The purpose of this challenge is to create an application that exposes a small api on the Google Cloud Platform.

Specification:  
- Write a short api in python GET using the following API path..../nxchallenge/astarisborn (done, see link below)  
- When the api is called, a json response is returned as such { "response" : "Hello World" }(done)  
- Host this api on 2 node containers within a Kubernetes cluster (done)  
- Provide build scripts for your solution (done)  
- Bonus: Add identity management (giving permission to some accounts and removing others) (done)

Identity Management

1. Open the IAM page in the GCP Console.
2. Click Add
3. Enter the email address of a new member to whom you have not granted any Cloud IAM role previously.
4. You can remove access my going to IAM and removing the access.

<http://35.233.242.208/nxchallenge/astarisborn>

As shown in the illustration on the next page, a Kubernetes Cluster with two nodes running the application is currently setup on GCP.

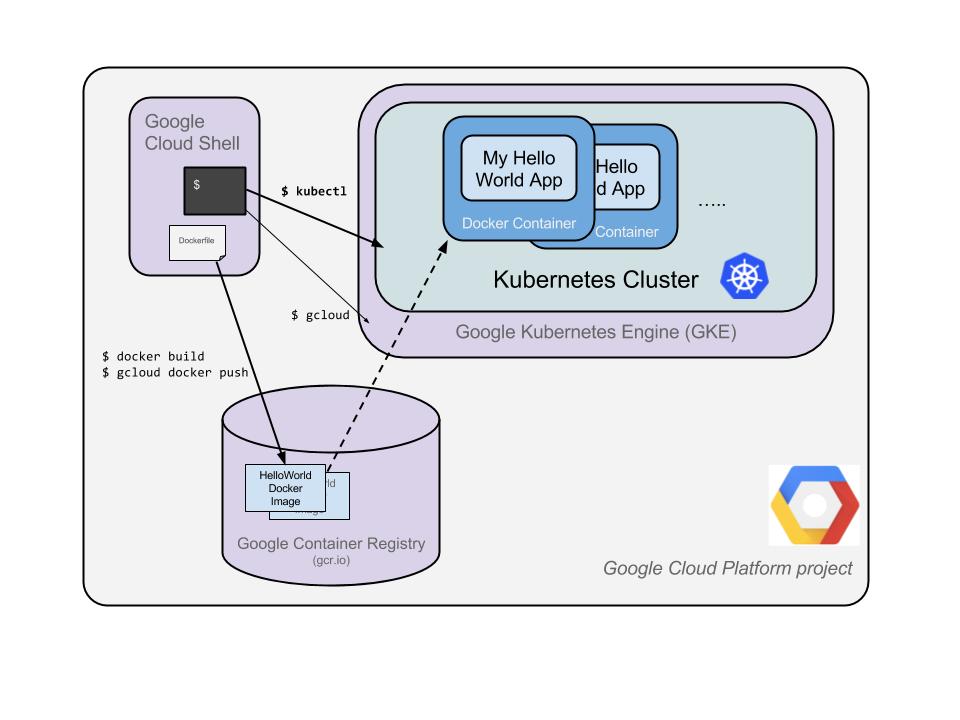
The choice of the location of the nodes and such information can be specified by running specific ‘gcloud’ commands that specify the zones and location of the machines. Gcloud provides a host of commands under its documentation that I used in this project as seen on the following pages.

This whole exercise was designed around the configuration of GCP. Containerization was used to preserve the portability of the code so that it could be easily updated, documented, maintained, and deployed.

An additional benefit to docker is that you can create a single “master” version of your image to quickly deploy if need be; and hence the reason I created two Docker Containers within a Kubernetes Cluster.

If you are doing co-development with a team, you can provide them with the resources they need without compromising any information on your own network; thus, enabling developers a productive and continuous feedback loop.

Python was used as the language and Flask as the templating framework. Though flask is a lightweight framework, the object of this exercise as a devops engineer is to show importance on the infrastructure than the code itself. In the real world, I would use a more robust framework like Django and perhaps an ElasticSearch database to handle thousands, or perhaps hundreds of thousands of get requests concurrently so that the quality of enterprise grade performance isn’t compromised.



**Readme / Steps:**

**1.Create a container package (by creating 3 files):**

Container package will help us create docker containers. This file has code and the docker file

Dockerfile:

Requirements.txt:

Python File (hello-world.py)

**2. Setup and Installing Libraries:**

**Installing these libraries will help us to locally build the container and deploy on cloud**

Install google cloud sdk

gcloud components install kubectl

Install docker (separate way for all the machines)

**3. Set default:**

gcloud config set project PROJECT\_ID

gcloud config set project authentic-codex-212623

gcloud config set compute/zone us-central1-b

export PROJECT\_ID="$(gcloud config get-value project -q)"

**4.Build container image:**

This step will help us build our container locally.

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

1. **Authenticate CR**

We authenticate Container registry to upload our locally built container

sudo gcloud auth configure-docker

1. **Push image to CR**

Uploading the local docker image to container registry

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

**7.Create a container cluster**

Creating a kubernetes cluster

sudo gcloud container clusters create hello-cluster --num-nodes=2 –-zone=us-west1-b

**8.Deploying the application:**

Deploying the application on the GKE using the kubectl command

sudo kubectl run hello-web --image=gcr.io/${PROJECT\_ID}/hello-app:v1 --port 8080

**9.Expose to internet**

sudo kubectl expose deployment hello-web --type=LoadBalancer --port 80 --target-port 8080

**10. Get the external IP and test the application by calling:**

kubectl get service