Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2021, 8(11):28-38

Research Article ISSN: 2394 - 658X

Deploying Python APIs on GCP GKE with HELM: A Comprehensive Guide

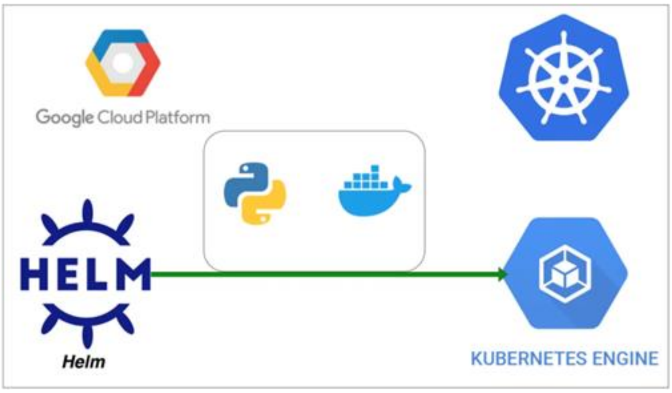
Bhargav Bachina

ABSTRACT

Kubernetes has rapidly become a cornerstone technology in modern IT infrastructure, with widespread adoption across various industries. However, deploying applications on Kubernetes entails managing numerous objects such as deployments, configmaps, and secrets, each defined in manifest files. While this approach works for initial deployments, it becomes cumbersome for repeated deployments. Enter Helm, the Kubernetes package manager designed to simplify application deployment, enhance security, and provide configurability. Helm streamlines the deployment process by enabling users to package and manage Kubernetes applications efficiently. In this paper, we explore deploying Python APIs on Google Cloud Platform's Google Kubernetes Engine (GKE) using Helm, covering essential prerequisites, Dockerization, container image pushing, Helm chart creation, GKE cluster configuration, Helm chart installation, accessing the deployed API, cluster cleanup, and concluding insights. Through this exploration, we aim to provide a comprehensive guide for deploying applications on Kubernetes with Helm, empowering users to leverage the full potential of Kubernetes in their projects.

Key words: Kubernetes, Programming, Software Development, Cloud Computing, Google Cloud Platform

INTRODUCTION



KUBERNETES has rapidly emerged as a fundamental technology in contemporary IT infrastructure, finding widespread adoption across diverse industries. However, the process of deploying applications on Kubernetes involves the management of numerous entities like deployments, configmaps, and secrets, each meticulously defined within manifest files. While this method suffices for initial deployments, it becomes unwieldy for iterative deployment cycles. Herein enters Helm, the Kubernetes package manager engineered to streamline application deployment, bolster security, and offer configurability. Helm simplifies the deployment workflow by allowing users to efficiently package and administer Kubernetes applications. This paper delves into the deployment of Python APIs on Google Cloud Platform's Google Kubernetes Engine (GKE) using Helm, covering essential prerequisites, Dockerization, container image pushing, Helm chart creation, GKE cluster configuration, Helm chart installation, API accessibility, cluster cleanup, and key insights. Through this exploration, our aim is to furnish a comprehensive guide for deploying applications on Kubernetes with Helm, empowering users to harness the full capabilities of Kubernetes in their projects.

- Example Project
- Prerequisites
- Install gcloud CLI and Configure
- Dockerize the Project
- Pushing Docker Image to Container Registry
- Building a Helm Chart
- Creating GKE Cluster
- Configure Kuebctl with GKE Cluster
- Installing Helm Chart on GKE Cluster
- Access the Installed API
- Deleting the Cluster
- Summary
- Conclusion

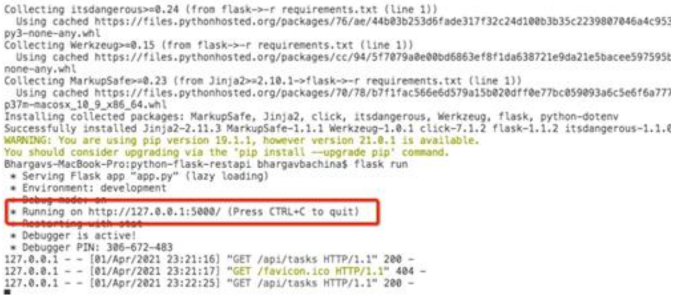
EXAMPLE PROJECT

Here is the Github link for the example project you can just clone and run on your machine.

// clone the projectgit clone https://github.com/bbachi/python-flask-restapi.git

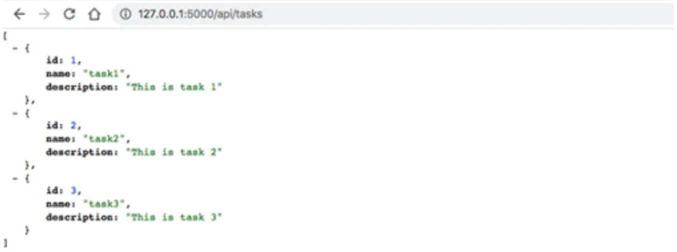
You need to run the following commands to install the required dependencies and start the project.

// install dependencies (Mac OS)python3 -m pip install -r requirements.txt



```
Collecting itsdangerous>=0.24 (from flask->-r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953
py3-none-any.whl
Collecting Werkzeug>=0.15 (from flask->-r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b
none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask->-r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/70/78/b7f1fac566e6d579a15b020dff0e77bc059093a6c5e6f6a777
p37m-macosx_10_9_x86_64.whl
Installing collected packages: MarkupSafe, Jinja2, click, itsdangerous, Werkzeug, flask, python-dotenv
Successfully installed Jinja2-2.11.3 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 flask-1.1.2 itsdangerous-1.1.0
WARNING: You are using pip version 19.1.1, however version 21.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ flask run
 * Serving Flask app "app.py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 306-672-483
127.0.0.1 - - [01/Apr/2021 23:21:16] "GET /api/tasks HTTP/1.1" 200 -
127.0.0.1 - - [01/Apr/2021 23:21:17] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [01/Apr/2021 23:22:25] "GET /api/tasks HTTP/1.1" 200 -
```

Figure 1: Flask Run



```
[
  - {
        id: 1,
        name: "task1",
        description: "This is task 1"
    },
  - {
        id: 2,
        name: "task2",
        description: "This is task 2"
    },
  - {
        id: 3,
        name: "task3",
        description: "This is task 3"
    }
]
```

Figure 2: API Running on port 5000.

PREREQUISITES

If you are new to Python REST API, please go through the below link on how to develop and build the Python REST API with Flask Restful.

***How To Write REST API With Python and Flask* (https://medium.com/bb-tutorials-and-thoughts/how-to-write-rest-api-with-python-and-flask-71ab42d253c5**

The other prerequisites to this post are Docker essentials and Kubernetes essentials. We are not going to discuss the basics such as what a container is or what Kubernetes is, rather, we will see how to build a Kubernetes cluster on GCP GKE. Below are the prerequisites you should know before going through this article.

A. Docker Essentials

You need to understand Docker concepts such as creating images, container management, etc. Below are some of the links that you can understand about Docker if you are new.

- Docker Docs (https://docs.docker.com/)
- Docker — A Beginner's guide to Dockerfile with a sample project (https://medium.com/bb-tutorials-and-thoughts/docker-a-beginners-guide-to-dockerfile-with-a-sample-project-6c1ac1f17490)
- Docker — Image creation and Management (https://medium.com/bb-tutorials-and-thoughts/docker-image-creation-and-management-9d91e4c277b1)
- Docker — Container Management with Examples (https://medium.com/bb-tutorials-and-thoughts/docker-container-management-with-examples-c280906158a8)
- Understanding Docker Volumes with an example (https://medium.com/bb-tutorials-and-thoughts/understanding-docker-volumes-with-an-example-d898cb5e40d7)

B. Kubernetes Essentials

You need to understand Kubernetes' essentials as well along with Docker essentials. Here are some of the docs to help you understand the concepts of Kubernetes.

- Kubernetes Docs (https://kubernetes.io/docs/concepts/)
- How to Get Started with Kubernetes (https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-kubernetes-e06ea82d23b)
- Some Example Projects (https://medium.com/bb-tutorials-and-thoughts/docker/home)

C. HELM Essentials

If you are new to HELM or want to get started, here is the article.

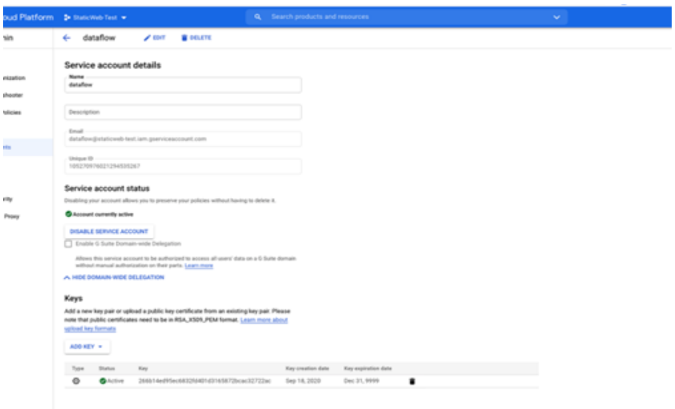
***How To Get Started with HELM* (https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-helm-b3babb30611f)**

D. GCP Prerequisites

- Create a new project.
- You need to create a Billing Account
- Link Billing Account With this project.
- Enable All the APIs that we need to run the dataflow on GCP.
- Download the Google SDK

E. Service Account

Need to create a service account so that when you run the application from your local machine it can invoke the GCP dataflow pipeline with owner permissions.



Service account details

Name: dataflow

Description

Email: dataflow@staticweb-test.iam.gserviceaccount.com

Unique ID: 105270976021294535267

Service account status

Disabling your account allows you to preserve your policies without having to delete it.

Account currently active

DISABLE SERVICE ACCOUNT

□ Enable G Suite Domain-wide Delegation

Allows this service account to be authorized to access all users' data on a G Suite domain without manual authorization on their parts. Learn more

HIDE DOMAIN-WIDE DELEGATION

Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA_X509_PEM format. Learn more about upload key formats

ADD KEY

Type	Status	Key	Key creation date	Key expiration date
	Active	[illegible]	Sep 18, 2020	Dec 31, 9999

Figure 3: Service Account

You must generate the key and download and set the environment variable called **GOOGLE_APPLICATION_CREDENTIALS.**

export GOOGLE_APPLICATION_CREDENTIALS="/Users/bhargavbachina/gcp-credentials/gcp-dataflow-service-account.json"

Finally, you can run the following command to log in to your GCP account.

gcloud auth login

INSTALL GCLOUD CLI AND CONFIGURE

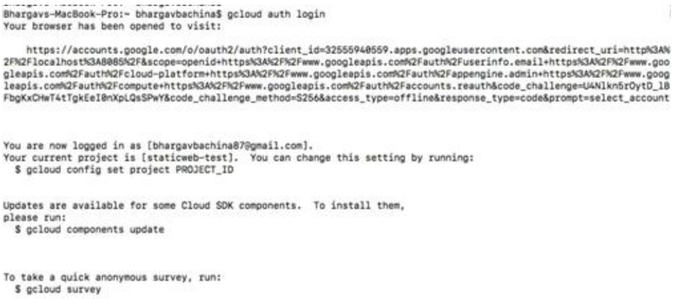
Once you have the GCP Account you can install the gcloud command-line tool. You can go to the below documentation and install gcloud CLI based on your operation system. You can configure gcloud CLI with your project.

The gcloud CLI is a part of the **Google Cloud SDK** (https://cloud.google.com/sdk/docs). You must **download and install the SDK** (https://cloud.google.com/sdk/docs/install) on your system and **initialize it** (https://cloud.google.com/sdk/docs/initializing) before you can use the gcloud command-line tool.

// initializinggcloud init

// auth logingcloud auth login

Once you run the above command, it opens up a browser for you to login into your GCP and you get the response as below.

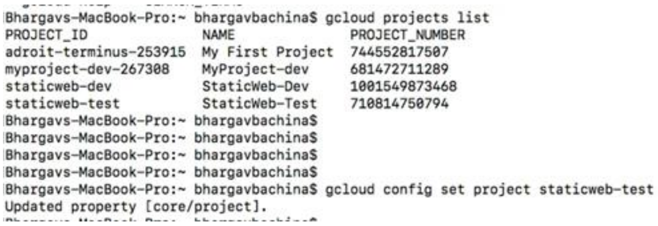


```
Bhargavs-MacBook-Pro:~ bhargavbachina$ gcloud auth login
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%
2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.goo
gleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.goog
leapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&code_challenge=U4Nlkn5rOytD_18
FbgKxCHwT4tTgkEeI0nXpLQsSPwY&code_challenge_method=S256&access_type=offline&response_type=code&prompt=select_account


You are now logged in as [bhargavbachina87@gmail.com].
Your current project is [staticweb-test].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID


Updates are available for some Cloud SDK components.  To install them,
please run:
  $ gcloud components update



To take a quick anonymous survey, run:
  $ gcloud survey
```

Figure 4: gcloud auth login

You can list the projects with the following command.gcloud projects list

You can set the current project with the following command.gcloud config set project staticweb-test



```
Bhargavs-MacBook-Pro:~ bhargavbachina$ gcloud projects list
PROJECT_ID              NAME              PROJECT_NUMBER
adroit-terminus-253915  My First Project  744552817507
myproject-dev-267308    MyProject-dev     681472711289
staticweb-dev           StaticWeb-Dev     1001549873468
staticweb-test          StaticWeb-Test    710814750794
Bhargavs-MacBook-Pro:~ bhargavbachina$
Bhargavs-MacBook-Pro:~ bhargavbachina$
Bhargavs-MacBook-Pro:~ bhargavbachina$
Bhargavs-MacBook-Pro:~ bhargavbachina$
Bhargavs-MacBook-Pro:~ bhargavbachina$ gcloud config set project staticweb-test
Updated property [core/project].
```

Figure 5: Configuring the project

DOCKERIZE THE PROJECT

We have seen how to build the project and tun the application in a normal way. Let's see how we can create a Dockerfile and run the same application in Docker.

We need to create a Dockerfile that creates a Docker image. Here is the file which starts with **FROM** command and with the base image **python:3.7.** You need to set up a working directory and then copy the contents from the local directory to the docker directory.

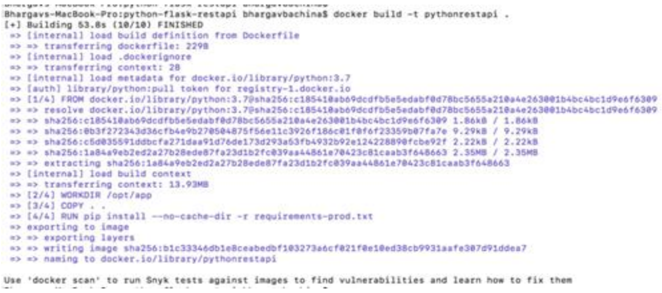
https://gist.github.com/bbachi/a3b16764484f6a0481b78d54f925ee07#file-dockerfile

Then, you need to run the install command so that it installs all the dependencies. Finally, you need to give the command when the container is instantiated.

// change directorycd python-flask-restapi

// build the imagedocker build -t pythonrestapi

// list the imagedocker images



```
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ docker build -t pythonrestapi .
[+] Building 53.8s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 229B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.7
 => [auth] library/python:pull token for registry-1.docker.io
 => [1/4] FROM docker.io/library/python:3.7@sha256:c185410ab69dcdfb5e5edabf0d78bc5655a210a4e263001b4bc4bc1d9e6f6309
 => => resolve docker.io/library/python:3.7@sha256:c185410ab69dcdfb5e5edabf0d78bc5655a210a4e263001b4bc4bc1d9e6f6309
 => => sha256:c185410ab69dcdfb5e5edabf0d78bc5655a210a4e263001b4bc4bc1d9e6f6309 1.86kB / 1.86kB
 => => sha256:0b3f272343d36cfb4e9b270504875f56e11c3926f186c01f0f6f23359b07fa7e 9.29kB / 9.29kB
 => => sha256:c5d035591ddbcfa271daa91d76de173d293a53fb4932b92e124228890fcbe92f 2.22kB / 2.22kB
 => => sha256:1a84a9eb2ed2a27b28ede87fa23d1b2fc039aa44861e70423c81caab3f648663 2.35MB / 2.35MB
 => => extracting sha256:1a84a9eb2ed2a27b28ede87fa23d1b2fc039aa44861e70423c81caab3f648663
 => [internal] load build context
 => => transferring context: 13.93MB
 => [2/4] WORKDIR /opt/app
 => [3/4] COPY . .
 => [4/4] RUN pip install --no-cache-dir -r requirements-prod.txt
 => exporting to image
 => => exporting layers
 => => writing image sha256:b1c33346db1e8ceabedbf103273a6cf021f0e10ed38cb9931aafe307d91ddea7
 => => naming to docker.io/library/pythonrestapi

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Figure 6: docker build

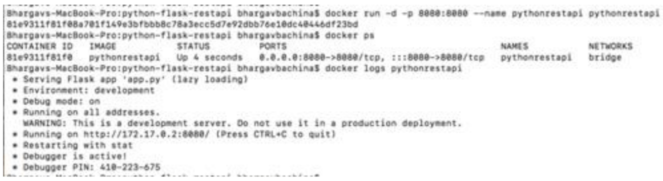
Now, we have the docker image, and let’s run the container and once it is up and running you can access the application at http://127.0.0.1:8080/

// run the containerdocker run -d -p 8080:8080 --name pythonrestapi pythonrestapi

// list the containerdocker ps

// logsdocker logs pythonrestapi

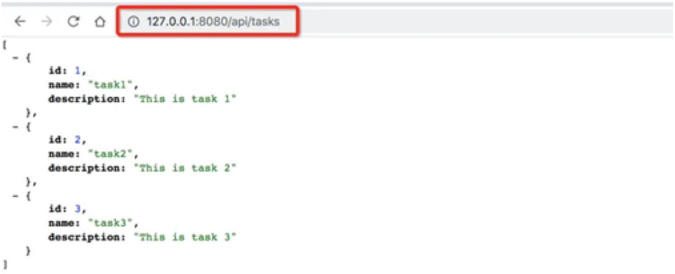
// exec into running containerdocker exec -it pythonrestapi /bin/sh



```
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ docker run -d -p 8080:8080 --name pythonrestapi pythonrestapi
81e9311f81f08a701f149e3bfbbb8c78a3ecc5d7e92dbb76e10dc40446df23bd
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ docker ps
CONTAINER ID   IMAGE           STATUS         PORTS                                       NAMES           NETWORKS
81e9311f81f0   pythonrestapi   Up 4 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   pythonrestapi   bridge
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ docker logs pythonrestapi
 * Serving Flask app 'app.py' (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 410-223-675
```

Figure 7: Running Container on the Docker

You can see that the API running on port 8080.

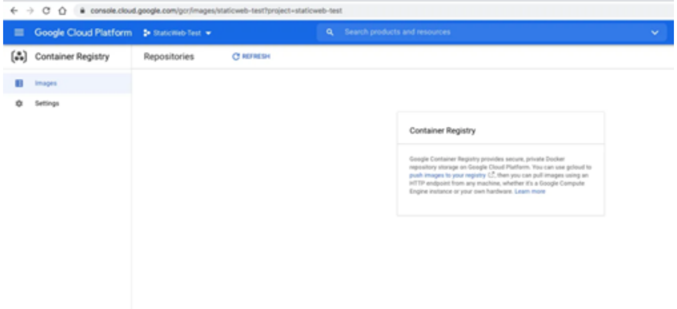


```
[
 - {
       id: 1,
       name: "task1",
       description: "This is task 1"
   },
 - {
       id: 2,
       name: "task2",
       description: "This is task 2"
   },
 - {
       id: 3,
       name: "task3",
       description: "This is task 3"
   }
]
```

Figure 8: API Running on port 8080

PUSHING DOCKER IMAGE TO CONTAINER REGISTRY

GCP GKE works with any Docker registry such as Docker Hub, etc. But, in this post, we see how we can use a GCP container registry to store our Docker images. Make sure you enable the relevant API on GCP. If you log in and go to the Container Registry you can see the empty registry.



console.cloud.google.com/gcr/images/staticweb-test?project=staticweb-test
Google Cloud Platform
StaticWeb Test
Search products and resources
Container Registry
Repositories
REFRESH
Images
Settings

Container Registry

Google Container Registry provides secure, private Docker repository storage on Google Cloud Platform. You can use gcloud to push images to your registry, then you can pull images using an HTTP endpoint from any machine, whether it's a Google Compute Engine instance or your own hardware. Learn more

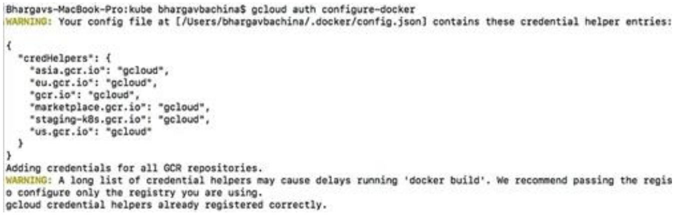
Figure 9: Container Registry Console

The first thing we need to do is to enable the API with the following command.

gcloud services enable containerregistry.googleapis.com

You need to configure docker with the following command. **You can see the full documentation here** (https://cloud.google.com/container-registry/docs/advanced-authentication)

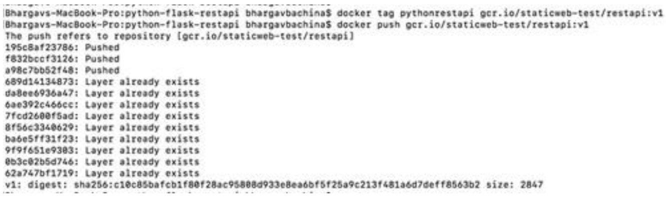
gcloud auth configure-docker



```
Bhargavs-MacBook-Pro:kube bhargavbachina$ gcloud auth configure-docker
WARNING: Your config file at [/Users/bhargavbachina/.docker/config.json] contains these credential helper entries:

{
  "credHelpers": {
    "asia.gcr.io": "gcloud",
    "eu.gcr.io": "gcloud",
    "gcr.io": "gcloud",
    "marketplace.gcr.io": "gcloud",
    "staging-k8s.gcr.io": "gcloud",
    "us.gcr.io": "gcloud"
  }
}
Adding credentials for all GCR repositories.
WARNING: A long list of credential helpers may cause delays running 'docker build'. We recommend passing the regis
o configure only the registry you are using.
gcloud credential helpers already registered correctly.
```

Figure 10: Configuring Docker

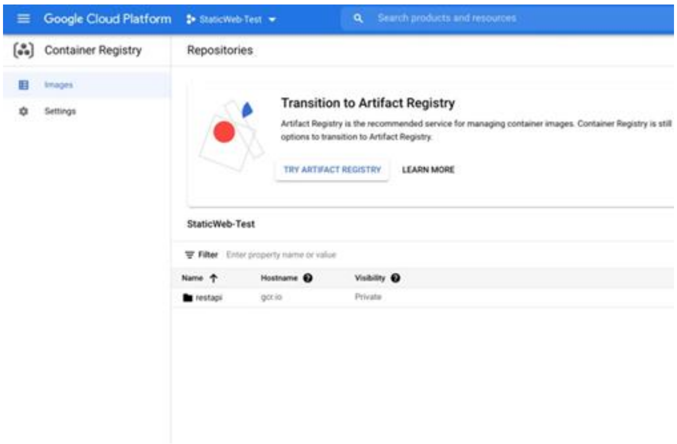
As we have seen in the Example Project section, let’s build the Docker image with the following command.
docker build -t pythonrestapi.
Let’s tag the local docker image with the registry name by using the command:
docker tag [SOURCE_IMAGE] [HOSTNAME]/[PROJECT-ID]/[IMAGE]:[TAG]
// run this commanddocker tag pythonrestapi gcr.io/staticweb-test/restapi:v1
Finally, push the image into the GCP container registry
docker push gcr.io/staticweb-test/restapi:v1



```
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ docker tag pythonrestapi gcr.io/staticweb-test/restapi:v1
Bhargavs-MacBook-Pro:python-flask-restapi bhargavbachina$ docker push gcr.io/staticweb-test/restapi:v1
The push refers to repository [gcr.io/staticweb-test/restapi]
195c8af23786: Pushed
f832bccf3126: Pushed
a98c7bb52f48: Pushed
689d14134873: Layer already exists
da8ee6936a47: Layer already exists
6ae392c466cc: Layer already exists
7fcd2600f5ad: Layer already exists
8f56c3340629: Layer already exists
ba6e5ff31f23: Layer already exists
9f9f651e9303: Layer already exists
0b3c02b5d746: Layer already exists
62a747bf1719: Layer already exists
v1: digest: sha256:c10c85bafcb1f80f28ac95808d933e8ea6bf5f25a9c213f481a6d7deff8563b2 size: 2847
```

Figure 11: Docker push

You can see the repository in the console as well.



Transition to Artifact Registry

Artifact Registry is the recommended service for managing container images. Container Registry is still options to transition to Artifact Registry.

StaticWeb-Test

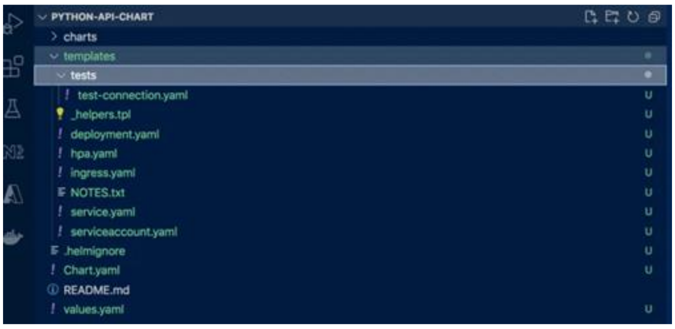
Filter Enter property name or value

Name ↑	Hostname	Visibility
restapi	gcr.io	Private

Figure 12: Container registry

BUILDING A HELM CHART

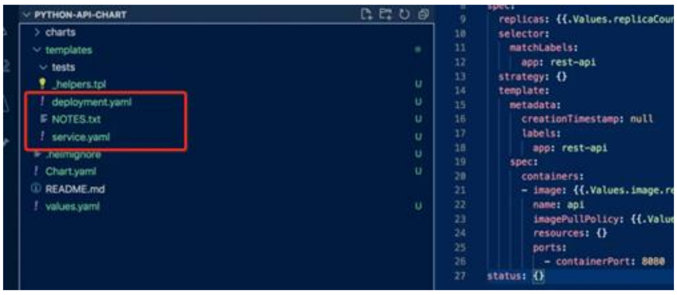
Let’s start with this command so that it creates the default structure for us.
helm create python-api-chart



```
PYTHON-API-CHART
  > charts
  v templates
    v tests
      test-connection.yaml     U
    _helpers.tpl               U
    deployment.yaml            U
    hpa.yaml                   U
    ingress.yaml               U
    NOTES.txt                  U
    service.yaml               U
    serviceaccount.yaml        U
  .helmignore                  U
  Chart.yaml                   U
  README.md
  values.yaml                  U
```

Figure 13: HELM Create Python-API-Chart

If you look at the example project all we need is deployment and service objects. We don’t need hpa, ingress, serviceaccount, etc. I am going to delete all these files and keep only the deployment.yml and service.yml files as below.



```
  spec:
    replicas: {{.Values.replicaCour
    selector:
      matchLabels:
        app: rest-api
    strategy: {}
    template:
      metadata:
        creationTimestamp: null
        labels:
          app: rest-api
      spec:
        containers:
        - image: {{.Values.image.re
          name: api
          imagePullPolicy: {{.Value
          resources: {}
          ports:
            - containerPort: 8080
  status: {}
```

Figure 14: Deployment and Service Objects

You can pass some default values to the Chart with the help of values.yaml file. We have some default values such as image, number of replicas, name, etc.

https://gist.github.com/bbachi/e2e3aff8fc64071adb5c5b9719c863d1#file-values-yaml

You can pass these values in the deployment and service manifest files below.

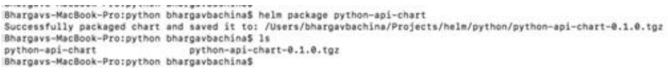
https://gist.github.com/bbachi/27dd1edac6c18faad80caf3e40eb95da#file-deployment-yaml

Chart.yaml contains the information about the Chart such as name, version, description, etc.

https://gist.github.com/bbachi/0b82a7b9416eacea9630147f22bea203#file-chart-yaml

We can package the files and directories of a chart into a single archive file. You can just pack it with the following command.

helm package python-api-chart



```
Bhargavs-MacBook-Pro:python bhargavbachina$ helm package python-api-chart
Successfully packaged chart and saved it to: /Users/bhargavbachina/Projects/helm/python/python-api-chart-0.1.0.tgz
Bhargavs-MacBook-Pro:python bhargavbachina$ ls
python-api-chart              python-api-chart-0.1.0.tgz
Bhargavs-MacBook-Pro:python bhargavbachina$
```

Figure 15: HELM Package

CREATING GKE CLUSTER

We have pushed the Docker image into the container registry and it's time to create a GKE Cluster. You can create the cluster with the following command.

gcloud container clusters create my-cluster

But, we will see how we can create Cluster through the console. Go to the GKE dashboard in the GCP console. Click the button to create a cluster.

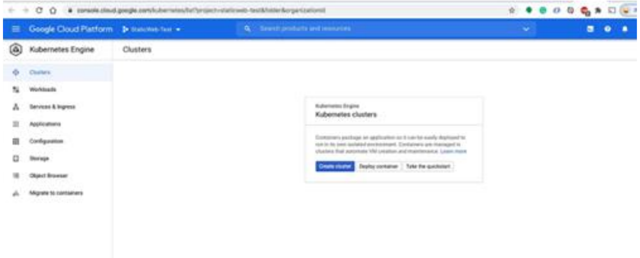
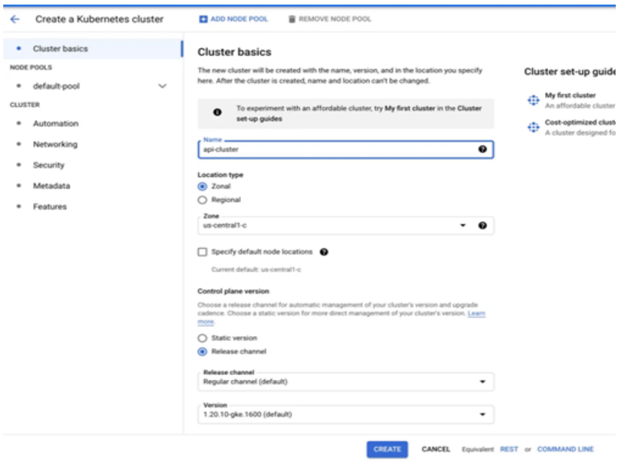


Figure 16: GKE Dashboard

It asks you some basic questions such as location type, version of Kubernetes, etc.



Cluster basics

The new cluster will be created with the name, version, and in the location you specify here. After the cluster is created, name and location can't be changed.

To experiment with an affordable cluster, try **My first cluster** in the **Cluster set-up guides**

Name: api-cluster

Location type
● Zonal
○ Regional

Zone: us-central1-c

□ Specify default node locations

Current default: us-central1-c

Control plane version

Choose a release channel for automatic management of your cluster's version and upgrade cadence. Choose a static version for more direct management of your cluster's version. Learn more.

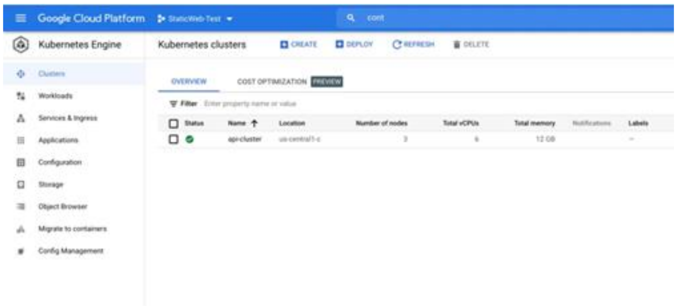
○ Static version
● Release channel

Release channel: Regular channel (default)

Version: 1.20.10-gke.1600 (default)

Figure 17: Creating a GKE Cluster

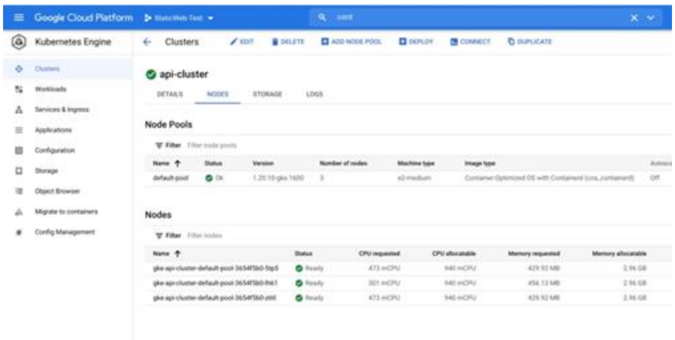
Once you click on the create button on the above screen it creates the cluster with three Nodes.



☐ Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
☐ ✓	api-cluster	us-central1-c	3	6	12 GB		—

Figure 18: Cluster Created

You can click on the cluster and go to the details section where you can see the nodes.



Node Pools

Filter Filter node pools

Name ↑	Status	Version	Number of nodes	Machine type	Image type	Autosca
default-pool	✓ Ok	1.20.10-gke.1600	3	e2-medium	Container-Optimized OS with Containerd (cos_containerd)	Off

Nodes

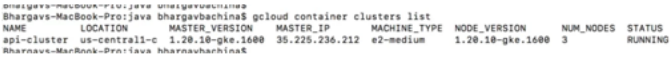
Filter Filter nodes

Name ↑	Status	CPU requested	CPU allocatable	Memory requested	Memory allocatable
gke-api-cluster-default-pool-3654f5b0-5tp5	✓ Ready	473 mCPU	940 mCPU	429.92 MB	2.96 GB
gke-api-cluster-default-pool-3654f5b0-lh61	✓ Ready	301 mCPU	940 mCPU	456.13 MB	2.96 GB
gke-api-cluster-default-pool-3654f5b0-z6ll	✓ Ready	473 mCPU	940 mCPU	429.92 MB	2.96 GB

Figure 19: Three nodes

You can list the clusters with the following command.

gcloud container clusters list

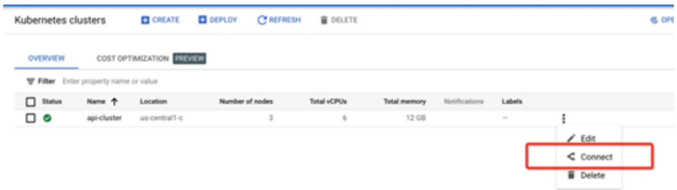


```
Bhargavs-MacBook-Pro:java bhargavbachina$
Bhargavs-MacBook-Pro:java bhargavbachina$ gcloud container clusters list
NAME         LOCATION       MASTER_VERSION    MASTER_IP       MACHINE_TYPE  NODE_VERSION      NUM_NODES  STATUS
api-cluster  us-central1-c  1.20.10-gke.1600  35.225.236.212  e2-medium     1.20.10-gke.1600  3          RUNNING
Bhargavs-MacBook-Pro:java bhargavbachina$
```

Figure 20: Cluster is in running state

CONFIGURE KUEBCTL WITH GKE CLUSTER

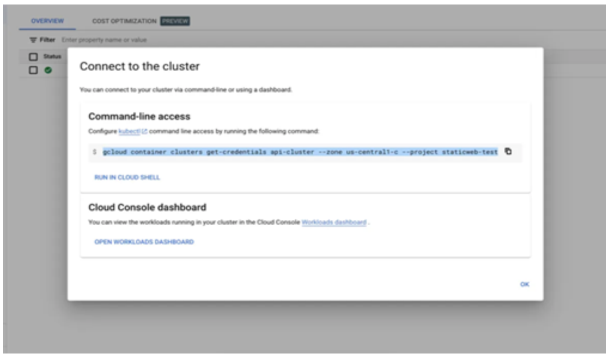
Kubectl is the command-line utility for the Kubernetes. You need to install kubectl before you configure it. Click on the connect button on the console so that it gives you a command to configure kubectl with the GKE Cluster.



☐ Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
☐ ✓	api-cluster	us-central1-c	3	6	12 GB		—

Figure 21: Cluster

Just copy the below command and run it on your terminal.



Connect to the cluster

You can connect to your cluster via command-line or using a dashboard.

Command-line access

Configure kubectl command line access by running the following command:

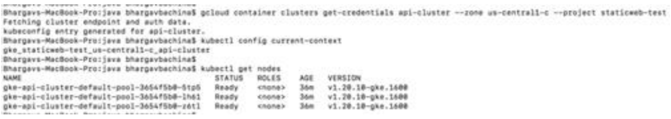
```
$ gcloud container clusters get-credentials api-cluster --zone us-central1-c --project staticweb-test
```

Cloud Console dashboard

You can view the workloads running in your cluster in the Cloud Console Workloads dashboard .

Figure 22: Connecting to Cluster

gcloud container clusters get-credentials api-cluster --zone us-central1-c --project staticweb-test
Once you run this command, kubectl is configured to use GKE Cluster and you can even get the nodes from the cluster.

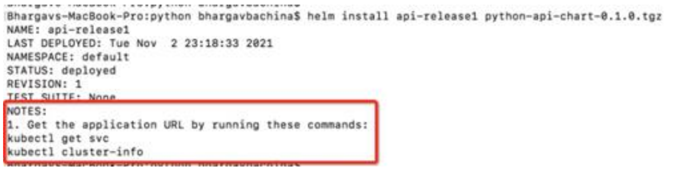


```
Bhargavs-MacBook-Pro:java bhargavbachina$ gcloud container clusters get-credentials api-cluster --zone us-central1-c --project staticweb-test
Fetching cluster endpoint and auth data.
kubeconfig entry generated for api-cluster.
Bhargavs-MacBook-Pro:java bhargavbachina$ kubectl config current-context
gke_staticweb-test_us-central1-c_api-cluster
Bhargavs-MacBook-Pro:java bhargavbachina$
Bhargavs-MacBook-Pro:java bhargavbachina$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
gke-api-cluster-default-pool-3654f5b0-5tp5   Ready    <none>   36m   v1.20.10-gke.1600
gke-api-cluster-default-pool-3654f5b0-lh61   Ready    <none>   36m   v1.20.10-gke.1600
gke-api-cluster-default-pool-3654f5b0-z6tl   Ready    <none>   36m   v1.20.10-gke.1600
```

Figure 23: kubectl is configured with GKE cluster

INSTALLING HELM CHART ON GKE CLUSTER

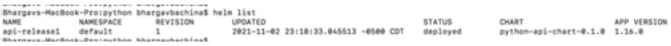
We have packaged the Chart into a single archive file. We can install it with the following command.
helm install api-release1 python-api-chart-0.1.0.tgz



```
Bhargavs-MacBook-Pro:python bhargavbachina$ helm install api-release1 python-api-chart-0.1.0.tgz
NAME: api-release1
LAST DEPLOYED: Tue Nov  2 23:18:33 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the application URL by running these commands:
kubectl get svc
kubectl cluster-info
```

Figure 24: Installed

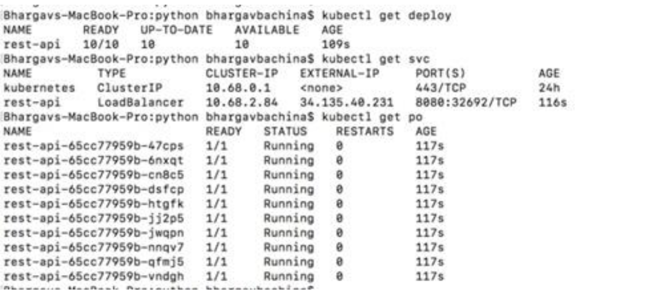
You can see the notes on how to access the application because we have Notes.txt in the Chart folder below. You can put any information that helps the developers who use your chart.
https://gist.github.com/bbachi/f3273a5a13226853b6f675ebef0feec4#file-notes-txt
You can verify the release with this command helm list



```
Bhargavs-MacBook-Pro:python bhargavbachina$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
api-release1    default         1               2021-11-02 23:18:33.045513 -0500 CDT    deployed        python-api-chart-0.1.0  1.16.0
```

Figure 25: HELM List

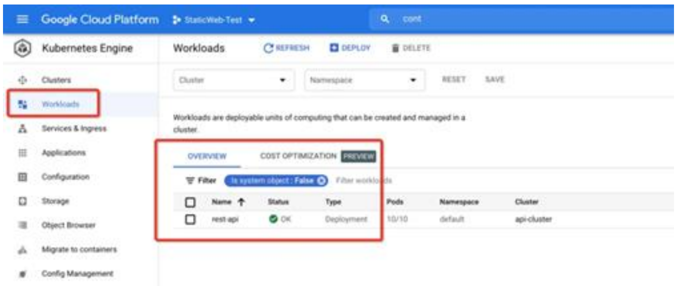
You can verify the pods and service installed and running in your cluster with the following commands.
// deploymentkubectl get deploy
//podskubectl get po
// servicekubectl get svc



```
Bhargavs-MacBook-Pro:python bhargavbachina$ kubectl get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
rest-api   10/10   10           10          109s
Bhargavs-MacBook-Pro:python bhargavbachina$ kubectl get svc
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
kubernetes   ClusterIP      10.68.0.1     <none>          443/TCP          24h
rest-api     LoadBalancer   10.68.2.84    34.135.40.231   8080:32692/TCP   116s
Bhargavs-MacBook-Pro:python bhargavbachina$ kubectl get po
NAME                        READY   STATUS    RESTARTS   AGE
rest-api-65cc77959b-47cps   1/1     Running   0          117s
rest-api-65cc77959b-6nxqt   1/1     Running   0          117s
rest-api-65cc77959b-cn8c5   1/1     Running   0          117s
rest-api-65cc77959b-dsfcp   1/1     Running   0          117s
rest-api-65cc77959b-htgfk   1/1     Running   0          117s
rest-api-65cc77959b-jj2p5   1/1     Running   0          117s
rest-api-65cc77959b-jwqpn   1/1     Running   0          117s
rest-api-65cc77959b-nnqv7   1/1     Running   0          117s
rest-api-65cc77959b-qfmj5   1/1     Running   0          117s
rest-api-65cc77959b-vndgh   1/1     Running   0          117s
```

Figure 26: Successfully Installed

You can verify the same on the GKE cluster by logging into GKE console and clicking on the workloads as below. As you see there are 10 pods running for this deployment.

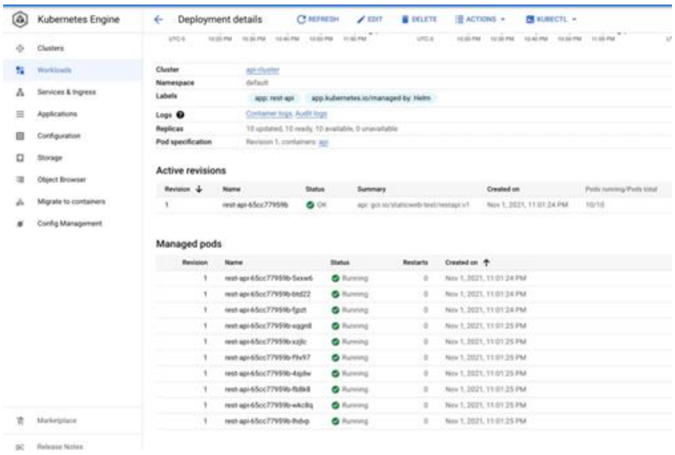


Workloads are deployable units of computing that can be created and managed in a cluster.

□	Name	Status	Type	Pods	Namespace	Cluster
□	rest-api	OK	Deployment	10/10	default	api-cluster

Figure 27: Workloads

You can see more on the details page.



Cluster	api-cluster
Namespace	default
Labels	app: rest-api app.kubernetes.io/managed-by: Helm
Logs	Container logs, Audit logs
Replicas	10 updated, 10 ready, 10 available, 0 unavailable
Pod specification	Revision 1, containers: api

Active revisions

Revision	Name	Status	Summary	Created on	Pods running/Pods total
1	rest-api-65cc77959b	OK	api: gcr.io/staticweb-test/restapi:v1	Nov 1, 2021, 11:01:24 PM	10/10

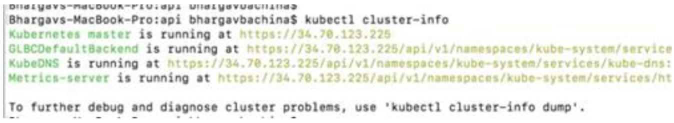
Managed pods

Revision	Name	Status	Restarts	Created on
1	rest-api-65cc77959b-5xxw6	Running	0	Nov 1, 2021, 11:01:24 PM
1	rest-api-65cc77959b-btd22	Running	0	Nov 1, 2021, 11:01:24 PM
1	rest-api-65cc77959b-fgzt	Running	0	Nov 1, 2021, 11:01:24 PM
1	rest-api-65cc77959b-xqgn8	Running	0	Nov 1, 2021, 11:01:25 PM
1	rest-api-65cc77959b-xzjlc	Running	0	Nov 1, 2021, 11:01:25 PM
1	rest-api-65cc77959b-f9x97	Running	0	Nov 1, 2021, 11:01:25 PM
1	rest-api-65cc77959b-4sjdw	Running	0	Nov 1, 2021, 11:01:25 PM
1	rest-api-65cc77959b-fb8k8	Running	0	Nov 1, 2021, 11:01:25 PM
1	rest-api-65cc77959b-wkc8q	Running	0	Nov 1, 2021, 11:01:25 PM
1	rest-api-65cc77959b-fhdvp	Running	0	Nov 1, 2021, 11:01:25 PM

Figure 28: Details Page

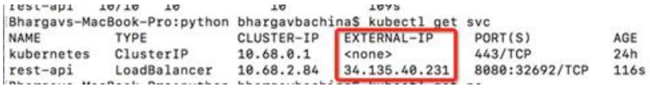
ACCESS THE INSTALLED APPLICATION

We have created deployment and services and now we need to access this deployment from the browser. We need to get the public IP address of the Kubernetes with this command kubectl cluster-info



```
Bhargavs-MacBook-Pro:api bhargavbachina$ kubectl cluster-info
Kubernetes master is running at https://34.70.123.225
GLBCDefaultBackend is running at https://34.70.123.225/api/v1/namespaces/kube-system/service
KubeDNS is running at https://34.70.123.225/api/v1/namespaces/kube-system/services/kube-dns:
Metrics-server is running at https://34.70.123.225/api/v1/namespaces/kube-system/services/ht

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Figure 29: kubectl cluster-info



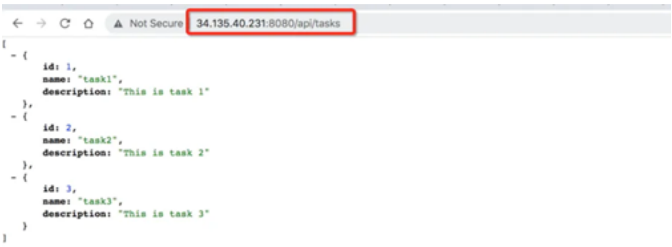
```
Bhargavs-MacBook-Pro:python bhargavbachina$ kubectl get svc
NAME         TYPE           CLUSTER-IP   EXTERNAL-IP     PORT(S)           AGE
kubernetes   ClusterIP      10.68.0.1    <none>          443/TCP           24h
rest-api     LoadBalancer   10.68.2.84   34.135.40.231   8080:32692/TCP    116s
```

Figure 30: External IP

Let’s access an API with this IP address **34.135.40.231** (where loadbalancer is running) and port **8080** (where the service port is mapped) with the below URLs. Make sure that you use HTTP instead of HTTPS.

http://34.135.40.231:8080/api/tasks

Accessing the API in the browser



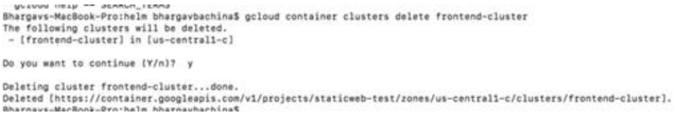
```
[
 - {
       id: 1,
       name: "task1",
       description: "This is task 1"
   },
 - {
       id: 2,
       name: "task2",
       description: "This is task 2"
   },
 - {
       id: 3,
       name: "task3",
       description: "This is task 3"
   }
]
```

Figure 31: Accessing the application

DELETING THE CLUSTER

You should delete the cluster if you don’t want to incur charges. You can either delete it either on the console or with the below command.

gcloud container clusters delete frontend-cluster



```
Bhargavs-MacBook-Pro:helm bhargavbachina$ gcloud container clusters delete frontend-cluster
The following clusters will be deleted.
 - [frontend-cluster] in [us-central1-c]

Do you want to continue (Y/n)?  y

Deleting cluster frontend-cluster...done.
Deleted [https://container.googleapis.com/v1/projects/staticweb-test/zones/us-central1-c/clusters/frontend-cluster].
```

Figure 32: Deleting Cluster

SUMMARY

- Kubernetes is one of the rapidly growing technology and all the companies are adopting it nowadays.
- When you run any application on Kubernetes you need to deploy so many objects such as deployment, configmap, secrets, etc. You need to define all these objects in the manifest.yml file and send these files to the Kubernetes API server.
- Deploying your application one time is ok but if you want to deploy the application again and again you need to send all the manifest files to the Kubernetes API server again and again. The helm is the tool that solves this problem.
- Helm is the package manager for Kubernetes and provides solutions for package management, security, configurability while deploying applications to Kubernetes.
- Helm's mainly focuses on three things when you are managing applications in the cluster: Security, Configurability, and Reusability.
- A Chart in a Helm is nothing but a packaged version of your application. A chart is a set of files and directories that follows some specification for describing the resources to be installed into Kubernetes.
- Helm uses the same configuration by default that kubectl uses and you can change it by exposing the environment variable either $KUBECONFIG or HELM_KUBECONTEXT.
- We can package the files and directories of a chart into a single archive file with this command helm package <package-name>

CONCLUSION

In conclusion, Kubernetes stands as a rapidly evolving technology that enjoys widespread adoption across industries. However, deploying applications on Kubernetes necessitates managing a multitude of objects, from deployments to configmaps and secrets, defined within manifest files. While initial deployments are manageable, repetitive deployments pose challenges in sending manifest files repeatedly to the Kubernetes API server. Enter Helm, the Kubernetes package manager, designed to address this issue. Helm streamlines application deployment by offering solutions for package management, security, and configurability within Kubernetes environments. With a primary focus on security, configurability, and reusability, Helm simplifies the management of applications within the cluster. Utilizing charts, Helm packages application resources into a structured format, facilitating efficient deployment. Moreover, Helm aligns with kubectl's configuration by default, offering flexibility through environment variables. By packaging chart files and directories into a single archive file, Helm provides a seamless deployment experience for Kubernetes applications.

REFERENCES

[1]. Docker Documentation https://docs.docker.com/trusted-content/official-images/

[2]. Kubernetes Documentation https://kubernetes.io/docs/home/

[3]. Helm Documentation https://helm.sh/docs/

[4]. Google Cloud Documentation https://cloud.google.com/docs

[5]. Python Documentation https://docs.python.org/3/

[6]. GCP GKE Documentation https://cloud.google.com/kubernetes-engine/docs