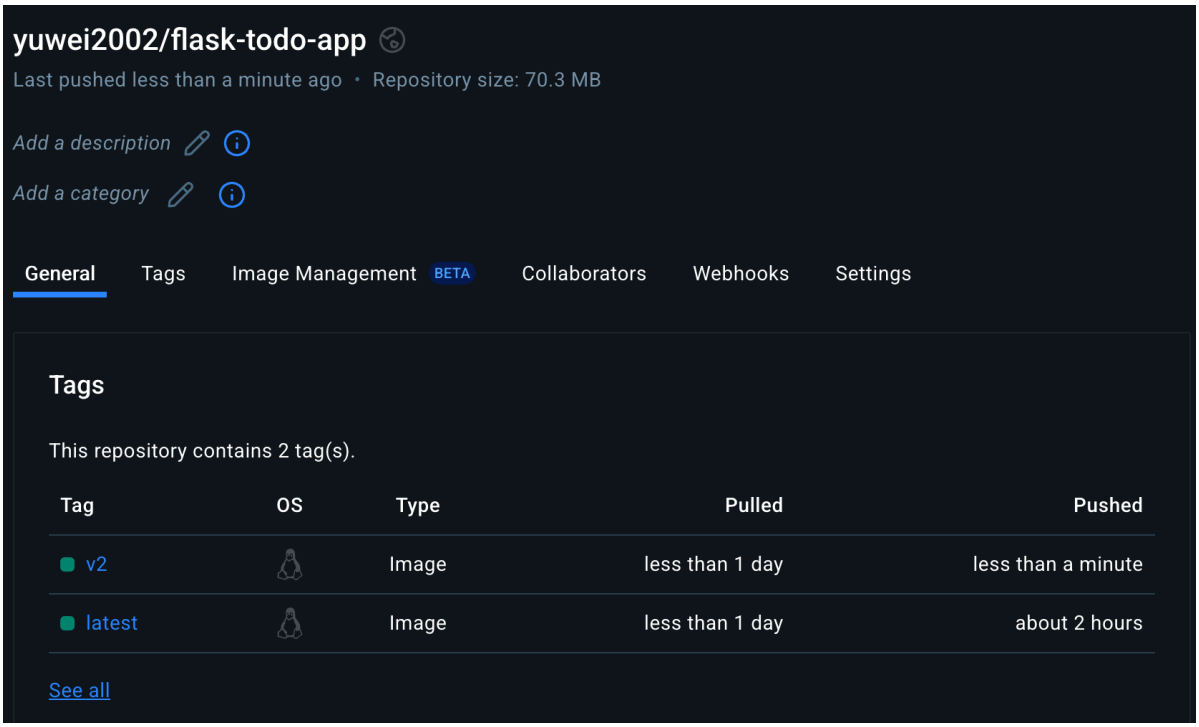
```
        ports:
          - containerPort: 5050
        env:
          - name: MONGO_HOST
            value: "mongodb"
          - name: MONGO_PORT
            value: "27017"
        resources:
          requests:
            cpu: "0.3"
            memory: "256Mi"
          limits:
            cpu: "0.5"
            memory: "512Mi"
```

2. Apply YAML

```
kubectl apply -f app-cloud.yaml -n eks-todo-list-app
```

3. Update the Docker image for the deployment to a new version

```
docker build -t yuwei2002/flask-todo-app:v2 .
docker push yuwei2002/flask-todo-app:v2
```

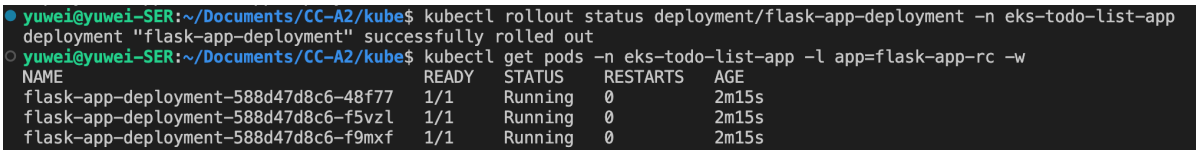


4. Update YAML to use new docker: Change tag in image
5. Apply YAML

```
kubectl apply -f kube/app-cloud.yaml -n eks-todo-list-app
```

6. Monitor the rolling update progress

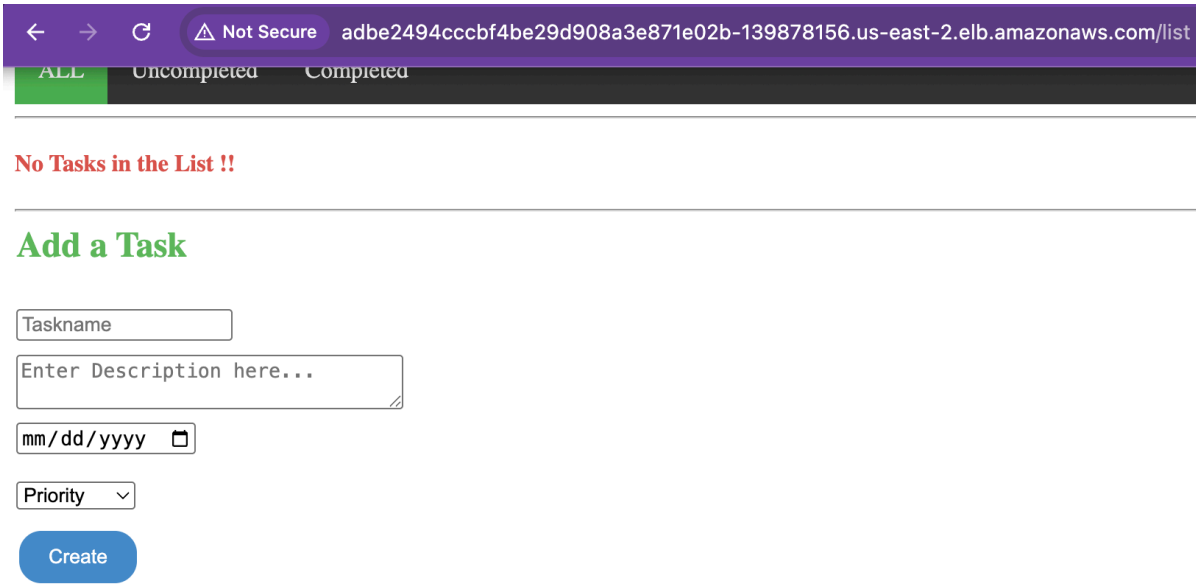
```
kubectl rollout status deployment/flask-app-deployment -n eks-todo-list-app
# Watch pods being updated
kubectl get pods -n eks-todo-list-app -l app=flask-app-rc -w
```



7. Test the appliction:

```
kubectl get service flask-app -n eks-todo-list-app
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get service flask-app -n eks-todo-list-app
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
     AGE
flask-app   LoadBalancer   10.100.105.37   adbe2494cccbf4be29d908a3e871e02b-139878156.us-east-2.elb.amazonaws.com   80:32566/T
CP   57m
```



7. Ensure application using New Docker image version

```
● yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl describe deployment flask-app-deploym
  ent -n eks-todo-list-app
  Name:                   flask-app-deployment
  Namespace:              eks-todo-list-app
  CreationTimestamp:      Wed, 12 Mar 2025 23:29:25 -0400
  Labels:                 <none>
  Annotations:            deployment.kubernetes.io/revision: 1
  Selector:               app=flask-app-rc
  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-app-rc
    Containers:
     flask-app:
      Image:      yuwei2002/flask-todo-app:v2
      Port:       5050/TCP
      Host Port:  0/TCP
      Limits:
        cpu:     500m
        memory:  512Mi
      Requests:
        cpu:     300m
        memory:  256Mi
      Environment:
        MONGO_HOST:  mongodb
        MONGO_PORT:  27017
      Mounts:        <none>
    Volumes:         <none>
    Node-Selectors:  <none>
    Tolerations:     <none>
  Conditions:
    Type           Status  Reason
    ----           ------  ------
    Available      True    MinimumReplicasAvailable
    Progressing    True    NewReplicaSetAvailable
```

Part 7: Health monitoring:

1. Add health and ready for probe in app.py

```
from flask import Flask, render_template,request,redirect,url_for # For flask imple
from pymongo import MongoClient # Database connector
from bson.objectid import ObjectId # For ObjectId to work
from bson.errors import InvalidId # For catching InvalidId exception for ObjectId
import os
import threading # Add threading for timer functionality

mongodb_host = os.environ.get('MONGO_HOST', 'localhost')
mongodb_port = int(os.environ.get('MONGO_PORT', '27017'))
client = MongoClient(mongodb_host, mongodb_port)    #Configure the connectic
```

```
db = client.camp2016    #Select the database
todos = db.todo #Select the collection

app = Flask(__name__)
title = "TODO with Flask(V2)"
heading = "ToDo Reminder"
#modify=ObjectId()

# Global variables to control health and readiness status
app_healthy = True
app_ready = True
ready_timer = None # Timer to track auto-reset

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

@app.route("/list")
def lists ():
    #Display the all Tasks
    todos_l = todos.find()
    a1="active"
    return render_template('index.html',a1=a1,todos=todos_l,t=title,h=heading)

@app.route("/")
@app.route("/uncompleted")
def tasks ():
    #Display the Uncompleted Tasks
    todos_l = todos.find({"done":"no"})
    a2="active"
    return render_template('index.html',a2=a2,todos=todos_l,t=title,h=heading)


@app.route("/completed")
def completed ():
    #Display the Completed Tasks
```

```
    todos_l = todos.find({"done":"yes"})
    a3="active"
    return render_template('index.html',a3=a3,todos=todos_l,t=title,h=heading)

@app.route("/done")
def done ():
    #Done-or-not ICON
    id=request.values.get("_id")
    task=todos.find({"_id":ObjectId(id)})
    if(task[0]["done"]=="yes"):
        todos.update_one({"_id":ObjectId(id)}, {"$set": {"done":"no"}})
    else:
        todos.update_one({"_id":ObjectId(id)}, {"$set": {"done":"yes"}})
    redir=redirect_url()    # Re-directed URL i.e. PREVIOUS URL from where it cam

#  if(str(redir)=="http://localhost:5000/search"):
#      redir+="?key="+id+"&refer="+refer

    return redirect(redir)

#@app.route("/add")
#def add():
#  return render_template('add.html',h=heading,t=title)

@app.route("/action", methods=['POST'])
def action ():
    #Adding a Task
    name=request.values.get("name")
    desc=request.values.get("desc")
    date=request.values.get("date")
    pr=request.values.get("pr")
    todos.insert_one({ "name":name, "desc":desc, "date":date, "pr":pr, "done":"no
    return redirect("/list")

@app.route("/remove")
def remove ():
    #Deleting a Task with various references
```

```
    key=request.values.get("_id")
    todos.delete_one({"_id":ObjectId(key)})
    return redirect("/")

@app.route("/update")
def update ():
    id=request.values.get("_id")
    task=todos.find({"_id":ObjectId(id)})
    return render_template('update.html',tasks=task,h=heading,t=title)

@app.route("/action3", methods=['POST'])
def action3 ():
    #Updating a Task with various references
    name=request.values.get("name")
    desc=request.values.get("desc")
    date=request.values.get("date")
    pr=request.values.get("pr")
    id=request.values.get("_id")
    todos.update_one({"_id":ObjectId(id)}, {'$set':{ "name":name, "desc":desc, "d
    return redirect("/")

@app.route("/search", methods=['GET'])
def search():
    #Searching a Task with various references

    key=request.values.get("key")
    refer=request.values.get("refer")
    if(refer=="id"):
        try:
            todos_l = todos.find({refer:ObjectId(key)})
            if not todos_l:
                return render_template('index.html',a2=a2,todos=todos_l,t=title,h=heac
        except InvalidId as err:
            pass
            return render_template('index.html',a2=a2,todos=todos_l,t=title,h=headin
    else:
        todos_l = todos.find({refer:key})
```

```
    return render_template('searchlist.html',todos=todos_l,t=title,h=heading)

@app.route("/about")
def about():
    return render_template('credits.html',t=title,h=heading)

# livenessProbe
@app.route("/health")
def health():
    if app_healthy:
        return {"status": "healthy"}, 200
    else:
        return {"status": "unhealthy"}, 500
# readinessProbe
@app.route("/ready")
def ready():
    if not app_ready:
        return {"status": "not ready", "error": "Readiness manually disabled"}, 503
    try:
        client.server_info() # Check if MongoDB connection is still alive
        return {"status": "ready"}, 200
    except Exception as e:
        return {"status": "not ready", "error": str(e)}, 503

# reset timer for /ready
def reset_ready():
    global app_ready, ready_timer
    app_ready = True
    ready_timer = None
    print("Readiness automatically reset to True after 30 seconds")

# Test endpoints to simulate failures
@app.route("/toggle-health")
def toggle_health():
    global app_healthy
    app_healthy = not app_healthy
    status = "unhealthy" if not app_healthy else "healthy"
```

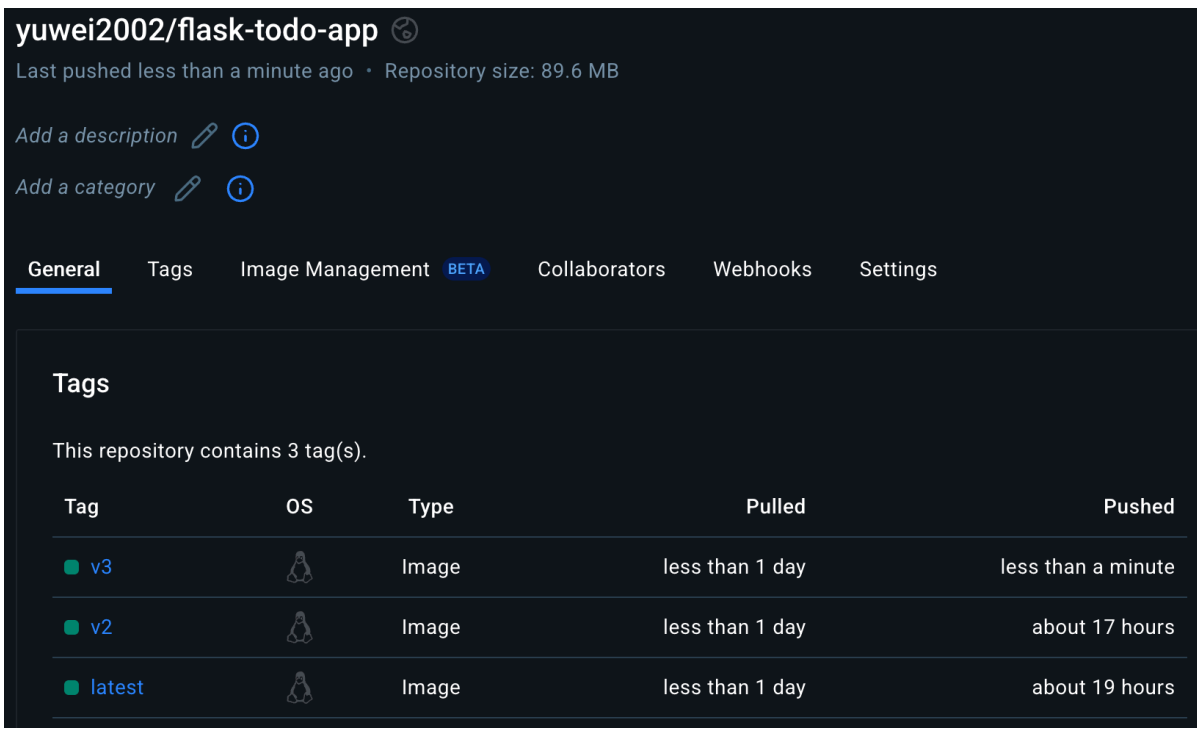
```
    return {"message": f"Health status toggled to {status}"}, 200

# Test endpoint to toggle readiness status
@app.route("/toggle-ready")
def toggle_ready():
    global app_ready, ready_timer
    app_ready = not app_ready
    status = "not ready" if not app_ready else "ready"
    # If toggled to not ready, set timer to reset after 30 seconds
    if not app_ready:
        if ready_timer:
            ready_timer.cancel()
        ready_timer = threading.Timer(30.0, reset_ready)
        ready_timer.daemon = True  # Make thread daemon so it won't block app sh
        ready_timer.start()
        return {"message": f"Readiness status toggled to {status}. Will reset in 30 s
    else:
        if ready_timer:
            ready_timer.cancel()
            ready_timer = None
        return {"message": f"Readiness status toggled to {status}"}, 200

if __name__ == "__main__":
    env = os.environ.get('FLASK_ENV', 'development')
    port = int(os.environ.get('PORT', 5050))
    debug = False if env == 'production' else True
    app.run(host='0.0.0.0', port=port, debug=debug)
    # Careful with the debug mode..
```

2. Build a new docker for app.py and push to Dockerhub

```
docker build -t yuwei2002/flask-todo-app:v3 .
docker push yuwei2002/flask-todo-app:v3
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/app$ docker build -t yuwei2002/flask-todo-app:v3 .
[+] Building 4.1s (10/10) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                              0.0s
 => => transferring dockerfile: 588B                                              0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                0.2s
 => [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:d1fd807555208707ec95b2    0.0s
 => [internal] load build context                                                 0.0s
 => => transferring context: 6.84kB                                               0.0s
 => CACHED [2/4] WORKDIR /app                                                     0.0s
 => [3/4] COPY . /app                                                             0.0s
 => [4/4] RUN pip install --no-cache-dir --upgrade pip &&     pip install --no    3.8s
 => exporting to image                                                            0.1s
 => => exporting layers                                                           0.1s
 => => writing image sha256:67a9c2b269053128a0f5a0bb640bac2c7846db72525ce2e742    0.0s
 => => naming to docker.io/yuwei2002/flask-todo-app:v3                            0.0s
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/app$ docker push yuwei2002/flask-todo-app:v3
The push refers to repository [docker.io/yuwei2002/flask-todo-app]
ce1de7ed81ec: Pushed
9f97b91bbb57: Pushed
14eb7e6c0f1b: Layer already exists
70fff1cc441b: Layer already exists
62f5a620d172: Layer already exists
4a0e2c276a6a: Layer already exists
5f1ee22ffb5e: Layer already exists
v3: digest: sha256:2f87d1b38fa00cdd23434ef6347d9e7cede759582e479f1b82957fb955ab4825 size: 1786
```



3. Add livenessProbe and readinessProbe to app-cloud.yaml

```yaml
# Flask-app
# Service
apiVersion: v1
kind: Service
metadata:
  name: flask-app
spec:
  selector:
    app: flask-app-rc
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5050
  type: LoadBalancer
---
# Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app-deployment
  namespace: eks-todo-list-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: flask-app-rc
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: flask-app-rc
    spec:
```

```yaml
      containers:
        - name: flask-app
          image: yuwei2002/flask-todo-app:v3 # latest
          ports:
            - containerPort: 5050
          env:
            - name: MONGO_HOST
              value: "mongodb"
            - name: MONGO_PORT
              value: "27017"
          resources:
            requests:
              cpu: "0.3"
              memory: "256Mi"
            limits:
              cpu: "0.5"
              memory: "512Mi"
          imagePullPolicy: Always
          livenessProbe: # Http Liveness Probe
            httpGet:
              path: /health
              port: 5050
            initialDelaySeconds: 30
            periodSeconds: 10
            timeoutSeconds: 5
            failureThreshold: 3
          readinessProbe: # Http Readiness Probe
            httpGet:
              path: /ready
              port: 5050
            initialDelaySeconds: 15
            periodSeconds: 5
            timeoutSeconds: 3
            successThreshold: 1
            failureThreshold: 3
```

4. Apply the updated YAML TODO

```
kubectl apply -f app-cloud-copy.yaml -n eks-todo-list-app
```

```
^Cyuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods -n eks-todo-list-app
NAME                                    READY   STATUS    RESTARTS   AGE
flask-app-deployment-68d8f9f787-9d744   1/1     Running   0          89s
flask-app-deployment-68d8f9f787-ltwg5   1/1     Running   0          89s
flask-app-deployment-68d8f9f787-pbppl   1/1     Running   0          89s
mongodb-6f84848985-bc8rz                1/1     Running   0          5m17s
```

5. Get external IP of deployment

```
kubectl get service flask-app -n eks-todo-list-app

# a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazonaws.co
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get service flask-app -n eks-todo-list-app
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
     AGE
flask-app   LoadBalancer   10.100.98.145   a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazonaws.com   80:31128/T
CP   5m18s
```

6. Monitor health of a pod (using pod name)

```
kubectl describe pod flask-app-deployment-68d8f9f787-9d744 -n eks-todo-list
```

```
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  10m                default-scheduler  Successfully assigned eks-todo-list-app/flask-app-deployment-68d
8f9f787-9d744 to ip-192-168-36-118.us-east-2.compute.internal
  Normal   Pulled     10m                kubelet            Successfully pulled image "yuwei2002/flask-todo-app:v3" in 1.503
s (1.503s including waiting). Image size: 53425846 bytes.
  Normal   Pulling    102s (x2 over 10m)  kubelet            Pulling image "yuwei2002/flask-todo-app:v3"
  Normal   Created    102s (x2 over 10m)  kubelet            Created container flask-app
  Normal   Started    102s (x2 over 10m)  kubelet            Started container flask-app
```

7. Trigger an active readiness prob via `curl`

```
curl http://a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazo
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ curl http://a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazonaws.com/tog
gle-ready
{
  "message": "Readiness status toggled to not ready. Will reset in 30 seconds."
}
```

To better test the liveness prob, we set the duration of “not ready” to 30s.

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods -n eks-todo-list-app -w
NAME                                    READY   STATUS    RESTARTS   AGE
flask-app-deployment-68d8f9f787-9d744   1/1     Running   0          4m44s
flask-app-deployment-68d8f9f787-ltwg5   1/1     Running   0          4m44s
flask-app-deployment-68d8f9f787-pbppl   1/1     Running   0          4m44s
mongodb-6f84848985-bc8rz                1/1     Running   0          8m32s
flask-app-deployment-68d8f9f787-ltwg5   0/1     Running   0          4m46s
flask-app-deployment-68d8f9f787-ltwg5   1/1     Running   0          5m6s
```

8. Trigger an active readiness prob via `curl`

```
curl http://a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazo
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ curl http://a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazonaws.com/tog
gle-health
{
  "message": "Health status toggled to unhealthy"
}
```

Once liveness prob detect failure, it will restart the pod.

```
yuwei@yuwei-SER:~/Documents/CC-A2/kube$ kubectl get pods -n eks-todo-list-app -w
NAME                                    READY   STATUS    RESTARTS   AGE
flask-app-deployment-68d8f9f787-9d744   1/1     Running   0          8m54s
flask-app-deployment-68d8f9f787-ltwg5   1/1     Running   0          8m54s
flask-app-deployment-68d8f9f787-pbppl   1/1     Running   0          8m54s
mongodb-6f84848985-bc8rz                1/1     Running   0          12m
flask-app-deployment-68d8f9f787-9d744   0/1     Running   1 (1s ago)   9m12s
flask-app-deployment-68d8f9f787-9d744   1/1     Running   1 (15s ago)   9m26s
```

Result of health monitor:

```
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  10m                  default-scheduler  Successfully assigned eks-todo-list-app/flask-app-deployment-68d
8f9f787-9d744 to ip-192-168-36-118.us-east-2.compute.internal
  Normal   Pulled     10m                  kubelet            Successfully pulled image "yuwei2002/flask-todo-app:v3" in 1.503
s (1.503s including waiting). Image size: 53425846 bytes.
  Normal   Pulling    102s (x2 over 10m)   kubelet            Pulling image "yuwei2002/flask-todo-app:v3"
  Normal   Created    102s (x2 over 10m)   kubelet            Created container flask-app
  Normal   Started    102s (x2 over 10m)   kubelet            Started container flask-app
  Warning  Unhealthy  102s (x3 over 2m2s)  kubelet            Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Killing    102s                 kubelet            Container flask-app failed liveness probe, will be restarted
  Normal   Pulled     102s                 kubelet            Successfully pulled image "yuwei2002/flask-todo-app:v3" in 173ms
 (173ms including waiting). Image size: 53425846 bytes.
```

Step 8: Alerting:

1. Install Prometheus

```
wget https://github.com/prometheus/prometheus/releases/download/v3.2.1/pror

tar xvfz prometheus-*.tar.gz
```

```
cd prometheus-3.2.1.linux-amd64/
```

2. Configure Prometheus

   Step 1: modify prometheus.yml

```
global:
  scrape_interval: 15s # collect metrics from targets every 15 seconds
  evaluation_interval: 15s

rule_files:
  - "probe-alerts.yml"

alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager:9093

scrape_configs:
  - job_name: "all-kubernetes-pods"
    kubernetes_sd_configs:
      - role: pod
    relabel_configs:
      # Add namespace as a label
      - source_labels: [__meta_kubernetes_namespace]
        action: replace
        target_label: kubernetes_namespace
      # Add pod name as a label
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: kubernetes_pod_name

  # Uses direct HTTP checks instead
  - job_name: "blackbox"
    metrics_path: /metrics
    static_configs:
```

```
      - targets:
          - localhost:9115

  # Direct health check
  - job_name: "app-health-check"
    metrics_path: /health
    scrape_interval: 10s
    static_configs:
      - targets:
          - a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazonav
    relabel_configs:
      - source_labels: [__address__]
        target_label: instance
      - target_label: __address__
        replacement: a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb
```

step 2: Create probe-alerts.yml

```
touch probe-alerts.yml
```

```
groups:
  - name: probe-alerts
    rules:
      - alert: ProbeFailureThresholdExceeded
        expr: probe_success == 0
        for: 45s
        labels:
          severity: critical
        annotations:
          summary: "Probe failure detected"
          description: "Probe {{ $labels.instance }} has been failing for 45 seconds,

      - alert: HighProbeFailureRate
        expr: sum(rate(probe_success{job="blackbox"}[5m]) == 0) / sum(rate(prob
        for: 45s
        labels:
```

```
        severity: warning
      annotations:
        summary: "High probe failure rate detected"
        description: "More than 10% of probes are failing in the last 10 minutes."

    - alert: SlowResponseTime
      expr: probe_duration_seconds > 1
      for: 45s
      labels:
        severity: warning
      annotations:
        summary: "Slow probe response time"
        description: "Probe {{ $labels.instance }} response time is above 1 second
```

3. Create a webhooks

   Step 1: Following https://api.slack.com/messaging/webhooks to create a webhooks

   Step 2: Test if success

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}'
```

4. Install Alertmanager

```
wget https://github.com/prometheus/alertmanager/releases/download/v0.28.1/al

tar xvfz alertmanager-*.tar.gz

cd alertmanager-0.28.1.linux-amd64/
```

5. Configure Alertmanager

   Step 1: Modify alertmanager.yml

```
global:
  resolve_timeout: 1m # Faster resolution notifications
  slack_api_url: "https://hooks.slack.com/services/T08AJ5GLFKM/B08J048A2P6
```

```
route:
  group_by: ["alertname", "job"]
  group_wait: 10s
  group_interval: 45s
  repeat_interval: 4h
  receiver: "slack-notifications"

receivers:
  - name: "slack-notifications"
    slack_configs:
      - channel: "#alerts"
        send_resolved: true
        title: "{{ range .Alerts }}{{ .Annotations.summary }}\n{{ end }}"
        text: "{{ range .Alerts }}{{ .Annotations.description }}\n{{ end }}"
        color: '{{ if eq .Status "firing" }}danger{{ else }}good{{ end }}'
```

6. Testing the Alerting System

   Step 1: Start alertmanager

```
./alertmanager --config.file=alertmanager.yml &
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/alertmanager-0.28.1.linux-amd64$ ./alertmanager --confi
g.file=alertmanager.yml &
[1]+  Done                    ./alertmanager --config.file=alertmanager.yml
[1] 1382593
yuwei@yuwei-SER:~/Documents/CC-A2/alertmanager-0.28.1.linux-amd64$ time=2025-03-15T15:45:
19.485Z level=INFO source=main.go:191 msg="Starting Alertmanager" version="(version=0.28.
1, branch=HEAD, revision=b2099eaa2c9ebc25edb26517cb9c732738e93910)"
time=2025-03-15T15:45:19.485Z level=INFO source=main.go:192 msg="Build context" build_con
text="(go=go1.23.7, platform=linux/amd64, user=root@fa3ca569dfe4, date=20250307-15:05:18,
 tags=netgo)"
time=2025-03-15T15:45:19.487Z level=INFO source=cluster.go:185 msg="setting advertise add
ress explicitly" component=cluster addr=10.12.66.12 port=9094
time=2025-03-15T15:45:19.488Z level=INFO source=cluster.go:674 msg="Waiting for gossip to
 settle..." component=cluster interval=2s
time=2025-03-15T15:45:19.502Z level=INFO source=coordinator.go:112 msg="Loading configura
tion file" component=configuration file=alertmanager.yml
time=2025-03-15T15:45:19.503Z level=INFO source=coordinator.go:125 msg="Completed loading
 of configuration file" component=configuration file=alertmanager.yml
time=2025-03-15T15:45:19.504Z level=INFO source=tls_config.go:347 msg="Listening on" addr
ess=[::]:9093
time=2025-03-15T15:45:19.504Z level=INFO source=tls_config.go:350 msg="TLS is disabled."
http2=false address=[::]:9093
time=2025-03-15T15:45:21.488Z level=INFO source=cluster.go:699 msg="gossip not settled" c
omponent=cluster polls=0 before=0 now=1 elapsed=2.000077995s
time=2025-03-15T15:45:29.490Z level=INFO source=cluster.go:691 msg="gossip settled; proce
eding" component=cluster elapsed=10.002617714s
yuwei@yuwei-SER:~/Documents/CC-A2/alertmanager-0.28.1.linux-amd64$
```

Step 2: Start prometheus

```
cd ../prometheus-3.2.1.linux-amd64/

./prometheus --config.file=prometheus.yml &
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/prometheus-3.2.1.linux-amd64$ ./prometheus --config.fil
e=prometheus.yml &
[1]+  Done                    ./prometheus --config.file=prometheus.yml
[1] 1383081
yuwei@yuwei-SER:~/Documents/CC-A2/prometheus-3.2.1.linux-amd64$ time=2025-03-15T15:46:05.
623Z level=INFO source=main.go:640 msg="No time or size retention was set so using the de
fault time retention" duration=15d
time=2025-03-15T15:46:05.623Z level=INFO source=main.go:687 msg="Starting Prometheus Serv
er" mode=server version="(version=3.2.1, branch=HEAD, revision=804c49d58f3f3784c77c9c8ec1
7c9062092cae27)"
```

```
time=2025-03-15T15:46:05.632Z level=INFO source=main.go:1486 msg="Completed loading of co
nfiguration file" db_storage=930ns remote_storage=1.08µs web_handler=200ns query_engine=6
00ns scrape=320.448µs scrape_sd=46.15µs notify=77.05µs notify_sd=10.11µs rules=485.587µs
tracing=2.92µs filename=prometheus.yml totalDuration=1.377512ms
time=2025-03-15T15:46:05.632Z level=INFO source=main.go:1213 msg="Server is ready to rece
ive web requests."
time=2025-03-15T15:46:05.632Z level=INFO source=manager.go:175 msg="Starting rule manager
..." component="rule manager"
yuwei@yuwei-SER:~/Documents/CC-A2/prometheus-3.2.1.linux-amd64$
```

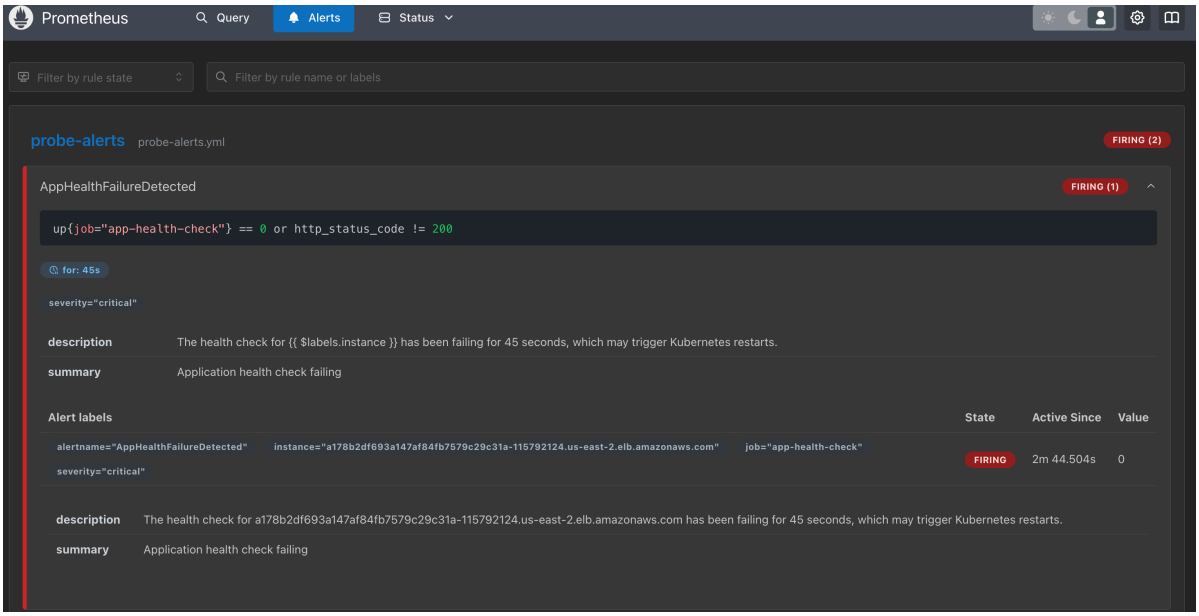
Step 3: Trigger failure

```
curl http://a178b2df693a147af84fb7579c29c31a-115792124.us-east-2.elb.amazo
```

```
yuwei@yuwei-SER:~/Documents/CC-A2/prometheus-3.2.1.linux-amd64$ curl http://a178b2df693a1
47af84fb7579c29c31a-115792124.us-east-2.elb.amazonaws.com/toggle-health
{
  "message": "Health status toggled to unhealthy"
}
```

Step 4: Watch firing:

Open the webui (http://localhost:9090)



Slack Update: