KUBERNETES SIMPLE NODE JS APPLICATION

[Document subtitle]

Kaushik Dey

Deploy a Simple Node.js-based Microservice in Kubernetes

Problem Statement:

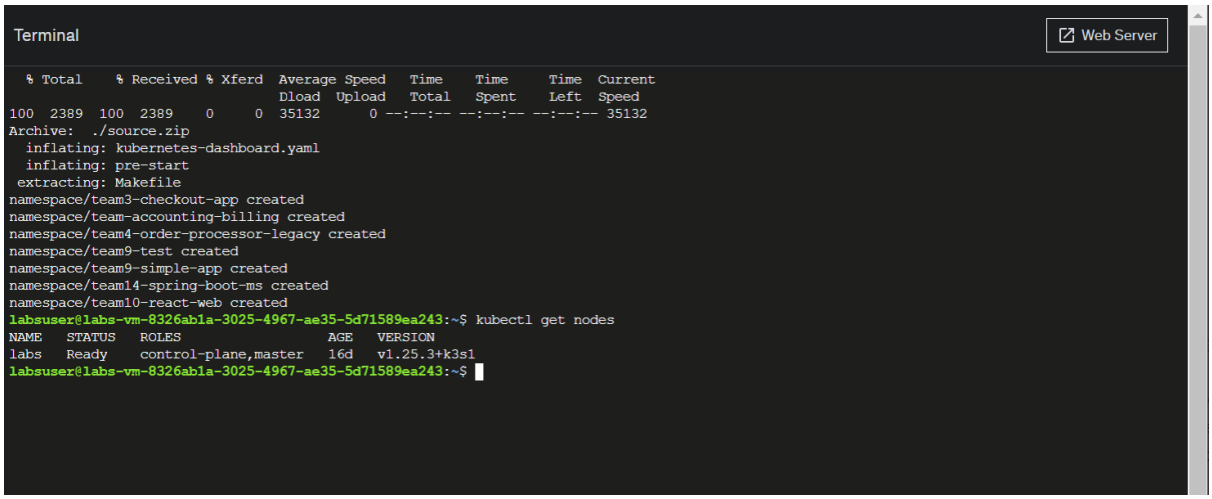
The development team needs help deploying a simple microservice in an existing Kubernetes cluster that has already been created for testing purposes. The development team already created a simple Node.js microservice and packaged it in a Docker container. They don't know which commands must be executed to deploy this container to Kubernetes. As your DevOps Engineering Manager, I count on your Kubernetes expertise to help the team in this sprint.

You will need to use the `kubectl` CLI tool to manage the Kubernetes cluster.
Please start by getting familiar with the test Kubernetes cluster. You will need to deploy the microservice and expose it to the outside for testing purposes. To help the team better visualize the Kubernetes objects, please install the Kubernetes Dashboard. Once everything is working, make sure to generate the Kubernetes YAML manifest files needed to deploy the microservice. Make sure to use the imperative approach for managing Kubernetes objects.

Gather information about the Kubernetes cluster

1. Run the following command using the terminal to get a list of all nodes in the cluster:

   kubectl get nodes



There is a single node being displayed. This indicates that this Kubernetes cluster has a single node that acts at the same time as a master and worker node.

2. Select the name of the node and copy it.
3. Run the following command to get detailed information about the node.
   Replace `<NODE_NAME>` with the value you have copied previously.

   ```
   kubectl describe node labs
   ```

4. Scroll down until you reach the section *Allocated resources*. Locate the information regarding the CPU and the memory.

```
  kube-system                 traefik-9c6dc6686-n29fr                   0 (0%)        0 (0%)      0 (0%)           0 (0%)         8m7s
  kube-system                 metrics-server-5c8978b444-9wbvt           100m (5%)     0 (0%)      70Mi (1%)        0 (0%)         8m18s
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests    Limits
  --------           --------    ------
  cpu                200m (10%)  0 (0%)
  memory             140Mi (3%)  170Mi (4%)
  ephemeral-storage  0 (0%)      0 (0%)
  hugepages-1Gi      0 (0%)      0 (0%)
  hugepages-2Mi      0 (0%)      0 (0%)
Events:
  Type     Reason                   Age                From                   Message
  ----     ------                   ----               ----                   -------
  Normal   Starting                 8m30s              kube-proxy
  Normal   NodeAllocatableEnforced   16d                kubelet                Updated Node Allocatable limit across pods
  Normal   Starting                 16d                kubelet                Starting kubelet.
  Warning  InvalidDiskCapacity      16d                kubelet                invalid capacity 0 on image filesystem
  Normal   NodeHasSufficientMemory  16d (x2 over 16d)  kubelet                Node labs status is now: NodeHasSufficientMemory
  Normal   NodeHasNoDiskPressure    16d (x2 over 16d)  kubelet                Node labs status is now: NodeHasNoDiskPressure
  Normal   NodeHasSufficientPID     16d (x2 over 16d)  kubelet                Node labs status is now: NodeHasSufficientPID
  Normal   Starting                 8m33s              kubelet                Starting kubelet.
  Warning  InvalidDiskCapacity      8m33s              kubelet                invalid capacity 0 on image filesystem
  Normal   NodeHasSufficientMemory  8m33s              kubelet                Node labs status is now: NodeHasSufficientMemory
  Normal   NodeHasNoDiskPressure    8m33s              kubelet                Node labs status is now: NodeHasNoDiskPressure
  Normal   NodeHasSufficientPID     8m33s              kubelet                Node labs status is now: NodeHasSufficientPID
  Warning  CheckLimitsForResolvConf 8m32s              kubelet                Resolv.conf file '/run/systemd/resolve/resolv.conf' contains se
arch line consisting of more than 3 domains!
  Warning  Rebooted                 8m32s              kubelet                Node labs has been rebooted, boot id: 2b1808c9-28cb-4982-a30e-3
5234dcf8f04
  Normal   NodeReady                8m32s              kubelet                Node labs status is now: NodeReady
  Normal   NodeAllocatableEnforced   8m32s              kubelet                Updated Node Allocatable limit across pods
  Normal   Synced                   8m31s              cloud-node-controller  Node synced successfully
  Normal   RegisteredNode           8m19s              node-controller        Node labs event: Registered Node labs in Controller
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
```

Create a new Kubernetes namespace

1. Use the following command to create a new namespace named `team-12-customers`:

   `kubectl create namespace team-12-customers`

```
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$ kubectl create namespace team-12-customers
namespace/team-12-customers created
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
```

2. To display a list of all available workspaces within the Kubernetes cluster, enter the following command

   `kubectl get namespaces`

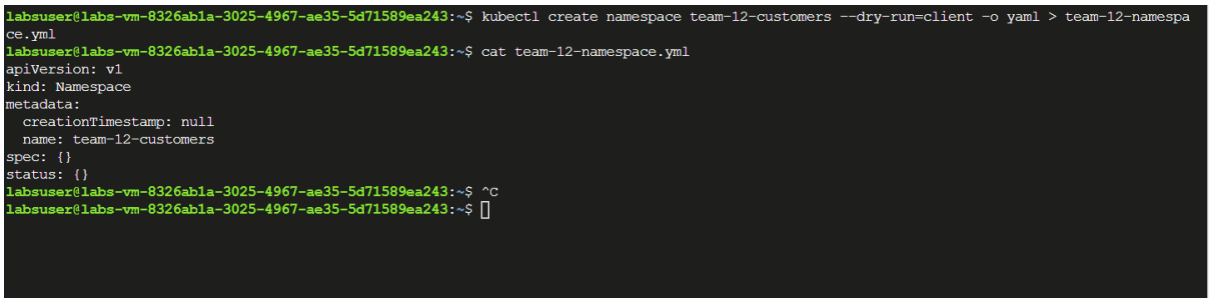
```
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$ kubectl get namespaces
NAME                          STATUS   AGE
default                       Active   16d
kube-system                   Active   16d
kube-public                   Active   16d
kube-node-lease               Active   16d
team3-checkout-app            Active   16m
team-accounting-billing       Active   16m
team4-order-processor-legacy  Active   16m
team9-test                    Active   16m
team9-simple-app              Active   16m
team14-spring-boot-ms         Active   16m
team10-react-web              Active   16m
team-12-customers             Active   2m54s
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
```

3. To store the configuration required to create the namespace in a Kubernetes manifest file, use the same command which you have used to create the namespace, but this time use the `--dry-run` flag to prevent the execution and redirect the output as YAML to a file named `team-12-namespace.yml`. The entire command will look as follows:

```
kubectl create namespace team-12-customers --dry-run=client -o yaml
> team-12-namespace.yml
```

4. To view the contents of the manifest file, use the `cat` command as follows:

```
cat team-12-namespace.yml
```

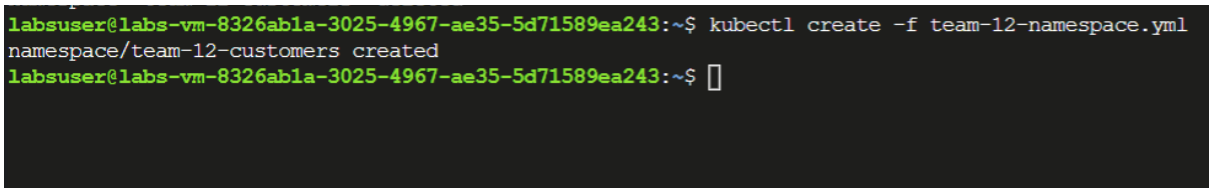


5. To test that the manifest file works, first delete the namespace using the following command:

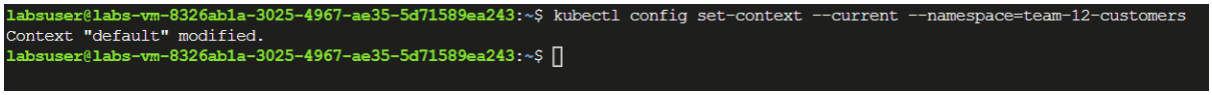
kubectl delete namespace team-12-customers

6. Use the `kubectl` command to create the namespace from the file which contains the manifest.

kubectl create -f team-12-namespace.yml



7. Since the team will work only from this workspace, set it as the default workspace. The following message should appear as a response indicating that the context has been successfully changed:



Create a Kubernetes deployment

Deploy a Simple Node.js-based Microservice in Kubernetes

1. Use `kubectl` to create a new deployment named `simple-api-deployment`. With the `--image` option, specify the Docker image and the tag.

```
kubectl create deployment simple-api-deployment --
image=veveritaengineering/simple-api:1
```

```
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$ kubectl create deployment simple-api-deployment --image=veveritaengineering/simple-ap
i:1
deployment.apps/simple-api-deployment created
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
```

2. Use the following `kubectl` command to verify if the deployment was created:

   kubectl get deployments

```
deployment.apps/simple-api-deployment created                                s
NAME                    READY   UP-TO-DATE   AVAILABLE   AGEectl get deployments
simple-api-deployment   1/1     1            1           91s
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
```

3. List the pods created by the deployment using the following command:
   `kubectl get pods`

```
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
simple-api-deployment-6f77554f75-qdm9x  1/1     Running   0          4m22s
labsuser@labs-vm-8326ab1a-3025-4967-ae35-5d71589ea243:~$
```

Since the application has only one replica, only one pod has been created.

4. To store the configuration required to create the deployment in a Kubernetes manifest file, use the same command which you have used to create the deployment, but this time use the `--dry-run` flag to prevent the execution and store the output as YAML to a file named `team-12-deployment.yml`. The entire command will look as follows:

```
kubectl create deployment simple-api-deployment --
image=veveritaengineering/simple-api:1  --dry-run=client -o yaml >
team-12-deployment.yml
```

   If the execution has been successful, no message will be displayed.

5. To view the contents of the manifest file, use the `cat` command as follows:

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ cat team-12-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: simple-api-deployment
  name: simple-api-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: simple-api-deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: simple-api-deployment
    spec:
      containers:
      - image: veveritaengineering/simple-api:1
        name: simple-api
        resources: {}
status: {}
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

6. To test that the manifest file works, first delete the deployment using `kubectl delete`

   kubectl delete deployment/simple-api-deployment
   A successful message should be displayed. When trying to get all deployments with `kubectl get deployments` no results should appear.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl delete deployment/simple-api-deployment
deployment.apps "simple-api-deployment" deleted
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get deployments
No resources found in team-12-customers namespace.
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

7. Use the `kubectl` command to create the deployment from the file containing the manifest.
   kubectl create -f team-12-deployment.yml
   A success message should be displayed indicating that the deployment has been successfully created.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl create -f team-12-deployment.yml
deployment.apps/simple-api-deployment created
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

Expose deployment

1. Create a Kubernetes service that exposes the deployment on port 3000. Ensure that the service type is LoadBalancer.

kubectl expose deployment simple-api-deployment --type=LoadBalancer --port=3000

The response to the command should indicate that the service has been created successfully.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl expose deployment simple-api-deployment --type=LoadBalancer --port=3000
service/simple-api-deployment exposed
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

2. Get information about the services available using the following command:
   kubectl get services

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get services
NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
simple-api-deployment   LoadBalancer   10.43.221.225   10.0.2.2      3000:30560/TCP   119s
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

   Since no service name has been specified, the default name for the service is `simple-api-deployment`.

3. To access the application, find the local port where container port 3000 has been mapped. In the example below, the local port is 30560.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get services
NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
simple-api-deployment   LoadBalancer   10.43.221.225   10.0.2.2      3000:30560/TCP   119s
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

4. Use `curl` to access the application. Replace LOCAL_PORT with the local port where the service is accessible.

5. To store the configuration required to create the service in a Kubernetes manifest file, use the same command which you have used to create the service, but this time use the `--dry-run` flag to prevent the execution and store the output as YAML to a file named `team-12-service-loadbalancer.yml`. The entire command will look as follows:

kubectl expose deployment simple-api-deployment --type=LoadBalancer --port=3000 --dry-run=client -o yaml > team-12-service-loadbalancer.yml

   If the execution has been successful, no message will be displayed.

6. To view the contents of the manifest file, use the `cat` command as follows.
   The contents should be as follows:

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ cat team-12-service-loadbalancer.yml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: simple-api-deployment
  name: simple-api-deployment
spec:
  ports:
  - port: 3000
    protocol: TCP
    targetPort: 3000
  selector:
    app: simple-api-deployment
  type: LoadBalancer
status:
  loadBalancer: {}
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

7. To test that the manifest file works, first delete the service using `kubectl delete:`

   kubectl delete service/simple-api-deployment

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl delete service/simple-api-deployment
service "simple-api-deployment" deleted
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get services
No resources found in team-12-customers namespace.
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

8. Use the `kubectl` command to create the service from the file containing the manifest.

   kubectl create -f team-12-service-loadbalancer.yml

A success message should be displayed indicating that the service has been successfully created.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl create -f team-12-service-loadbalancer.yml
service/simple-api-deployment created
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

9. To get information about the available services, use the following command:

```
kubectl get services
```

The response should look as follows:

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get services
NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
simple-api-deployment LoadBalancer   10.43.229.74   10.0.2.2      3000:31743/TCP   2m33s
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

10. Notice that the local port of the service has changed.

Deploy Kubernetes Dashboard

1. Locate the kubernetes-dashboard.yaml file and verify its content using the cat command.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ cat kubernetes-dashboard.yaml
# Copyright 2017 The Kubernetes Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

apiVersion: v1
kind: Namespace
metadata:
  name: kubernetes-dashboard

---

apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
```

2. Use the `kubectl` command to deploy the Kubernetes Dashboard from the file containing the manifest.

```
kubectl create -f kubernetes-dashboard.yaml
```

Successful execution is indicated by the creation of various Kubernetes objects, as in the screenshot below:

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl create -f kubernetes-dashboard.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

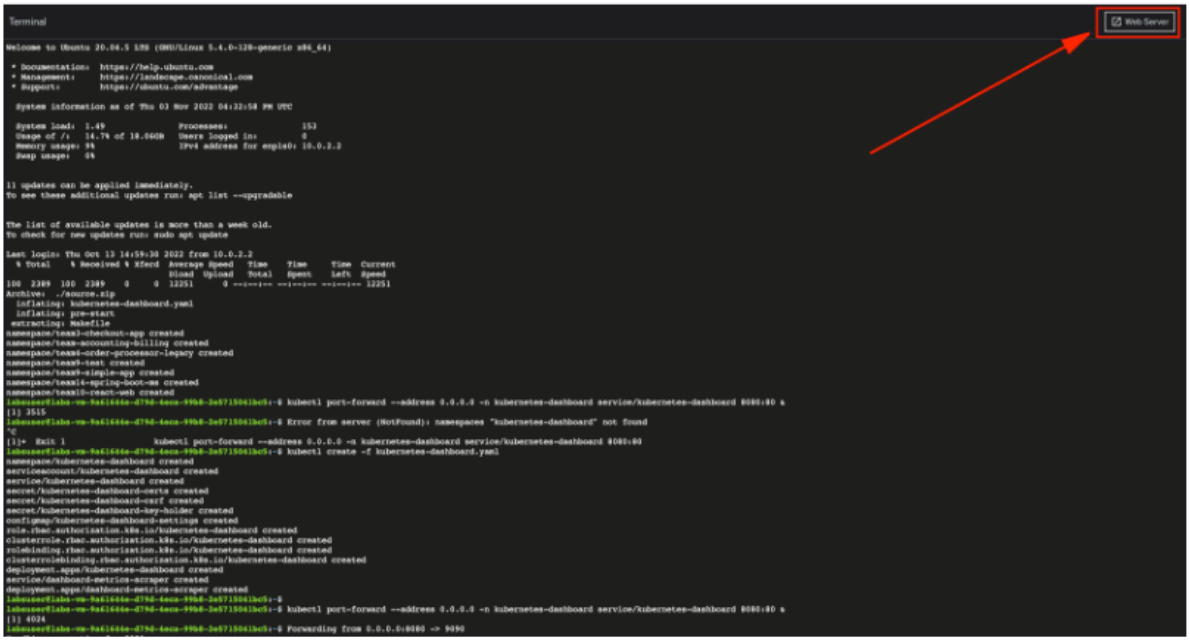
3. Use the following command to expose the Kubernetes Dashboard:

   kubectl port-forward --address 0.0.0.0 -n kubernetes-dashboard service/kubernetes-dashboard 8080:80 &

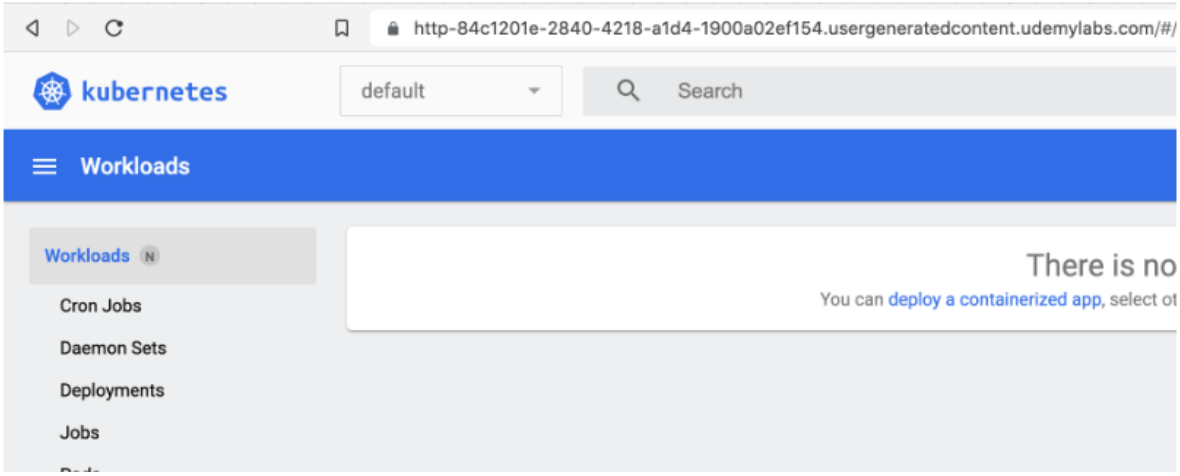
```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl port-forward --address 0.0.0.0 -n kubernetes-dashboard service/kubernetes-dashboard 8080:80 &
[1] 131158
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ Forwarding from 0.0.0.0:8080 -> 9090

labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

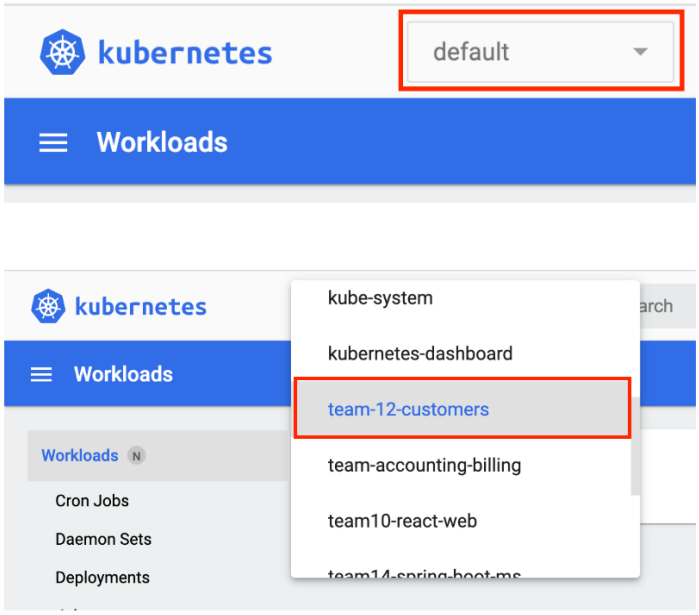
4. From the web terminal window, click the **Web Server** button to open a new browser tab with the Kubernetes Dashboard.



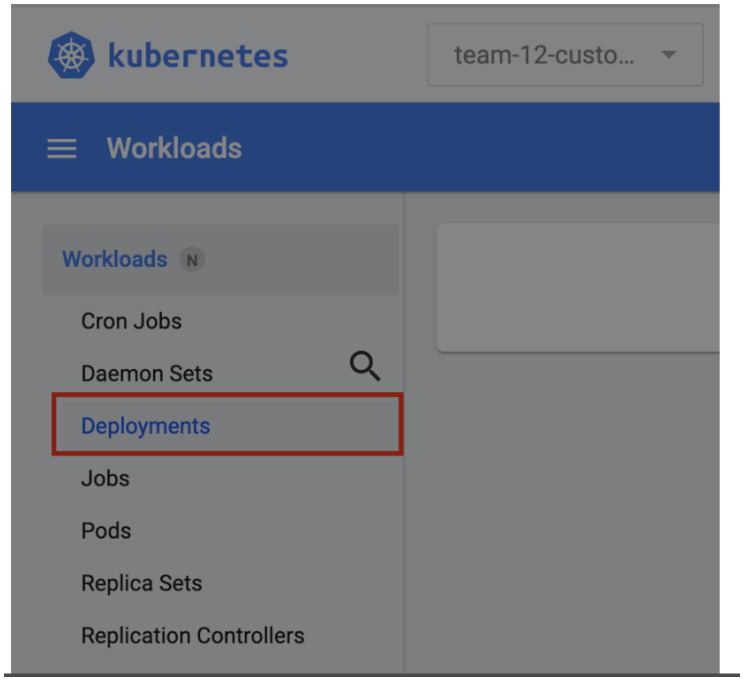
The Kubernetes Dashboard should look as follows:

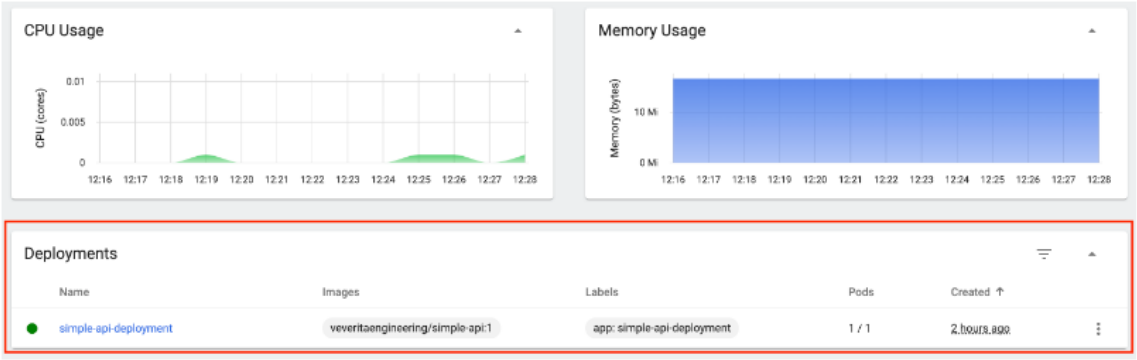


5. From the Kubernetes Dashboard, identify the select list where `default` namespace is currently active. Select the `team-12-customers` namespace.



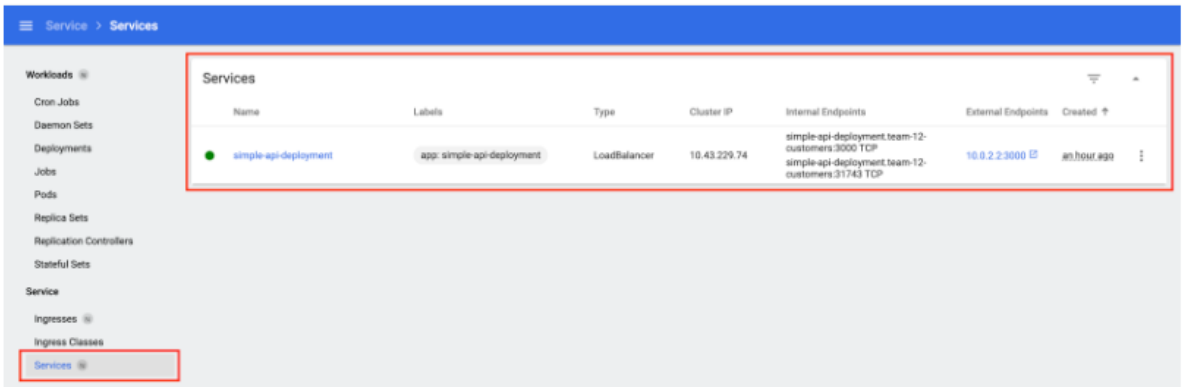
6. From the left-side menu, select **Deployments** to inspect the deployment previously created





7. From the left-side menu, select Pods to inspect the existing pods. The deployment has created a single replica, so the list will only contain a single pod.

8. From the left-side menu, select Services and locate the load balancer service that exposes the application.



Scale up the application!

1. Using the `kubectl scale` command, increase the replicas from 1 to 4 for the `simple-api-deployment`:

   kubectl scale --replicas=4 deployment simple-api-deployment

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl scale --replicas=4 deployment simple-api-deployment
deployment.apps/simple-api-deployment scaled
```

2. Verify that the four pods are now available by running the following command:

   `kubectl get pods`

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
simple-api-deployment-b7fd88f4-5kkgg  1/1     Running   0          159m
simple-api-deployment-b7fd88f4-9x9v8  1/1     Running   0          27s
simple-api-deployment-b7fd88f4-7jrnm  1/1     Running   0          27s
simple-api-deployment-b7fd88f4-6jnlw  1/1     Running   0          27s
```

3. `Use kubectl get services to get the local port of the service.`

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get services
NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
simple-api-deployment LoadBalancer   10.43.229.74   10.0.2.2      3000:31743/TCP   124m
```

4. Run the `curl` command against the application a few times and observe that the application is served by different pods.

```
curl http://localhost:LOCAL_PORT
```

5. For debugging purposes, the application will display the name of the pod that is sending the response.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"1","pod":"simple-api-deployment-b7fd88f4-7jrnm"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"1","pod":"simple-api-deployment-b7fd88f4-6jnlw"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"1","pod":"simple-api-deployment-b7fd88f4-9x9v8"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"1","pod":"simple-api-deployment-b7fd88f4-5kkgg"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

Deploy a new version of the application

1. Verify the current version of the deployment by using `kubectl describe` to get details about the deployment:

   kubectl describe deployment simple-api-deployment

   The following details will be provided in regard to the deployment. Notice which image and tag are being used:

- Image: veveritaengineering/simple-api
- Tag: 1

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl describe deployment simple-api-deployment
Name:                   simple-api-deployment
Namespace:              team-12-customers
CreationTimestamp:      Mon, 31 Oct 2022 17:13:11 +0000
Labels:                 app=simple-api-deployment
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=simple-api-deployment
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=simple-api-deployment
  Containers:
   simple-api:
    Image:        veveritaengineering/simple-api:1
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   simple-api-deployment-b7fd88f4 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  30m   deployment-controller  Scaled up replica set simple-api-deployment-b7fd88f4 to 4
```

2. Use the command `kubectl set image` to change the tag for the `simple-api-deployment` deployment.

kubectl set image deployment/simple-api-deployment simple-api=veveritaengineering/simple-api:2

A successful change will be indicated by message as in the image below:

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl set image deployment/simple-api-deployment simple-api=veveritaengineering/simple-api:2
deployment.apps/simple-api-deployment image updated
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

3. Verify that the version change has been made by using `kubectl describe`

kubectl describe deployment simple-api-deployment.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl describe deployment simple-api-deployment
Name:                   simple-api-deployment
Namespace:              team-12-customers
CreationTimestamp:      Mon, 31 Oct 2022 17:13:11 +0000
Labels:                 app=simple-api-deployment
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=simple-api-deployment
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=simple-api-deployment
  Containers:
   simple-api:
    Image:        veveritaengineering/simple-api:2
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   simple-api-deployment-5d54dbc8c9 (4/4 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m20s  deployment-controller  Scaled up replica set simple-api-deployment-5d54dbc8c9 to 1
  Normal  ScalingReplicaSet  3m20s  deployment-controller  Scaled down replica set simple-api-deployment-b7fd88f4 to 3
  Normal  ScalingReplicaSet  3m20s  deployment-controller  Scaled up replica set simple-api-deployment-5d54dbc8c9 to 2
  Normal  ScalingReplicaSet  3m17s  deployment-controller  Scaled down replica set simple-api-deployment-b7fd88f4 to 1
  Normal  ScalingReplicaSet  3m17s  deployment-controller  Scaled up replica set simple-api-deployment-5d54dbc8c9 to 4
  Normal  ScalingReplicaSet  3m16s  deployment-controller  Scaled down replica set simple-api-deployment-b7fd88f4 to 0
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

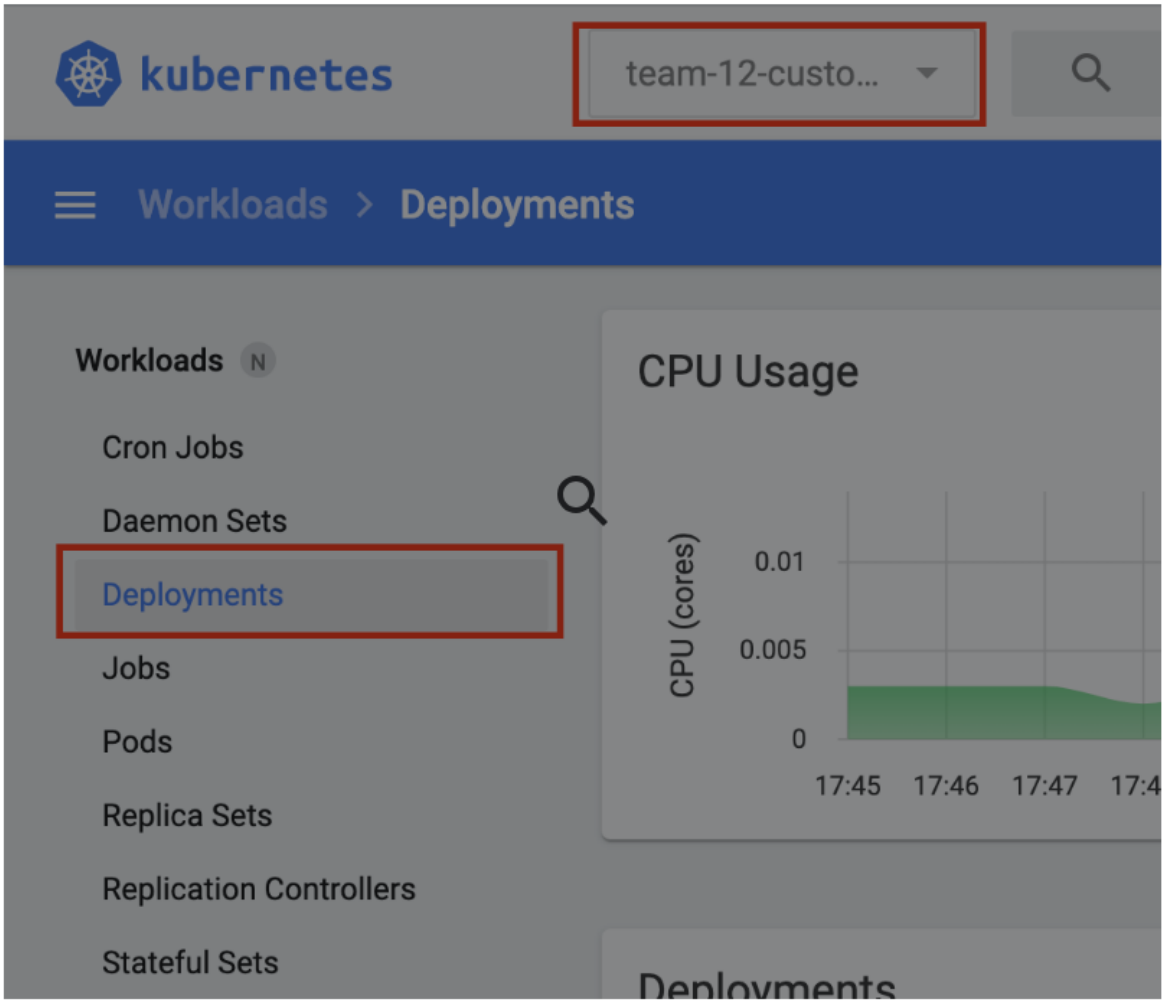
4. Run the curl command against the application a few times and observe if the application version has changed:

For debugging purposes, the application will display the current application version in the response.

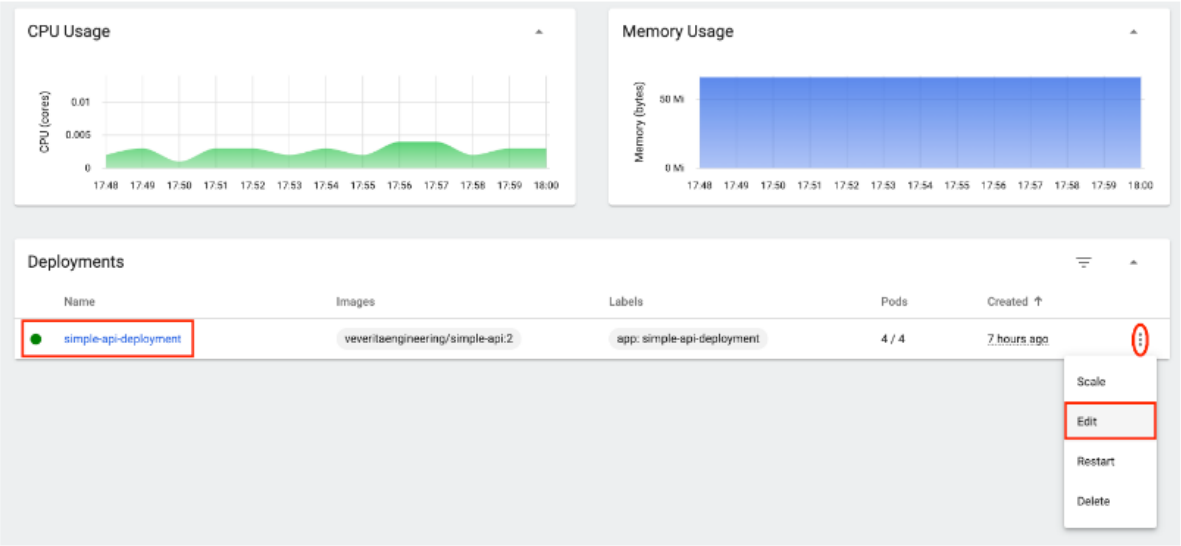
```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"2","pod":"simple-api-deployment-5d54dbc8c9-g6nc9"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"2","pod":"simple-api-deployment-5d54dbc8c9-g6nc9"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"2","pod":"simple-api-deployment-5d54dbc8c9-rr9xg"}labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

Scale down the application

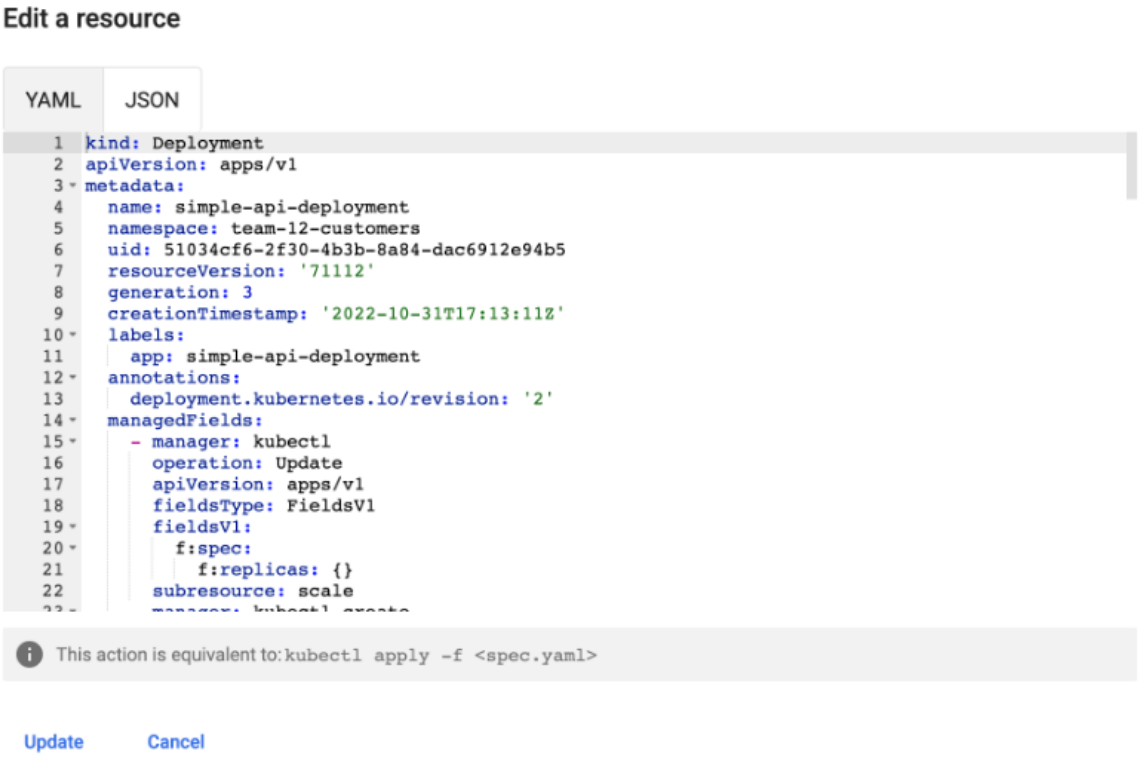
1. Having the **team-12-customers** namespace selected, open the Kubernetes Dashboard and select **Deployments** from the left-side menu.



2. Identify the `simple-api-deployment` and click on the tree dots to open the context menu. From the list, select **Edit**.



This will display the manifest for the selected deployment.



3. Identify the spec section of the manifest which describes the desired state of the deployment. Currently the specification requires 4 replicas.

Edit a resource

YAML JSON

```
102                 f:status: {}
103                 f:type: {}
104             f:observedGeneration: {}
105             f:readyReplicas: {}
106             f:replicas: {}
107             f:updatedReplicas: {}
108         subresource: status
109 spec:
110   replicas: 4
111   selector:
112     matchLabels:
113       app: simple-api-deployment
114   template:
115     metadata:
116       creationTimestamp: null
117       labels:
118         app: simple-api-deployment
119     spec:
120       containers:
121         - name: simple-api
122           image: veveritaengineering/simple-api:2
123           resources: {}
```

This action is equivalent to: kubectl apply -f <spec.yaml>

4. To scale down the deployment, change the value of replicas from 4 to 2. Click on **Update** to apply the new configuration.

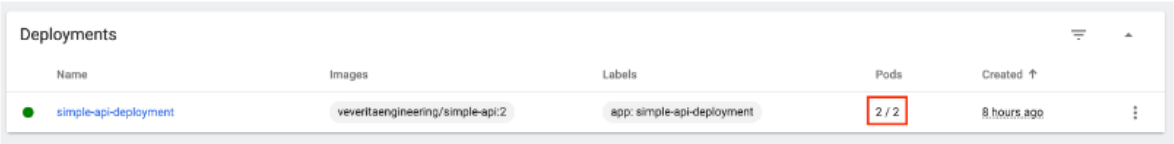
YAML JSON

```
102                 f:status: {}
103                 f:type: {}
104             f:observedGeneration: {}
105             f:readyReplicas: {}
106             f:replicas: {}
107             f:updatedReplicas: {}
108         subresource: status
109 spec:
110   replicas: 2
111   selector:
112     matchLabels:
113       app: simple-api-deployment
114   template:
115     metadata:
116       creationTimestamp: null
117       labels:
118         app: simple-api-deployment
119     spec:
120       containers:
121         - name: simple-api
122           image: veveritaengineering/simple-api:2
123           resources: {}
```

This action is equivalent to: kubectl apply -f <spec.yaml>

Update Cancel

5. . Refresh the Deployments page and observe that the number of pods has been scaled down from 4 to 2.



Make changes through the Kubernetes manifest file

1. Use vim to open the manifest file:

   vim team-12-deployment.yml

2. Identify the lines in the spec that require changes:

   replicas: 1

   - image: veveritaengineering/simple-api:1

Terminal

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: simple-api-deployment
  name: simple-api-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: simple-api-deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: simple-api-deployment
    spec:
      containers:
      - image: veveritaengineering/simple-api:1
        name: simple-api
        resources: {}
status: {}
~
```

3. Press the key i to enter INSERT mode. When the editor is in insert mode, the following text will be visible at the bottom of the screen.

```
~
-- INSERT --
```

4. Navigate the file with the arrow keys and change the value for replicas from 1 to 2 and the image tag from 1 to 3.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: simple-api-deployment
  name: simple-api-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: simple-api-deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: simple-api-deployment
    spec:
      containers:
      - image: veveritaengineering/simple-api:3
        name: simple-api
        resources: {}
status: {}
~
```

5. Press the Esc key to exit the insert mode. Type `:x` followed by pressing the Enter key to save and exit.

```
~
~
~
:x
```

6. Use the `kubectl replace` command to replace the deployment from the file containing the manifest.

   kubectl replace -f team-12-deployment.yml

7. A message will indicate that the action has been successful.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl replace -f team-12-deployment.yml
deployment.apps/simple-api-deployment replaced
```

Inspect application logs

1. Replicate the error by trying to access the application using `curl`. Replace LOCAL_PORT with the local port where the load balancer service is accessible.

   ```
   curl http://localhost:LOCAL_PORT
   ```

   The following error should appear:

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
curl: (7) Failed to connect to localhost port 31743: Connection refused
```

2. List all the pods using the following command

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
simple-api-deployment-6fbbdc88c6-248zr  1/1     Running   0          75m
simple-api-deployment-6fbbdc88c6-xrph4  1/1     Running   0          75m
```

3. 3. Use the `kubectl logs` command to get the logs of the pod. Replace <POD_NAME> with the name of the pod you have copied.

   `kubectl logs <POD_NAME>`

Running the command should show the logs generated by the application

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl logs simple-api-deployment-6fbbdc88c6-248zr
The API is listening on port 3001
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

4. Modify the load balancer service to communicate with the application on port 3001. Using `vim`, edit the manifest file `team-12-service-loadbalancer.yml`.

   vim team-12-service-loadbalancer.yml

5. Locate the port and target port which needs to be changed.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: simple-api-deployment
  name: simple-api-deployment
spec:
  ports:
  - port: 3000
    protocol: TCP
    targetPort: 3000
  selector:
    app: simple-api-deployment
  type: LoadBalancer
status:
  loadBalancer: {}
~
```

6. Press the key `i` to enter INSERT mode. When the editor is in insert mode, the following text will be visible at the bottom of the screen.

```
~
-- INSERT --
```

7. Navigate the file with the arrow keys and change the values from 3000 to 3001.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: simple-api-deployment
  name: simple-api-deployment
spec:
  ports:
  - port: 3001
    protocol: TCP
    targetPort: 3001
  selector:
    app: simple-api-deployment
  type: LoadBalancer
status:
  loadBalancer: {}
~
```

8. Press the Esc key to exit the insert mode. Type `:x` followed by pressing the Enter key to save and exit.

```
~
~
~
:x
```

9. Use the `kubectl replace` command to replace the service from the file containing the manifest.

```
kubectl replace -f team-12-service-loadbalancer.yml
```

A message will indicate that the action has been successful.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl replace -f team-12-service-loadbalancer.yml
service/simple-api-deployment replaced
```

10. Use the previously used curl command to verify if the application is accessible.

No error message should appear and the application version should be visible in the response.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ curl http://localhost:31743
{"status":"UP","version":"3","pod":"simple-api-deployment-6fbbdc88c6-xrph4"}labsuser@l
```

Delete Kubernetes objects

1. Use the `kubectl delete` command to delete the load balancer service by specifying the `team-12-service-loadbalancer.yml` manifest file.
`kubectl delete -f team-12-service-loadbalancer.yml`
The response should indicate that the service has been deleted.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl delete -f team-12-service-loadbalancer.yml
service "simple-api-deployment" deleted
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

2. Use the `kubectl delete` command to delete the deployment specifying the `team-12-deployment.yml` manifest file.
`kubectl delete -f team-12-deployment.yml`
The response should indicate that the deployment has been deleted.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl delete -f team-12-deployment.yml
deployment.apps "simple-api-deployment" deleted
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```

3. Use the `kubectl delete` command to delete the namespace specifying the `team-12-namespace.yml` manifest file.
`kubectl delete -f team-12-namespace.yml`
The response should indicate that the namespace has been deleted.

```
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$ kubectl delete -f team-12-namespace.yml
namespace "team-12-customers" deleted
labsuser@labs-vm-84c1201e-2840-4218-a1d4-1900a02ef154:~$
```