DISTRIBUTED SYSTEMS ASSIGNMENT REPORT

ASSIGNMENT REPORT

Assignment ID: Assignment4 - Exploring Kubernetes

Student Name: 王谦益

Student ID: 12111003

DESIGN

Task 0: K8s Deployment & Service

1. **modify the root API**

    1. create `app.py` and form root API, which returns pod name, pod IP, node name

    ```
    @app.route('/')
    def hello():
       pod_name = os.getenv('POD_NAME', 'unknown')
       pod_ip = os.getenv('POD_IP', 'unknown')
       node_name = os.getenv('NODE_NAME', 'unknown')
       return f"Hello from {pod_name}, IP: {pod_ip}, Node: {node_name}\n"
    ```

    2. initial `Dockerfile` and `requirements.txt`

    ```
    FROM python:3-slim
    WORKDIR /app
    COPY requirements.txt app.py ./
    RUN pip3 install --no-cache-dir -r requirements.txt
    CMD [ "python", "app.py" ]
    ```

    3. write `kind-config.yaml` file to create a 4-node k8s cluster (1 control plane + 3 worker nodes)

    ```
    kind: Cluster
    apiVersion: kind.x-k8s.io/v1alpha4
    ```

```
name: flask-cluster
nodes:
- role: control-plane
- role: worker
- role: worker
- role: worker
```

4. use `t0.yaml` to create a deployment and service
5. run `docker build -t flask-app:1.0.0 .` to get the image
6. run `kind create cluster --name t0 --config kind-config.yaml` to create the cluster
7. run `kind load docker-image flask-app:1.0.0 --name t0` to load the image into the cluster
8. run `kubectl apply -f t0.yaml` to create the deployment and service
9. run `kubectl get svc` to get `ip` address and run `docker exec -it t0-control-plane /bin/bash` to start the container
10. run `curl <ip>` to get the result

2. **graceful shutdown**

```
def graceful_shutdown(signum, frame):
   print("Gracefully shutting down...")
   sys.exit(0)

signal.signal(signal.SIGTERM, graceful_shutdown)
```

3. **greet-with-info API**
    1. add API in `app.py`

```
@app.route('/chat/<username>')
def greet_with_info(username):
   return f"Hello {username}, welcome to the Flask app!\n"
```

    2. fix `t0.yaml` by changing the `image: flask-app:1.0.0` to `image: flask-app:1.0.1`
    3. run `docker build -t flask-app:1.0.1 .` to get the new image
    4. run `kind load docker-image flask-app:1.0.1 --name t0` to update the image into the cluster
    5. run `kubectl apply -f t0.yaml` to change the deployment and service

6. run `kubectl get svc` to get `ip` address and run `docker exec -it t0-control-plane /bin/bash` to start the container

7. run `curl <ip>` and `curl <ip>/chat/<username>` to get the result

Task 1: K8s Pod Scheduling

1. delete the deployment and service of t0

2. write `kind-config.yaml` file to createa a 6-node k8s cluster (1 control plane + 5 worker nodes)

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
   labels:
      usage: normal
- role: worker
   labels:
      usage: normal
      capability: powerful
- role: worker
   kubeadmConfigPatches:
   - |
      kind: JoinConfiguration
      nodeRegistration:
      kubeletExtraArgs:
         node-labels: "usage=normal,capability=powerful"
      taints:
      - key: class
         value: "vip"
         effect: NoSchedule
- role: worker
   labels:
      usage: backup
- role: worker
   labels:
      usage: backup
```

3. use `t1.yaml` to create a deployment and service

4. run `kind create cluster --name t1 --config kind-config.yaml` to create the cluster

5. run `kind load docker-image flask-app:1.0.1 --name t1` to load the image into the cluster

6. run `kubectl apply -f t1.yaml` to create the deployment and service

7. run `kubectl scale deployment flask-app --replicas=<i>` from 1 to 5 and run `kubectl get pods -o wide` each time to get the result

8. improve `t1.yaml` by adding code

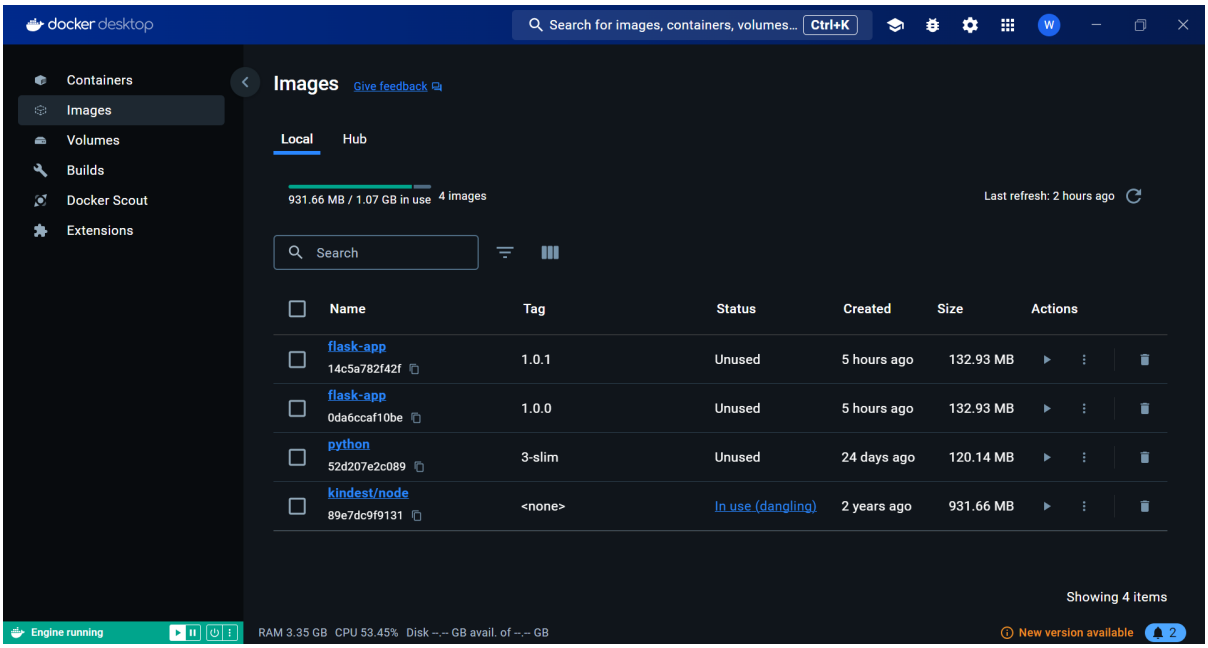
```
tolerations:
- key: "class"
   operator: "Equal"
   value: "vip"
   effect: "NoSchedule"
```

9. run `kubectl apply -f t1.yaml` to update the deployment and service and run `kubectl get pods -o wide` to get the result

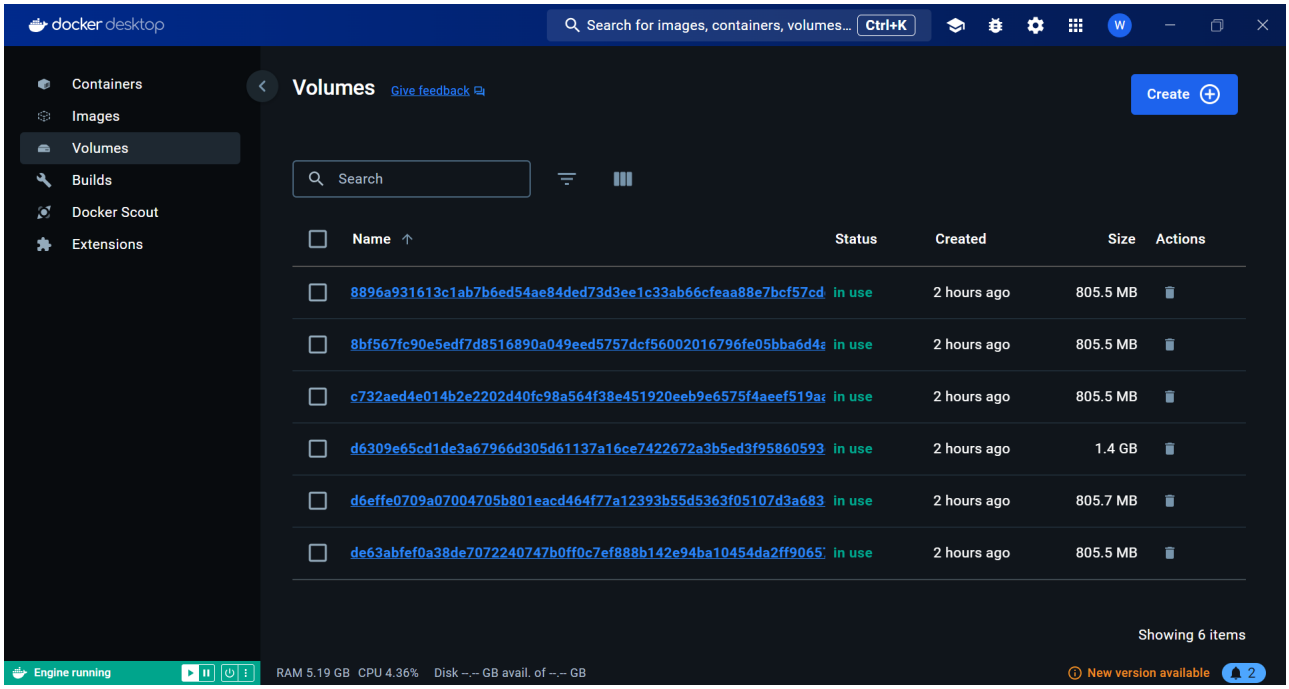
RUNNING RESULT

docker

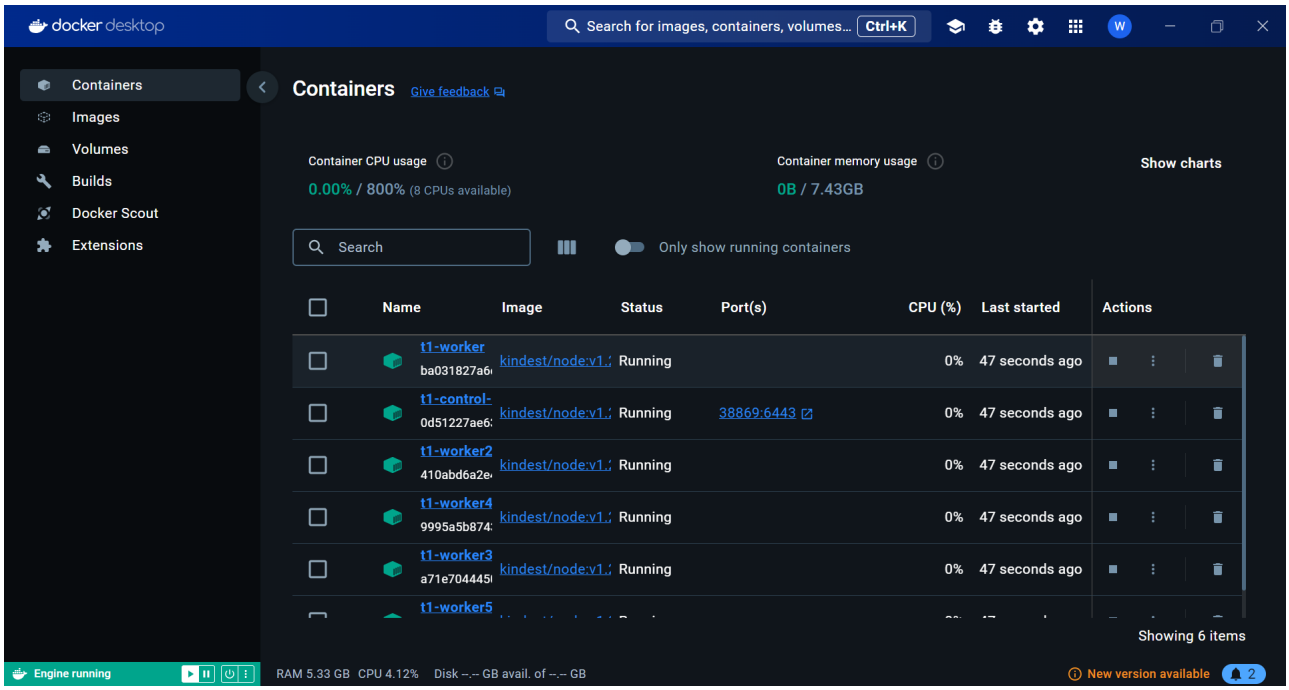
1. docker images
    - flask-app:1.0.0: only root API
    - flask-app:1.0.1: both root and greet-with-info API



2. docker volumes



3. docker containers(only t1 is running because t0 is already deleted)



t0

1. get into t0 floder, create cluster t0, load image 1.0.0, apply t0.yaml

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc$ cd codebase/t0
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kind create cluster --name t0 --config kind-config.yaml
Creating cluster "t0" ...
 ✓ Ensuring node image (kindest/node:v1.27.3) 🖼
 ✓ Preparing nodes 📦📦📦📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
 ✓ Joining worker nodes 🚜
Set kubectl context to "kind-t0"
You can now use your cluster with:

kubectl cluster-info --context kind-t0

Have a question, bug, or feature request? Let us know! https://kind.sigs.k8s.io/#community 🙂
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl config use-context kind-t0
Switched to context "kind-t0".
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kind load docker-image flask-app:1.0.0 --name t0
Image: "flask-app:1.0.0" with ID "sha256:0da6ccaf10bea5c25c2b26122e4a66bc07a73fbcf861ce3091e0dd1a41d83c20" not yet present on node "t0-worker", loading...
Image: "flask-app:1.0.0" with ID "sha256:0da6ccaf10bea5c25c2b26122e4a66bc07a73fbcf861ce3091e0dd1a41d83c20" not yet present on node "t0-worker2", loading...
Image: "flask-app:1.0.0" with ID "sha256:0da6ccaf10bea5c25c2b26122e4a66bc07a73fbcf861ce3091e0dd1a41d83c20" not yet present on node "t0-worker3", loading...
Image: "flask-app:1.0.0" with ID "sha256:0da6ccaf10bea5c25c2b26122e4a66bc07a73fbcf861ce3091e0dd1a41d83c20" not yet present on node "t0-control-plane", loading...
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl apply -f t0.yaml
deployment.apps/flask-app created
service/flask-service-t0 created
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-5c685644f4-cpnhg   1/1     Running   0          5s    10.244.3.2   t0-worker2   <none>           <none>
flask-app-5c685644f4-f8fc4   1/1     Running   0          5s    10.244.2.2   t0-worker    <none>           <none>
flask-app-5c685644f4-g2zvc   1/1     Running   0          5s    10.244.2.3   t0-worker    <none>           <none>
flask-app-5c685644f4-lsj89   1/1     Running   0          5s    10.244.1.2   t0-worker3   <none>           <none>
```

2. use `kubectl get svc` to get the ip address, start container, and curl the ip address, load new image 1.0.1, apply t0.yaml to update

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl get svc
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
flask-service-t0  ClusterIP   10.96.189.85   <none>        80/TCP    10s
kubernetes        ClusterIP   10.96.0.1      <none>        443/TCP   57s
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ docker exec -it t0-control-plane /bin/bash
root@t0-control-plane:/# curl 10.96.189.85
Hello from flask-app-5c685644f4-cpnhg, IP: 10.244.3.2, Node: t0-worker2
root@t0-control-plane:/# curl 10.96.189.85
Hello from flask-app-5c685644f4-g2zvc, IP: 10.244.2.3, Node: t0-worker
root@t0-control-plane:/# curl 10.96.189.85
Hello from flask-app-5c685644f4-lsj89, IP: 10.244.1.2, Node: t0-worker3
root@t0-control-plane:/#
exit
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kind load docker-image flask-app:1.0.1 --name t0
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t0-worker", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t0-worker2", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t0-worker3", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t0-control-plane", loading...
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl apply -f t0.yaml
deployment.apps/flask-app configured
service/flask-service-t0 unchanged
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl get pods -o wide
NAME                         READY   STATUS        RESTARTS   AGE     IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-5c685644f4-cpnhg   1/1     Terminating   0          2m21s   10.244.3.2   t0-worker2   <none>           <none>
flask-app-5c685644f4-f8fc4   1/1     Terminating   0          2m21s   10.244.2.2   t0-worker    <none>           <none>
flask-app-5c685644f4-g2zvc   1/1     Terminating   0          2m21s   10.244.2.3   t0-worker    <none>           <none>
flask-app-5c685644f4-lsj89   1/1     Terminating   0          2m21s   10.244.1.2   t0-worker3   <none>           <none>
flask-app-845c8df6bf-cfmjv   1/1     Running       0          9s      10.244.3.3   t0-worker2   <none>           <none>
flask-app-845c8df6bf-pgbz2   1/1     Running       0          8s      10.244.1.4   t0-worker3   <none>           <none>
flask-app-845c8df6bf-s446b   1/1     Running       0          9s      10.244.2.4   t0-worker    <none>           <none>
flask-app-845c8df6bf-smzm2   1/1     Running       0          9s      10.244.1.3   t0-worker3   <none>           <none>
```

3. search ip, start container and curl the ip address again

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ kubectl get svc
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
flask-service-t0  ClusterIP   10.96.189.85   <none>        80/TCP    2m33s
kubernetes        ClusterIP   10.96.0.1      <none>        443/TCP   3m20s
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t0$ docker exec -it t0-control-plane /bin/bash
root@t0-control-plane:/# curl 10.96.189.85
Hello from flask-app-845c8df6bf-s446b, IP: 10.244.2.4, Node: t0-worker
root@t0-control-plane:/# curl 10.96.189.85
Hello from flask-app-845c8df6bf-smzm2, IP: 10.244.1.3, Node: t0-worker3
root@t0-control-plane:/# curl 10.96.189.85
Hello from flask-app-845c8df6bf-s446b, IP: 10.244.2.4, Node: t0-worker
root@t0-control-plane:/# curl 10.96.189.85/chat/a
Hello a, welcome to the Flask app!
root@t0-control-plane:/# curl 10.96.189.85/chat/b
Hello b, welcome to the Flask app!
root@t0-control-plane:/# curl 10.96.189.85/chat/c
Hello c, welcome to the Flask app!
root@t0-control-plane:/#
exit
```

t1

1. get into t1 floder, create cluster t1, load image 1.0.1, apply t0.yaml

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc$ cd codebase/t1
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kind create cluster --name t1 --config kind-config.yaml
Creating cluster "t1" ...
 ✓ Ensuring node image (kindest/node:v1.27.3) 🖼
 ✓ Preparing nodes 📦📦📦📦📦📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
 ✓ Joining worker nodes 🚜
Set kubectl context to "kind-t1"
You can now use your cluster with:

kubectl cluster-info --context kind-t1

Have a question, bug, or feature request? Let us know! https://kind.sigs.k8s.io/#community 🙂
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl config use-context kind-t1
Switched to context "kind-t1".
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kind load docker-image flask-app:1.0.1 --name t1
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t1-worker", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t1-control-plane", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t1-worker2", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t1-worker4", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t1-worker3", loading...
Image: "flask-app:1.0.1" with ID "sha256:14c5a782f42f4a485e41a05a990983420b54e50653edec5f4de7b0f758df5bc9" not yet present on node "t1-worker5", loading...
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl apply -f t1.yaml
deployment.apps/flask-app created
service/flask-service-t1 created
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-kcw7x   1/1     Running   0          3s    10.244.1.2   t1-worker2   <none>           <none>
```

2. scale deployment from 1 to 3 and get pods each time

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl scale deployment flask-app --replicas=1
deployment.apps/flask-app scaled
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS        RESTARTS   AGE     IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-72rlg   1/1     Terminating   0          5m53s   10.244.2.2   t1-worker    <none>           <none>
flask-app-7c77b965dc-kcw7x   1/1     Running       0          7m59s   10.244.1.2   t1-worker2   <none>           <none>
flask-app-7c77b965dc-kts4s   1/1     Terminating   0          6m3s    10.244.3.2   t1-worker5   <none>           <none>
flask-app-7c77b965dc-mb7tv   1/1     Terminating   0          5m57s   10.244.5.2   t1-worker4   <none>           <none>
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl scale deployment flask-app --replicas=2
deployment.apps/flask-app scaled
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-kcw7x   1/1     Running   0          8m44s   10.244.1.2   t1-worker2   <none>           <none>
flask-app-7c77b965dc-r94gp   1/1     Running   0          2s      10.244.3.3   t1-worker5   <none>           <none>
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl scale deployment flask-app --replicas=3
deployment.apps/flask-app scaled
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-kcw7x   1/1     Running   0          8m47s   10.244.1.2   t1-worker2   <none>           <none>
flask-app-7c77b965dc-r94gp   1/1     Running   0          5s      10.244.3.3   t1-worker5   <none>           <none>
flask-app-7c77b965dc-w4drz   1/1     Running   0          1s      10.244.5.3   t1-worker4   <none>           <none>
```

3. scale deployment from 3 to 5 and get pods each time, find a pod is not ready

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl scale deployment flask-app --replicas=4
deployment.apps/flask-app scaled
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-9th7g   1/1     Running   0          2s      10.244.2.3   t1-worker    <none>           <none>
flask-app-7c77b965dc-kcw7x   1/1     Running   0          8m51s   10.244.1.2   t1-worker2   <none>           <none>
flask-app-7c77b965dc-r94gp   1/1     Running   0          9s      10.244.3.3   t1-worker5   <none>           <none>
flask-app-7c77b965dc-w4drz   1/1     Running   0          5s      10.244.5.3   t1-worker4   <none>           <none>
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl scale deployment flask-app --replicas=5
deployment.apps/flask-app scaled
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-9th7g   1/1     Running   0          6s      10.244.2.3   t1-worker    <none>           <none>
flask-app-7c77b965dc-fqq65   0/1     Pending   0          1s      <none>       <none>       <none>           <none>
flask-app-7c77b965dc-kcw7x   1/1     Running   0          8m55s   10.244.1.2   t1-worker2   <none>           <none>
flask-app-7c77b965dc-r94gp   1/1     Running   0          13s     10.244.3.3   t1-worker5   <none>           <none>
flask-app-7c77b965dc-w4drz   1/1     Running   0          9s      10.244.5.3   t1-worker4   <none>           <none>
```

4. improve t1.yaml and apply t1.yaml to update

```
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl apply -f t1.yaml
deployment.apps/flask-app configured
service/flask-service-t1 unchanged
(dcc_ass4) (base) win@DESKTOP-Q3V6AG6:~/dcc/codebase/t1$ kubectl get pods -o wide
NAME                         READY   STATUS        RESTARTS   AGE   IP           NODE         NOMINATED NODE   READINESS GATES
flask-app-7c77b965dc-9th7g   1/1     Terminating   0          20m   10.244.2.3   t1-worker    <none>           <none>
flask-app-7c77b965dc-kcw7x   1/1     Terminating   0          29m   10.244.1.2   t1-worker2   <none>           <none>
flask-app-7c77b965dc-r94gp   1/1     Terminating   0          20m   10.244.3.3   t1-worker5   <none>           <none>
flask-app-7c77b965dc-w4drz   1/1     Terminating   0          20m   10.244.5.3   t1-worker4   <none>           <none>
flask-app-847998c9c9-xvzxq   1/1     Running       0          7s    10.244.4.2   t1-worker3   <none>           <none>
```

PROBLEMS

when write `kind-config.yaml` for t1 in the first time, I use:

```
- role: worker
   labels:
      usage: normal
```

```
        capability: powerful
    taints:
        - key: class
          value: "vip"
          effect: NoSchedule
```

but there's error:

```
ERROR: failed to create cluster: unable to decode config: yaml: unmarshal errors
```

then I check the slides and change the code to:

```
- role: worker
    kubeadmConfigPatches:
    - |
      kind: JoinConfiguration
      nodeRegistration:
        kubeletExtraArgs:
          node-labels: "usage=normal,capability=powerful"
        taints:
        - key: class
          value: "vip"
          effect: NoSchedule
```

then it works

Advice on Future Cloud Computing Lab

As for me, I think the lab is good, but I think it can be improved by adding more details and examples about cloud platform or cloud security measures and data storage solutions maybe.