Experiment-10

Kubernetes Engine: Qwik Start: Deploy a containerized application to a Kubernetes Engine cluster.

Steps to Deploy a Containerized Application to GKE

Set Up Google Cloud SDK and Kubernetes Tools

1. **Install Google Cloud SDK**
   If you haven't already installed the Google Cloud SDK, follow the instructions here:
   Google Cloud SDK Installation
2. **Install kubectl**
   kubectl is the Kubernetes command-line tool used to manage Kubernetes clusters. The Cloud SDK includes kubectl, so if you have the SDK installed, you already have kubectl.

Create a Google Cloud Project

1. **Create a new Google Cloud project** (if you don't already have one):
   - Go to the Google Cloud Console.
   - Click on **Select a Project** > **New Project**.
   - Name your project and click **Create**.

2. **Set your project in the gcloud CLI**:

gcloud config set project PROJECT_ID

Enable Required APIs

1. **Enable Kubernetes Engine API**:

gcloud services enable container.googleapis.com

2. **Enable Compute Engine API** (if not already enabled):

gcloud services enable compute.googleapis.com

Create a Kubernetes Cluster

1. **Create the Kubernetes Engine cluster**:

gcloud container clusters create my-cluster --zone us-central1-a

Replace my-cluster with your desired cluster name and adjust the zone if necessary.

2. **Get the credentials for your cluster**: This command configures kubectl to use the cluster you just created.

gcloud container clusters get-credentials my-cluster --zone us-central1-a

Create a Containerized Application

1. **Create a Dockerfile** for your application. Below is an example for a simple web application using Node.js:

Dockerfile:

dockerfile

CopyEdit

```
FROM node:14

WORKDIR /usr/src/app

COPY . .

RUN npm install

EXPOSE 8080

CMD [ "npm", "start" ]
```

2. **Build the Docker image**:

```
docker build -t gcr.io/PROJECT_ID/my-app:v1 .
```

Replace PROJECT_ID with your Google Cloud project ID.

3. **Push the image to Google Container Registry**:

```
docker push gcr.io/PROJECT_ID/my-app:v1
```

Create a Kubernetes Deployment

1. **Create a Kubernetes Deployment configuration file** (deployment.yaml) for your containerized application.

deployment.yaml:

```
apiVersion: apps/v1

kind: Deployment

metadata:
```

```
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: gcr.io/PROJECT_ID/my-app:v1
        ports:
        - containerPort: 8080
```

Replace PROJECT_ID with your project ID.

2. **Apply the Deployment to the Kubernetes cluster**:

```
kubectl apply -f deployment.yaml
```

Expose the Application via a Service

1. **Create a Service to expose the application** (either LoadBalancer or ClusterIP for internal access).

Example **Service.yaml** for external exposure (LoadBalancer type):

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

2. **Apply the Service configuration**:

```
kubectl apply -f service.yaml
```

3. **Get the external IP address**: It may take a few moments for the LoadBalancer to be provisioned.

```
kubectl get svc
```

The **EXTERNAL-IP** column will show the public IP once the LoadBalancer is provisioned.

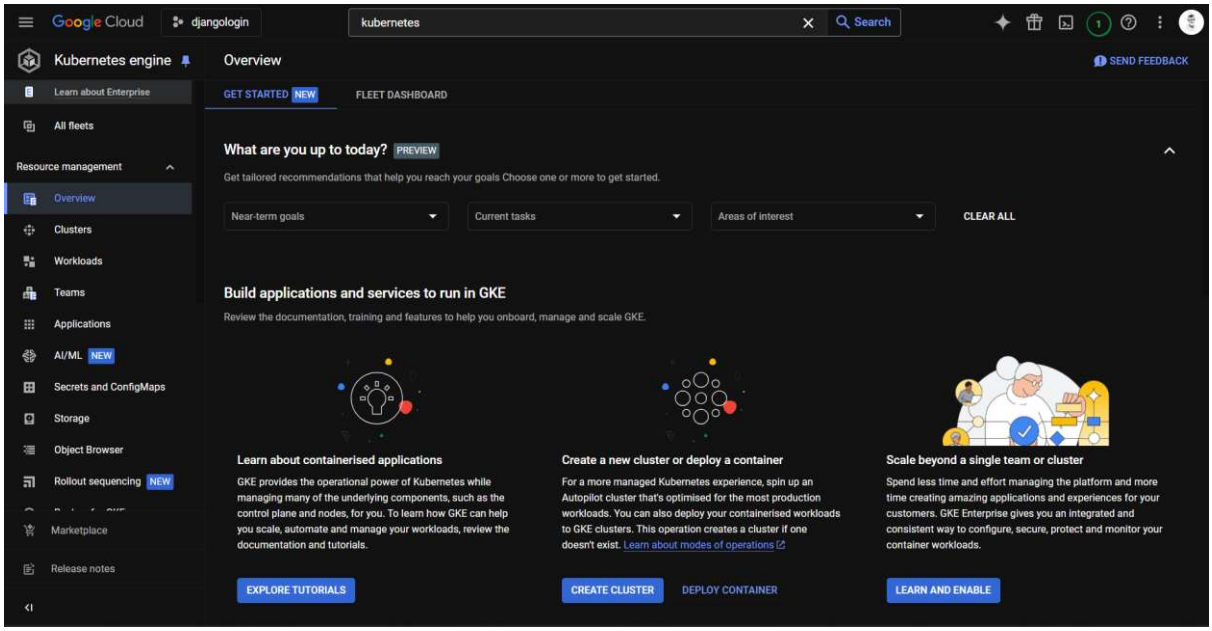
Verify the Application

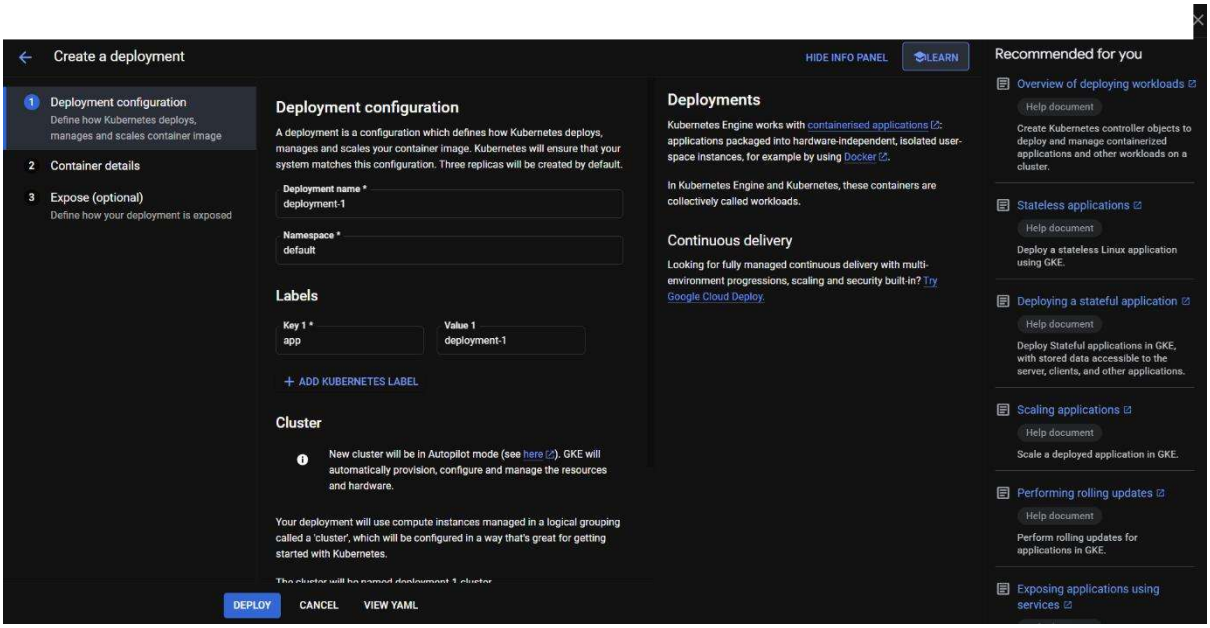
1. Open a web browser and navigate to the external IP address to verify your application is running.
2. You can also use kubectl to get the status of your pods and services:

kubectl get pods

kubectl get svc

Output




Create a deployment
HIDE INFO PANEL
LEARN
1 Deployment configuration
Define how Kubernetes deploys, manages and scales container image
2 Container details
3 Expose (optional)
Define how your deployment is exposed
Deployment configuration
A deployment is a configuration which defines how Kubernetes deploys, manages and scales your container image. Kubernetes will ensure that your system matches this configuration. Three replicas will be created by default.
Deployment name *
deployment-1
Namespace *
default
Labels
Key 1 *
app
Value 1
deployment-1
+ ADD KUBERNETES LABEL
Cluster
New cluster will be in Autopilot mode (see here). GKE will automatically provision, configure and manage the resources and hardware.
Your deployment will use compute instances managed in a logical grouping called a 'cluster', which will be configured in a way that's great for getting started with Kubernetes.
DEPLOY
CANCEL
VIEW YAML
Deployments
Kubernetes Engine works with containerised applications: applications packaged into hardware-independent, isolated user-space instances, for example by using Docker.
In Kubernetes Engine and Kubernetes, these containers are collectively called workloads.
Continuous delivery
Looking for fully managed continuous delivery with multi-environment progressions, scaling and security built-in? Try Google Cloud Deploy.
Recommended for you
Overview of deploying workloads
Help document
Create Kubernetes controller objects to deploy and manage containerized applications and other workloads on a cluster.
Stateless applications
Help document
Deploy a stateless Linux application using GKE.
Deploying a stateful application
Help document
Deploy Stateful applications in GKE, with stored data accessible to the server, clients, and other applications.
Scaling applications
Help document
Scale a deployed application in GKE.
Performing rolling updates
Help document
Perform rolling updates for applications in GKE.
Exposing applications using services