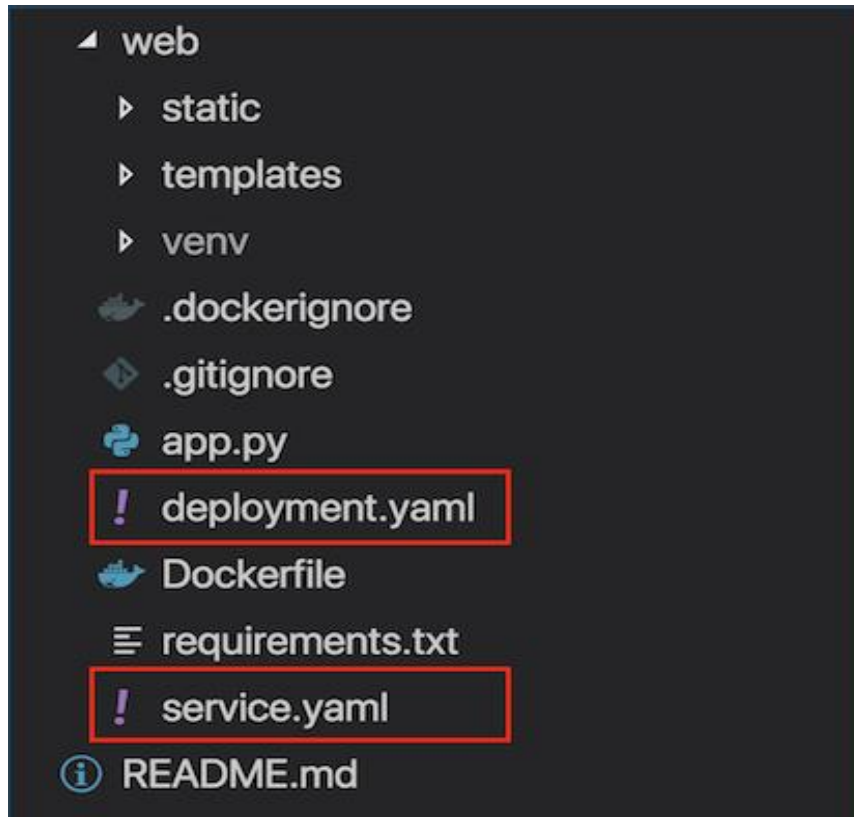


DEPLOY IN KUBERNETES CLUSTER

TEAM ID: PNT2022TMID22391

CREATE CONFIGURATION FILES FOR KUBERNETES

STEP 1: Once the image is successfully uploaded to the private registry, go to your project directory and create two files: deployment.yaml and service.yaml.



STEP 2 : In the deployment.yaml file, paste this code:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: flask-node-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flasknode
  template:
    metadata:
      labels:
        app: flasknode
    spec:
      containers:
```

```
- name: flasknode
  image: registry.ng.bluemix.net/flask-node/app
  imagePullPolicy: Always
  ports:
  - containerPort: 5000
```

STEP 3 : In the service.yaml file, paste this code:

```
apiVersion: v1
kind: Service
metadata:
  name: flask-node-deployment
spec:
  ports:
  - port: 5000
    targetPort: 5000
  selector:
    app: flasknode
```

DEPLOY YOUR APPLICATION TO KUBERNETES :

STEP 1 : Target the IBM Cloud Kubernetes Service region where you want to work.

```
ibmcloud cs region-set uk-south
```

STEP 2 : Set the context for the cluster in your CLI.

a. Get the command to set the environment variable and download the Kubernetes configuration files.

```
ibmcloud cs cluster-config cluster_kunal
```

b. Set the KUBECONFIG environment variable. Copy the output from the previous command and paste it in your terminal. The command output should look similar to the following.

```
> export KUBECONFIG=/Users/$USER/.bluemix/plugins/container-  
service/clusters/< cluster_name >/< cluster_configuration_file.yaml>
```

STEP 3 : Verify that you can connect to your cluster by listing your worker nodes.

```
kubectl get nodes
```

STEP 4 : Create the deployment.

```
kubectl create -f deployment.yaml
```

STEP 5 : Create the service.

```
kubectl create -f service.yaml
```

STEP 6 :Look at the Kubernetes dashboard from the IBM Kubernetes Service overview page.

The screenshot shows the IBM Kubernetes Service Overview dashboard. The left sidebar contains navigation links for Cluster, Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Workloads, Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets, Ingresses, Services, Config and Storage, and Config Maps. The main content area displays several tables:

- Deployments:** A table with columns Name, Labels, Pods, Age, and Images. It shows one deployment: `flask-node-deployment` with label `app: flasknode`, 1/1 pods, 5 minutes age, and image `registry.ng.bluemix.net/flask-node/app`.
- Pods:** A table with columns Name, Node, Status, Restarts, Age, CPU (cores), and Memory (bytes). It shows one pod: `flask-node-deployment-Sod9c6f8bc-d5n6x` on node `10.47.79.201`, status `Running`, 0 restarts, 5 minutes age, 0 CPU cores, and 19.352 Mi memory.
- Replica Sets:** A table with columns Name, Labels, Pods, Age, and Images. It shows one replica set: `flask-node-deployment-Sod9c6f8bc` with label `app: flasknode` and `pod-template-hash: 1785279267`, 1/1 pods, 5 minutes age, and image `registry.ng.bluemix.net/flask-node/app`.
- Services:** A table with columns Name, Labels, Cluster IP, Internal endpoints, External endpoints, and Age. It shows two services: `kubernetes` (component: `apiserver`, provider: `kubernetes`) and `flask-node-deployment` (flask-node-deployment:5000 TCP).

STEP 7 : Finally, go to your browser and ping the Public IP of your worker node.

