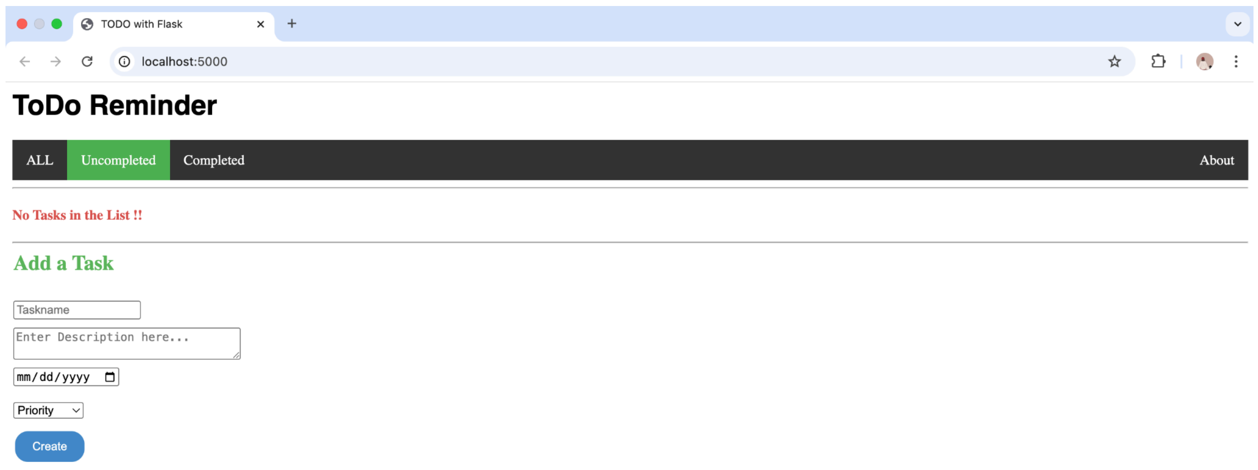
Part 1: Creating an Application

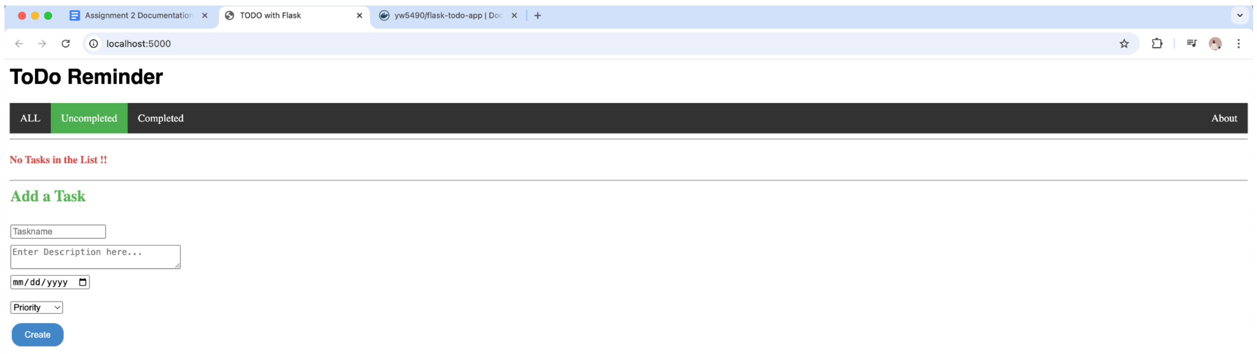
1. Download source code.
2. Create a Python virtual environment using the following command:
   - `python3 -m venv todo`
   - `source todo/bin/activate`
   - `pip install -r requirements.txt`
3. Run the Web Application with: `python app.py`
4. Visit http://localhost:5000 to check if the Web Application is running successfully as shown in the screenshot below:



Part 2: Containerizing the Application on Docker

1. Write `Dockerfile` and `docker-compose.yml`.
2. Run the following command to build the Docker image:
   - `docker build -t yw5490/flask-todo-app:latest .`
3. Create and start Docker containers with: `docker compose up`

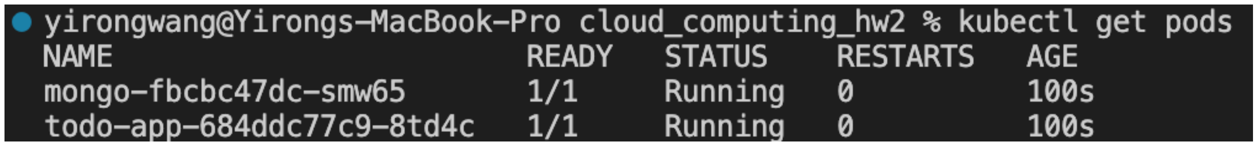
4. Visit http://localhost:5000 to check if the Web Application is running successfully as shown in the screenshot below:



5. Stop and remove Docker containers with: `docker compose down`
6. Push the Docker image to Docker Hub using:
   - `docker push yw5490/flask-todo-app:latest`
   - Link to Docker Hub Repository

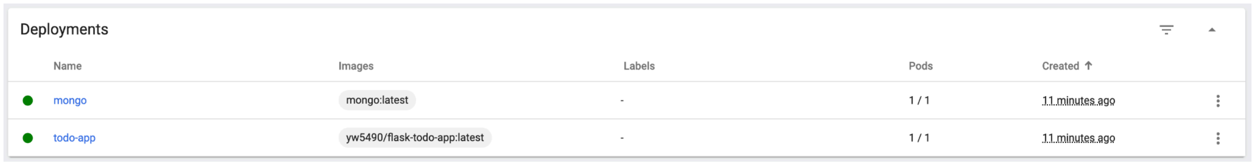
Part 3: Deploying the Application on Minikube

1. Write the `todo-app.yaml` and `mongo.yaml` file. Place the two files into a folder called `kube`.
2. Start Minikube using: `minikube start`
3. Deploy the Web Application using: `kubectl apply -f kube/`
4. Check that the pods have been successfully created using: `kubectl get pods`



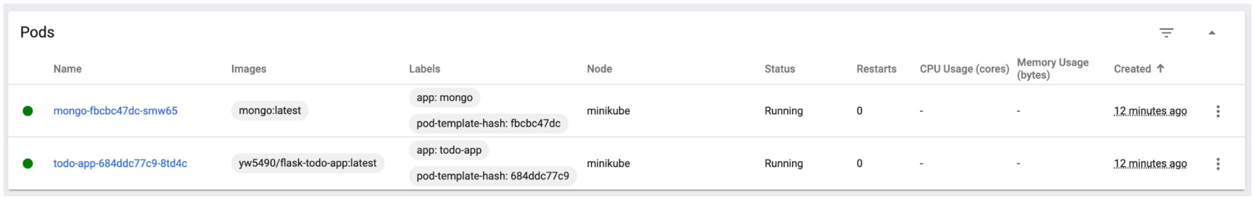
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-smw65         1/1     Running   0          100s
todo-app-684ddc77c9-8td4c     1/1     Running   0          100s
```

5. Check the status of Deployments and Pods on Kubernetes Dashboard using: `minikube dashboard`



Deployments

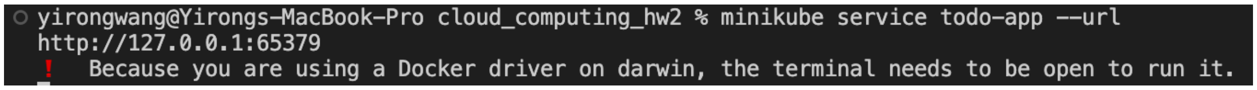
	Name	Images	Labels	Pods	Created ↑
●	mongo	mongo:latest	-	1 / 1	11 minutes ago
●	todo-app	yw5490/flask-todo-app:latest	-	1 / 1	11 minutes ago



Pods

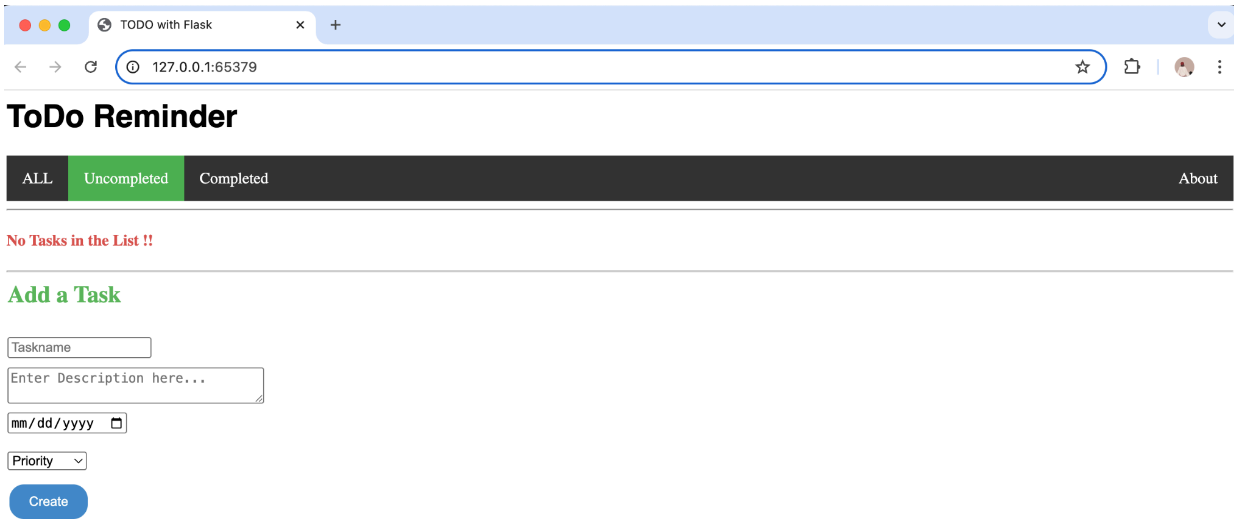
	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
●	mongo-fbcbc47dc-smw65	mongo:latest	app: mongo pod-template-hash: fbcbc47dc	minikube	Running	0	-	-	12 minutes ago
●	todo-app-684ddc77c9-8td4c	yw5490/flask-todo-app:latest	app: todo-app pod-template-hash: 684ddc77c9	minikube	Running	0	-	-	12 minutes ago

6. Test the Web Application by visiting the service URL provided when running the following command: `minikube service todo-app --url`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % minikube service todo-app --url
http://127.0.0.1:65379
!   Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

7. The Web Application should run successfully on Minikube as shown in the screenshot below:



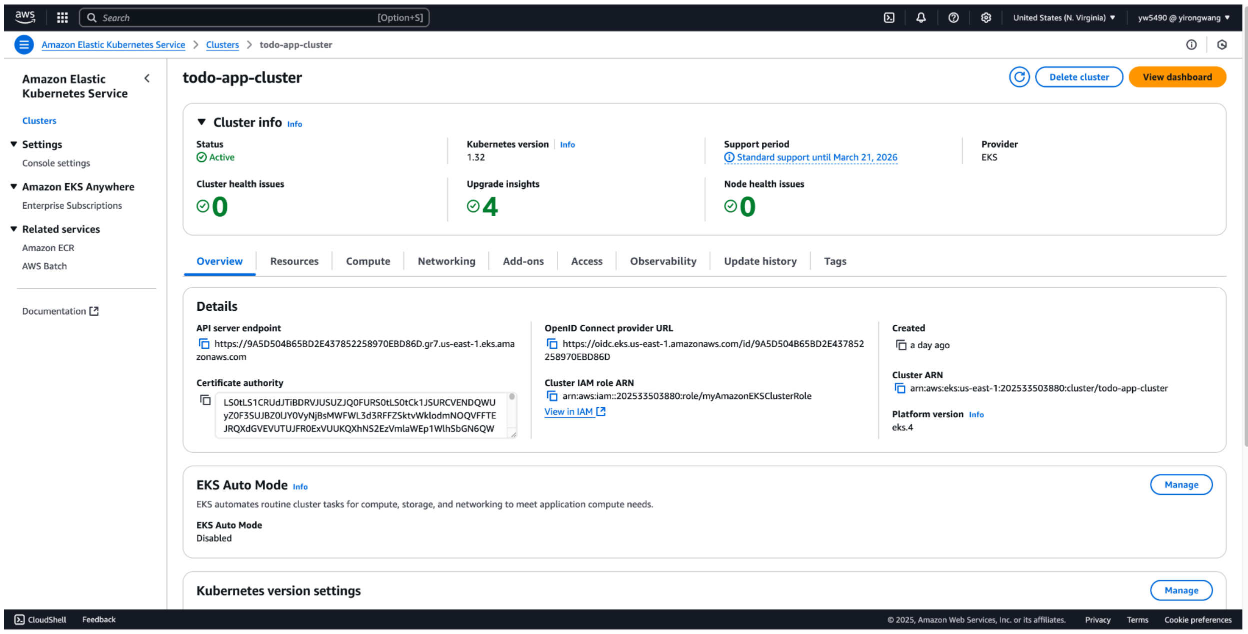
ToDo Reminder

No Tasks in the List !!

Add a Task

Part 4: Deploying the Application on AWS EKS

1. Create an AWS EKS cluster using this guide.



Amazon Elastic Kubernetes Service > Clusters > todo-app-cluster

todo-app-cluster

Cluster info

Status: Active

Kubernetes version: 1.32

Support period: Standard support until March 21, 2026

Provider: EKS

Cluster health issues: 0

Upgrade insights: 4

Node health issues: 0

Details

API server endpoint: https://9A5D504B65BD2E437852258970EBD86D.gr7.us-east-1.eks.amazonaws.com

Certificate authority: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJY0VyNjBsMWFWL3d3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QW

OpenID Connect provider URL: https://oidc.eks.us-east-1.amazonaws.com/id/9A5D504B65BD2E437852258970EBD86D

Cluster IAM role ARN: arn:aws:iam::202533503880:role/myAmazonEKSClusterRole

Created: a day ago

Cluster ARN: arn:aws:eks:us-east-1:202533503880:cluster/todo-app-cluster

Platform version: eks.4

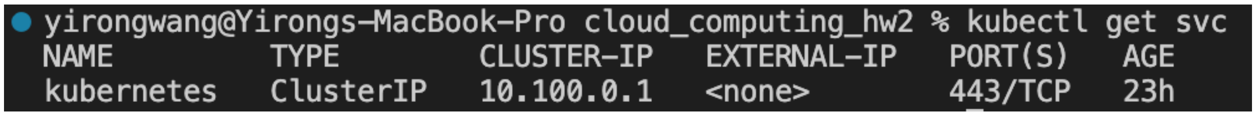
EKS Auto Mode

EKS automates routine cluster tasks for compute, storage, and networking to meet application compute needs.

EKS Auto Mode: Disabled

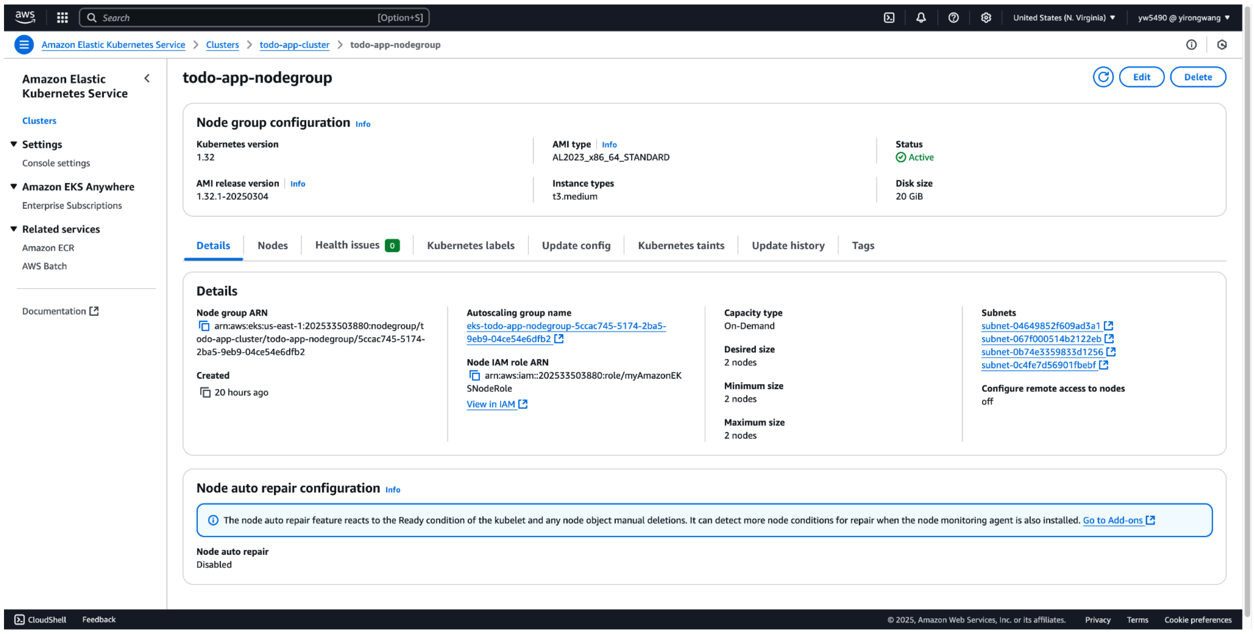
Kubernetes version settings

2. Configure the Kubernetes CLI (kubectl) to connect to the EKS cluster using: `aws eks update-kubeconfig --region us-east-1 --name todo-app-cluster`
3. Test the configuration using: `kubectl get svc`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   23h
```

4. Create EC2 Linux managed node group using this guide.



todo-app-nodegroup

Node group configuration

Kubernetes version: 1.32
AMI release version: 1.32.1-20250304
AMI type: AL2023_x86_64_STANDARD
Instance types: t3.medium
Status: Active
Disk size: 20 GiB

Details

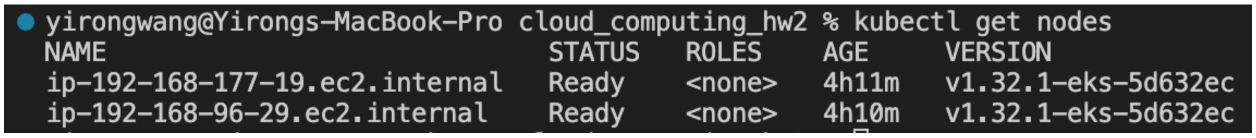
Node group ARN: arn:aws:eks:us-east-1:202533503880:nodegroup/todo-app-cluster/todo-app-nodegroup/5ccac745-5174-2ba5-9eb9-04ce54e6dfb2
Created: 20 hours ago
Autoscaling group name: eks-todo-app-nodegroup-5ccac745-5174-2ba5-9eb9-04ce54e6dfb2
Node IAM role ARN: arn:aws:iam::202533503880:role/myAmazonEKSNodeRole
Capacity type: On-Demand
Desired size: 2 nodes
Minimum size: 2 nodes
Maximum size: 2 nodes
Subnets: subnet-04649852f609ad3a1, subnet-067f000514b2122eb, subnet-0b74e3359833d1256, subnet-0c4fe7d56901fbebf
Configure remote access to nodes: off

Node auto repair configuration

The node auto repair feature reacts to the Ready condition of the kubelet and any node object manual deletions. It can detect more node conditions for repair when the node monitoring agent is also installed. Go to Add-ons

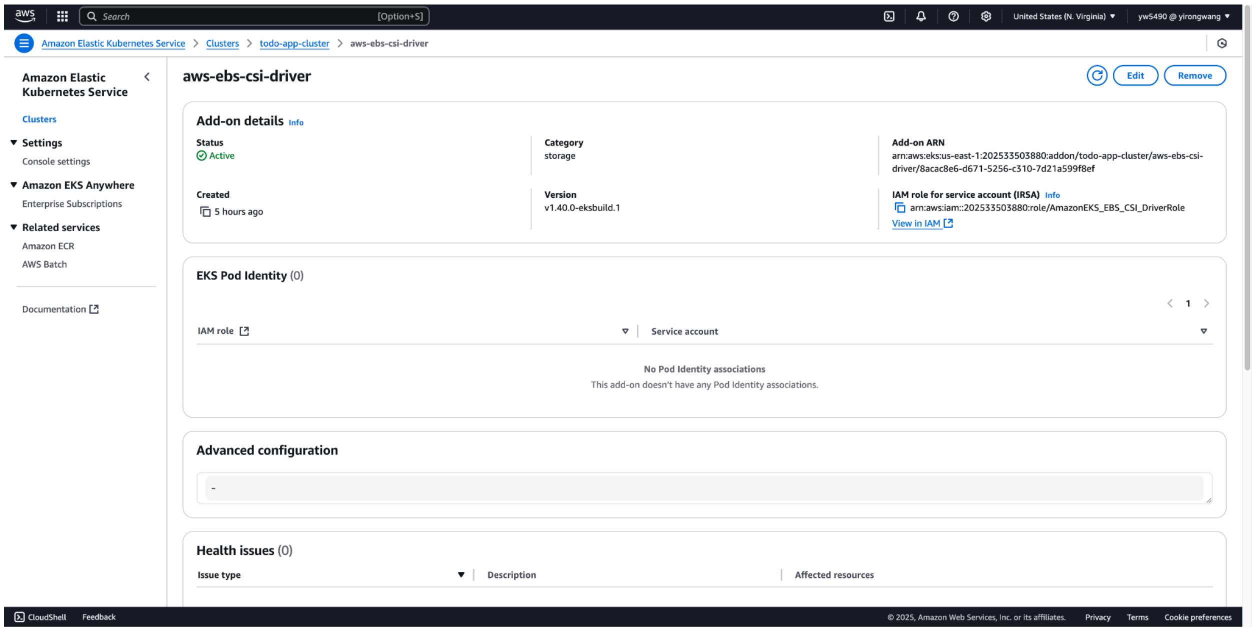
Node auto repair: Disabled

5. Check that nodes are created and are in "Ready" status using: `kubectl get nodes`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get nodes
NAME                             STATUS   ROLES    AGE     VERSION
ip-192-168-177-19.ec2.internal   Ready    <none>   4h11m   v1.32.1-eks-5d632ec
ip-192-168-96-29.ec2.internal    Ready    <none>   4h10m   v1.32.1-eks-5d632ec
```

6. Set up the Amazon EBS CSI driver through the AWS EKS add-on to utilize Amazon EBS volume as the persistent volume we used to store MongoDB data.



aws-ebs-csi-driver

Add-on details

Status: Active
Created: 5 hours ago
Category: storage
Version: v1.40.0-eksbuild.1
Add-on ARN: arn:aws:eks:us-east-1:202533503880:addon/todo-app-cluster/aws-ebs-csi-driver/8acac8e6-d671-5256-c310-7d21a599f8ef
IAM role for service account (IRSA): arn:aws:iam::202533503880:role/AmazonEKS_EBS_CSI_DriverRole

EKS Pod Identity (0)

IAM role | Service account

No Pod Identity associations
This add-on doesn't have any Pod Identity associations.

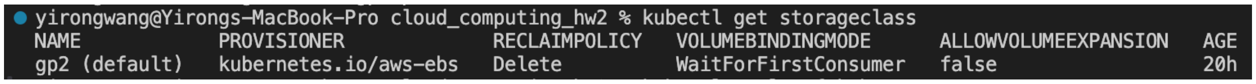
Advanced configuration

-

Health issues (0)

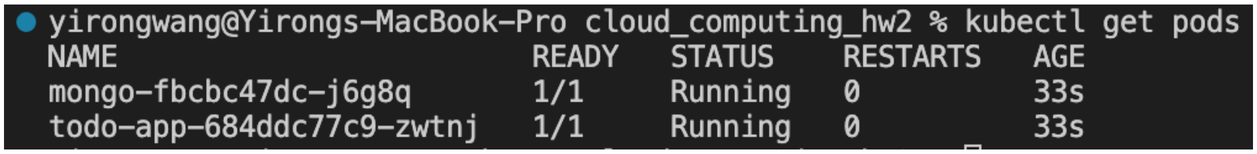
Issue type | Description | Affected resources

7. Make `gp2` the default storage class for any PersistentVolumeClaim that does not specify a storage class: `kubectl patch storageclass gp2 -p '{"metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'`



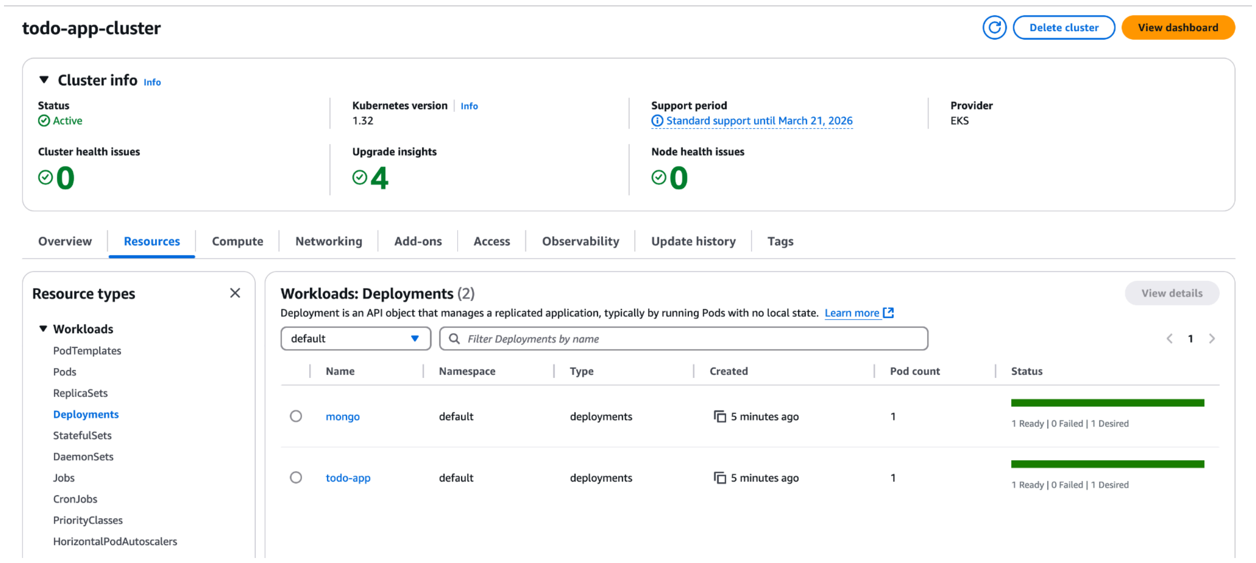
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get storageclass
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
gp2 (default)   kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   false                  20h
```

8. Build and push a multi-architecture image to the existing Docker Hub Repository with the `Dockerfile` written in Part 2 using: `docker buildx build --platform linux/amd64,linux/arm64 -t yw5490/flask-todo-app:latest --push .`
9. Deploy the Web Application using: `kubectl apply -f kube/`
10. Check that the pods have been successfully created using: `kubectl get pods`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-j6g8q       1/1     Running   0          33s
todo-app-684ddc77c9-zwtnj   1/1     Running   0          33s
```

11. Check the status of Deployments and Pods on AWS EKS Console:



todo-app-cluster

Cluster info Info

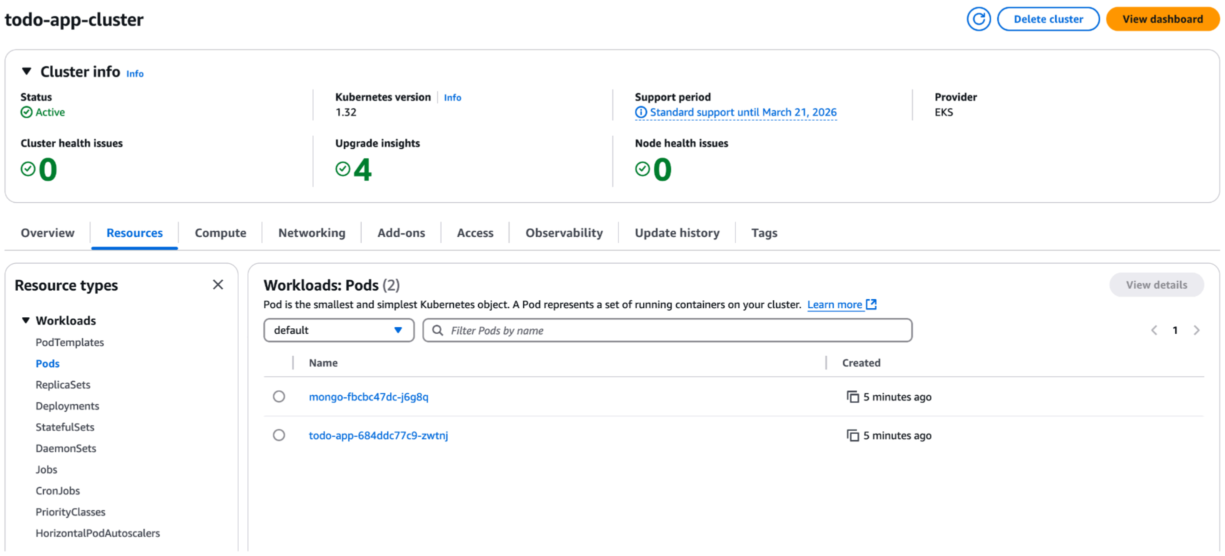
Status Active	**Kubernetes version** Info 1.32	**Support period** Standard support until March 21, 2026	**Provider** EKS
Cluster health issues 0	**Upgrade insights** 4	**Node health issues** 0	

Resource types

Workloads: Deployments (2)

Deployment is an API object that manages a replicated application, typically by running Pods with no local state. Learn more

Name	Namespace	Type	Created	Pod count	Status
mongo	default	deployments	5 minutes ago	1	1 Ready | 0 Failed | 1 Desired
todo-app	default	deployments	5 minutes ago	1	1 Ready | 0 Failed | 1 Desired



todo-app-cluster

Cluster info Info

Status Active	**Kubernetes version** Info 1.32	**Support period** Standard support until March 21, 2026	**Provider** EKS
Cluster health issues 0	**Upgrade insights** 4	**Node health issues** 0	

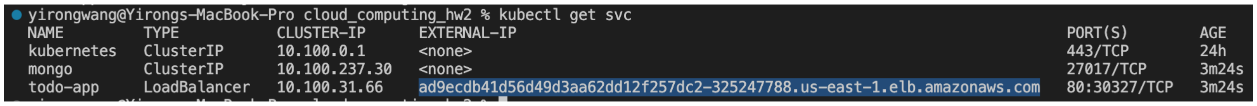
Resource types

Workloads: Pods (2)

Pod is the smallest and simplest Kubernetes object. A Pod represents a set of running containers on your cluster. Learn more

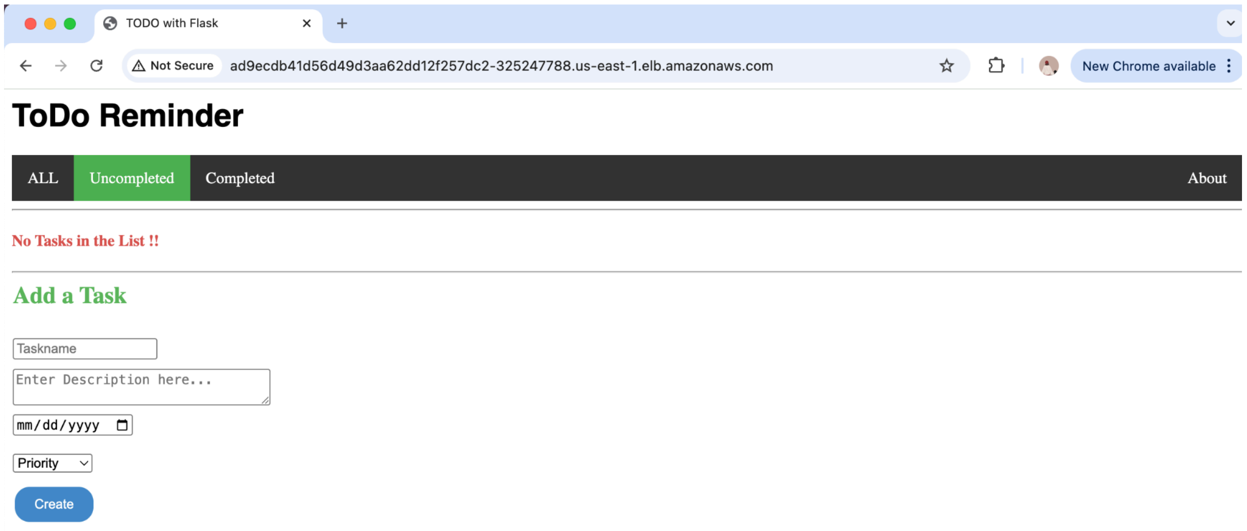
Name	Created
mongo-fbcbc47dc-j6g8q	5 minutes ago
todo-app-684ddc77c9-zwtnj	5 minutes ago

12. Test the Web Application by accessing the EXTERNAL-IP provided when running the following command: `kubectl get svc`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get svc
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)         AGE
kubernetes   ClusterIP      10.100.0.1      <none>                                                                    443/TCP         24h
mongo        ClusterIP      10.100.237.30   <none>                                                                    27017/TCP       3m24s
todo-app     LoadBalancer   10.100.31.66    ad9ecdb41d56d49d3aa62dd12f257dc2-325247788.us-east-1.elb.amazonaws.com    80:30327/TCP    3m24s
```

13. The Web Application should run successfully on AWS EKS as shown in the screenshot below:



ToDo Reminder

No Tasks in the List !!

Add a Task

Taskname

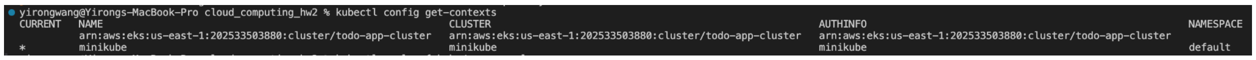
Enter Description here...

mm/dd/yyyy

Priority

Part 5: Replication Controller Feature

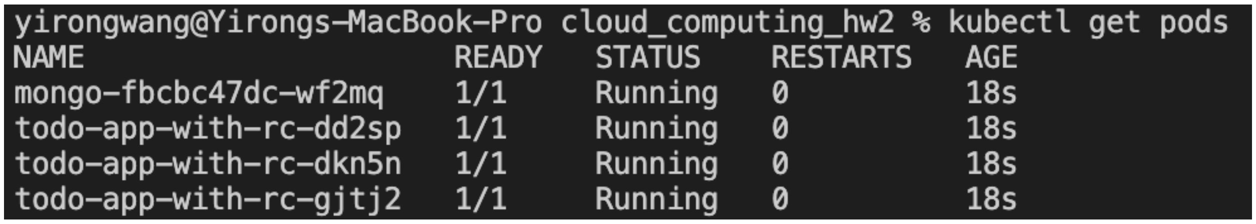
1. Write the `todo-app-rc.yaml` file, specify that 3 replicas of the application are always running.
2. Place the file into a folder called `kube-rc`, together with the `mongo.yaml` file we wrote in Part 3 and a `todo-app-load-balancer.yaml` file defining the Load Balancer Service, copied from Part 3.
3. Start Minikube using: `minikube start`
4. Check the current Kubernetes context using: `kubectl config get-contexts`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl config get-contexts
CURRENT   NAME                                                          CLUSTER                                                       AUTHINFO                                                      NAMESPACE
          arn:aws:eks:us-east-1:202533503880:cluster/todo-app-cluster   arn:aws:eks:us-east-1:202533503880:cluster/todo-app-cluster   arn:aws:eks:us-east-1:202533503880:cluster/todo-app-cluster
*         minikube                                                      minikube                                                      minikube                                                      default
```

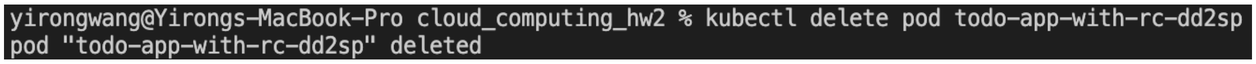
5. If the current context is not Minikube, set it to Minikube using: `kubectl config use-context minikube`
6. Deploy the Web Application with Replication Controller using: `kubectl apply -f kube-rc/`

7. Verify that 3 replicas of the application are created and running using: `kubectl get pods`



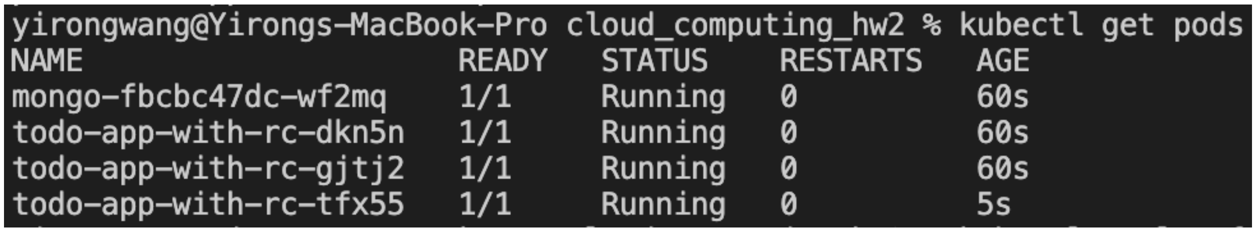
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-wf2mq     1/1     Running   0          18s
todo-app-with-rc-dd2sp    1/1     Running   0          18s
todo-app-with-rc-dkn5n    1/1     Running   0          18s
todo-app-with-rc-gjtj2    1/1     Running   0          18s
```

8. Test the Replication Controller by intentionally deleting one of the pods: `kubectl delete pod <pod-name>`



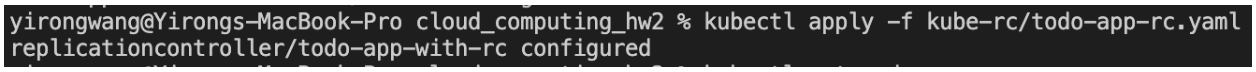
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl delete pod todo-app-with-rc-dd2sp
pod "todo-app-with-rc-dd2sp" deleted
```

9. Use `kubectl get pods` to verify that a new pod is automatically created by the Replication Controller to maintain that 3 replicas are running for the application:



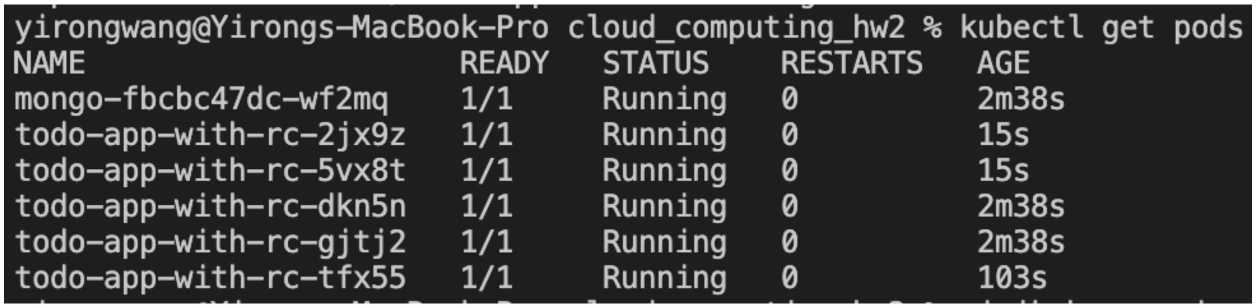
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-wf2mq     1/1     Running   0          60s
todo-app-with-rc-dkn5n    1/1     Running   0          60s
todo-app-with-rc-gjtj2    1/1     Running   0          60s
todo-app-with-rc-tfx55    1/1     Running   0          5s
```

10. Update the `todo-app-rc.yaml` file to set the desired number of replicas to 5.
11. Apply the changes to the running Replication Controller using: `kubectl apply -f kube-rc/todo-app-rc.yaml`



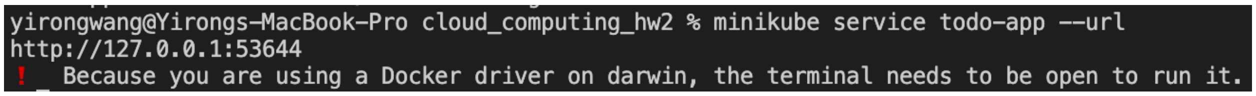
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl apply -f kube-rc/todo-app-rc.yaml
replicationcontroller/todo-app-with-rc configured
```

12. Use `kubectl get pods` to verify that the Replication Controller scales up the number of replicas to 5:



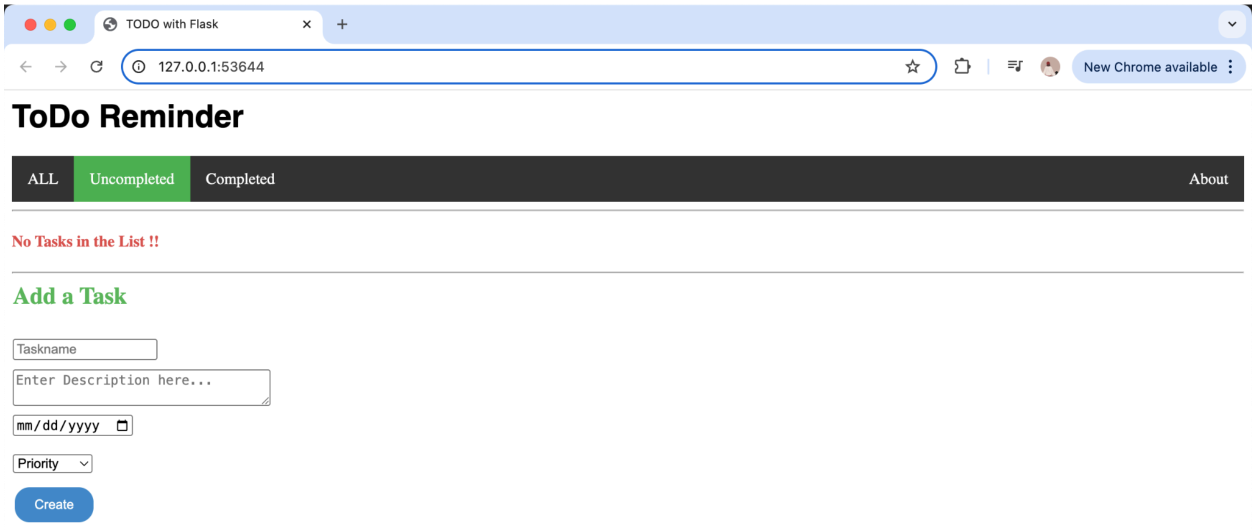
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-wf2mq     1/1     Running   0          2m38s
todo-app-with-rc-2jx9z    1/1     Running   0          15s
todo-app-with-rc-5vx8t    1/1     Running   0          15s
todo-app-with-rc-dkn5n    1/1     Running   0          2m38s
todo-app-with-rc-gjtj2    1/1     Running   0          2m38s
todo-app-with-rc-tfx55    1/1     Running   0          103s
```

13. Test the Web Application by visiting the service URL provided when running the following command: `minikube service todo-app --url`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % minikube service todo-app --url
http://127.0.0.1:53644
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

14. The Web Application should run successfully on Minikube as shown in the screenshot below:



ToDo Reminder

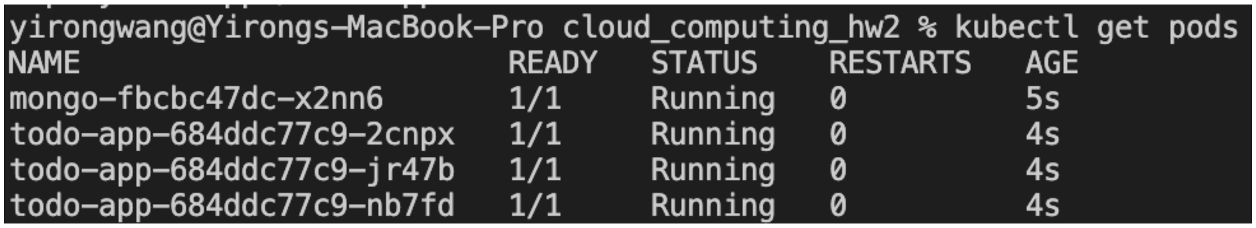
No Tasks in the List !!

Add a Task

Part 6: Rolling Update Strategy

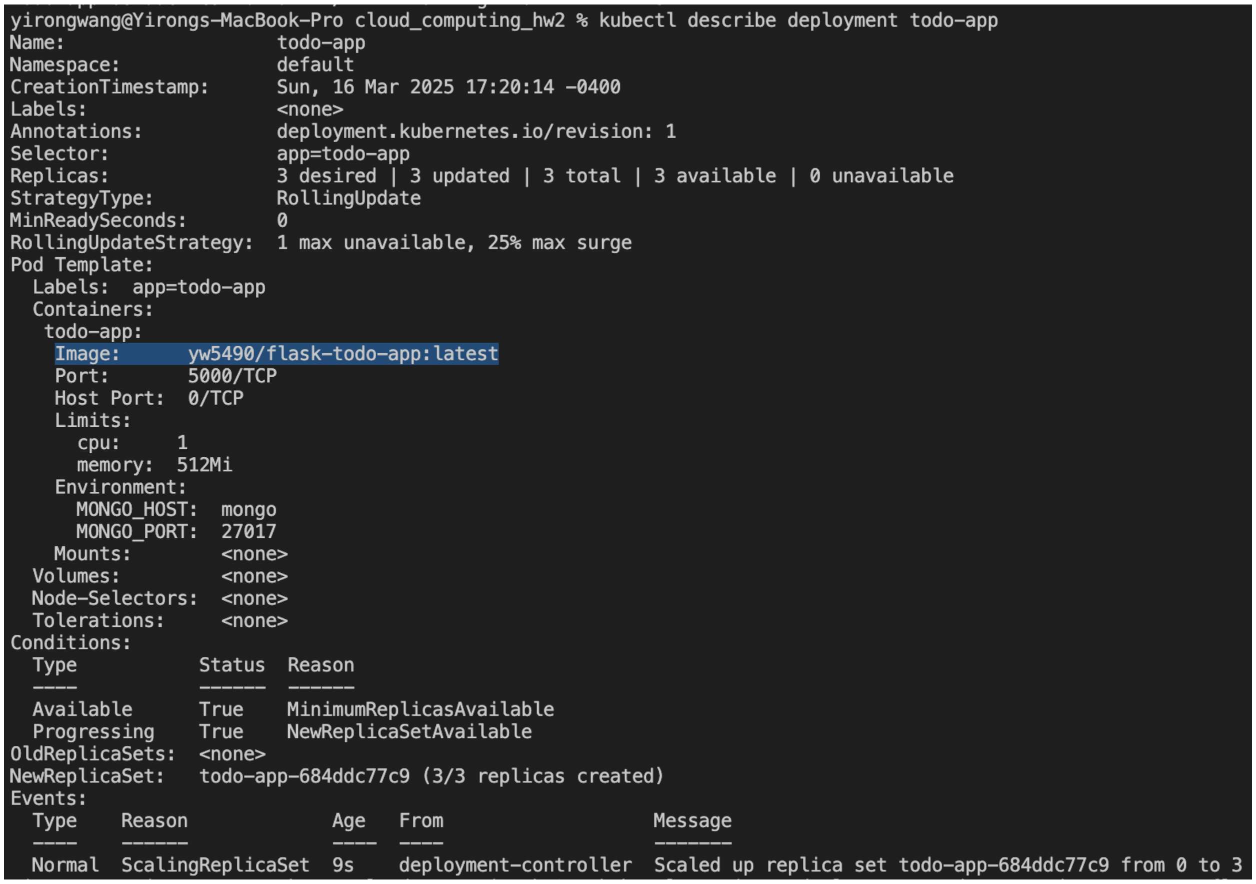
1. Update the `todo-app.yaml` file to set the update strategy to Rolling Update, and set the maximum number of pods that can be unavailable during the update to 1.
2. Push a new version of the Docker image to the existing Docker Hub Repository using the following command:
   - `docker tag yw5490/flask-todo-app:latest yw5490/flask-todo-app:v2`
   - `docker push yw5490/flask-todo-app:v2`
3. Start Minikube using: `minikube start`
4. Deploy the Web Application using: `kubectl apply -f kube/`

5. Check that the pods have been successfully created using: `kubectl get pods`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-x2nn6       1/1     Running   0          5s
todo-app-684ddc77c9-2cnpx   1/1     Running   0          4s
todo-app-684ddc77c9-jr47b   1/1     Running   0          4s
todo-app-684ddc77c9-nb7fd   1/1     Running   0          4s
```

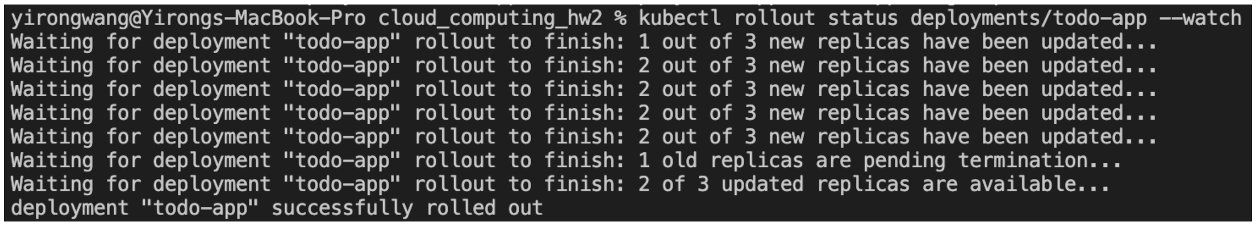
6. Check the current version of the Docker image used by the deployment: `kubectl describe deployment todo-app`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl describe deployment todo-app
Name:                   todo-app
Namespace:              default
CreationTimestamp:      Sun, 16 Mar 2025 17:20:14 -0400
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=todo-app
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 25% max surge
Pod Template:
  Labels:  app=todo-app
  Containers:
   todo-app:
    Image:      yw5490/flask-todo-app:latest
    Port:       5000/TCP
    Host Port:  0/TCP
    Limits:
      cpu:     1
      memory:  512Mi
    Environment:
      MONGO_HOST:  mongo
      MONGO_PORT:  27017
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   todo-app-684ddc77c9 (3/3 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  9s    deployment-controller  Scaled up replica set todo-app-684ddc77c9 from 0 to 3
```

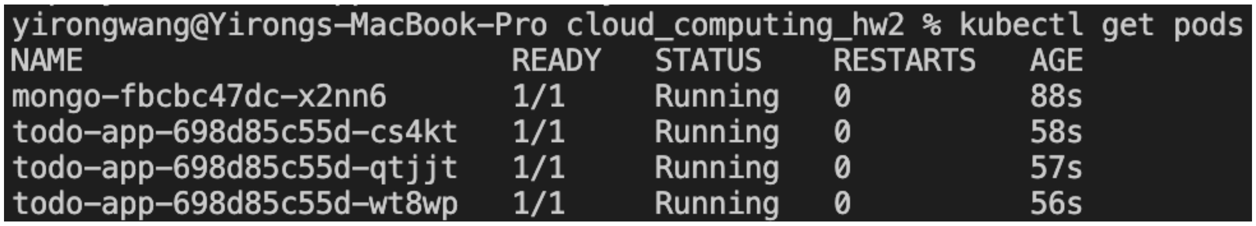
7. Trigger the Rolling Update by updating the deployment with the new Docker image version using: `kubectl set image deployments/todo-app todo-app=yw5490/flask-todo-app:v2`

8. Monitor the Rolling Update progress using: `kubectl rollout status deployments/todo-app --watch`



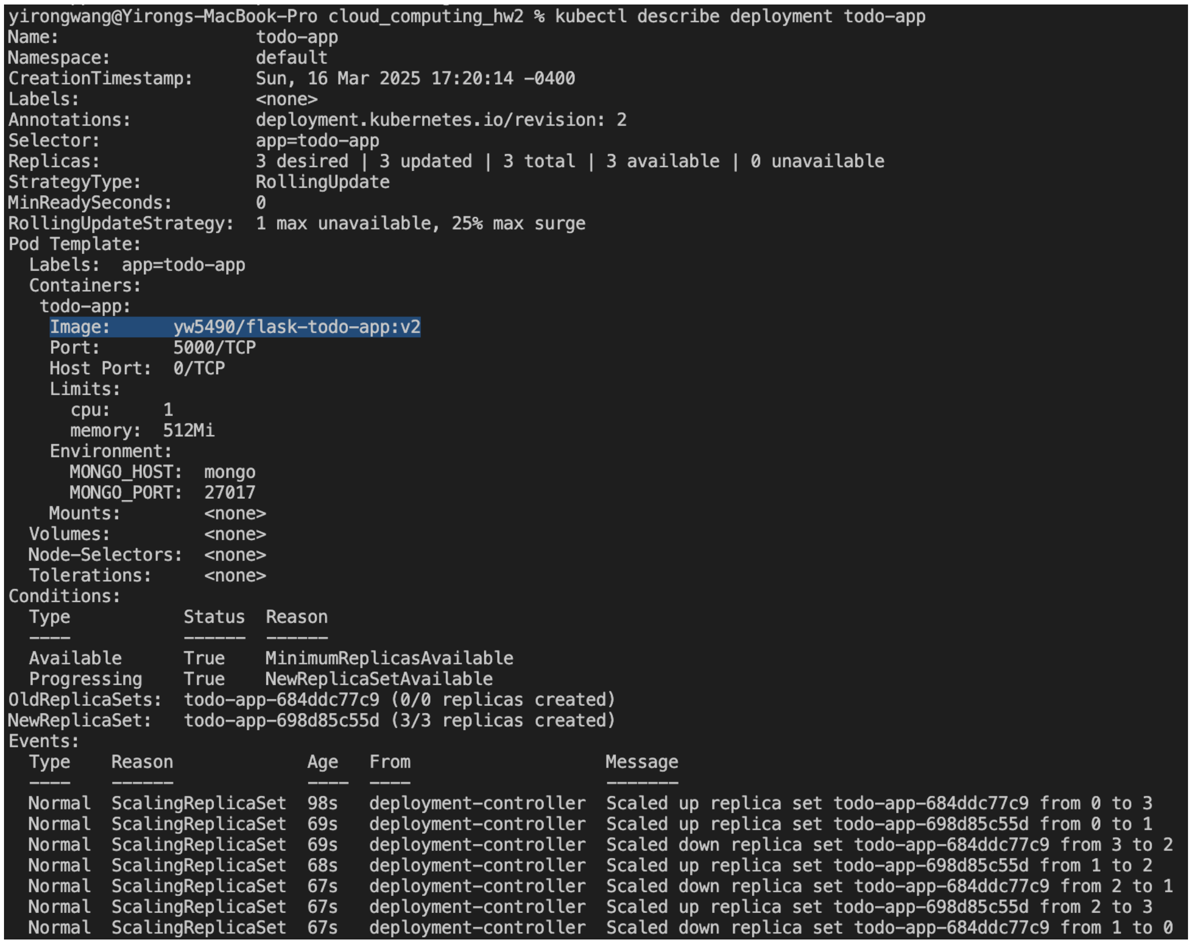
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl rollout status deployments/todo-app --watch
Waiting for deployment "todo-app" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "todo-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "todo-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "todo-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "todo-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "todo-app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "todo-app" rollout to finish: 2 of 3 updated replicas are available...
deployment "todo-app" successfully rolled out
```

9. Check that the pods have been successfully updated using: `kubectl get pods`



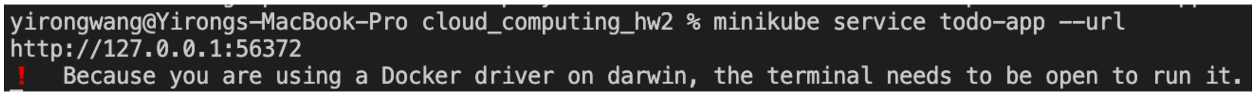
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-x2nn6        1/1     Running   0          88s
todo-app-698d85c55d-cs4kt    1/1     Running   0          58s
todo-app-698d85c55d-qtjjt    1/1     Running   0          57s
todo-app-698d85c55d-wt8wp    1/1     Running   0          56s
```

10. Check that the updated application deployment is running with the new Docker image version: `kubectl describe deployment todo-app`



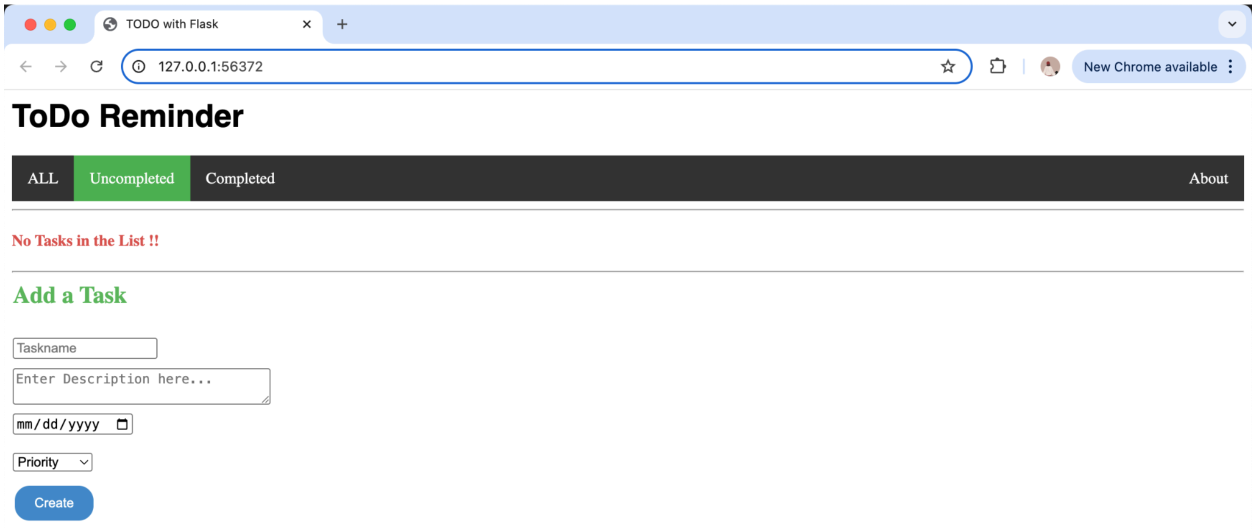
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl describe deployment todo-app
Name:                   todo-app
Namespace:              default
CreationTimestamp:      Sun, 16 Mar 2025 17:20:14 -0400
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=todo-app
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 25% max surge
Pod Template:
  Labels:  app=todo-app
  Containers:
   todo-app:
    Image:      yw5490/flask-todo-app:v2
    Port:       5000/TCP
    Host Port:  0/TCP
    Limits:
      cpu:     1
      memory:  512Mi
    Environment:
      MONGO_HOST:  mongo
      MONGO_PORT:  27017
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  todo-app-684ddc77c9 (0/0 replicas created)
NewReplicaSet:   todo-app-698d85c55d (3/3 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  98s   deployment-controller  Scaled up replica set todo-app-684ddc77c9 from 0 to 3
  Normal  ScalingReplicaSet  69s   deployment-controller  Scaled up replica set todo-app-698d85c55d from 0 to 1
  Normal  ScalingReplicaSet  69s   deployment-controller  Scaled down replica set todo-app-684ddc77c9 from 3 to 2
  Normal  ScalingReplicaSet  68s   deployment-controller  Scaled up replica set todo-app-698d85c55d from 1 to 2
  Normal  ScalingReplicaSet  67s   deployment-controller  Scaled down replica set todo-app-684ddc77c9 from 2 to 1
  Normal  ScalingReplicaSet  67s   deployment-controller  Scaled up replica set todo-app-698d85c55d from 2 to 3
  Normal  ScalingReplicaSet  67s   deployment-controller  Scaled down replica set todo-app-684ddc77c9 from 1 to 0
```

11. Test the Web Application by visiting the service URL provided when running the following command: `minikube service todo-app --url`



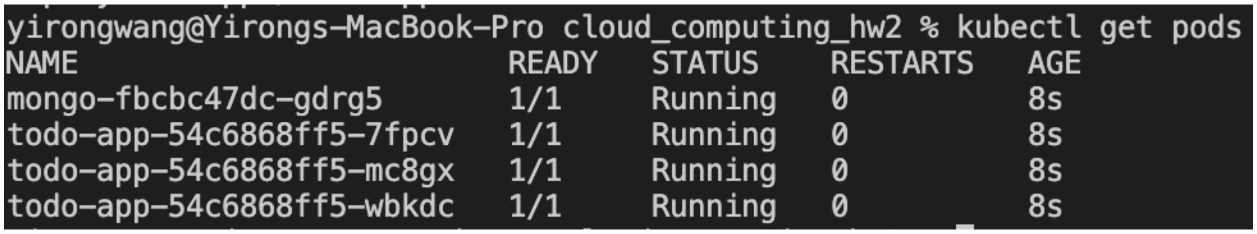
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % minikube service todo-app --url
http://127.0.0.1:56372
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

12. The Web Application should run successfully on Minikube as shown in the screenshot below:



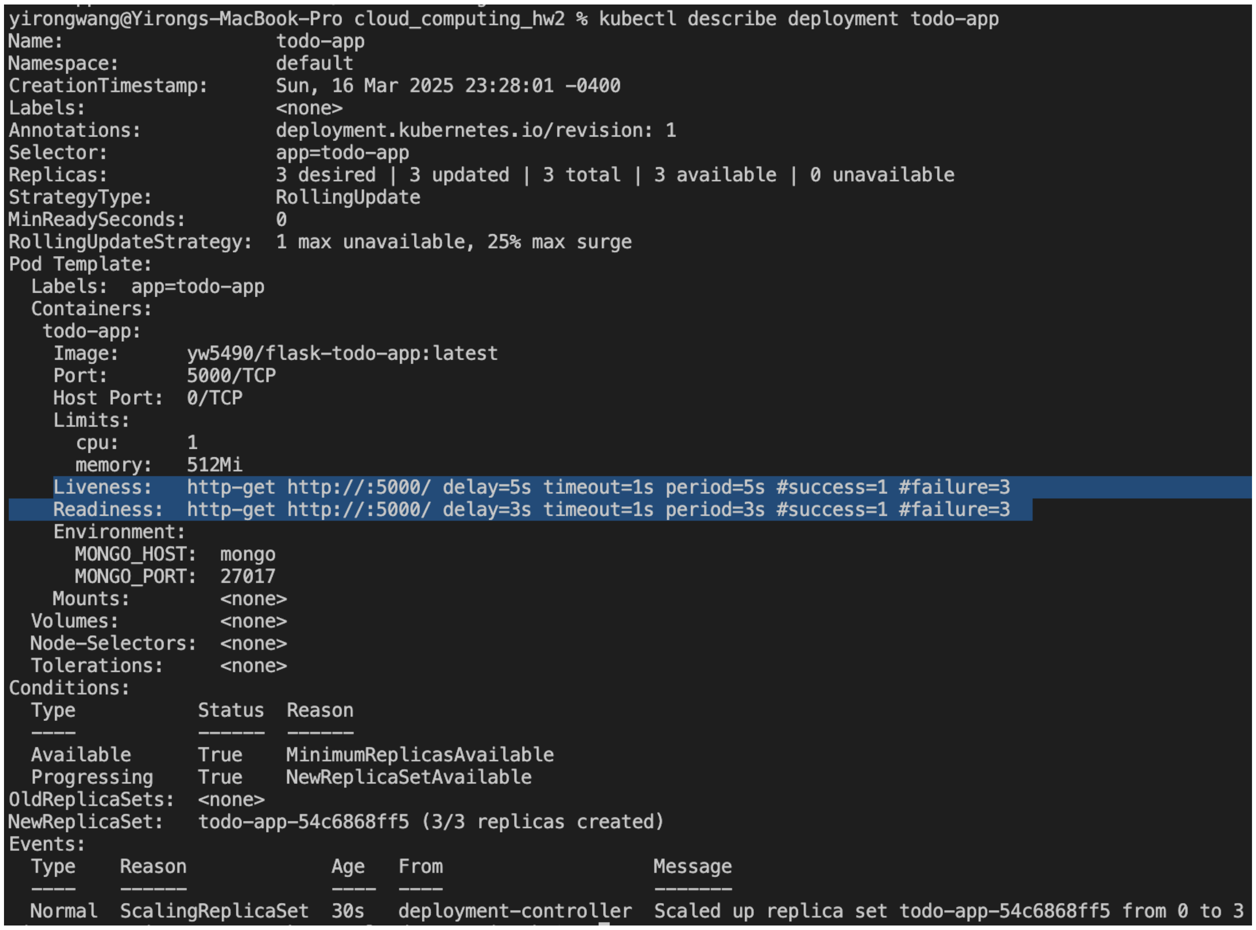
Part 7: Health Monitoring

1. Update the `todo-app.yaml` file to set up a liveness Probe and a readiness Probe for the pods.
2. Start Minikube using: `minikube start`
3. Deploy the Web Application using: `kubectl apply -f kube/`
4. Check that the pods have been successfully created using: `kubectl get pods`



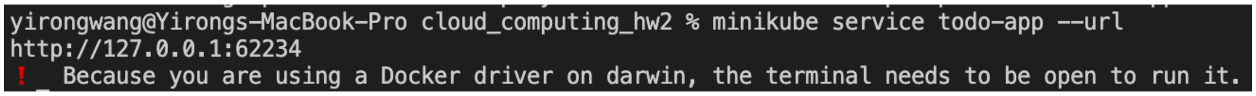
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-gdrg5       1/1     Running   0          8s
todo-app-54c6868ff5-7fpcv   1/1     Running   0          8s
todo-app-54c6868ff5-mc8gx   1/1     Running   0          8s
todo-app-54c6868ff5-wbkdc   1/1     Running   0          8s
```

5. Check that the liveness Probe and the readiness Probe has been successfully configured using: `kubectl describe deployment todo-app`



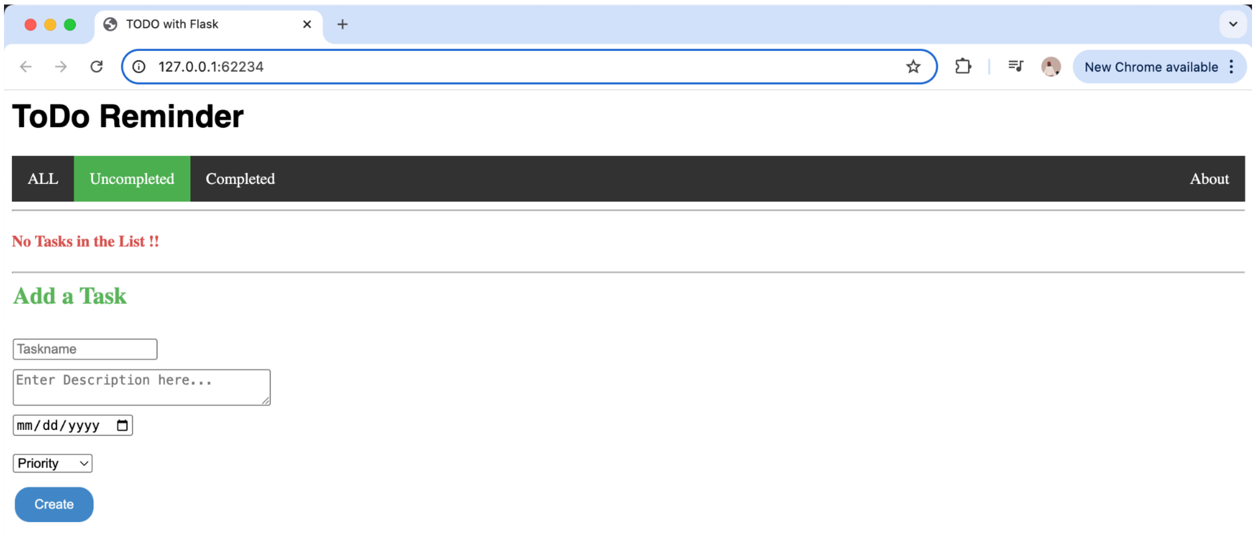
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl describe deployment todo-app
Name:                   todo-app
Namespace:              default
CreationTimestamp:      Sun, 16 Mar 2025 23:28:01 -0400
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=todo-app
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 25% max surge
Pod Template:
  Labels:  app=todo-app
  Containers:
   todo-app:
    Image:      yw5490/flask-todo-app:latest
    Port:       5000/TCP
    Host Port:  0/TCP
    Limits:
      cpu:      1
      memory:   512Mi
    Liveness:   http-get http://:5000/ delay=5s timeout=1s period=5s #success=1 #failure=3
    Readiness:  http-get http://:5000/ delay=3s timeout=1s period=3s #success=1 #failure=3
    Environment:
      MONGO_HOST:  mongo
      MONGO_PORT:  27017
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   todo-app-54c6868ff5 (3/3 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  30s   deployment-controller  Scaled up replica set todo-app-54c6868ff5 from 0 to 3
```

6. Test the Web Application by visiting the service URL provided when running the following command: `minikube service todo-app --url`

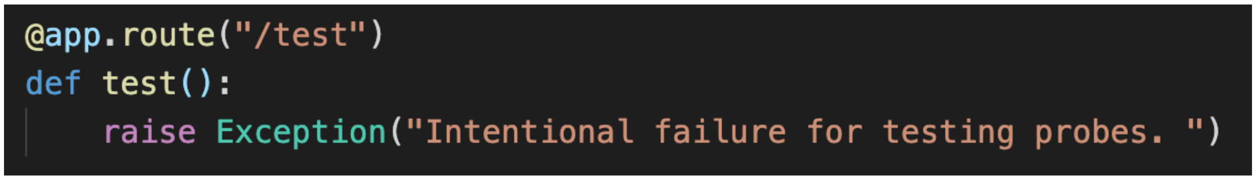


```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % minikube service todo-app --url
http://127.0.0.1:62234
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

7. The Web Application should run successfully on Minikube as shown in the screenshot below:

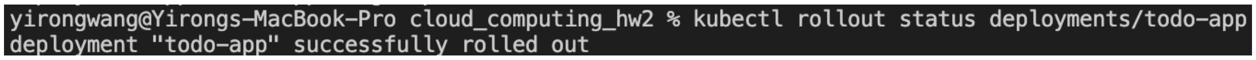


8. To test the health monitoring system, add an intentional failure to the application code as below:



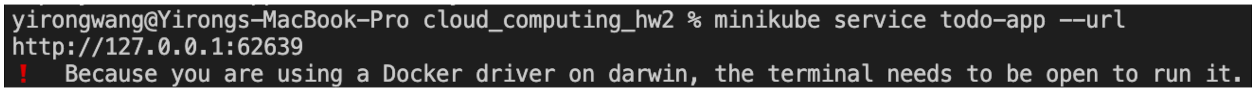
```
@app.route("/test")
def test():
    raise Exception("Intentional failure for testing probes. ")
```

9. Build and push a new Docker image version to the existing Docker Hub Repository with the updated application code using: `docker buildx build --platform linux/amd64,linux/arm64 -t yw5490/flask-todo-app:v3 --push .`
10. Update the deployment with the new Docker image version using: `kubectl set image deployments/todo-app todo-app=yw5490/flask-todo-app:v3`
11. Check that the new image version has been rolled out successfully using: `kubectl rollout status deployments/todo-app`



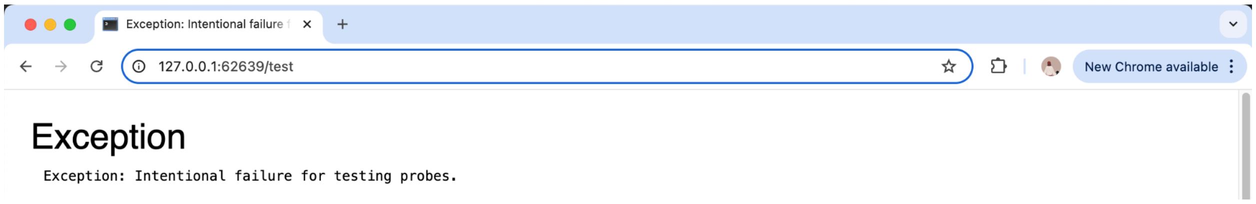
```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl rollout status deployments/todo-app
deployment "todo-app" successfully rolled out
```

12. Test the new application code by visiting the service URL provided when running the following command: `minikube service todo-app --url`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % minikube service todo-app --url
http://127.0.0.1:62639
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

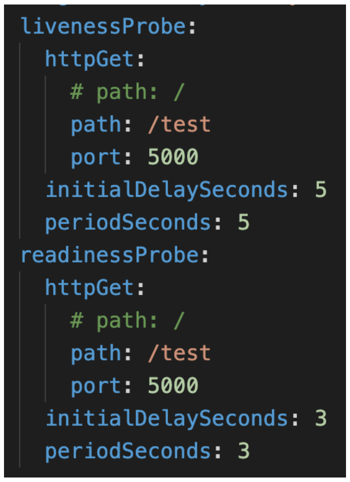
13. The Web Application should run into an Exception when visiting the `/test` route:



Exception

```
Exception: Intentional failure for testing probes.
```

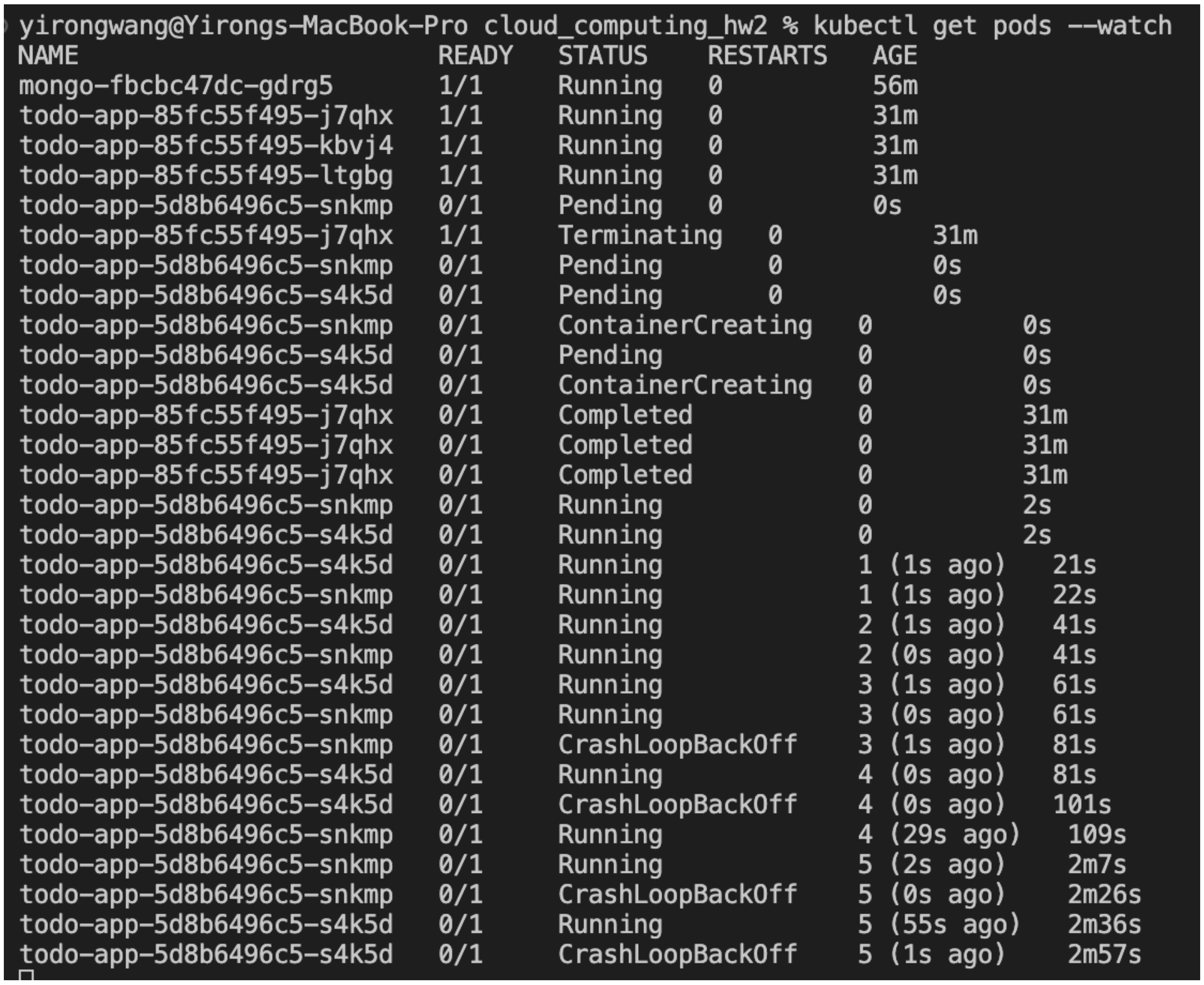
14. Temporarily modify `todo-app.yaml` to set the Docker image version to `yw5490/flask-todo-app:v3` (so that it doesn't conflict with the Docker image version update we completed above) and explicitly hit the failing endpoint `/test` as shown below:



```yaml
livenessProbe:
  httpGet:
    # path: /
    path: /test
    port: 5000
  initialDelaySeconds: 5
  periodSeconds: 5
readinessProbe:
  httpGet:
    # path: /
    path: /test
    port: 5000
  initialDelaySeconds: 3
  periodSeconds: 3
```

15. Apply the temporary changes using: `kubectl apply -f kube/todo-app.yaml`

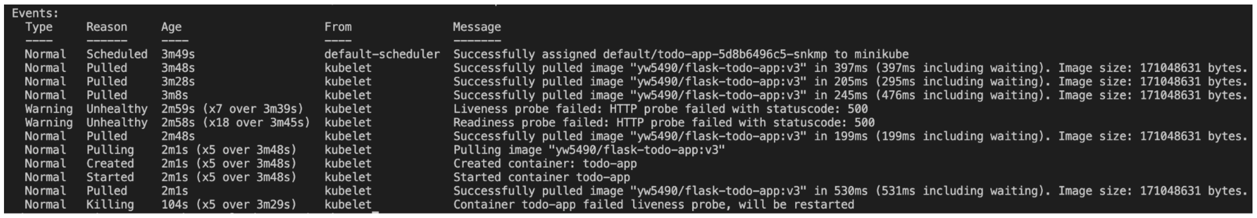
16. Monitor the health of the pods using: `kubectl get pods --watch`



```
yirongwang@Yirongs-MacBook-Pro cloud_computing_hw2 % kubectl get pods --watch
NAME                           READY   STATUS    RESTARTS   AGE
mongo-fbcbc47dc-gdrg5          1/1     Running   0          56m
todo-app-85fc55f495-j7qhx      1/1     Running   0          31m
todo-app-85fc55f495-kbvj4      1/1     Running   0          31m
todo-app-85fc55f495-ltgbg      1/1     Running   0          31m
todo-app-5d8b6496c5-snkmp      0/1     Pending   0          0s
todo-app-85fc55f495-j7qhx      1/1     Terminating   0          31m
todo-app-5d8b6496c5-snkmp      0/1     Pending       0          0s
todo-app-5d8b6496c5-s4k5d      0/1     Pending       0          0s
todo-app-5d8b6496c5-snkmp      0/1     ContainerCreating   0          0s
todo-app-5d8b6496c5-s4k5d      0/1     Pending             0          0s
todo-app-5d8b6496c5-s4k5d      0/1     ContainerCreating   0          0s
todo-app-85fc55f495-j7qhx      0/1     Completed           0          31m
todo-app-85fc55f495-j7qhx      0/1     Completed           0          31m
todo-app-85fc55f495-j7qhx      0/1     Completed           0          31m
todo-app-5d8b6496c5-snkmp      0/1     Running             0          2s
todo-app-5d8b6496c5-s4k5d      0/1     Running             0          2s
todo-app-5d8b6496c5-s4k5d      0/1     Running             1 (1s ago)   21s
todo-app-5d8b6496c5-snkmp      0/1     Running             1 (1s ago)   22s
todo-app-5d8b6496c5-s4k5d      0/1     Running             2 (1s ago)   41s
todo-app-5d8b6496c5-snkmp      0/1     Running             2 (0s ago)   41s
todo-app-5d8b6496c5-s4k5d      0/1     Running             3 (1s ago)   61s
todo-app-5d8b6496c5-snkmp      0/1     Running             3 (0s ago)   61s
todo-app-5d8b6496c5-snkmp      0/1     CrashLoopBackOff    3 (1s ago)   81s
todo-app-5d8b6496c5-s4k5d      0/1     Running             4 (0s ago)   81s
todo-app-5d8b6496c5-s4k5d      0/1     CrashLoopBackOff    4 (0s ago)   101s
todo-app-5d8b6496c5-snkmp      0/1     Running             4 (29s ago)   109s
todo-app-5d8b6496c5-snkmp      0/1     Running             5 (2s ago)    2m7s
todo-app-5d8b6496c5-snkmp      0/1     CrashLoopBackOff    5 (0s ago)    2m26s
todo-app-5d8b6496c5-s4k5d      0/1     Running             5 (55s ago)   2m36s
todo-app-5d8b6496c5-s4k5d      0/1     CrashLoopBackOff    5 (1s ago)    2m57s
```

17. Verify that Kubernetes takes the appropriate action when a Probe fails using:

`kubectl describe pod <pod-name>`



```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  3m49s                default-scheduler  Successfully assigned default/todo-app-5d8b6496c5-snkmp to minikube
  Normal   Pulled     3m48s                kubelet            Successfully pulled image "yw5490/flask-todo-app:v3" in 397ms (397ms including waiting). Image size: 171048631 bytes.
  Normal   Pulled     3m28s                kubelet            Successfully pulled image "yw5490/flask-todo-app:v3" in 205ms (295ms including waiting). Image size: 171048631 bytes.
  Normal   Pulled     3m8s                 kubelet            Successfully pulled image "yw5490/flask-todo-app:v3" in 245ms (476ms including waiting). Image size: 171048631 bytes.
  Warning  Unhealthy  2m59s (x7 over 3m39s)  kubelet          Liveness probe failed: HTTP probe failed with statuscode: 500
  Warning  Unhealthy  2m58s (x18 over 3m45s) kubelet          Readiness probe failed: HTTP probe failed with statuscode: 500
  Normal   Pulled     2m48s                kubelet            Successfully pulled image "yw5490/flask-todo-app:v3" in 199ms (199ms including waiting). Image size: 171048631 bytes.
  Normal   Pulling    2m1s (x5 over 3m48s)  kubelet           Pulling image "yw5490/flask-todo-app:v3"
  Normal   Created    2m1s (x5 over 3m48s)  kubelet           Created container: todo-app
  Normal   Started    2m1s (x5 over 3m48s)  kubelet           Started container todo-app
  Normal   Pulled     2m1s                 kubelet            Successfully pulled image "yw5490/flask-todo-app:v3" in 530ms (531ms including waiting). Image size: 171048631 bytes.
  Normal   Killing    104s (x5 over 3m29s)  kubelet           Container todo-app failed liveness probe, will be restarted
```