**SWE 645**

**Component-based Software Development**

**Assignment 3**

By:

Abhishek Samuel Daniel

Keerthan Srinivas

Navaneeth Krishna

Thanuj Pathapati

This readme is for explaining how we implemented the extra credit assignment. It is divided into two parts. The first explains the creation of an ingress into the HW3 service application using a DNS.

The second explains how we created a simple NGINX web server application and deployed it onto a cluster and created an ingress for it as well.

**Section 1**

**Part 1: Creating a static IP address.**

* To create a static IP address on GCP we need to head over to the VPC Network homepage.
* Under the VPC Network category choose the IP address subcategory. At the top of this page is an option to reserve a static IP address. Click on it.
* An IP creation page opens up. Give a name to your IP address. In the network type select Premium and in the IP Version select IPv4.
* In the region type select the same region you used to create the cluster and the Jenkins VM.
* Leave the attached section as none (default).
* Click on reserve this will create a static IP address.

A screenshot of a computer

Description automatically generated

* The next step is to assign this IP address to the Load Balancer we created for the myapp-svc in the main project.
* To do this we need to open the service yaml file for the web application and under the load balancer specify this static IP address as the external IP to be assigned to the Load Balancer.

A screenshot of a computer program

Description automatically generated

* The reason for assigning a static IP address is that whenever the service is deleted and recreated it assigns a new external IP to the load balancer. And since the ingress object is using the previous IP, it leads to the webpage not loading.
* So setting a static external IP will always ensure that the new load balancer always points to the same IP every single time it is created.

**Part 2: Creating a Domain.**

* To register our very own domain we need to head over to the Cloud Domains section under Network Services.
* On the top of the page we can see a “Register Domain” .
* A screenshot of a web page

  Description automatically generatedEnter the domain name you would like to create. Something abc.com. This will list all the available domain names similar to the one you wanted to create. Choose one from the options and click continue.
* Next we have to create our very own DNS. Under the DNS Provider menu we have to choose “Use Cloud DNS” and in the drop down choose “Set up new zone”.
* This will open a side panel in which the zone name is already entered, you can leave it as it is or could change if you wish to. Then click on Save and Continue.
* Next we turn on Privacy protection and provide our contact information.
* Finally we click on REGISTER.
* And Voila we have our very own domain.

**Part 3: Creating Record sets for our DNS.**

* Click on the zone you created in the previous step. And then click on “Add Standard”. Here you will already see the domain you created in the previous step.
* Here you will give a DNS name which appended to your domain.
* In the IPV4 address section we give the static IP address of the Load Balancer.
* Then click on create.

In total we should have created

1. A Domain.
2. A DNS Zone (through the domain).
3. A standard/DNS Name.
4. An IPv4 address.

**Part 4: Creating the Ingress.**

* Next head over to the “Services and Ingress” section under the Kubernetes Engine service.
* Select the service you want to create an ingress to then select “Create Ingress” which can be found on the top of the page.
* This will redirect you to a new page where we need to configure our ingress.
* Here, give a name to your ingress.
* Then select the Host and path rules section, copy and paste your new DNS name/standard that you have created in the previous part and in the “Backend service” choose the service for which you wanted to create an ingress. In my case, the myapp-svc service. Leave everything else as default and click on create.
* Your ingress takes a few seconds to create. After creation you can see your URL in the frontends column.
* Click on it and you should be able to see your application.

A screenshot of a chat

Description automatically generated

**Section 2**

**Part1: Creating the Hello World Web Application**

* To create simple Web Application we first need to create an HTML template that displays the message hello world to the web browser.

A screen shot of a computer program

Description automatically generated

* This is a simple boiler plate code. That displays Hello World!!! .

**Part 2: Creating the Dockerfile:**

* A screen shot of a computer

  Description automatically generatedSince this is an NGINX web server web application we need to use the nginx image from docker hub and move out html template to the /usr/share/nginx/html/index.html in the nginx image.

**Part 3: Building the Docker Image**

* To build our image we need to open a terminal in the directory where the Dockerfile is present.
* In the terminal give the following commands.
  + docker build -t <IMAGE\_NAME> .
  + docker tag <IMAGE\_NAME> <REPOSITORY\_NAME>/<IMAGE\_NAME>
  + docker push <REPOSITORY\_NAME>/<IMAGE\_NAME>

**Part 4: Creating the YAML files.**

* Since we need to deploy the web application onto a cluster, we need to create deployment and service yaml files.
* Here is the deployment.yaml file.

A screen shot of a computer program

Description automatically generated

Since the specification for this extra credit was to create deployment in a separate namespace to the main project, I deployed these pods in the namespace nginx-app-namspace. It will be deployed onto 3 nodes and the image it will use is the asdpkp/nginx-webapp with container name and port as nginx-cont and 80 respectively.

* A screen shot of a computer

  Description automatically generatedThis is my Service.yaml:

I followed the same process as I did in Section 1 and assigned a static IP to the Load Balancer created for this service.

**Part 5: Create a GitHub repo.**

* Create a new github repository and put all these files into that repository using the following commands.
  + git init
  + git add –all
  + git commit -am “First Commit”
  + git push <URL\_TO\_GITHUB\_REPO>

A screenshot of a computer

Description automatically generated

**Part 6: Creating a Jenkinsfile.**

* Now that we have created deployment and service files, we need to create Jenkinsfile to create the CI/CD pipeline.
* My Jenkinsfile looks like this.

A screenshot of a computer program

Description automatically generated

It functions the same way as it does for the main project.

**Part 7: Creating the CI/CD pipeline.**

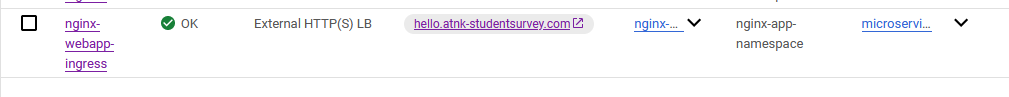
* Similar to the main project, we need to create a new item and select the pipeline option from the below options and give a name to our pipeline.
* We need to select the “Poll SCM” option and give the following as the input: - \* \* \* \* \*
* Under the Pipeline definition page, we must select the “Pipeline script from SCM” option and in the SCM selection window we select Git.
* This will open a Repositories section. In repository URL give the url to your GitHub repo and in credential dropdown choose the same credential you did for the main project i.e Docker Credentials.
* Change the branch from \*/master to \*/main.
* Click on Apply and then on save. The Jenkins file is automatically from the github repository.
* The build process starts automatically and deploys the pods to the cluster.

A screenshot of a computer

Description automatically generated

**Part 8: Creating an Ingress.**

* Follow the same steps as in section one to create a DNS Name/ Standard and assign your load balancer static IP address to the DNS name.
* Then select nginx-webapp-svc and select on create ingress which can be found under the Kubernetes Engine -> Service and Ingress section.
* Follow the same steps as in Section 1 to assign the newly created DNS Name to the front end and select the service as the backend.
* Click create and your ingress wll be created.
* It will take a few seconds but the DNS Name you assigned to the ingress will show up under the Frontends column, click on it and it will redirect you to your Hello World web page.



**Links:**

GitHub Repo: <https://github.com/ASDPKP/MAIN-HW3-SWE645.git>

MyApp DNS: <http://gmu.atnk-studentsurvey.com/>

NGINX-APP DNS: <http://hello.atnk-studentsurvey.com/>

NGINX-APP IP: [34.86.147.13:80](http://34.86.147.13:80)