Published in Towards Data Science

Moez Ali Follow

May 28, 2020 · 10 min read · Listen

Deploy Machine Learning Pipeline on Google Kubernetes Engine

A step-by-step beginner's guide to containerize and deploy ML pipeline on Google Kubernetes Engine

RECAP

In our last post on deploying a machine learning pipeline in the cloud, we demonstrated how to develop a machine learning pipeline in PyCaret, containerize it with Docker and serve as a web app using Microsoft Azure Web App Services. If you haven't heard about PyCaret before, please read this announcement to learn more

deploy a machine learning pipeline on Google Kubernetes Engine.

👉 Learning Goals of this Tutorial

- Learn what is a Container, what is Docker, what is Kubernetes, and what is Google Kubernetes Engine?
- Build a Docker image and upload it on Google Container Registry (GCR).
- Create clusters and deploy a machine learning pipeline with a Flask app as a web service.
- See a web app in action that uses a trained machine learning pipeline to predict new data points in real-time.

Previously we demonstrated how to deploy a ML pipeline on Heroku PaaS and how to deploy a ML pipeline on Azure Web Services with a Docker container.

This tutorial will cover the entire workflow starting from building a docker image, uploading it onto Google Container Registry and then deploying the pre-trained machine learning pipeline and Flask app onto Google Kubernetes Engine (GKE).

💻 Toolbox for this tutorial

PyCaret

PyCaret is an open source, low-code machine learning library in Python that is used to train and deploy machine learning pipelines and models into production. PyCaret can be installed easily using pip.

```
pip install pycaret
```

Flask

Google Cloud Platform

Google Cloud Platform (GCP), offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail and YouTube. If you do not have an account with GCP, you can sign-up here. If you are signing up for the first time you will get free credits for 1 year.

Let's get started.

Before we get into Kubernetes, let's understand what a container is and why we would need one?

https://www.freepik.com/free-photos-vectors/cargo-ship

Have you ever had the problem where your code works fine on your computer but when a friend tries to run the exact same code, it doesn't work? If your friend is repeating the exact same steps, he or she should get the same results, right? The one-word answer to this is ***the environment.*** Your friend's environment is different than yours.

What does an environment include? → Programing language such as Python and all the libraries and dependencies with the exact versions using which application was built

reproduce the results anywhere. Hence, ***a container*** is a type of software that packages up an application and all its dependencies so the application runs reliably from one computing environment to another.

> *What's Docker then?*

Docker is a company that provides software (also called Docker) that allows users to build, run and manage containers. While Docker's container are the most common, there are other less famous *alternatives* such as LXD and LXC that provides container solution.

Now that you understand containers and docker specifically, let's understand what Kubernetes is all about.

What is Kubernetes?

Kubernetes is a powerful open-source system developed by Google back in 2014, for managing containerized applications. In simple words, Kubernetes is a system for running and coordinating containerized applications across a cluster of machines. It is a platform designed to completely manage the life cycle of containerized applications.



Photo by chuttersnap on Unsplash

Features

✔ **Load Balancing:** Automatically distributes the load between containers.

✔ **Scaling:** Automatically scale up or down by adding or removing containers when demand changes such as peak hours, weekends and holidays.

✔ **Storage:** Keeps storage consistent with multiple instances of an application.

✔ **Self-healing** Automatically restarts containers that fail and kills containers that don't respond to your user-defined health check.

✔ **Automated Rollouts** you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all of their resources to the

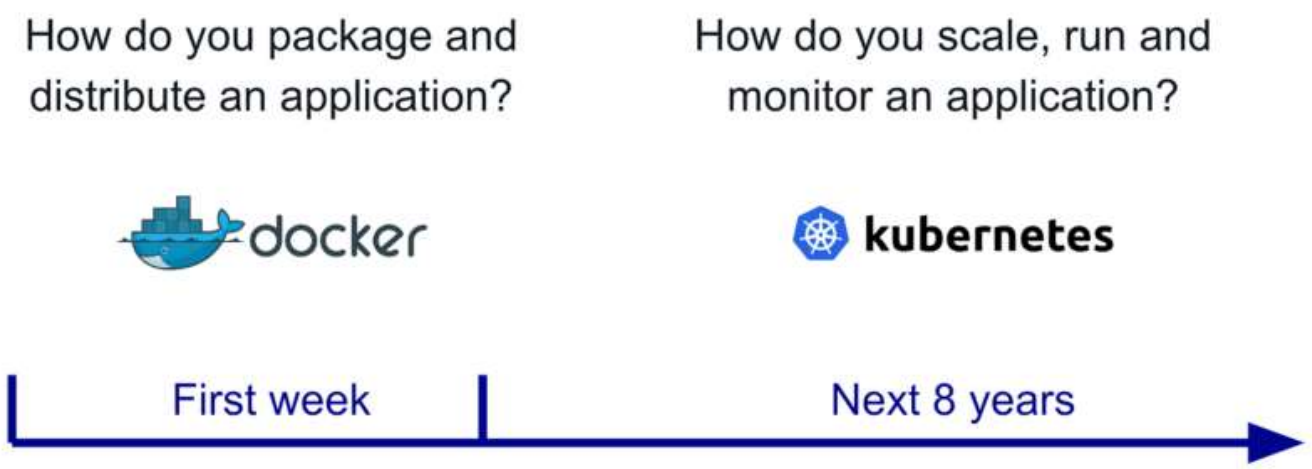
machines to support an enterprise level ML application with varied workloads during day and night. As simple as it may sound, it is a lot of work to do manually.

You need to start the right containers at the right time, figure out how they can talk to each other, handle storage considerations, and deal with failed containers or hardware. This is the problem Kubernetes is solving by allowing large numbers of containers to work together in harmony, reducing the operational burden.

> It's a mistake to compare Docker with Kubernetes. These are two different technologies. Docker is a software that allows you to containerize applications while Kubernetes is a container management system that allows to create, scale and monitor hundreds and thousands of containers.

In the lifecycle of any application, Docker is used for packaging the application at the time of deployment, while kubernetes is used for rest of the life for managing the application.





Lifecycle of an application deployed through Kubernetes / Docker

What is Google Kubernetes Engine?

Google Kubernetes Engine is implementation of *Google's open source Kubernetes* on Google Cloud Platform. Simple!

Other popular alternatives to GKE are Amazon ECS and Microsoft Azure Kubernetes Service.

One final time, do you understand this?

- **A Container** is a type of software that packages up an application and all its dependencies so the application runs reliably from one computing environment to another.
- **Docker** is a software used for building and managing containers.
- **Kubernetes** is an open-source system for managing containerized applications in a clustered environment.
- **Google Kubernetes Engine** is an implementation of the open source Kubernetes framework on Google Cloud Platform.

patient charges using demographic and basic patient health risk metrics at the time of hospitalization.

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

(data source)

Objective

To build and deploy a web application where the demographic and health information of a patient is entered into a web-based form which then outputs a predicted charge amount.

Tasks

- Train and develop a machine learning pipeline for deployment.
- Build a web app using a Flask framework. It will use the trained ML pipeline to generate predictions on new data points in real-time.
- Build a docker image and upload a container onto Google Container Registry (GCR).
- Create clusters and deploy the app on Google Kubernetes Engine.

Since we have already covered the first two tasks in our initial tutorial, we will quickly recap them and then focus on the remaining items in the list above. If you are interested in learning more about developing a machine learning pipeline in Python using PyCaret and building a web app using a Flask framework, please read this

pipeline which will be used as part of our web app. The Machine Learning Pipeline can be developed in an Integrated Development Environment (IDE) or Notebook. We have used a notebook to run the below code:

```
# Import dataset from pycaret repository
from pycaret.datasets import get_data
insurance = get_data('insurance')

# Initialize environment
from pycaret.regression import *
r1 = setup(insurance, target = 'charges', session_id = 123,
           normalize = True,
           polynomial_features = True, trigonometry_features = True,
           feature_interaction=True,
           bin_numeric_features= ['age', 'bmi'])

# Train a linear regression model
lr = create_model('lr')

# save transformation pipeline and model
save_model(lr, model_name = 'c:/username/pycaret-deployment-azure/deployment_28042020')
```

deployment_28042020.py hosted with ❤ by GitHub view raw

When you save a model in PyCaret, the entire transformation pipeline based on the configuration defined in the **setup()** function is created . All inter-dependencies are orchestrated automatically. See the pipeline and model stored in the 'deployment_28042020' variable:



```
                                display_types=True, features_todrop=[],
                                ml_usecase='regression',
                                numerical_features=[], target='charges',
                                time_features=[])),
                ('imputer',
                 Simple_Imputer(categorical_strategy='not_available',
                                numeric_strategy='mean',
                                target_variable=None)),
                ('new_levels1',
                 New_Catagorical_Levels...
                ('dummy', Dummify(target='charges')),
                ('fix_perfect', Remove_100(target='charges')),
                ('clean_names', Clean_Colum_Names()),
                ('feature_select', Empty()), ('fix_multi', Empty()),
                ('dfs',
                 DFS_Classic(interactions=['multiply'], ml_usecase='regression',
                             random_state=123, subclass='binary',
                             target='charges',
                             top_features_to_pick_percentage=None)),
                ('pca', Empty())],
         verbose=False),
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False),
None]
```

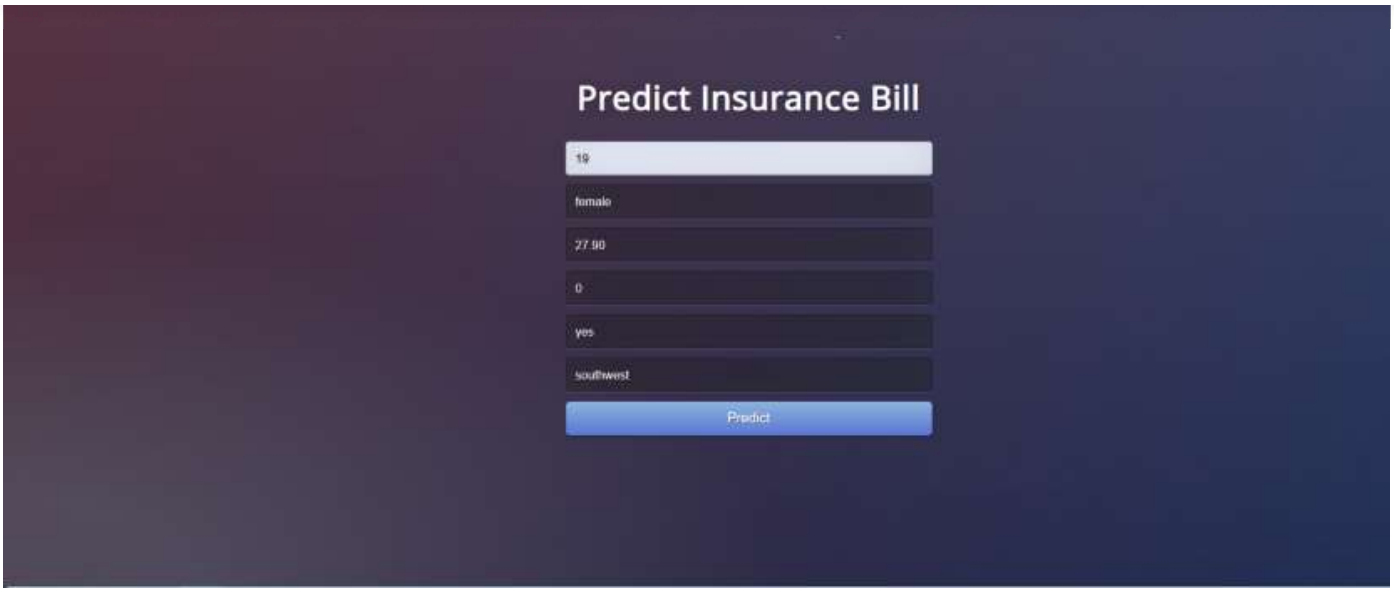
Machine Learning Pipeline created using PyCaret

👉 Build a Web Application

This tutorial is not focused on building a Flask application. It is only discussed here for completeness. Now that our machine learning pipeline is ready we need a web application that can connect to our trained pipeline to generate predictions on new data points in real-time. We have created the web application using Flask framework in Python. There are two parts of this application:

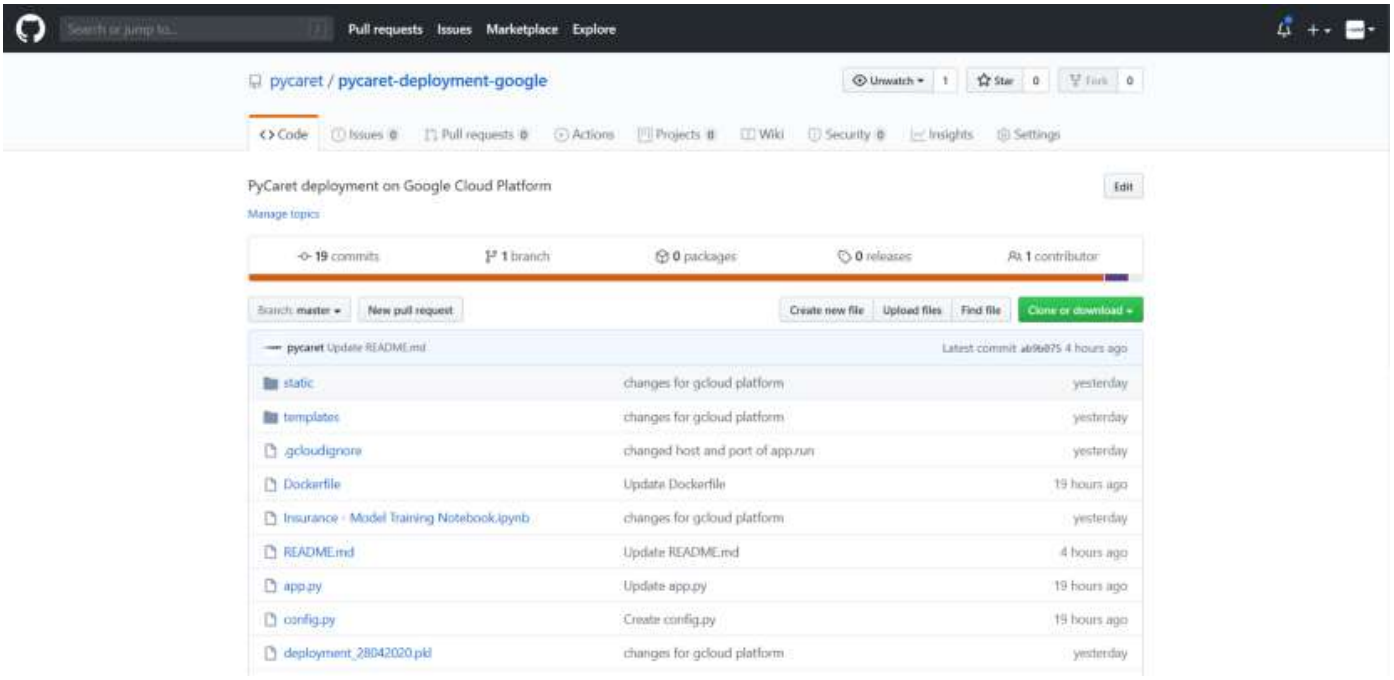
- Front-end (designed using HTML)
- Back-end (developed using Flask)

This is how our web application looks:



Web application on local machine

If you haven't followed along so far, no problem. You can simply fork this repository from GitHub. This is how your project folder should look at this point:

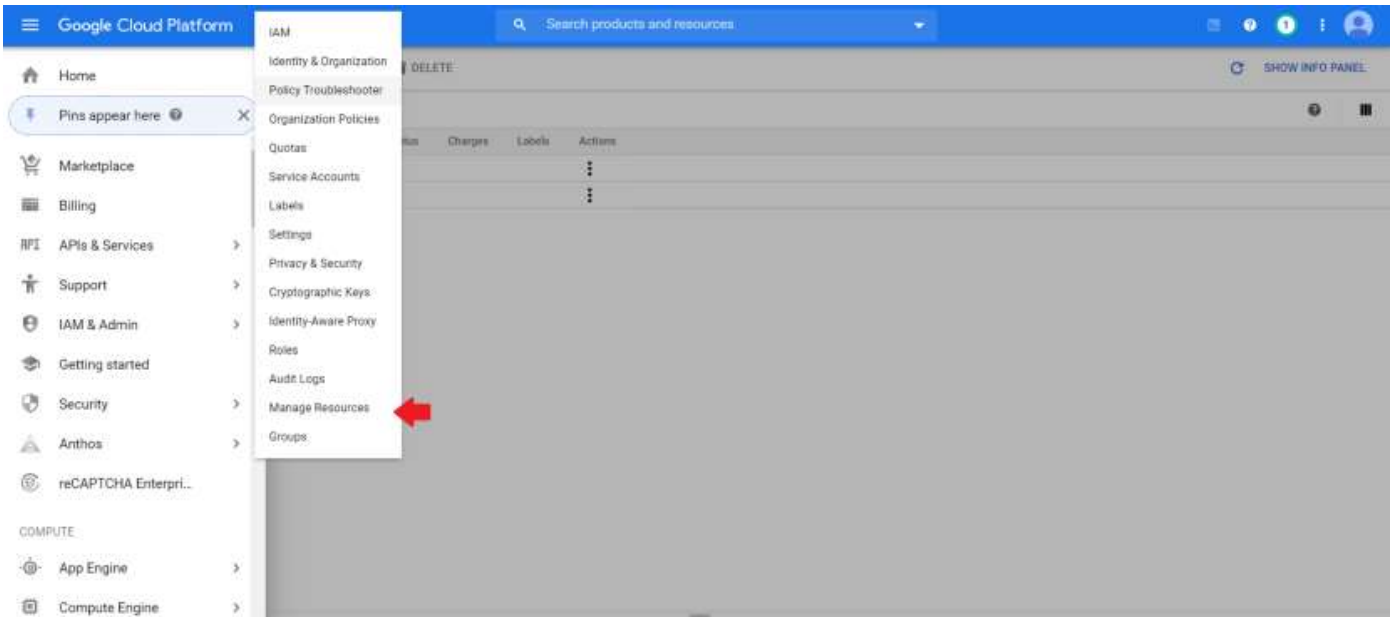


https://www.github.com/pycaret/pycaret-deployment-google

Now that we have a fully functional web application, we can start the process of

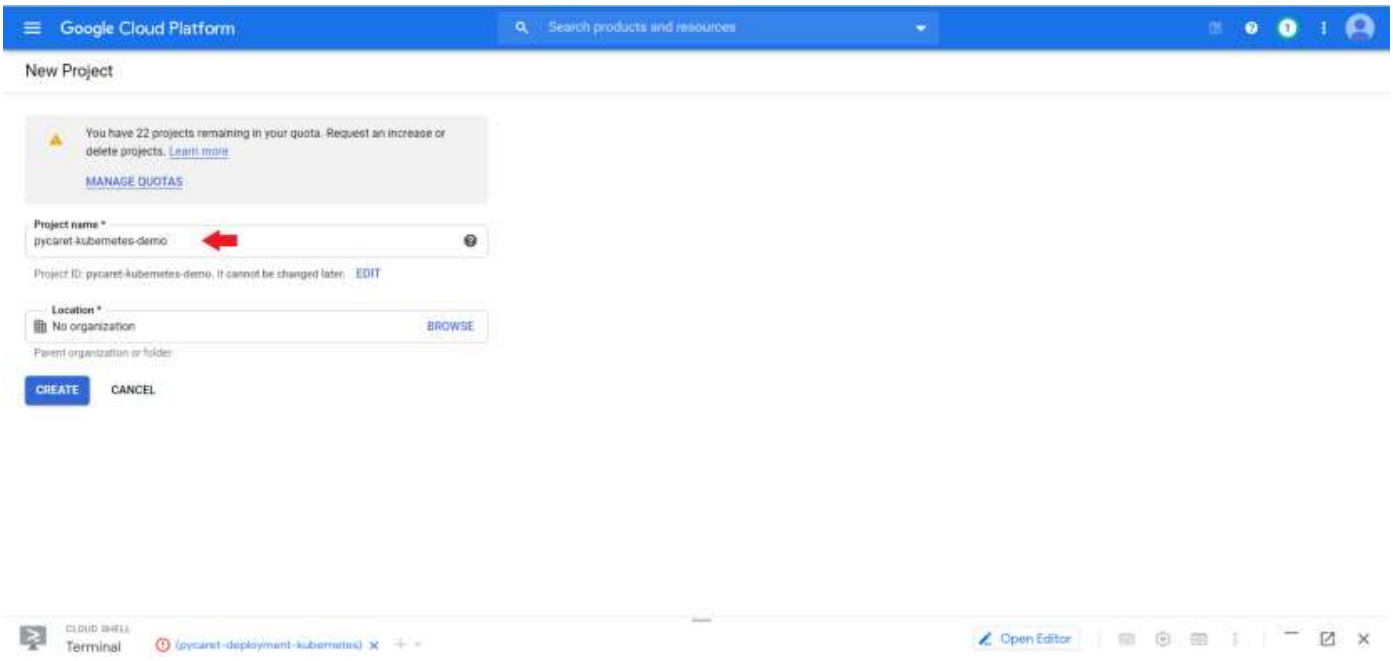
👉 Step 1 — Create a new project in GCP Console

Sign-in to your GCP console and go to Manage Resources

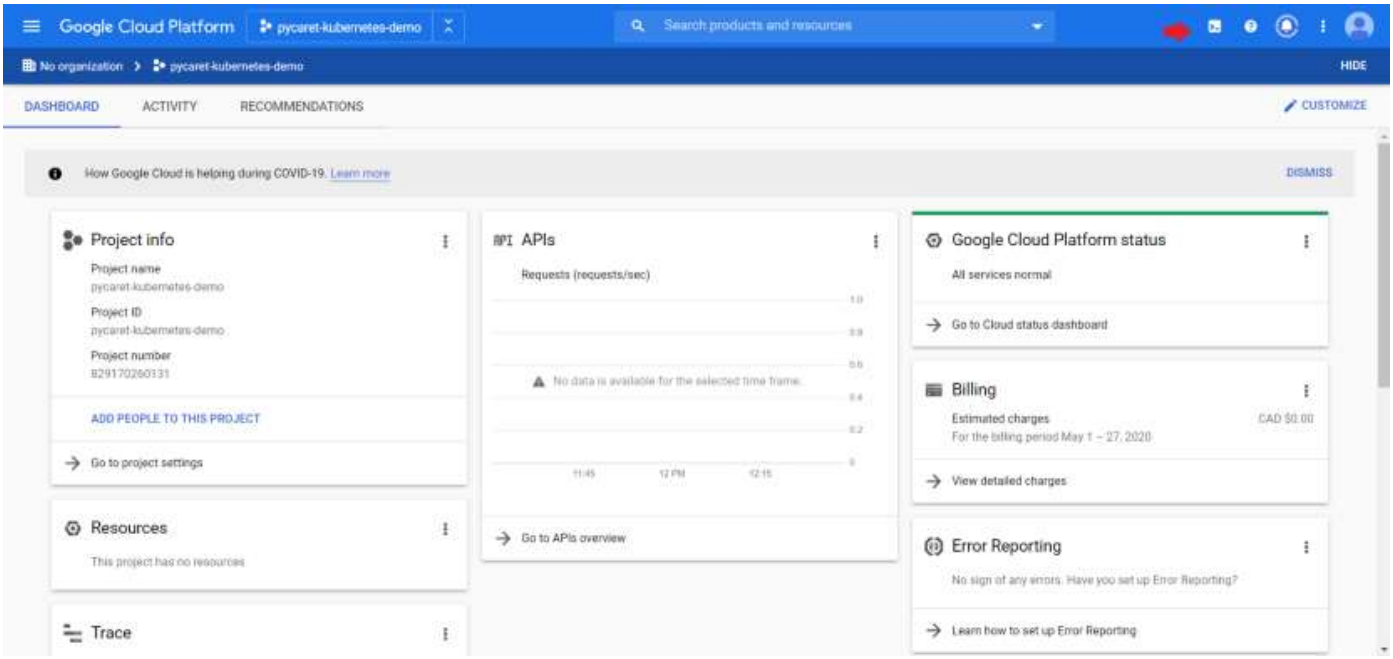


Google Cloud Platform Console → Manage Resources

Click on **Create New Project**



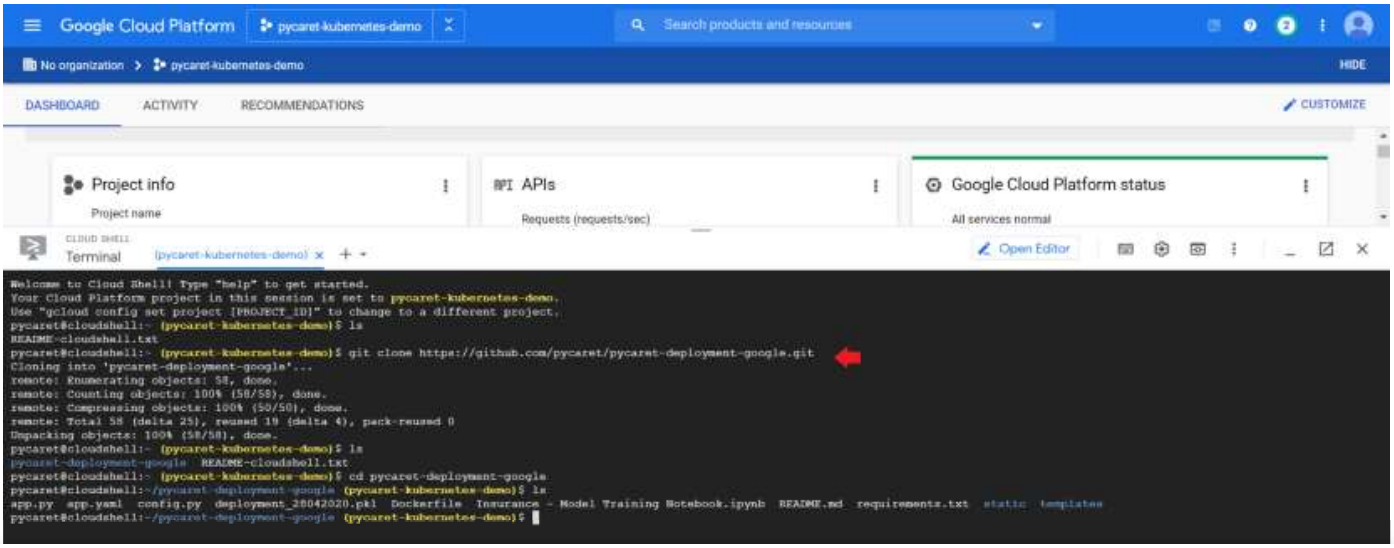
Google Cloud Platform Console → Manage Resources → Create New Project



Google Cloud Platform (Project Info Page)

Execute the following code in Cloud Shell to clone the GitHub repository used in this tutorial.

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



👉 Step 3— Set Project ID Environment Variable

Execute the following code to set the PROJECT_ID environment variable.

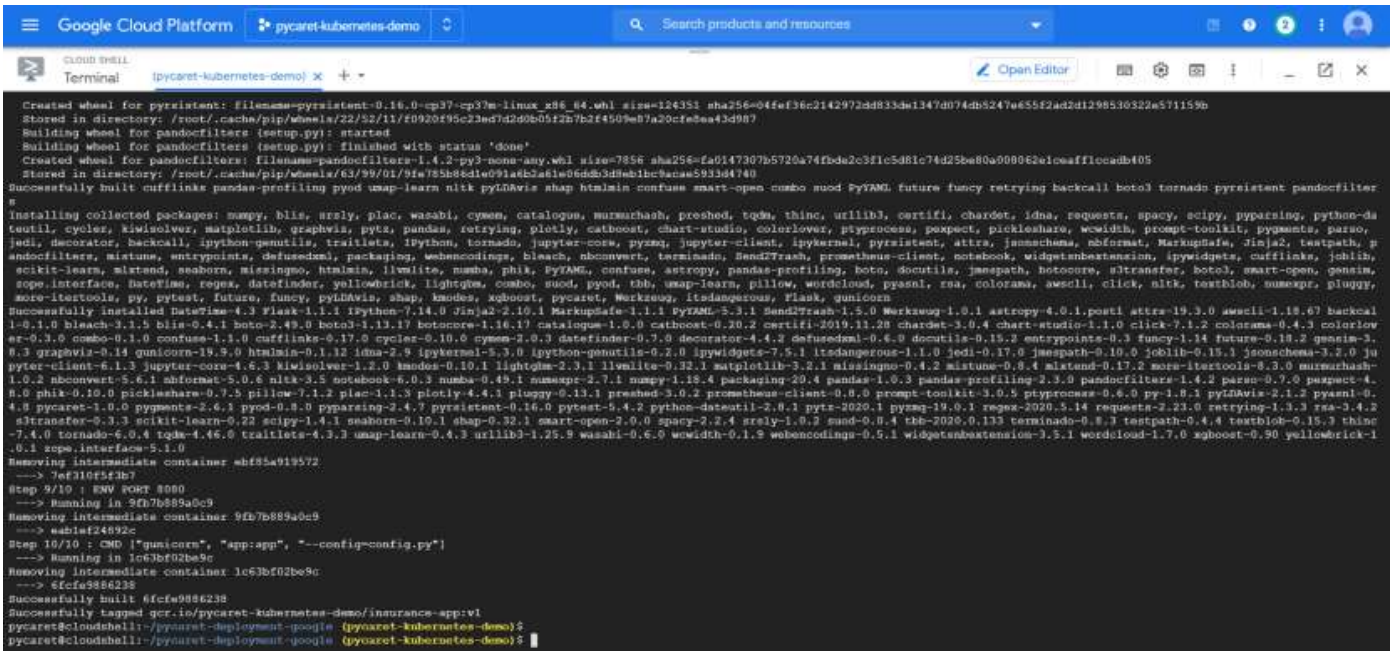
```
export PROJECT_ID=pycaret-kubernetes-demo
```

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

👉 Step 4— Build the docker image

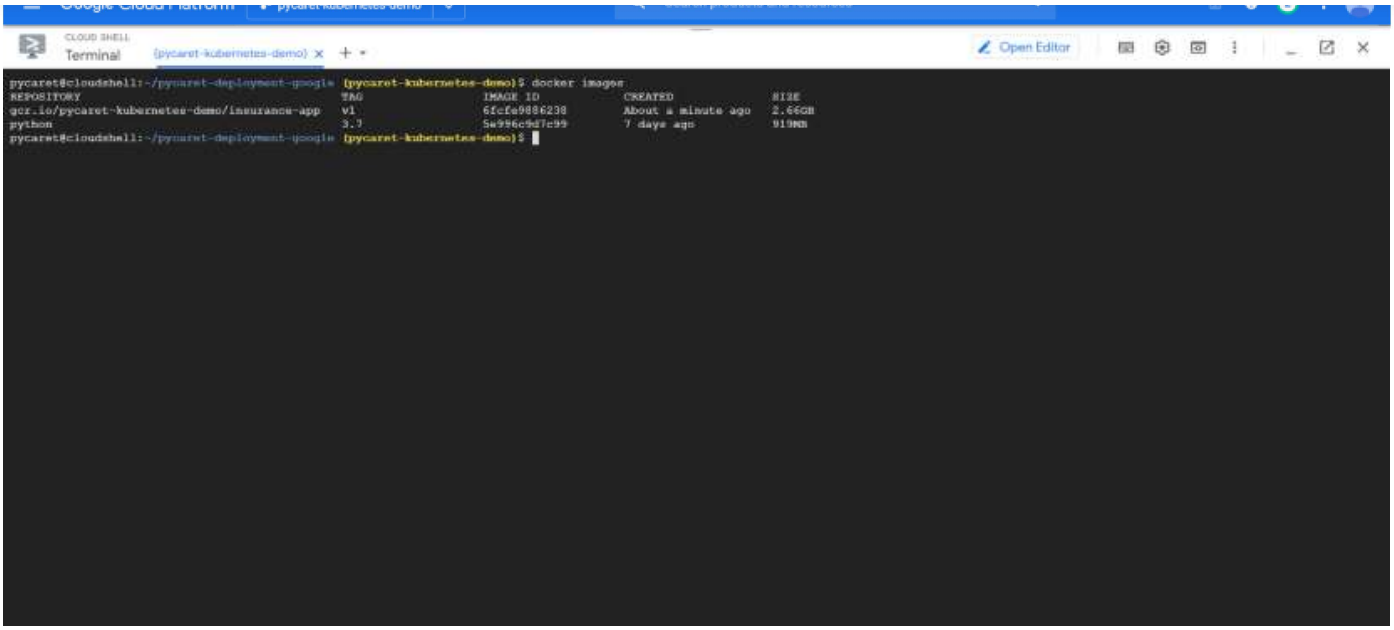
Build the docker image of the application and tag it for uploading by executing the following code:

```
docker build -t gcr.io/${PROJECT_ID}/insurance-app:v1 .
```



Message returned when docker build is successful

You can check the available images by running the following code:



Output of “docker images” command on Cloud Shell

👉 Step 5— Upload the container image

1. Authenticate to 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:

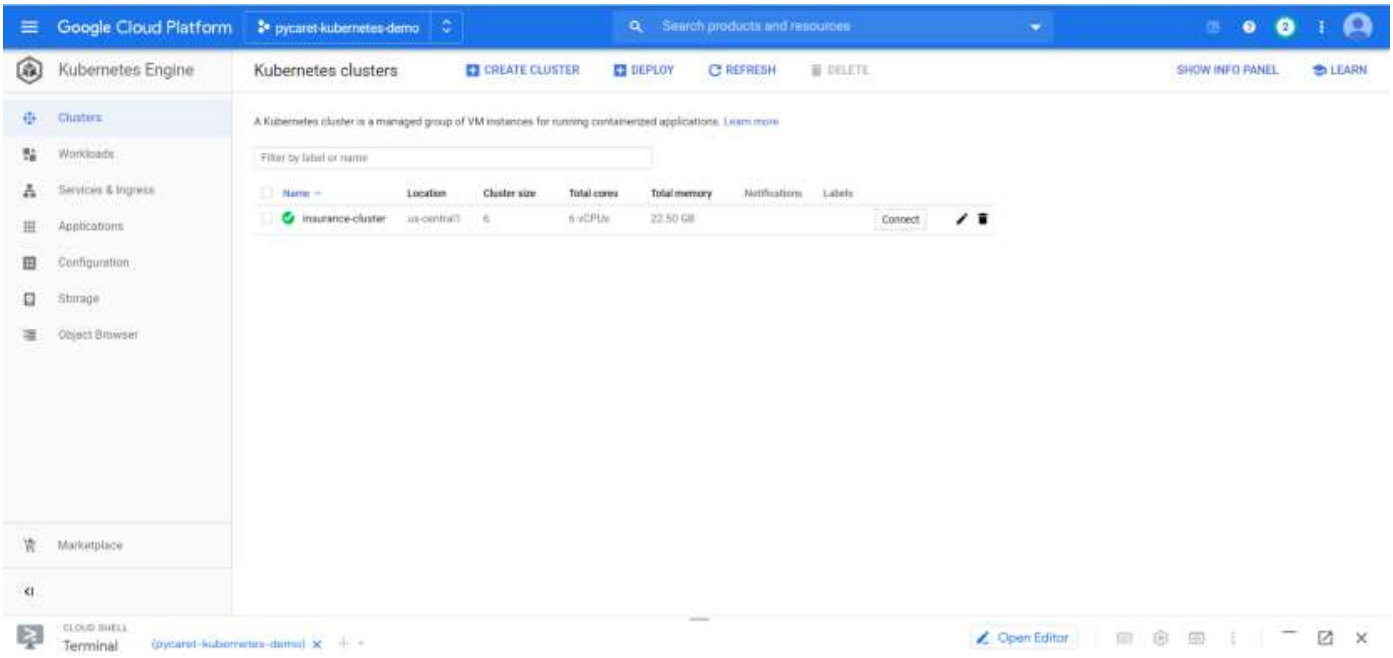
```
docker push gcr.io/${PROJECT_ID}/insurance-app:v1
```

👉 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 container clusters create insurance-cluster --num-nodes=2
```

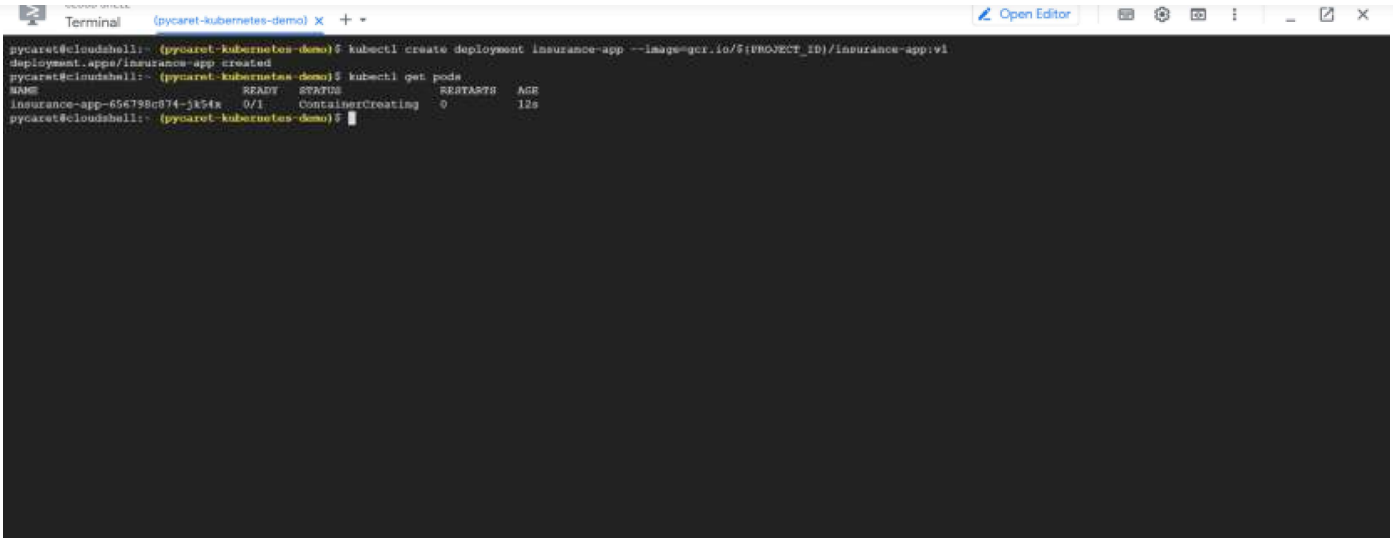


Google Cloud Platform → Kubernetes Engine → Clusters

👉 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
```



Output returned on creating deployment through kubectl

👉 Step 8— Expose your application to the internet

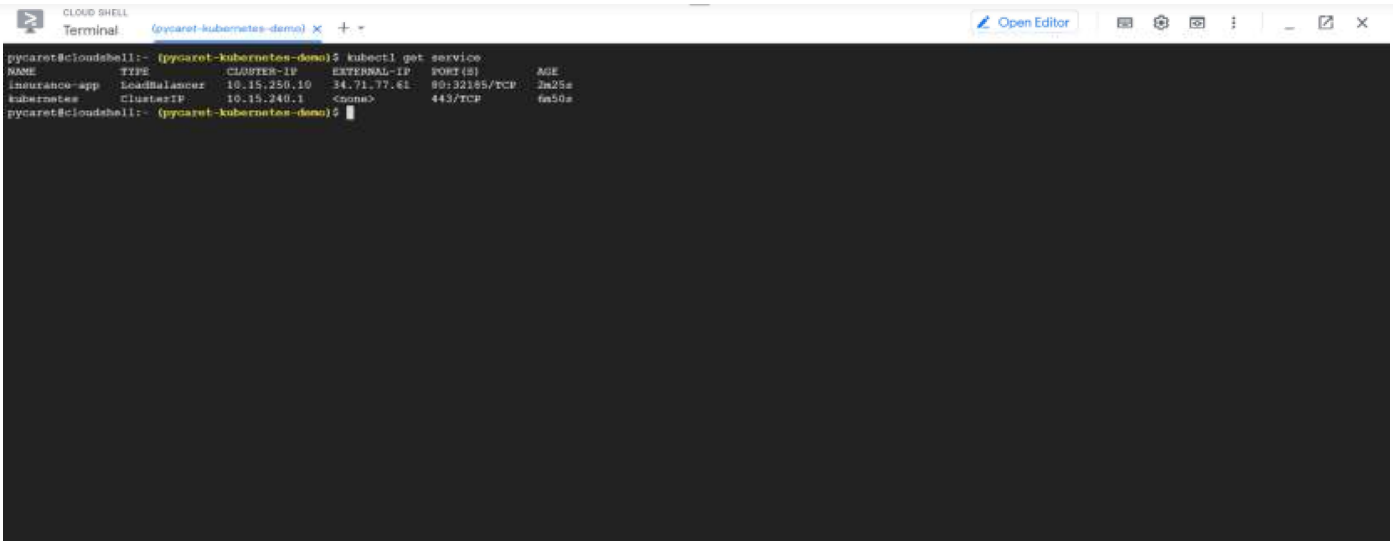
By default, the containers you run on GKE are not accessible from the internet because they do not have external IP addresses. Execute the following code to expose the application to the internet:

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

👉 Step 9— Check Service

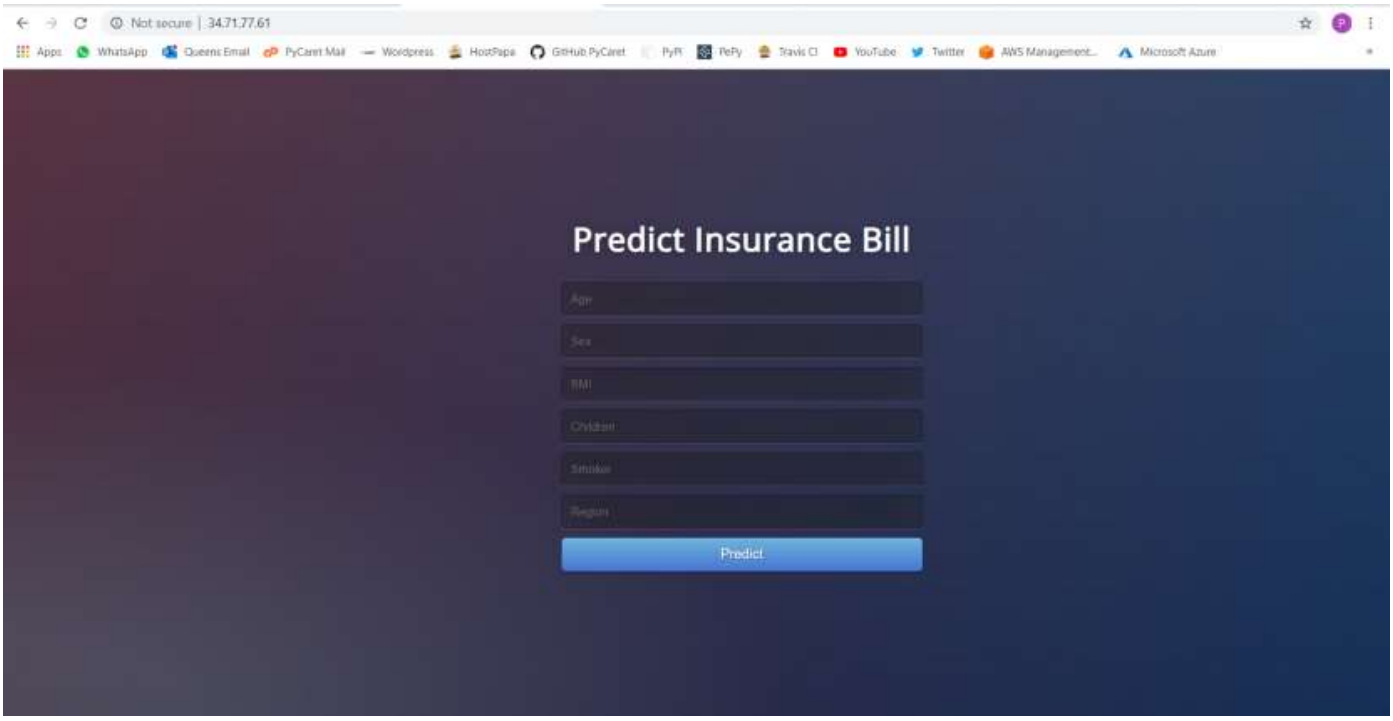
Execute the following code to get the status of the service. **EXTERNAL-IP** is the web address you can use in browser to view the published app.

```
kubectl get service
```



Cloud Shell → kubectl get service

👉 Step 10— See the app in action on http://34.71.77.61:8080



Final app uploaded on http://34.71.77.61:8080

Note: By the time this story is published, the app will be removed from the public address to restrict resource consumption.

Link to GitHub Repository for Heroku Deployment

PyCaret 1.0.1 is coming!

We have received overwhelming support and feedback from the community. We are actively working on improving PyCaret and preparing for our next release. **PyCaret 1.0.1 will be bigger and better**. If you would like to share your feedback and help us improve further, you may fill this form on the website or leave a comment on our GitHub or LinkedIn page.

Follow our LinkedIn and subscribe to our YouTube channel to learn more about PyCaret.

Want to learn about a specific module?

As of the first release 1.0.0, PyCaret has the following modules available for use. Click on the links below to see the documentation and working examples in Python.

Classification
Regression
Clustering
Anomaly Detection
Natural Language Processing
Association Rule Mining

Also see:

PyCaret getting started tutorials in Notebook:

Clustering
Anomaly Detection
Natural Language Processing
Association Rule Mining



Would you like to contribute?

PyCaret is an open source project. Everybody is welcome to contribute. If you would like contribute, please feel free to work on open issues. Pull requests are accepted with unit tests on dev-1.0.1 branch.

Please give us ⭐ on our GitHub repo if you like PyCaret.

Medium : https://medium.com/@moez_62905/

LinkedIn : https://www.linkedin.com/in/profile-moez/

Twitter : https://twitter.com/moezpycaretorg1