Project IDS
Book Store using Google App Engine

LARREINEGABE Gerardo
RATNAPARKHI Gaurang
PERPETUA Patricio

10/04/2017

1 Technologies Used

- Google App Engine (https://cloud.google.com/appengine/)
- Google Cloud Postgresql (https://cloud.google.com/sql/docs/postgres/)
- Google Cloud SDK (https://cloud.google.com/sdk/)
- Docker (https://www.docker.com/get-docker)
- Kubernetes (https://kubernetes.io)
- Maven
- Git
- Java Spring Boot

2 Framework Used

- Yeoman (http://yeoman.io/)
- Node.js (https://nodejs.org/en/about/)

2.1 Yeoman

It is a generator ecosystem which helps us proscribing best practices and tools to stay productive. The workflow that it provides is a client-side stack, comprising tools and frameworks that can help quickly build web applications.

2.2 Node

Node.js is an event driven and therefore asynchronous I/O runtime library and environment that runs on the JavaScript interpreter created by Google V8. The truth is that it is very fashionable but not something new since there are libraries like Twisted that do exactly the same but if it is true that is the first based on JavaScript and has a great performance

3 Logic

The basic idea was to create a web application using JHipster and then deploy it on the Google Container engine using Kubernetes. The Google Cloud SQL was used as Postgresql database.

3.1 Why JHipster?

JHipster uses the following Client Side (Front end) technologies Yeoman, Webpack, Angular JS, Bootstrap. Also it uses the following Server Side (Back end) technologies Maven, Spring, Spring MVC REST, Spring data jpa. It uses a 'monolithic architecture' which contains both front-end and back-end code. It basically creates a decent architecture for the type of entities we want.

3.2 Step-by-Step Implementation

3.2.1 Application

- Create the environment on the local machine.
- Create the structure of our application
- Write logic for database entry and changes for UI.

3.2.2 Docker and Kubernetes

Following were the steps to deploy the web application [doc][kubb].

- Create Google Cloud Project and set the project as default

  ```
  gcloud config set project bookstoreids
  ```

- Create a Google SQL Instance

  ```
  gcloud beta sql instances create bookstoresql
  --region=europe-west1 --tier=db-f1-micro
  --authorized-networks=`curl -s ifconfig.co`
  --backup-start-time=01:00 --enable-bin-log
  --activation-policy=ALWAYS --storage-type=HDD
  --storage-size=10GB
  ```

- Create a Container Cluster

  ```
  gcloud container clusters create bookstore-1
  --zone=europe-west1-b --machine-type=g1-small --num-nodes=3
  ```

- Get the credentials for kubectl

  ```
  gcloud container clusters get-credentials bookstore-1
  ```

- Build and push a docker image

  ```
  docker image tag bookstore alarreine/bookstore:v1
  ```

  It is better to add a version

  ```
  gcloud docker -- push bookstore alarreine/bookstore:v1
  ```

- Create Credentials to use Cloud SQL Proxy

  ```
  gcloud iam service-accounts create bookstore-app
  --display-name="Bookstore App"
  ```

- Get access to the service account

  ```
  gcloud projects add-iam-policy-binding bookstoreids
   --member serviceAccount:service@created.com
   --role roles/editor
  ```

- Create the key to use the proxy

```
gcloud iam service-accounts keys create
--iam-account service@created.com aplication-credentials.json
```

- Add this key to the cluster Kubernetes

```
kubectl create secret generic cloudsql-oauth-credentials
--from-file=credentials.json=aplication-credentials.json
```

- Add this credential to the pod file [con]
- Deploy the cluster

```
kubectl apply -f bookstore
```

- Get the external IP

```
kubectl get services
```

4 Architecture

4.1 Google Container

The architecture of Google Container with Kubernetes Figure 1.

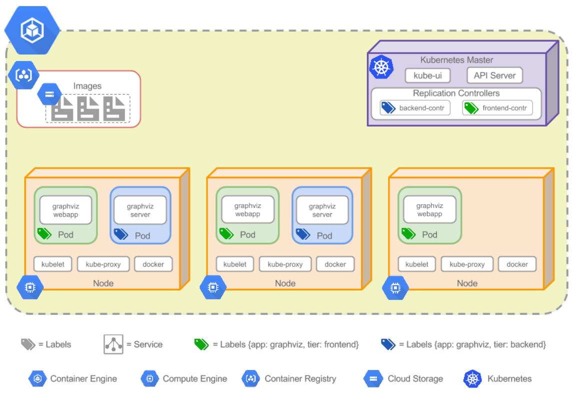

Figure 1: Compute Engine.[Ome]

4.2 Bookstore

The architecture of Bookstore is show the Figure 2 shows.

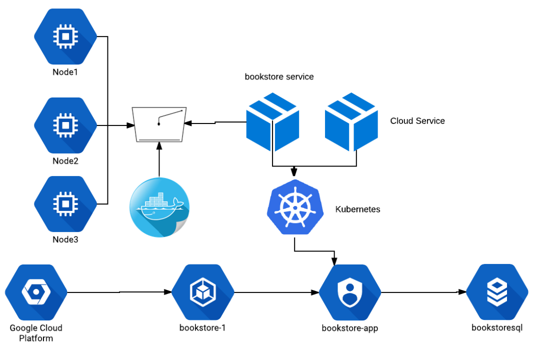

Figure 2: Architecture Bookstore.

We can see all the layers like:

- Bookstore-1: which is a computer container with 3 nodes.
- Bookstore-app: which is a Identity Access to authorize who can use our resource.
- Bookstoresql: which is the instance of Postgresql database.

Google Container uses Kubernetes to manage the container [gco] and the command *kubectl* Figure 3. is a command line interface for running commands against Kubernetes clusters.

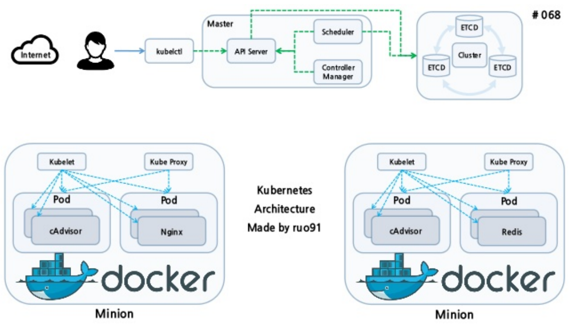

Figure 3: Kubernetes command.[kuba]

We have created a service for our application which has proxy which use *Bookstore-app* to get access to the *Booktoresql*.

4.3 Our implementation vs App Engine

They are basically the same. One provides a custom configuration and other does not.

App Engine is a Platform-as-a-Service (PaaS). It means that you simply deploy your code, and the platform does everything else for you; create instance, increase volumes if necessary, etc [gap].

Compute Engine is an Infrastructure-as-a-Service. You have to create and configure your own virtual machine instances. It gives you more flexibility and generally costs much less than App Engine. The drawback is that you have to manage your app and virtual machines yourself.

Container Engine is another level above Compute Engine, i.e. it's cluster of several Compute Engine instances which can be centrally managed. [gco]

References

[con] Connecting from google container engine.

[doc] Docker documentations.

[gap] Google app engine.

[gco] Google container.

[kuba] Kubectl.

[kubb] Kubernetes documentations.

[Ome] Omer Dawelbeit. Getting started with kubernetes on google container engine.