1. Deploy the node.js in docker using Kubernetes. Scale the application

* Create a Node.js server.
* Create a Docker container image using the Node.js
* Use GCP Container Registry
* Create a Kubernetes container cluster on GCP
* Create a Kubernetes pod.
* Create a Service and Allow External Traffic to the service
* Do Auto Scaling using your Pods to manage more traffic
* Scale up your services
* Modify the Node.js code
* Use and Update on GCP Container Registry
* Redeploy the Node.js code

We have already covered first 3 parts of the assignment in “ASSIGNMENT – 10” please refer to the assignment number 10 for creation of the docker file and pushing into GCR (google loud registry). Let’s look 2 different version that already pushed to the CGR. Tag with 0.1 is first container and 0.2 with update in application.Graphical user interface, text, application, email

Description automatically generated

We have 2 ways to create kubernetes container cluter. 1st by GCP console and 2nd is by gcloud shell. I prefer to doing with gcloud shell you also can perform same task with GCP console by Navigating through Menu -> Kubernates Engine -> Clusters and create the clusters. All you need to configure the settings according to use. So let’s get started with creating cluster with cloud shell. So, first we need certain command to do that as we are doing for compute engine we are using gcloud compute but here we use “gcloud container clusters create NAME –zone/--region” i.e. “gcloud container clusters create cluster-1 --zone=us-central1-c” it will take a while to creating container cluster and it will create a cluser with name “cluster-1” with defult configuration. Text

Description automatically generatedGraphical user interface, text, application, chat or text message

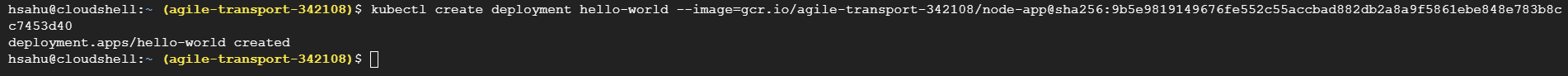
Description automatically generated

Now we need to connect to the cluster that we have created. Use comand “gcloud container clusters get-credentials NAME\_OF\_CLUSTER –zone/--region –project ID” i.e. “gcloud container clusters get-credentials cluster-1 –zone us-central1-c –project agile-transport-342108” Text

Description automatically generated

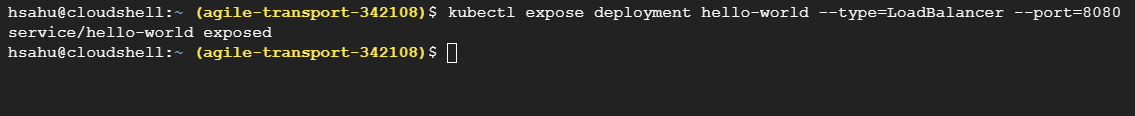
Now we need to create kubernetes pod and for creating same we use “kubectl” command inside the cloud shell. Let’s see some options available with kubectl commands Text

Description automatically generated

Let’s create a deployment using “kubectl create deployment NAME --image PATH\_OF\_IMAGE\_REGISTRY” i.e. “kubectl create deployment hello-world --image=gcr.io/agile-transport-342108/node-app:01” we can get full path of image from the container registry d Graphical user interface, text, application, email

Description automatically generated Text

Description automatically generated

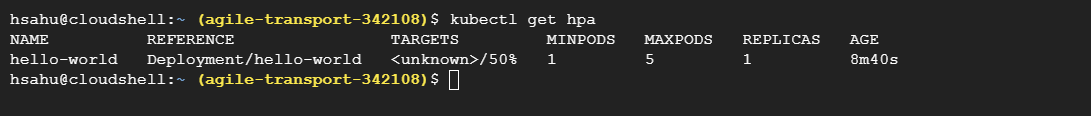
Now we have Cluster, Pods up and run and in next step we need to expose the service by spacifing --port command in kubectl we also can spacify --type of expose like LoadBalancer tool. 

Now we need to check services so Text

Description automatically generated

We have external ip address with us now we need to check wether our deployed service is working or not? Lets curl the request Text

Description automatically generated

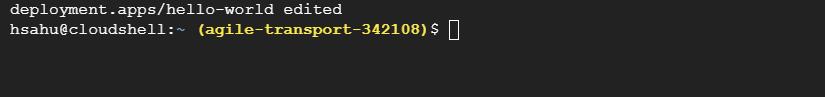
Now we need to autoscale the pods so that it can handle more traffic usign “kubectl autoscale deployment --max 5 --min 1 --cpu-percent 50”   Text

Description automatically generated

Now time to update the container in kubernetes using the command “kubectl edit deployment hello-world” it will open vi editor and chane the image file url with updated image url and save it by pressing “esc” then “:wq” and hit “enter”.



Text

Description automatically generated 

To update the image use command “kubectl get deployment” and “kubectl get rs” to get relese history Text

Description automatically generated

Now verify the changes made by curling the same link of services; Text

Description automatically generated Text

Description automatically generated