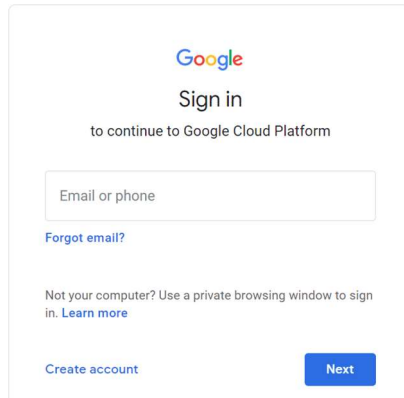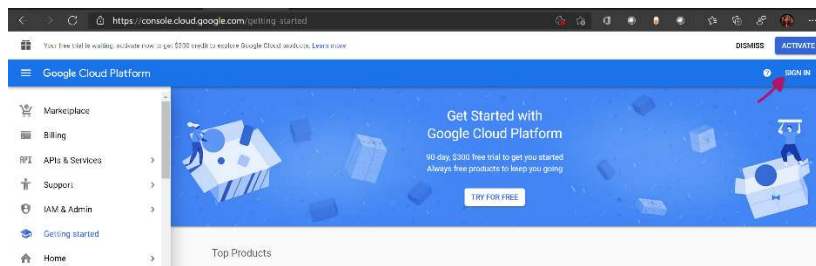# GKE Route-based Cluster

Sign up for free trial account on GCP
For this lab, you will need a google cloud account. Sign up for a free trial account on GCP.

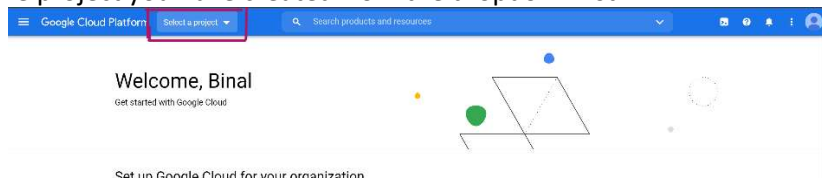1. Go to console.cloud.google.com and click 'Create Account' 'For myself':

2. Signin to your account. (Find signin in the top right corner.)

Select your project
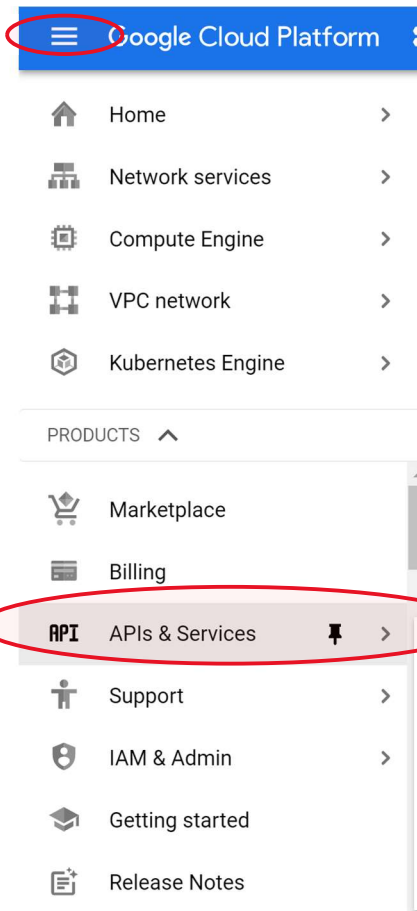If you haven't created a project yet, create a project.
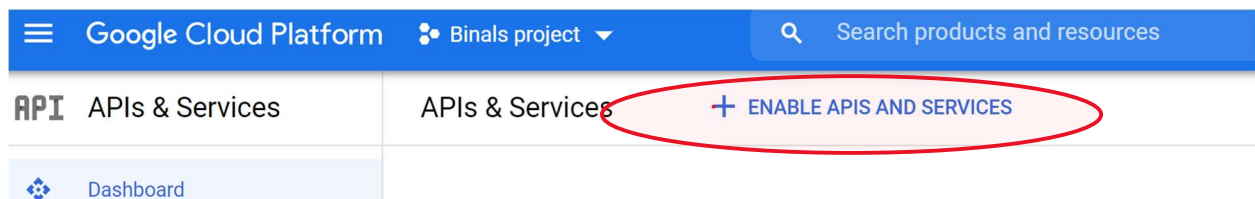Select the project you have created from the dropdown list.

Enable Cloud APIs
For each service you use, you need to ensure the Google cloud APIs are enabled. In this step, follow the steps to enable the Container Registry API.
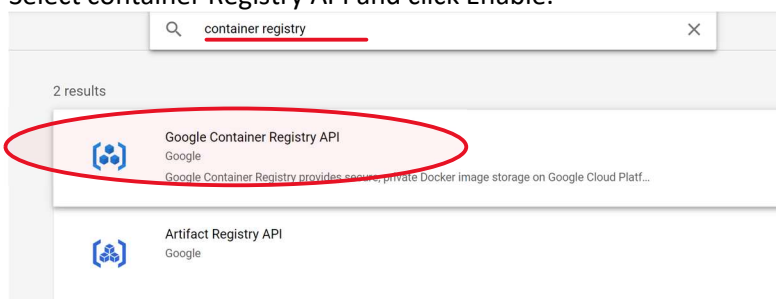
From the cloud console, go to the API and Services page.

Select **+Enable APIs and Services**.

Select container Registry API and click Enable.

Enable the following two APIs:

Kubernetes API
Compute Engine API

Setup cloud shell
Connect to the cloud shell, setup project variables and run the following steps to build an image.

- On the top bar to the right, Click **Activate** to connect to the cloud shell.

- Setup Project ID variable in cloud shell. Click on the Project dropdown in the top bar of the console to get the project ID.
    export PROJECT_ID=<your project ID>
    echo $PROJECT_ID

Build Container Image
Download the sample app:
We will use a sample hello app. Run the following command in your cloud shell.
    git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples

Create docker image:
The cloud shell includes Docker. Go to the hello-app directory and build docker image using the following commands:
    cd kubernetes-engine-samples/hello-app
    docker build -t gcr.io/${PROJECT_ID}/hello-app:v1 .
    docker images

Test locally:
    docker run --rm -p 8080:8080 gcr.io/${PROJECT_ID}/hello-app:v1
    Open a new terminal to test the app is running:
        curl http://localhost:8080

Push the image to container registry:
Google Container Registry is found at gcr.io/<PROJECT ID>. Upload the image to this registry.
    gcloud auth configure-docker
    docker push gcr.io/${PROJECT_ID}/hello-app:v1

Create GKE cluster
Next, create a GKE zonal route-based cluster.
    a. From the menu, navigate to the Kubernetes engine page.
    b. Click **+Create** to create a new cluster.
    c. On the 'Cluster basics' page, configure the following values:
        o Name: **hello-cluster**
        o Location type: Zonal
        o Master version: Release Channel
        o Release Channel: Rapid channel
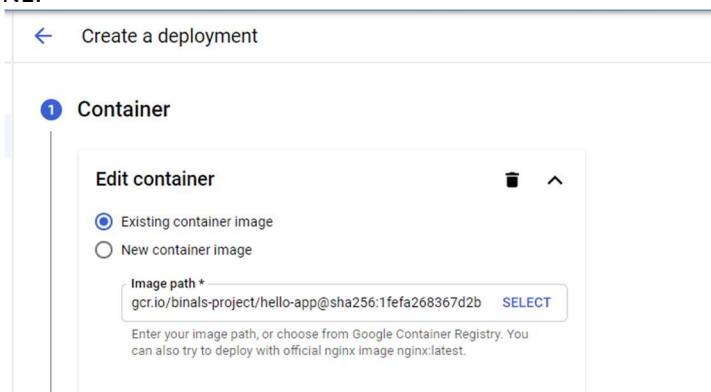
    d. On the left menu, select node pools and add the following configuration:
        i. Name: default-pool
        ii. Number of nodes: 3

      iii.   Select Enable autoscaling with minimum number of nodes 1 and maximum nodes 3.

      iv.   Under 'Nodes',

         1.   Machine Configuration: Series: N1

         2.   Machine type: n1-standard-1

         3.   Enable preemptible nodes

         4.   Networking: hello-nodes

e.   Under Cluster -> Networking

      i.   Select Public cluster

      ii.   Network: default

      iii.   Subnet: default

      iv.   Advanced networking:

         1.   Disable VPC-native routing

         2.   Disable HTTP Load balancing

f.   Click **Create** once you have configured the above values.

Deploy application

Next, deploy an app to the cluster.

- Go to the cluster menu and select the cluster you just created. Click DEPLOY from the top menu options.
- Select existing container image.
- For image path, select the image from the container registry you previously created, then click DONE.

Click DONE and CONTINUE to configure the application details as follows:

Application name: hello-app

Namespace: default

Labels: Key: app, Value: hello-app

Cluster: hello-cluster

Click DEPLOY.
Verify pods are successfully created and running.

Managed pods

Revision	Name	Status	Restarts	Created on ↑
1	hello-app-587fcc667c-kxg2n	✅ Running	0	Mar 1, 2021, 5:36:03 PM
1	hello-app-587fcc667c-6nxv7	✅ Running	0	Mar 1, 2021, 5:36:03 PM
1	hello-app-587fcc667c-n2rbw	✅ Running	0	Mar 1, 2021, 5:36:03 PM

Create a service for the app
To make the app accessible to the external world, from the deployment page, click EXPOSE to create a service. Configure the following information
External port: 80
Target port: 8080
Service type: Load balancer
Service name: hello-app-service
Click EXPOSE.

← Expose a deployment

Exposing a deployment creates a Kubernetes Service. A service lets your deployment receive traffic and defines

Port mapping

Port * ❓	Target port ❓	Protocol ❓
80	8080	TCP ▾

➕ ADD PORT MAPPING

Service type
Load balancer ▾ ❓

Service name
hello-app-service

EXPOSE   VIEW YAML

* indicates required field

Verify access to your application
On the GKE page, select Services and Ingress from the left menu.
Verify your service is up and running.

Services & Ingress   ⟳ REFRESH   ➕ CREATE INGRESS   🗑 DELETE

Cluster
cluster-demo , cluster-1 , a... ▾   Namespace
default, kube-node-lease, ... ▾   RESET   SAVE   BETA

SERVICES   INGRESS

Services are sets of Pods with a network endpoint that can be used for discovery and
load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to
Services.

≡ Is system object : False ⊗ Filter services and ingresses

☐	Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
☐	hello-app-service	✅ OK	External load balancer	104.197.36.80:80 ↗	3/3	default	cluster-1

You should see a public IP assigned to your external load balancer.
Access the service using the public IP address of the load balancer. The hello-app application Hello
World page should load successfully.

You have successfully deployed a GKE cluster on Google cloud.

Connect to the cluster:
From the Kubernetes Engine page, go to the Cluster menu and click on the cluster we just created. Click
CONNECT from the top menu bar.

Click RUN IN CLOUD SHELL. This should open the cloud shell. Run the command that is pasted in the shell to get the cluster credentials.

Run the following command to verify successful run:
        kubectl get pods

Update pods:
Update the hello app:
        In your cloud shell, you have cloned the hello app.
        Change to directory kubernetes-engine-samples/hello-app and open file main.go for editing.
        Update the version value from 1.0.0 to 2.0.0.

Ensure you have set the variable for project ID again.
export PROJECT_ID=<your project ID>
echo $PROJECT_ID

Build docker image:
        docker build -t gcr.io/${PROJECT_ID}/hello-app:v2 .

Push the image to the container registry:

docker push gcr.io/${PROJECT_ID}/hello-app:v2

Verify an updated image appears in the container registry with a label of v2.

Now we are ready to update the image on the pods:
You can update pod application from the workloads menu.
Click ACTIONS → ROLLING UPDATE

Select the right image from the container registry and paste it in the Container Image field.

Rolling update

Update workload Pods to a new application version.

Minimum seconds ready
2   ❓

Maximum surge
25%   ❓

Maximum unavailable
25%   ❓

Container images

Image of hello-app-sha256-1 *
gcr.io/binals-project/hello-app@sha256:051902ffe393f318efbb52a9981fdb47eea06e

* indicates required field

CANCEL   UPDATE

Watch the pods status until all pods are updated.

Managed pods

Revision	Name	Status	Restarts	Created on ↑
1	hello-app-587fcc667c-kxg2n	✅ Running	0	Mar 1, 2021, 5:36:03 PM
1	hello-app-587fcc667c-6nxv7	✅ Running	0	Mar 1, 2021, 5:36:03 PM
1	hello-app-587fcc667c-n2rbw	✅ Running	0	Mar 1, 2021, 5:36:03 PM
2	hello-app-76f7d58f7d-v5n6w	🔄 ContainerCreating	0	Mar 1, 2021, 6:42:18 PM

Managed pods

Revision	Name	Status	Restarts	Created on ↑
2	hello-app-76f7d58f7d-v5n6w	✅ Running	0	Mar 1, 2021, 6:42:18 PM
2	hello-app-76f7d58f7d-r899k	✅ Running	0	Mar 1, 2021, 6:42:23 PM
2	hello-app-76f7d58f7d-l8ccg	✅ Running	0	Mar 1, 2021, 6:42:27 PM

Delete Cluster

From the cluster menu, select the cluster you just created. Click DELETE. This will delete all the configuration related to the cluster.