**10-steps to deploy a ML pipeline on Google Kubernetes Engine:**

## **Step 1 — Create a new project in GCP Console**

Click on **Create New Project**

**Step 2 — Import Project Code**

Click the **Activate Cloud Shell**button at the top of the console window to open the Cloud Shell.

git clone <https://github.com/pycaret/pycaret-deployment-google.git>

**Step 3— Set Project ID Environment Variable**

export PROJECT\_ID=**pycaret-kubernetes-demo**

pycaret-kubernetes-demo is the name of the project we chose in step 1 above

**Step 4— Build the docker image**

Build the docker image of the application and tag it for uploading by executing the following code. Here gcr is Google Container Registry and v1 is the tag

docker build -t gcr.io/${PROJECT\_ID}/insurance-app:v1 .

Check Available images: docker images

**Step 5— Upload the container image**

1. Authenticate to [Container Registry](https://cloud.google.com/container-registry) (you need to run this only once):

gcloud auth configure-docker

2. Execute the following code to upload the docker image to Google Container Registry:

docker push gcr.io/${PROJECT\_ID}/insurance-app:v1

## **Step 6— Create Cluster**

Now that the container is uploaded, you need a cluster to run the container. A cluster consists of a pool of Compute Engine VM instances, running Kubernetes.

1. Set your project ID and Compute Engine zone options for the gcloud tool:

gcloud config set project $PROJECT\_ID   
gcloud config set compute/zone **us-central1**

2. Create a cluster by executing the following code:

gcloud container clusters create **insurance-cluster** --num-nodes=2

## **Step 7— Deploy Application**

To deploy and manage applications on a GKE cluster, you must communicate with the Kubernetes cluster management system. Execute the following command to deploy the application:

kubectl create deployment insurance-app --image=gcr.io/${PROJECT\_ID}/insurance-app:v1

## **Step 8— Expose your application to the internet**

By default, the containers you run on GKE are not accessible from the internet because they do not have external IP addresses. Execute the following code to expose the application to the internet:

kubectl expose deployment insurance-app --type=LoadBalancer --port 80 --target-port 8080

## **Step 9— Check Service**

Execute the following code to get the status of the service. **EXTERNAL-IP** is the web address you can use in browser to view the published app.

kubectl get service

Step 10— See the app in action on Master IP address you get from above command