

Tejaswini Pradip Srivastava - tps7866
Komal Bagwe - knb4003

Cloud Computing and Big Data Systems Fall 2025 Assignment 2

Part 1: Creating the Application:

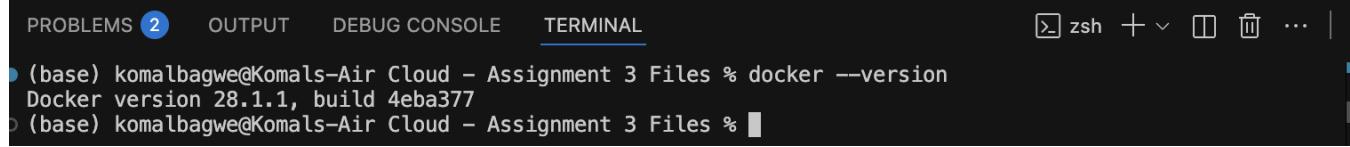
We have created the application using the source code and uploaded the repository on Github -
<https://github.com/komal-b/ToDoApp>

Part 2: Containerizing the Application on Docker:

Step 1: Install Docker

1. Download and install Docker from the official Docker website.
2. Verify installation:

docker --version



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL zsh + ⌂ ⌂ ... |  
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker --version  
Docker version 28.1.1, build 4eba377  
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files %
```

Step 2: Create a Dockerfile for Flask App

1. We created the following docker file and stored in this path: **Path: /Cloud - Assignment 3 Files/Dockerfile**

Step 3: Build the Docker Image

1. To build the docker image. We ran the following command in the same directory as the Dockerfile:

docker build -t flask-todo-app:latest .

```

● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker build -t flask-todo-app:latest .
[+] Building 0.8s (11/11) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 488B                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.10-slim          0.6s
=> [auth] library/python:pull token for registry-1.docker.io                 0.0s
=> [internal] load .dockerrignore                                           0.0s
=> => transferring context: 2B                                             0.0s
=> [1/5] FROM docker.io/library/python:3.10-slim@sha256:e0c4fae70d550834a40f6c3e0326e02cf239c2351 0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 7.34kB                                         0.0s
=> CACHED [2/5] WORKDIR /app                                              0.0s
=> CACHED [3/5] COPY requirements.txt                                     0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt        0.0s
=> [5/5] COPY . .                                                       0.0s
=> exporting to image                                                 0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:2cec1518c0d86808969bbb2e91f2cd7f3a853f29e46582919b7d68d94855e2d3 0.0s
=> => naming to docker.io/library/flask-todo-app:latest                  0.0s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/u4guzfpovrglob49kmdq0oqc3
What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview

```

Step 4: Verify the image:

1. We ran the following command to verify the docker image:

docker images

```

● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker images
REPOSITORY          IMAGE ID       CREATED      TAG      SIZE
flask-todo-app      2cec1518c0d8   About a minute ago   latest   161MB
cloud-assignment3files-web 47152d8cbe06   9 hours ago      latest   161MB

```

Step 5: Test Locally Using Docker Compose

1. To test the application locally with both containers (Flask and MongoDB), We have created docker-compose.yaml file.

Path: /Cloud - Assignment 3 Files/docker-compose.yaml

2. To run the application: **docker-compose up**

```

● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker-compose up
[+] Running 2/2
  ✓ Container mongodb      Created
  ✓ Container flask-todo-app Created
Attaching to flask-todo-app, mongodb

```

Step 6: Push the Docker Image to Docker Hub

1. Create a Docker Hub account at <https://hub.docker.com>

2. Log in using the Docker CLI:

docker login

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker login
Authenticating with existing credentials... [Username: komalbagwe31]
  Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files %
```

3. Tag your local image to match your Docker Hub repository:

**docker tag flask-todo-app:latest
<dockerhub-username>/flask-todo-app:latest**

a. For us : **docker tag flask-todo-app:latest
komalbagwe31/flask-todo-app**

4. Push the image to Docker Hub:

docker push <dockerhub-username>/flask-todo-app:latest

a. For us : **docker push komalbagwe31/flask-todo-app:latest**

5. Image will be publicly available on Docker Hub for use in deployments.

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker tag flask-todo-app:latest komalbagwe31/flask-todo-app
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % docker push komalbagwe31/flask-todo-app:latest
The push refers to repository [docker.io/komalbagwe31/flask-todo-app]
e3b68d20237e: Pushed
59ba9a89551c: Layer already exists
736ee4c8d468: Layer already exists
8cc3bd02b194: Layer already exists
938c55db4ae0: Layer already exists
772dad6c1281: Layer already exists
2ace991c990b: Layer already exists
27d9a8093fc: Layer already exists
latest: digest: sha256:9f238d5343c55b624856dae4b96d25ddf589267edf76b7e7a667be38d8ab2f5c size: 1993
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files %
```

hub Explore My Hub

Repositories / flask-todo-app / General

komalbagwe31 Docker Personal

Last pushed 2 minutes ago · Repository size: 95.9 MB · ⭐0 · ↓67

Add a description *(i)*

Add a category *(i)*

General Tags Image Management BETA Collaborators Webhooks Settings

Tags

This repository contains 0 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	2 minutes

[See all](#)

Repository overview (i) INCOMPLETE

An overview describes what your image does and how to run it. It displays in [the public view of your repository](#) once you have pushed some content.

[Add overview](#)

Using 0 of 1 private repositories.

Docker commands

To push a new tag to this repository:

```
docker push komalbagwe31/flask-todo-app:  
tagname
```

Public view

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

[Go to Docker Build Cloud →](#)

Part 3: Deploying the Application on Minikube:

Step 1: Install Minikube

1. Download and install Minikube
2. Verify installation:

```
minikube version
```

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % minikube version
minikube version: v1.36.0
commit: f8f52f5de11fc6ad8244afac475e1d0f96841df1
○ (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % █
```

Step 2: Start Minikube

1. Start a local Kubernetes cluster:

```
minikube start
```

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % minikube start
  ☺ minikube v1.36.0 on Darwin 14.1.2 (arm64)
  ⚡ minikube 1.37.0 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.37.0
 💡 To disable this notice, run: 'minikube config set WantUpdateNotification false'

  ⚡ Automatically selected the docker driver
  ⚡ Using Docker Desktop driver with root privileges
  ⚡ Starting "minikube" primary control-plane node in "minikube" cluster
  ⚡ Pulling base image v0.0.47 ...
  ⚡ Downloading Kubernetes v1.33.1 preload ...
    > preloaded-images-k8s-v18-v1...: 327.15 MiB / 327.15 MiB 100.00% 11.63 M
    > gcr.io/k8s-minikube/kicbase...: 463.69 MiB / 463.69 MiB 100.00% 11.50 M
  ⚡ Creating docker container (CPUs=2, Memory=2200MB) ...
  ⚡ Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
  ⚡ Configuring bridge CNI (Container Networking Interface) ...
  ⚡ Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  ⚡ Enabled addons: storage-provisioner, default-storageclass
  ⚡ Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

2. Verify the cluster is running:

```
kubectl get nodes
```

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl get nodes
  NAME      STATUS    ROLES      AGE      VERSION
  minikube  Ready     control-plane  5m8s   v1.33.1
```

Step 3a: MongoDB Deployment

1. A file **mongo-deployment.yaml** is created

Path: **/Cloud - Assignment 3 Files/mongo-deployment.yaml**

2. Apply the MongoDB deployment:

kubectl apply -f mongo-deployment.yaml

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl apply -f mongo-deployment.yaml
deployment.apps/mongo-deployment created
service/mongo created
○ (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % █
```

Step 3b: Flask Deployment

1. A file **deployment.yaml** is created.

Path: **/Cloud - Assignment 3 Files/deployment.yaml**

2. Apply the Flask deployment:

kubectl apply -f deployment.yaml

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl apply -f deployment.yaml
deployment.apps/flask-todo-deployment created
service/flask-todo-service created
○ (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % █
```

Step 4: Expose the Deployment

1. Minikube provides a URL to access the service:

minikube service flask-todo-service --url

```
○ (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % minikube service flask-todo-service --url
http://127.0.0.1:54026
! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
█
```

Pods:

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl get pods
NAME           READY   STATUS      RESTARTS   AGE
flask-todo-deployment-8684d4f8d6-5h8gb   0/1     ContainerCreating   0          102s
flask-todo-deployment-8684d4f8d6-tx54b   0/1     ContainerCreating   0          102s
mongo-deployment-84c58f889-fqwmj        1/1     Running      0          3m35s
█
```

Output:

ToDo Reminder

ALL Uncompleted Completed About

No Tasks in the List !!

Add a Task

Taskname

Enter Description here...

mm/dd/yyyy

Priority

Part 4: Deploying the Application on AWS EKS:

Step 1: Create an SSH Keys if not present

1. Check if you have any SSH keys:

```
ls ~/.ssh/
```

2. If not create one

```
ssh-keygen -t rsa -b 4096 -C "eks-key" -f ~/.ssh/eks_key
```

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % ssh-keygen -t rsa -b 4096 -C "eks-key" -f ~/.ssh/eks_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/komalbagwe/.ssh/eks_key
Your public key has been saved in /Users/komalbagwe/.ssh/eks_key.pub
The key fingerprint is:
SHA256:iVaCTBNmE5RaZI39phhtc19iFmkZ64w9wsFo0uHNKWE eks-key
The key's randomart image is:
+---[RSA 4096]---+
| o%B   .+
| *++oo  =.
| oE.+..+...
| .o.B++B+ .
| =+=S+=o
| .+.. ...
+---[SHA256]---+
```

3. Import the key in EC2

- a. Go to the **AWS Management Console → EC2 → Key Pairs**
- b. Click **Import key pair**
- c. We need to add name for key. I added **eks-key** as name

Key pairs (1) Info				
Actions Create key pair				
<input type="checkbox"/>	Name	Type	Created	Fingerprint
<input type="checkbox"/>	eks-key	rsa	2025/10/29 02:13 GMT-4	56:24:12:7f:2f:1f:b7:7b:b2:44:37:...

Step 2: Create an EKS Cluster

Command:

```
eksctl create cluster \
--name flask-eks-cluster \
--region us-east-1 \
--nodes 2 \
--node-type t3.medium \
--with-oidc \
--ssh-access \
--ssh-public-key eks-key \
--managed
```

```
(base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % eksctl create cluster \
--name flask-eks-cluster \
--region us-east-1 \
--nodes 2 \
--node-type t3.medium \
--with-oidc \
--ssh-access \
--ssh-public-key eks-key \
--managed

2025-10-29 02:15:21 [i] eksctl version 0.215.0
2025-10-29 02:15:21 [i] using region us-east-1
2025-10-29 02:15:22 [i] setting availability zones to [us-east-1f us-east-1d]
2025-10-29 02:15:22 [i] subnets for us-east-1f - public:192.168.0.0/19 private:192.168.64.0/19
2025-10-29 02:15:22 [i] subnets for us-east-1d - public:192.168.32.0/19 private:192.168.96.0/19
2025-10-29 02:15:22 [i] nodegroup "ng-a635343a" will use "" [AmazonLinux2023/1.32]
2025-10-29 02:15:22 [i] using EC2 key pair "eks-key"
2025-10-29 02:15:22 [!] Auto Mode will be enabled by default in an upcoming release of eksctl. This means
managed node groups and managed networking add-ons will no longer be created by default. To maintain current behavior, explicitly set 'autoModeConfig.enabled: false' in your cluster configuration. Learn more: ht
tps://eksctl.io/usage/auto-mode/
2025-10-29 02:15:22 [i] using Kubernetes version 1.32
2025-10-29 02:15:22 [i] creating EKS cluster "flask-eks-cluster" in "us-east-1" region with managed nodes
2025-10-29 02:15:22 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial m
```

Explanation:

1. **--name:** Cluster name
2. **--region:** AWS region
3. **--nodes:** Number of worker nodes

4. **--node-type**: EC2 instance type for nodes
5. **--managed**: Use managed node groups

This creates both the control plane and worker nodes.

Step 3: Create a deployment file

The **deployment.yaml** file we created had the service type set to **NodePort**, but in AWS EKS, we can't use **NodePort**. For AWS EKS, we need to use the service type **LoadBalancer** instead. So created new deployment file named **eks-deployment.yaml**

Apply the eks-deployment:

kubectl apply -f eks-deployment.yaml

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl apply -f eks-deployment.yaml
deployment.apps/flask-app-deployment unchanged
service/flask-app-service created
```

Apply the mongo-deployment.yaml

kubectl apply -f mongo-deployment.yaml

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl apply -f mongo-deployment.yaml
deployment.apps/mongo-deployment created
service/mongo created
```

Step 5: Test the Application

Get the external URL:

kubectl get service flask-app-service

```
● (base) komalbagwe@Komals-Air Cloud - Assignment 3 Files % kubectl get service flask-app-service
  NAME           TYPE      CLUSTER-IP   EXTERNAL-IP
  flaskservice   LoadBalancer  10.100.189.6  afa27381a58d64e289e8fe53faf245a-63172696.us-east-1.
  amazonaws.com  80:31001/TCP  11m
```

Output:

Not Secure afa27381a58d64e289e8fe53fafe245a-63172696.us-east-1.elb.amazonaws.com

Google Trends Lists - Learn Python... Constellation Jew... How to Register th... Automation Univers... Merch Titans Auto... LunaPic | Free Onli... Redbubble Popula... All Bookmarks

ToDo Reminder

No Tasks in the List !!

Add a Task

Taskname

Enter Description here...

mm/dd/yyyy

Priority

Part 5: Deployments and ReplicaSets:

1. Define the desired number of replicas in your deployment.yaml file under **spec.replicas**.
2. We have done this here: Line 6: **replicas: 2**

33 lines (32 loc) · 551 Bytes

Code Blame

Raw ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: flask-app-deployment
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: flask-app
10   template:
11     metadata:
12       labels:
13         app: flask-app
14     spec:
15       containers:
16         - name: flask-app
17           image: komalbagwe31/flask-todo-app:latest
18           ports:
19             - containerPort: 5000
20   ---
21
22   apiVersion: v1
23   kind: Service
24   metadata:
25     name: flask-app-service
26   spec:
27     type: LoadBalancer
28     selector:
29       app: flask-app
30     ports:
31       - protocol: TCP
32         port: 80
```



3. Apply the deployment using '**kubectl apply -f eks-deployment.yaml**' .

```

$ kubectl apply -f eks-deployment.yaml
deployment.apps/flask-app-deployment configured
service/flask-app-service unchanged
~ $ kubectl get deploy,rs,pods -o wide
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE      CONTAINERS   IMAGES
SELECTOR
deployment.apps/flask-app-deployment  2/2     1           2           3d21h   flask-app    komalbagwe31/flask-todo-app:latest
app=flask-app
deployment.apps/mongo-deployment     1/1     1           1           3d21h   mongo        mongo:6

NAME                                DESIRED   CURRENT   READY   AGE      CONTAINERS   IMAGES
SELECTOR
replicaset.apps/flask-app-deployment-665c67b84b  0         0         0         14m     flask-app    komalbagwe31/flask-todo-app@s
ha256:b0a7c0b94c577d95fbcae8185c25289bd63e8536fe3e76e7af469a501cd43b4b
replicaset.apps/flask-app-deployment-6956d8bf9c  2         2         2         2m30s   flask-app    komalbagwe31/flask-todo-app@s
ha256:b0a7c0b94c577d95fbcae8185c25289bd63e8536fe3e76e7af469a501cd43b4b
replicaset.apps/flask-app-deployment-6c57f4887d  1         1         0         27s     flask-app    komalbagwe31/flask-todo-app:l
atest
replicaset.apps/flask-app-deployment-777f68f68f  0         0         0         3d21h   flask-app    komalbagwe31/flask-todo-app:l
atest
replicaset.apps/flask-app-deployment-8695c58c85  0         0         0         33m     flask-app    komalbagwe31/flask-todo-app:l
atest
replicaset.apps/mongo-deployment-84c58f889    1         1         1         3d21h   mongo        mongo:6
                                         app=mongo, pod-template-hash=84c58f889

NAME          NOMINATED NODE   READINESS GATES   READY   STATUS   RESTARTS   AGE   IP          NODE
pod/flask-app-deployment-6956d8bf9c-2ctzb  ec2.internal <none>       <none>    1/1     Running   0          2m29s  192.168.3.46  ip-192-168-5-149.e
c2.internal
pod/flask-app-deployment-6956d8bf9c-7vqk6  ec2.internal <none>       <none>    1/1     Running   0          2m30s  192.168.40.140 ip-192-168-33-104.
ec2.internal
pod/flask-app-deployment-6c57f4887d-88kgz  ec2.internal <none>       <none>    0/1     CrashLoopBackOff 2 (13s ago)  27s   192.168.47.216 ip-192-168-33-104.
ec2.internal
pod/mongo-deployment-84c58f889-c855p    ec2.internal <none>       <none>    1/1     Running   0          3d21h  192.168.47.231 ip-192-168-33-104.
ec2.internal

```

4. Verify the ReplicaSet with '**kubectl get rs**' .

```

$ kubectl get rs
NAME          DESIRED   CURRENT   READY   AGE
flask-app-deployment-665c67b84b  0         0         0         15m
flask-app-deployment-6956d8bf9c  2         2         2         3m6s
flask-app-deployment-6c57f4887d  1         1         0         63s
flask-app-deployment-777f68f68f  0         0         0         3d21h
flask-app-deployment-8695c58c85  0         0         0         34m
mongo-deployment-84c58f889     1         1         1         3d21h

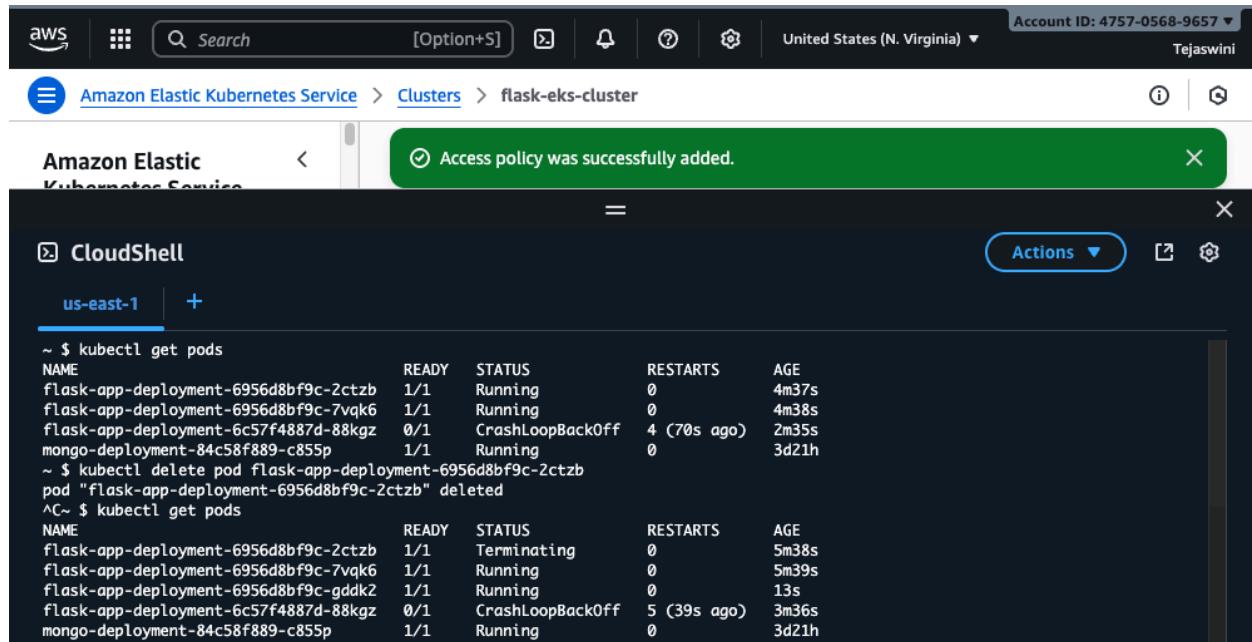
```

5. Delete one pod and observe Kubernetes automatically creating a replacement pod.

Kubectl get pods

kubectl delete pod <POD_NAME>

kubectl get pods



The screenshot shows the AWS CloudShell interface in a browser window. At the top, there's a navigation bar with the AWS logo, search bar, and account information (Account ID: 4757-0568-9657, United States (N. Virginia)). Below the navigation is a green success message: "Access policy was successfully added." The main area is a terminal window titled "CloudShell" with the region "us-east-1". The terminal output shows the following sequence of commands:

```

~ $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
flask-app-deployment-6956d8bf9c-2ctzb  1/1     Running   0          4m37s
flask-app-deployment-6956d8bf9c-7vqk6  1/1     Running   0          4m38s
flask-app-deployment-6c57f4887d-88kgz  0/1     CrashLoopBackOff 4 (70s ago)  2m35s
mongo-deployment-84c58f889-c855p      1/1     Running   0          3d21h

~ $ kubectl delete pod flask-app-deployment-6956d8bf9c-2ctzb
pod "flask-app-deployment-6956d8bf9c-2ctzb" deleted

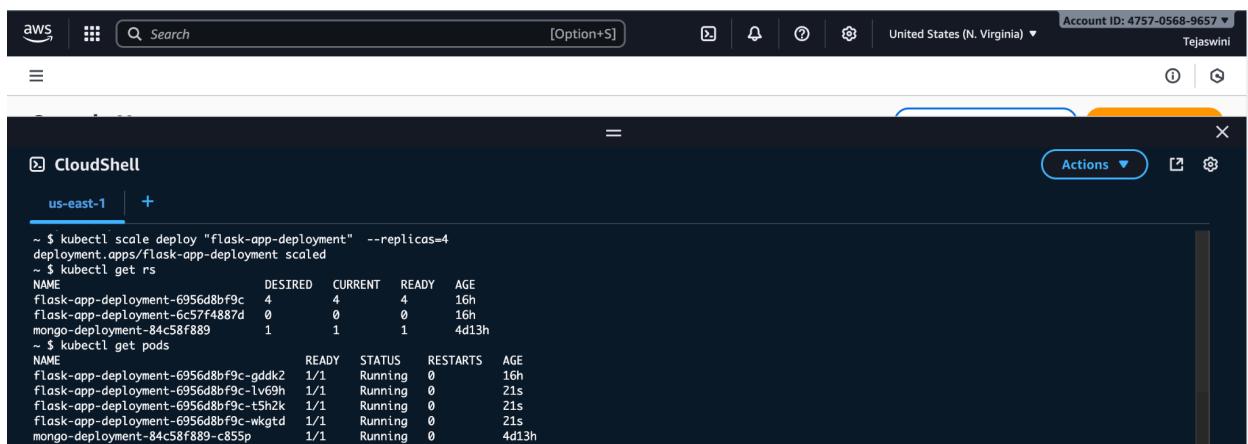
^C~ $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
flask-app-deployment-6956d8bf9c-2ctzb  1/1     Terminating   0          5m38s
flask-app-deployment-6956d8bf9c-7vqk6  1/1     Running   0          5m39s
flask-app-deployment-6956d8bf9c-gddk2  1/1     Running   0          13s
flask-app-deployment-6c57f4887d-88kgz  0/1     CrashLoopBackOff 5 (39s ago)  3m36s
mongo-deployment-84c58f889-c855p      1/1     Running   0          3d21h

```

6. Scale up or down using either 'kubectl scale deployment --replicas=n' or by editing the YAML file

- a. Scale up:

Kubectl scale deploy “flask-app-deployment” –replicas = 4



The screenshot shows the AWS CloudShell interface in a browser window. The terminal output shows the scaling of the "flask-app-deployment" and the subsequent state of the pods:

```

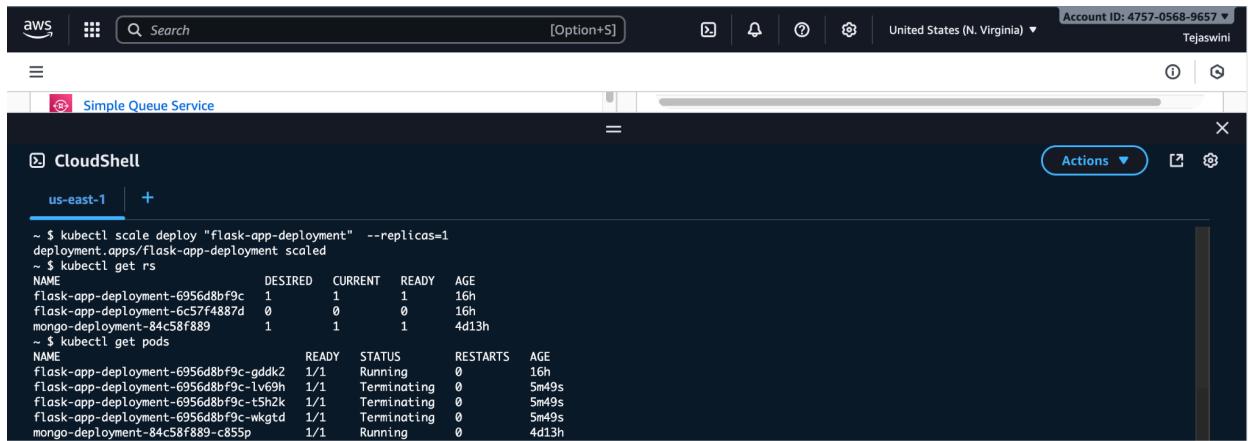
~ $ kubectl scale deploy "flask-app-deployment" --replicas=4
deployment.apps/flask-app-deployment scaled
~ $ kubectl get rs
NAME        DESIRED   CURRENT   READY   AGE
flask-app-deployment-6956d8bf9c   4         4         4       16h
flask-app-deployment-6c57f4887d   0         0         0       16h
mongo-deployment-84c58f889      1         1         1       4d13h

~ $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
flask-app-deployment-6956d8bf9c-gddk2  1/1     Running   0          16h
flask-app-deployment-6956d8bf9c-1v69h  1/1     Running   0          21s
flask-app-deployment-6956d8bf9c-t5h2k  1/1     Running   0          21s
flask-app-deployment-6956d8bf9c-wkgtd  1/1     Running   0          21s
mongo-deployment-84c58f889-c855p      1/1     Running   0          4d13h

```

b. Scale down:

Kubectl scale deploy “flask-app-deployment” –replicas = 1



The screenshot shows the AWS CloudShell interface in the Simple Queue Service (SQS) section. The terminal window displays the following command and its output:

```
~ $ kubectl scale deploy "flask-app-deployment" --replicas=1
deployment.apps/flask-app-deployment scaled
~ $ kubectl get rs
NAME          DESIRED   CURRENT   READY   AGE
flask-app-deployment-6956d8bf9c  1         1         1       16h
flask-app-deployment-6c57f4887d  0         0         0       16h
mongo-deployment-84c58f889     1         1         1       4d13h
~ $ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
flask-app-deployment-6956d8bf9c-gddk2  1/1     Running   0          16h
flask-app-deployment-6956d8bf9c-lv69h  1/1     Terminating   0          5m49s
flask-app-deployment-6956d8bf9c-t5h2k  1/1     Terminating   0          5m49s
flask-app-deployment-6956d8bf9c-wkgtd  1/1     Terminating   0          5m49s
mongo-deployment-84c58f889-c855p    1/1     Running   0          4d13h
```

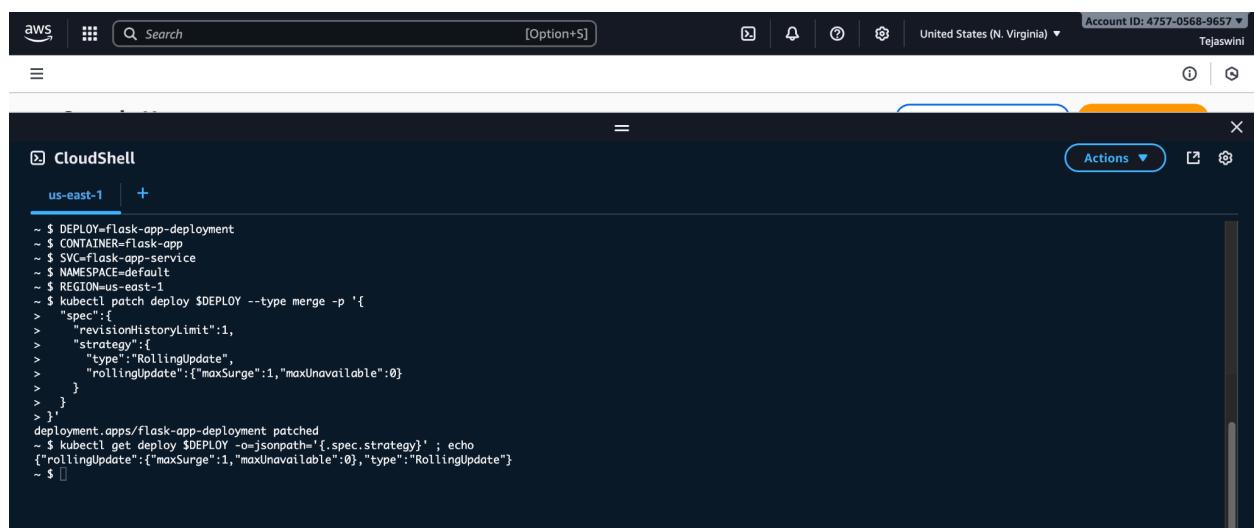
Part 6: Rolling update strategy:

1. Configure the Kubernetes deployment to use a rolling update strategy.

```
kubectl patch deploy $DEPLOY --type merge -p '{  
  "spec":{  
    "revisionHistoryLimit":1,  
    "strategy":{  
      "type":"RollingUpdate",  
      "rollingUpdate": {"maxSurge":1,"maxUnavailable":0}  
    }  
  }  
}'
```

```
kubectl get deploy $DEPLOY -o=jsonpath='{.spec.strategy}' ; echo
```

This keeps all existing pods serving while adding one new pod at a time.



The screenshot shows the AWS CloudShell interface in a browser window. The title bar includes the AWS logo, a search bar, and account information: Account ID: 4757-0568-9657, Region: United States (N. Virginia), and User: Tejaswini. The main area is a dark-themed terminal window titled 'CloudShell' with the region 'us-east-1'. The terminal displays the following command history:

```
~ $ DEPLOY=Flask-app-deployment  
~ $ CONTAINER=flask-app  
~ $ SVC=flask-app-service  
~ $ NAMESPACE=default  
~ $ REGION=us-east-1  
~ $ kubectl patch deploy $DEPLOY --type merge -p '{  
  "spec":{  
    "revisionHistoryLimit":1,  
    "strategy":{  
      "type":"RollingUpdate",  
      "rollingUpdate": {"maxSurge":1,"maxUnavailable":0}  
    }  
  }  
}'  
deployment.apps/flask-app-deployment patched  
~ $ kubectl get deploy $DEPLOY -o=jsonpath='{.spec.strategy}' ; echo  
{"rollingUpdate":{"maxSurge":1,"maxUnavailable":0},"type":"RollingUpdate"}  
~ $
```

2. Update the Docker image for the deployment to a new version,

```
kubectl set image deploy/$DEPLOY
flask-app=komalbagwe31/flask-todo-app:latest
```

3. Monitor roll-out:

```
kubectl rollout status deploy/flask-app-deployment
kubectl rollout history deploy/flask-app-deployment
```

Show RS + Pods during/after rollout

```
Kubectl get deploy,rs,pods -o wide
```

4. Prove pods run the new image:

```
kubectl get pods -l app=flask-app -o jsonpath='{range .items[*]}{.metadata.name}{" =>"}{.status.containerStatuses[0].image}{"\n"}{end}'
```

```

aws | [ ] Search [Option+S] | United States (N. Virginia) ▾ | Account ID: 4757-0568-9657 ▾ | Tejaswini
☰ | i | X
CloudShell | Actions ▾ | ☰
us-east-1 | +
~ $ kubectl get deploy flask-app-deployment -o jsonpath='{.spec.template.spec.containers[0].image}'; echo komalbagwe31/flask-todo-app:latest
~ $ kubectl patch deploy $DEPLOY --type='json' -p='[{"op":"replace","path":"/spec/template/spec/containers/0/imagePullPolicy","value":"Always"}]'
> []
deployment.apps/flask-app-deployment patched (no change)
~ $ kubectl rollout status deploy/$DEPLOY
deployment "flask-app-deployment" successfully rolled out
~ $ kubectl rollout status deploy/$DEPLOY
deployment "flask-app-deployment" successfully rolled out
~ $ kubectl rollout history deploy/$DEPLOY
deployment.apps/flask-app-deployment
REVISION CHANGE-CAUSE
18 <none>
19 <none>

~ $ kubectl get deploy,rs,pods -o wide
NAME READY UP-TO-DATE AVAILABLE AGE CONTAINERS IMAGES
deployment.apps/flask-app-deployment 1/1 1 1 4d14h flask-app komalbagwe31/flask-todo-app:lates
t app=flask-app
deployment.apps/mongo-deployment 1/1 1 1 4d14h mongo mongo:6

NAME DESIRED CURRENT READY AGE CONTAINERS IMAGES
SELECTOR
replicaset.apps/flask-app-deployment-6c57f4887d 0 0 0 16h flask-app komalbagwe31/flask-todo-app
:latest app=flask-app,pod-template-hash=6c57f4887d
replicaset.apps/flask-app-deployment-846555d6c4 1 1 1 4m58s flask-app komalbagwe31/flask-todo-app
:latest app=flask-app,pod-template-hash=846555d6c4
replicaset.apps/mongo-deployment-84c58f889 1 1 1 4d14h mongo mongo:6
app=mongo,pod-template-hash=84c58f889

NAME NOMINATED NODE READINESS GATES
pod/flask-app-deployment-846555d6c4-f8zw2 1/1 Running 0 4m58s 192.168.47.216 ip-192-168-33-104.ec2.internal
al <none> <none>
pod/mongo-deployment-84c58f889-c855p 1/1 Running 0 4d14h 192.168.47.231 ip-192-168-33-104.ec2.internal
al <none> <none>
```

AWS CloudShell interface in us-east-1. The terminal window shows the output of a `kubectl` command:

```
~ $ kubectl get pods -l app=flask-app -o jsonpath='{range .items[*]}{.metadata.name}{\" => image=\"}{.status.containerStatuses[0].image}{\" ; imageID=\"}{.status.containerStatuses[0].imageID}{\"\\n\"}{end}' flask-app-deployment-846555d6c4-f8zw2 => image=docker.io/komalbagwe31/flask-todo-app:latest ; imageID=docker.io/komalbagwe31/flask-todo-app@sha256:3f6776a56b41e2164029affa2b54b3aa0d10051c0150bd24cb0d356322136262
```

5. Hit the Service to validate app after update:

```
ELB=$(kubectl get svc flask-app-service -o jsonpath='{.status.loadBalancer.ingress[0].hostname}'); echo "http://$ELB" curl -I "http://$ELB"
```

AWS CloudShell interface in us-east-1. The terminal window shows the output of the command from step 5:

```
~ $ ELB=$(kubectl get svc $SVC -o jsonpath='{.status.loadBalancer.ingress[0].hostname}'); echo "http://$ELB"  
http://afa27381a58d64e289e8fe53fafe245a-63172696.us-east-1.elb.amazonaws.com  
~ $ curl -I "http://$ELB"  
HTTP/1.1 200 OK  
Server: Werkzeug/2.2.3 Python/3.10.19  
Date: Sun, 02 Nov 2025 21:02:31 GMT  
Content-Type: text/html; charset=utf-8  
Content-Length: 1629  
Connection: close
```

6. Verify pods in Kubernetes dashboard:

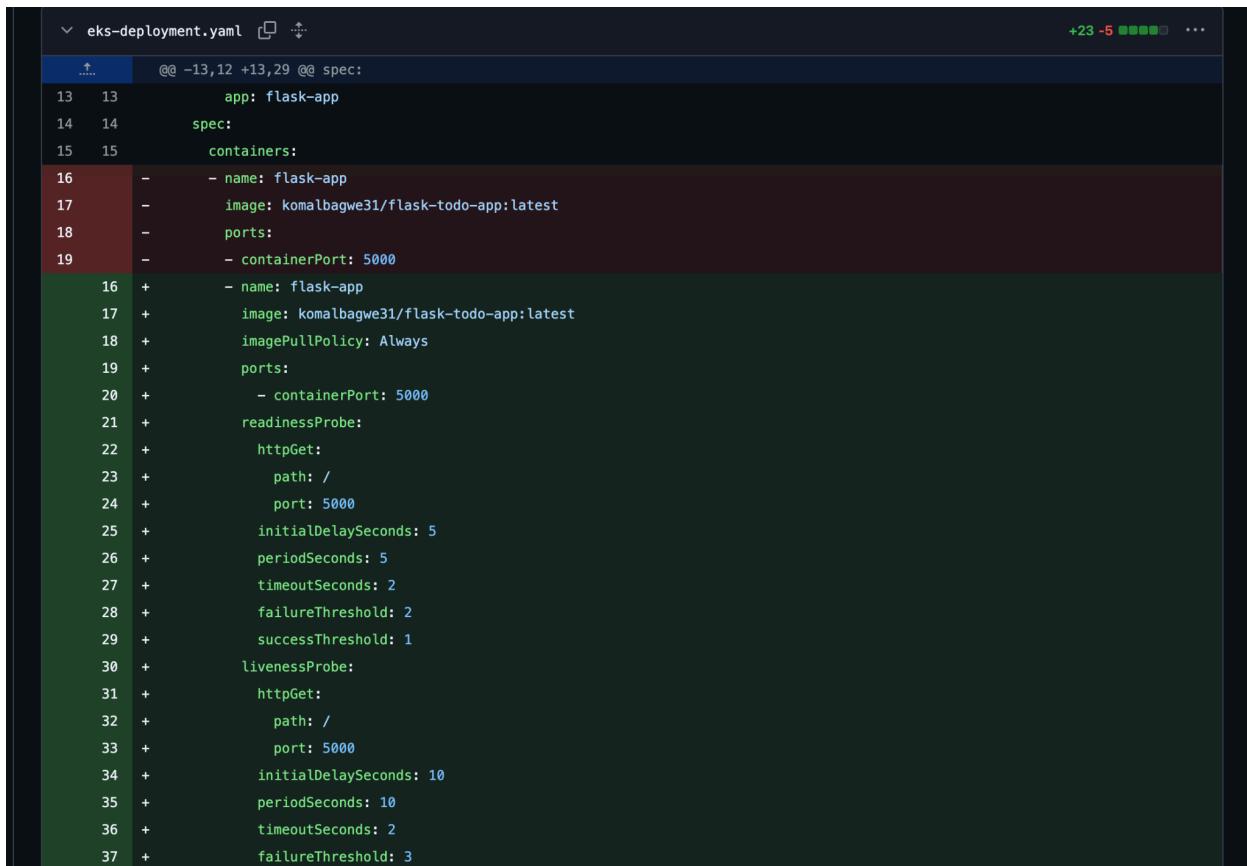
The screenshot shows the EKS cluster overview page. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 4757-0568-9657, Tejaswini). Below the navigation is the cluster name 'flask-eks-cluster'. On the left, a sidebar lists 'Amazon Elastic Kubernetes Service' (Dashboard, Clusters), 'Settings' (Dashboard settings, Console settings), 'Amazon EKS Anywhere' (Enterprise Subscriptions), and 'Related services' (Amazon ECR, AWS Batch). The main content area has tabs for Overview, Resources, Compute, Networking, Add-ons (1), Access, Observability, and Update history. The Overview tab is selected. It displays cluster info: Status (Active), Kubernetes version (1.32), Support period (Standard support until March 22, 2026), Provider (EKS). It also shows Cluster health (0), Upgrade insights (5), and Node health issues (0). Below this is a 'Details' section with fields for API server endpoint (https://078309BD906275566D9CEAC5613B599D.gr7.us-east-1.eks.amazonaws.com), Certificate authority (long ARN), OpenID Connect provider URL (https://oidc.eks.us-east-1.amazonaws.com/), Cluster IAM role ARN (arn:aws:iam::475705689657:role/eksctl-flask-eks-cluster-cluster-ServiceRole-ddV65eJIE1tp), and other metadata like Created (October 29, 2025, 02:15 UTC-04:00) and Cluster ARN (arn:aws:eks:us-east-1:475705689657:cluster/flask-eks-cluster). There's also an 'EKS Auto Mode' section with a Manage button.

Part 7: Health monitoring:

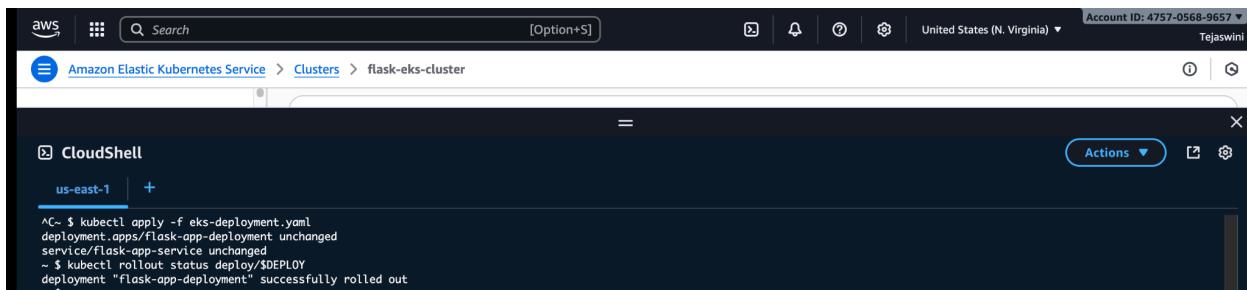
Step 1: Configure Kubernetes to monitor the health of the application by setting up probes for the pods

Updated **eks-deployment.yaml** file (lines: 21-37) to have readiness and liveness probes in the following way:

1. **Readiness probe fails** → Pod is marked **NotReady** and **removed from Service endpoints** (traffic stops).
2. **Liveness probe fails repeatedly (per failureThreshold)** → **Kubelet restarts the container.**



```
@@ -13,12 +13,29 @@ spec:
13   13     app: flask-app
14   14   spec:
15   15     containers:
16 -   - name: flask-app
17 -     image: komalbagwe31/flask-todo-app:latest
18 -     ports:
19 -       - containerPort: 5000
16 +   - name: flask-app
17 +     image: komalbagwe31/flask-todo-app:latest
18 +     imagePullPolicy: Always
19 +     ports:
20 +       - containerPort: 5000
21 +     readinessProbe:
22 +       httpGet:
23 +         path: /
24 +         port: 5000
25 +       initialDelaySeconds: 5
26 +       periodSeconds: 5
27 +       timeoutSeconds: 2
28 +       failureThreshold: 2
29 +     successThreshold: 1
30 +     livenessProbe:
31 +       httpGet:
32 +         path: /
33 +         port: 5000
34 +       initialDelaySeconds: 10
35 +       periodSeconds: 10
36 +       timeoutSeconds: 2
37 +       failureThreshold: 3
```



```
AWS | Search [Option+S] Account ID: 4757-0568-9657 ▾ Tejaswini
Amazon Elastic Kubernetes Service > Clusters > flask-eks-cluster
CloudShell us-east-1 Actions ▾
us-east-1 +
^C- $ kubectl apply -f eks-deployment.yaml
deployment.apps/flask-app-deployment unchanged
service/flask-app-service unchanged
~ $ kubectl rollout status deploy/$DEPLOY
deployment "flask-app-deployment" successfully rolled out
^C
```

Step 2: Monitoring:

1. Get pods

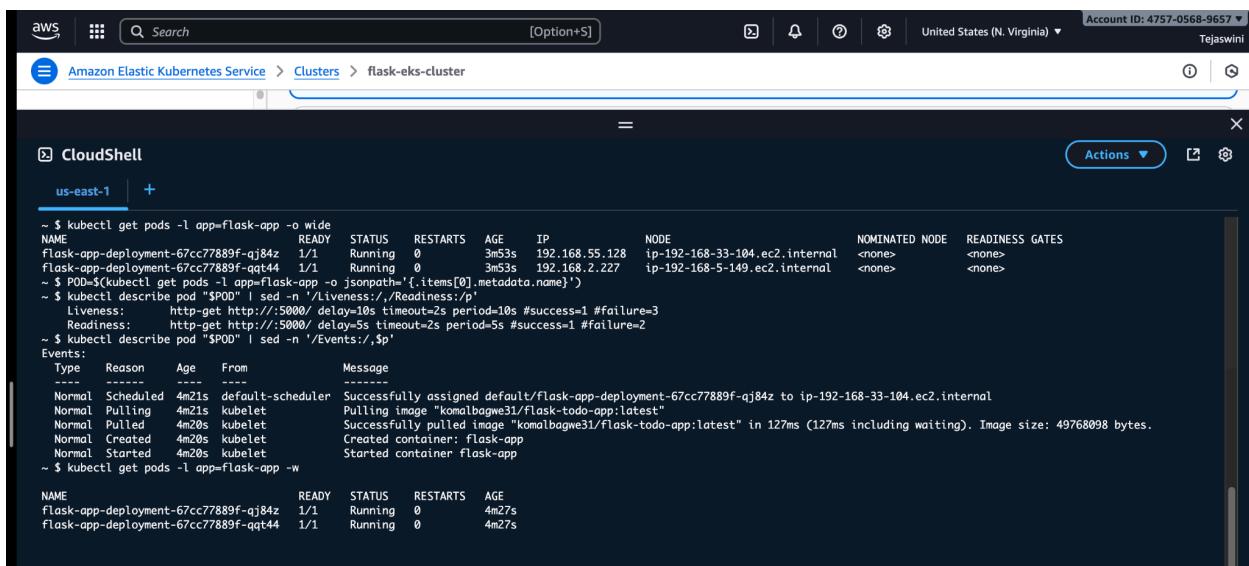
```
kubectl get pods -l app=flask-app -o wide
```

2. See probe configuration + last events

```
POD=$(kubectl get pods -l app=flask-app -o jsonpath='{.items[0].metadata.name}')
kubectl describe pod "$POD" | sed -n '/Liveness:/,/Readiness:/p'
kubectl describe pod "$POD" | sed -n '/Events:/,$p'
```

3. Live watch of status changes

```
kubectl get pods -l app=flask-app -w
```



The screenshot shows the AWS CloudShell interface in a browser window. The URL is [Amazon Elastic Kubernetes Service > Clusters > flask-eks-cluster](#). The CloudShell tab is active, showing the region as us-east-1. The terminal window displays the following command history:

```
$ kubectl get pods -l app=flask-app -o wide
NAME           READY   STATUS    RESTARTS   AGE     IP           NODE
flask-app-deployment-67cc77889f-qj84z  1/1    Running   0          3m53s  192.168.55.128  ip-192-168-33-104.ec2.internal
flask-app-deployment-67cc77889f-qqt44  1/1    Running   0          3m53s  192.168.2.227   ip-192-168-5-149.ec2.internal
$ POD=$(kubectl get pods -l app=flask-app -o jsonpath='{.items[0].metadata.name}')
$ kubectl describe pod "$POD" | sed -n '/Liveness:/,/Readiness:/p'
  Liveness: http-get http://:5000/ delay=10s timeout=2s period=10s #success=1 #failure=3
  Readiness: http-get http://:5000/ delay=5s timeout=2s period=5s #success=1 #failure=2
$ kubectl describe pod "$POD" | sed -n '/Events:/,$p'
Events:
  Type  Reason  Age   From            Message
  ----  -----  --   --             --
  Normal Scheduled 4m21s default-scheduler  Successfully assigned default/flask-app-deployment-67cc77889f-qj84z to ip-192-168-33-104.ec2.internal
  Normal Pulling  4m21s kubelet        Pulling image "komalbagwe31/flask-todo-app:latest"
  Normal Pulled   4m20s kubelet        Successfully pulled image "komalbagwe31/flask-todo-app:latest" in 127ms (127ms including waiting). Image size: 49768098 bytes.
  Normal Created  4m20s kubelet        Created container: flask-app
  Normal Started  4m20s kubelet        Started container flask-app
$ kubectl get pods -l app=flask-app -w
```

The terminal shows the output of the `-w` command, which provides real-time updates on the pod's status and events.

4. Verify in Dashboard:

AWS Console → EKS → Clusters → flask-eks-cluster → Workloads / Pods

The screenshot shows the AWS EKS Cluster dashboard for the 'flask-eks-cluster'. The left sidebar includes options for Dashboard, Clusters (selected), Settings, Amazon EKS Anywhere, Related services (Amazon ECR, AWS Batch), and Documentation. The main content area displays 'Cluster info' with status 'Active', Kubernetes version 1.32, support period until March 22, 2026, and provider EKS. Below this, the 'Resources' tab is selected, showing 'Workloads: Pods (11)'. A table lists the pods, all of which are currently running.

Name	Created	Status
aws-node-52rb4	October 29, 2025, 02:28 (UTC-04:00)	Running
aws-node-f2zq8	October 29, 2025, 02:28 (UTC-04:00)	Running
coredns-6b9575c64c-8w6bk	October 29, 2025, 02:24 (UTC-04:00)	Running
coredns-6b9575c64c-mz66p	October 29, 2025, 02:24 (UTC-04:00)	Running
flask-app-deployment-67cc77889f-qj84z	7 minutes ago	Running
flask-app-deployment-67cc77889f-qqt44	7 minutes ago	Running
kube-proxy-2qb75	October 29, 2025, 02:28 (UTC-04:00)	Running

5. Test Readiness:

- Make readiness check a bad path

```
kubectl patch deploy $DEPLOY --type='json' -p='[{"op":"replace","path":"/spec/template/spec/containers/0/readinessProbe/httpGet/path","value":"/nope"}]'
```

- Watch it go NotReady (READY column becomes 0/1)

```
kubectl get pods -l app=flask-app -w
```

- Confirm Service stops sending traffic to that pod

```
kubectl get endpoints $SVC -w
```

```

CloudShell
us-east-1 + Actions ▾

~ $ kubectl -n "$NS" patch deploy "$DEPLOY" --type='json' -p='[{"op":"replace","path":"/spec/template/spec/containers/0/readinessProbe/httpGet/path","value":"/nope"}]'
deployment.apps/flask-app-deployment patched
~ $ kubectl -n "$NS" get pods -l app=flask-app -w
NAME READY STATUS RESTARTS AGE
flask-app-deployment-67cc77889f-bhsxx 1/1 Terminating 0 39s
flask-app-deployment-67cc77889f-qj84z 1/1 Running 0 21h
flask-app-deployment-bcfc4bc-x5k2n 0/1 Running 0 7s
flask-app-deployment-bcfc4bc-zwj9m 0/1 Running 0 7s
^C- $ kubectl -n "$NS" get endpoints "$SVC" -w
NAME ENDPOINTS AGE
flask-app-service 192.168.55.128:5000 5d12h

```

d. Restore and then check rollout

```
kubectl -n "$NS" patch deploy "$DEPLOY" --type='json' -p='[{"op":"replace","path":"/spec/template/spec/containers/0/readinessProbe/httpGet/path","value":"/"}]'
```

```
kubectl -n "$NS" rollout status deploy/"$DEPLOY" --timeout=2m
```

```

CloudShell
us-east-1 + Actions ▾

~ $ kubectl -n "$NS" patch deploy "$DEPLOY" --type='json' -p='[{"op":"replace","path":"/spec/template/spec/containers/0/readinessProbe/httpGet/path","value":"/"}]'
deployment.apps/flask-app-deployment patched
~ $ kubectl -n "$NS" rollout status deploy/"$DEPLOY" --timeout=2m
Waiting for deployment "flask-app-deployment" rollout to finish: 1 of 2 updated replicas are available...
deployment "flask-app-deployment" successfully rolled out
~ $ kubectl -n "$NS" get pods -l app=flask-app -w
NAME READY STATUS RESTARTS AGE
flask-app-deployment-67cc77889f-kdpp 1/1 Running 0 14s
flask-app-deployment-67cc77889f-qj84z 1/1 Running 0 21h
flask-app-deployment-bcfc4bc-x5k2n 0/1 Terminating 0 111s
flask-app-deployment-bcfc4bc-zwj9m 0/1 Terminating 0 111s

```

6. Test Liveness:

- Make liveness check a bad path so it fails repeatedly

```
kubectl patch deploy $DEPLOY --type='json' -p='[{"op":"replace","path":"/spec/template/spec/containers/0/livenessProbe/httpGet/path","value":"/nope"} ]'
```

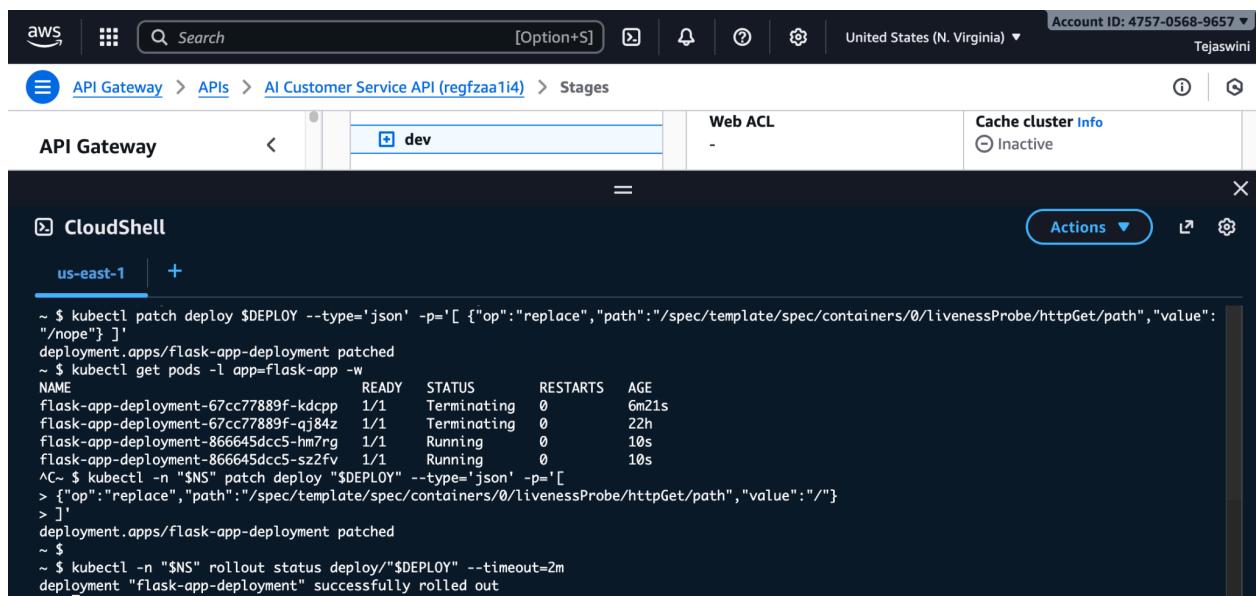
- Watch restarts increment

```
kubectl get pods -l app=flask-app -w
```

- Restore and then check rollout

```
kubectl -n "$NS" patch deploy "$DEPLOY" --type='json' -p='[ {"op":"replace","path":"/spec/template/spec/containers/0/livenessProbe/httpGet/path","value":"/"} ]'
```

```
kubectl -n "$NS" rollout status deploy/"$DEPLOY" --timeout=2m
```



```
aws | Search [Option+S] Account ID: 4757-0568-9657 ▾ United States (N. Virginia) ▾ Tejaswini
API Gateway > APIs > AI Customer Service API (regfzaa1i4) > Stages
API Gateway < dev Web ACL - Cache cluster Info
CloudShell Actions ▾
us-east-1 +
~ $ kubectl patch deploy $DEPLOY --type='json' -p='[ {"op":"replace","path":"/spec/template/spec/containers/0/livenessProbe/httpGet/path","value":"/nope"} ]'
deployment.apps/flask-app-deployment patched
~ $ kubectl get pods -l app=flask-app -w
NAME READY STATUS RESTARTS AGE
flask-app-deployment-67cc77889f-kdcpp 1/1 Terminating 0 6m21s
flask-app-deployment-67cc77889f-qj184z 1/1 Terminating 0 22h
flask-app-deployment-866645dcc5-hm7rg 1/1 Running 0 10s
flask-app-deployment-866645dcc5-sz2fv 1/1 Running 0 10s
^C~ $ kubectl -n "$NS" patch deploy "$DEPLOY" --type='json' -p='[ {"op":"replace","path":"/spec/template/spec/containers/0/livenessProbe/httpGet/path","value":"/"} ]'
deployment.apps/flask-app-deployment patched
~ $
~ $ kubectl -n "$NS" rollout status deploy/"$DEPLOY" --timeout=2m
deployment "flask-app-deployment" successfully rolled out
```

Part 8: Alerting (Extra Credit):

1. Install the monitoring stack:

We have used the **kube-prometheus-stack** Helm chart. It ships with default K8s alert rules like **KubePodNotReady**, **KubePodCrashLooping**, etc

```
export REGION=us-east-1
```

```
export CLUSTER=flask-eks-cluster
```

```
aws eks update-kubeconfig --region "$REGION" --name "$CLUSTER"
```

```
curl -fsSL
```

```
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

```
helm repo add prometheus-community
```

```
https://prometheus-community.github.io/helm-charts
```

```
helm repo update
```

```
helm upgrade --install kube-prometheus-stack
```

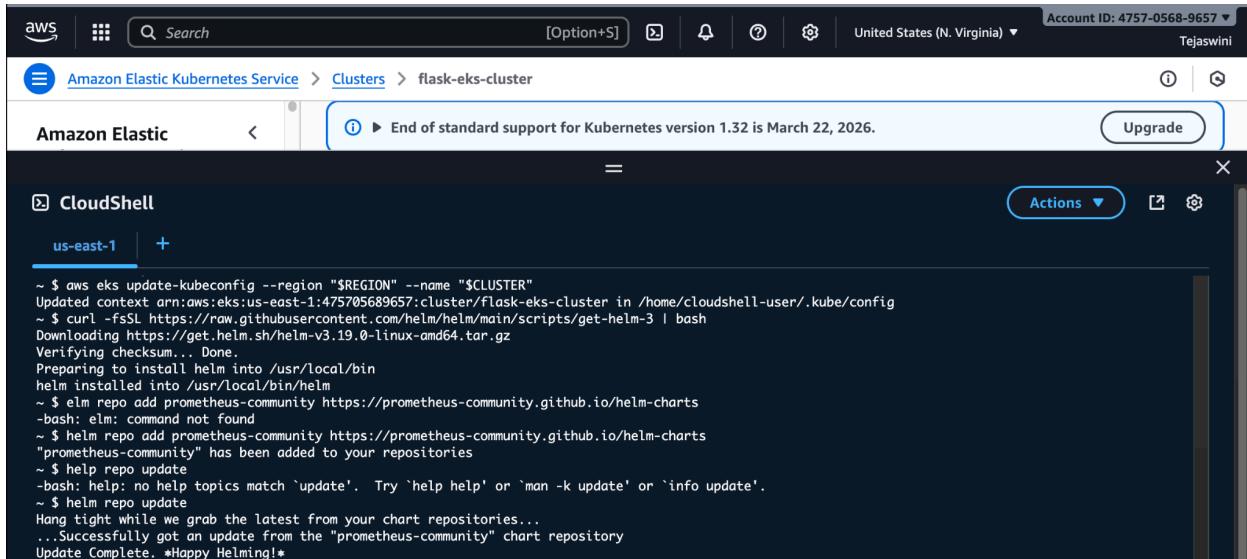
```
prometheus-community/kube-prometheus-stack -n monitoring  
--create-namespace
```

```
kubectl -n monitoring rollout status
```

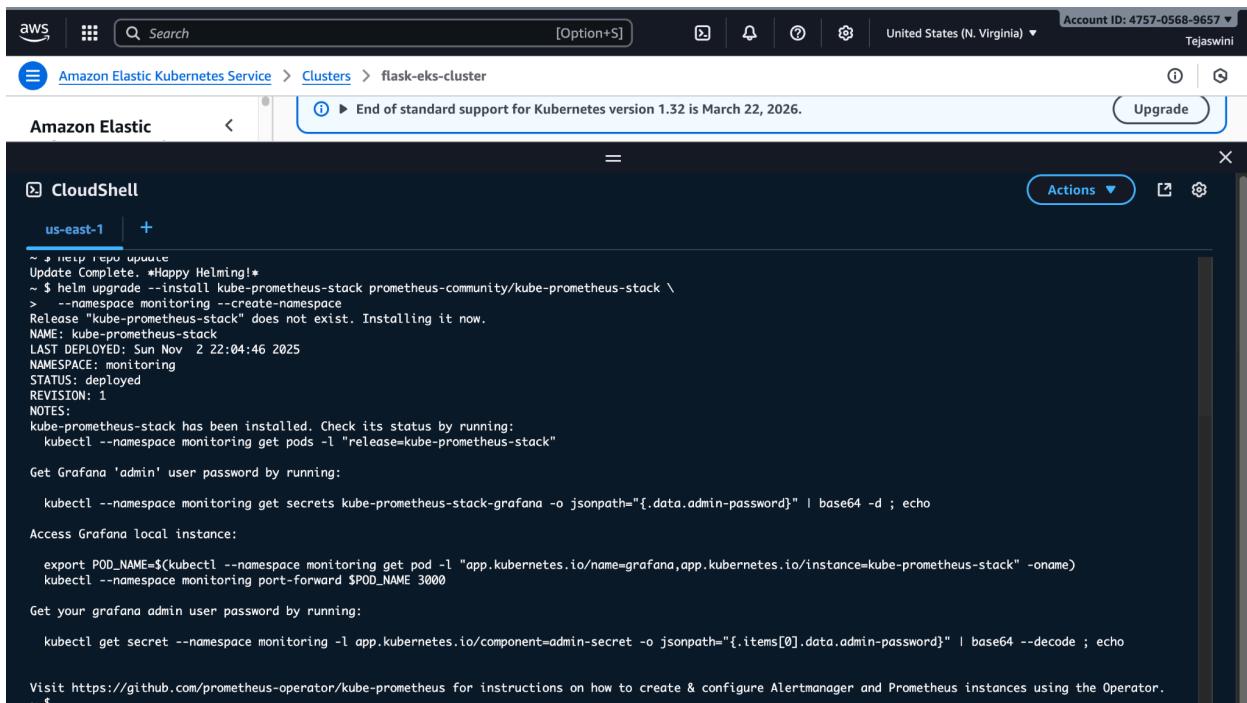
```
statefulset/kube-prometheus-stack-prometheus
```

```
kubectl -n monitoring rollout status
```

```
statefulset/kube-prometheus-stack-alertmanager
```



```
aws | Search [Option+S] Account ID: 4757-0568-9657 | United States (N. Virginia) | Tejaswini
Amazon Elastic Kubernetes Service > Clusters > flask-eks-cluster
Amazon Elastic < Upgrade Actions ▾
CloudShell
us-east-1 +
~ $ aws eks update-kubeconfig --region "$REGION" --name "$CLUSTER"
Updated context arn:aws:eks:us-east-1:475705689657:cluster/flask-eks-cluster in /home/cloudshell-user/.kube/config
~ $ curl -fsL https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
Downloading https://get.helm.sh/helm-v3.19.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
~ $ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
-bash: helm: command not found
~ $ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
~ $ help repo update
-bash: help: no help topics match `update'. Try `help help' or `man -k update' or `info update'.
~ $ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helming!*
```



```
aws | Search [Option+S] Account ID: 4757-0568-9657 | United States (N. Virginia) | Tejaswini
Amazon Elastic Kubernetes Service > Clusters > flask-eks-cluster
Amazon Elastic < Upgrade Actions ▾
CloudShell
us-east-1 +
~ + help helm upgrade
Update Complete. *Happy Helming!*
~ $ helm upgrade --install kube-prometheus-stack prometheus-community/kube-prometheus-stack \
> --namespace monitoring --create-namespace
Release "kube-prometheus-stack" does not exist. Installing it now.
NAME: kube-prometheus-stack
LAST DEPLOYED: Sun Nov  2 22:04:46 2025
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=kube-prometheus-stack"

Get Grafana 'admin' user password by running:
  kubectl --namespace monitoring get secrets kube-prometheus-stack-grafana -o jsonpath="{.data.admin-password}" | base64 -d ; echo

Access Grafana local instance:
  export POD_NAME=$(kubectl --namespace monitoring get pod -l "app.kubernetes.io/name=grafana,app.kubernetes.io/instance=kube-prometheus-stack" -o name)
  kubectl --namespace monitoring port-forward $POD_NAME 3000

Get your grafana admin user password by running:
  kubectl get secret --namespace monitoring -l app.kubernetes.io/component=admin-secret -o jsonpath=".items[0].data.admin-password" | base64 --decode ; echo

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
~ $
```

```
~ $ kubectl get pods -n monitoring
NAME                               READY   STATUS    RESTARTS   AGE
alertmanager-kube-prometheus-stack-alertmanager-0   2/2     Running   0          83s
kube-prometheus-stack-grafana-bf5cc8d8c-px7fz        3/3     Running   0          87s
kube-prometheus-stack-kube-state-metrics-557fd457c6-dbdkqm 1/1     Running   0          87s
kube-prometheus-stack-operator-6556d5f5f48-5jhkt      1/1     Running   0          87s
kube-prometheus-stack-prometheus-node-exporter-454kn  1/1     Running   0          87s
kube-prometheus-stack-prometheus-node-exporter-69wrq   1/1     Running   0          86s
prometheus-kube-prometheus-stack-prometheus-0         2/2     Running   0          83s
```

Prometheus:

**kubectl -n monitoring port-forward
svc/kube-prometheus-stack-prometheus 9090:9090**

```
~ $ kubectl -n monitoring port-forward svc/kube-prometheus-stack-prometheus 9090:9090
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
```

Prometheus

Query Alerts Status

Enter expression (press Shift+Enter for newlines)

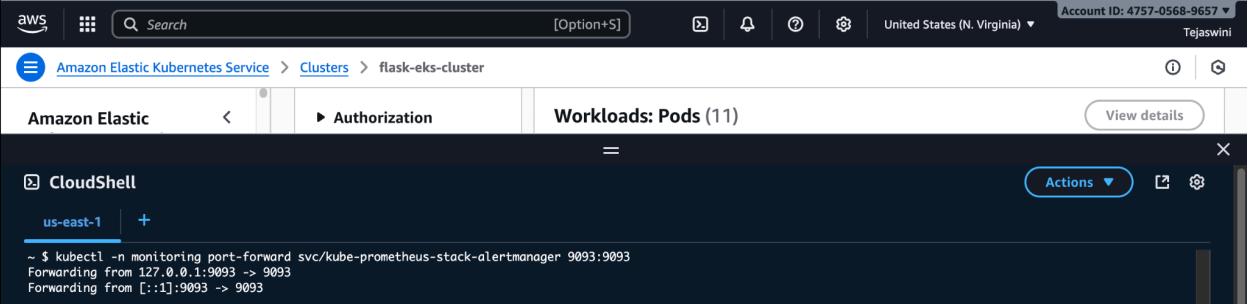
Execute

No data queried yet

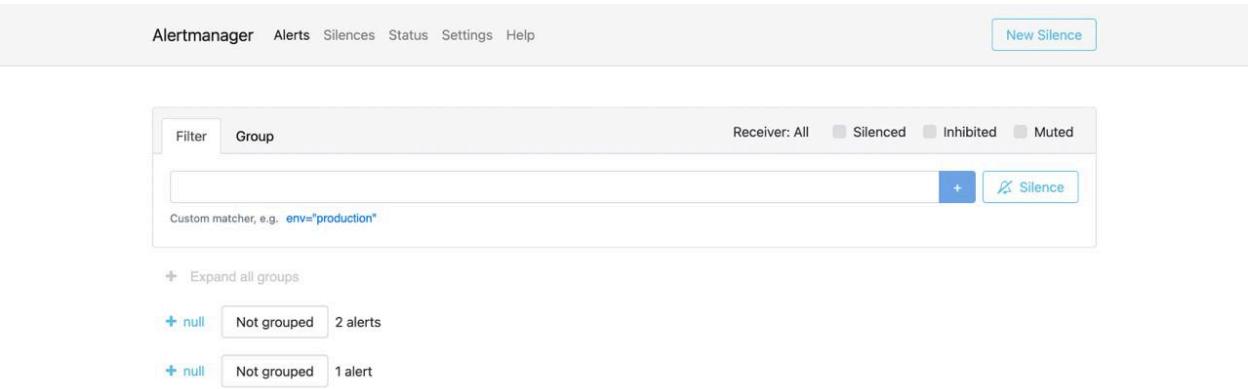
+ Add query

Alertmanager:

**kubectl -n monitoring port-forward
svc/kube-prometheus-stack-alertmanager 9093:9093**



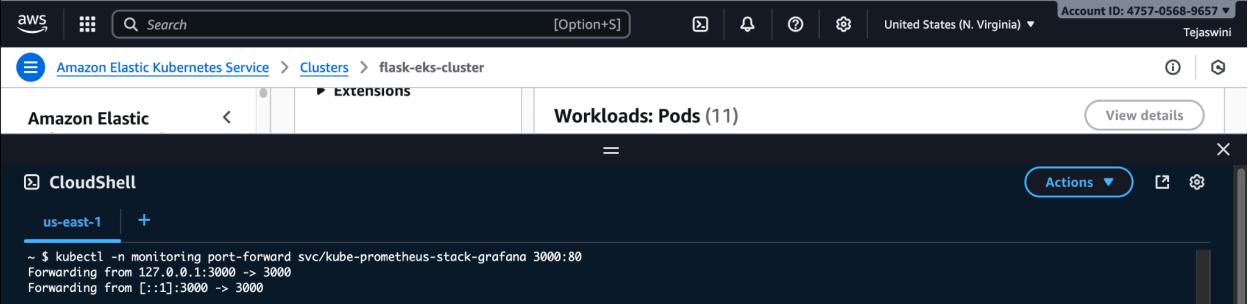
A screenshot of the AWS CloudShell interface. The title bar shows "Amazon Elastic Kubernetes Service > Clusters > flask-eks-cluster". The main area is titled "Workloads: Pods (11)". A terminal window is open with the command: ~ \$ kubectl -n monitoring port-forward svc/kube-prometheus-stack-alertmanager 9093:9093. The output shows two forwarding entries: Forwarding from 127.0.0.1:9093 -> 9093 and Forwarding from [::1]:9093 -> 9093.



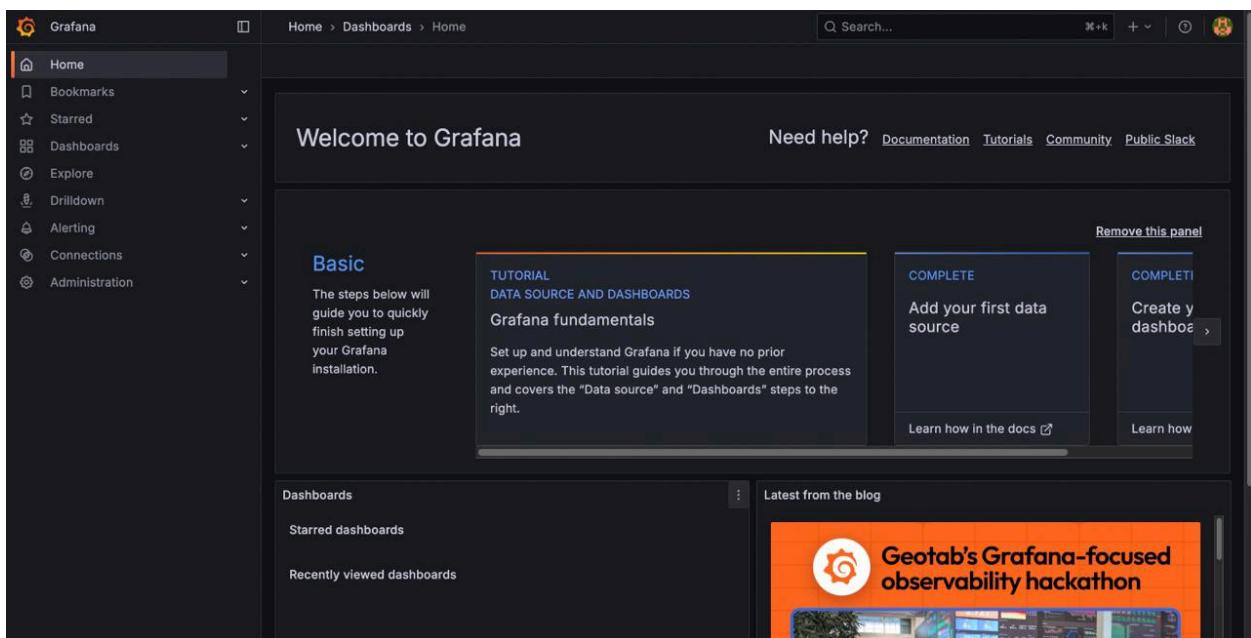
A screenshot of the Alertmanager interface. The top navigation bar includes links for Alertmanager, Alerts, Silences, Status, Settings, and Help. On the right, there is a "New Silence" button. The main area displays alert filtering options ("Filter", "Group") and a search bar. It shows a list of alerts grouped by receiver: "All" (2 alerts) and "Silenced" (1 alert). A "Custom matcher, e.g. env='production'" input field is also present.

Grafana:

kubectl -n monitoring port-forward svc/kube-prometheus-stack-grafana 3000:80



```
aws [Option+S] Search United States (N. Virginia) ▾ Account ID: 4757-0568-9657 ▾ Tejaswini
Amazon Elastic Kubernetes Service > Clusters > flask-eks-cluster
Amazon Elastic Workloads: Pods (11)
CloudShell Actions View details
us-east-1 +
~ $ kubectl -n monitoring port-forward svc/kube-prometheus-stack-grafana 3000:80
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```



2. Integrating Slack:

a. Added Webhook:

```
secret/slack-webhook configured
~ $ kubectl -n monitoring create secret generic slack-webhook --from-literal=url='https://hooks.slack.com/services/T09PV9KT2BH/B09PVA588UX/dBhnqz0NdZqkspx
W63UnIEUZ' --dry-run=client -o yaml | kubectl apply -f -
secret/slack-webhook configured
~ $
```

```
~ $ kubectl -n monitoring create secret generic slack-webhook --from-literal=url='https://hooks.slack.com/services/T09PV9KT2BH/B09PVA588UX/dBhnqz0NdZqkspx
W63UnIEUZ' --dry-run=client -o yaml | kubectl apply -f -
secret/slack-webhook configured
~ $ cat <<'YAML' | kubectl -n monitoring apply -f -
> apiVersion: monitoring.coreos.com/v1alpha1
> kind: AlertmanagerConfig
> metadata:
>   name: slack-notify
>   labels:
>     alertmanagerConfig: "true"
> spec:
> route:
>   receiver: slack-default
>   groupBy: ["job"]
>   groupWait: 30s
>   groupInterval: 5m
>   repeatInterval: 4h
> receivers:
>   - name: slack-default
>     slackConfigs:
>       - channel: "#alerts"
>         sendResolved: true
>       apiURL:
>         name: slack-webhook
>         key: url
>       title: "[{{ .Status | toUpper }}] {{ .CommonLabels.alertname }}"
>       text: >-
>         *Cluster:* {{ .ExternalURL }}
>         *Severity:* {{ .CommonLabels.severity }}
>         {{- range .Alerts }}
>           • *{{ .Labels.alertname }}* on `{{ .Labels.instance }}` - {{ .Annotations.summary }} {{ .Annotations.description }}
>           {{- end }}
> YAML
alertmanagerconfig.monitoring.coreos.com/slack-notify created
```

b. Test:

```
us-east-1 X + 
~ >
~ $ cat <<'YAML' | kubectl -n monitoring apply -f -
> apiVersion: monitoring.coreos.com/v1
> kind: PrometheusRule
> metadata:
>   name: test-slack-alert
>   labels: { release: kube-prometheus-stack }
> spec:
>   groups:
>     - name: test
>       rules:
>         - alert: AlwaysFiring
>           expr: vector(1)
>           for: 1m
>           labels:
>             severity: critical
>             namespace: monitoring # <-- required to match your routes
>           annotations:
>             summary: "Test Slack alert"
>             description: "This is a test alert from Alertmanager via Slack."
> YAML
prometheusrule.monitoring.coreos.com/test-slack-alert configured
```

```

us-east-1 ✘ + prometheusrule.monitoring.coreos.com/test-slack-alert configured
~ $ kubectl -n monitoring exec -it alertmanager-kube-prometheus-stack-alertmanager-0 -c alertmanager -- \
> wget -qO http://localhost:9093/api/v2/alerts | sed -n '1,200p'
[{"annotations": {"description": "This is a test alert from Alertmanager via Slack.", "summary": "Test Slack alert"}, "endsAt": "2025-11-03T01:13:26.033Z", "fingerprint": "00d08ca3b52b1744", "receivers": [{"name": "monitoring/slack-notify/slack-default"}, {"name": "monitoring/smtp-notify/ses-email"}], "startsAt": "2025-11-03T01:06:26.033Z", "status": "inhibitedBy": [], "mutedBy": []}, {"annotations": {"description": "KubeScheduler has disappeared from Prometheus target discovery.", "summary": "Target disappeared from Prometheus target discovery"}, "endsAt": "2025-11-03T01:12:43.895Z", "fingerprint": "4891a0dfaf68c63fb", "receivers": [{"name": "null"}], "startsAt": "2025-11-02T22:20:43.895Z", "status": "inhibitedBy": [], "mutedBy": []}, {"annotations": {"description": "KubeControllerManager has disappeared from Prometheus target discovery.", "summary": "KubeControllerManager has disappeared from Prometheus target discovery"}, "endsAt": "2025-11-03T01:14:09.910Z", "fingerprint": "5a0799653ad781b1", "receivers": [{"name": "null"}], "startsAt": "2025-11-02T22:20:39.910Z", "status": "inhibitedBy": [], "mutedBy": []}, {"annotations": {"description": "KubeControllerManagerDown", "summary": "KubeControllerManagerDown"}, "endsAt": "2025-11-03T01:14:01.259Z", "fingerprint": "7da6e02b588fb5", "receivers": [{"name": "null"}], "startsAt": "2025-11-02T22:05:31.259Z", "status": "inhibitedBy": [], "mutedBy": []}, {"annotations": {"description": "Watchdog", "summary": "Watchdog"}, "endsAt": "2025-11-03T01:10:09.912Z", "fingerprint": "http://kube-prometheus-stack-prometheus.monitoring:9090/graph?g0.expr=absent%28up%7bjob%3D%22kube-controller-manager%22%7D%3D%3D1%29\u0026g0.tab=1", "receivers": [{"name": "null"}], "startsAt": "2025-11-02T22:09:43.897Z", "status": "inhibitedBy": [], "mutedBy": []}, {"annotations": {"description": "An alert that should always be firing to certify that Alertmanager is working properly.", "summary": "An alert that should always be firing to certify that Alertmanager is working properly"}, "endsAt": "2025-11-03T01:14:01.259Z", "fingerprint": "7da6e02b588fb5", "receivers": [{"name": "null"}], "startsAt": "2025-11-02T22:05:31.259Z", "status": "inhibitedBy": [], "mutedBy": []}, {"annotations": {"description": "AlwaysFiring", "summary": "AlwaysFiring"}, "endsAt": "2025-11-03T01:06:56.122Z", "fingerprint": "http://kube-prometheus-stack-alertmanager-0 -c alertmanager --tail=200 | grep -i slackwebhooknotify|status=success", "receivers": [{"name": "null"}], "startsAt": "2025-11-03T01:06:57.260Z", "status": "WARN", "msg": "Notify attempt failed, will retry later", "severity": "warning"}, {"annotations": {"description": "AlwaysFiring", "summary": "AlwaysFiring"}, "endsAt": "2025-11-03T01:06:57.260Z", "fingerprint": "http://kube-prometheus-stack-smtp-notify/ses-email", "receivers": [{"name": "null"}], "startsAt": "2025-11-03T01:06:57.260Z", "status": "WARN", "msg": "Notify attempt failed, will retry later", "severity": "warning"}]
~ $ 

```

Slack output:

The screenshot shows a Slack interface for a channel named '# alerts'. The channel was created today. A message from Slackbot congratulates the channel's creation with a video thumbnail and the text 'YAY! CONGRATULATIONS!!!'.