Colin Dickerson
CSYE 6225
HW – Week 10

Setup

To Begin this exercise we're going to use google cloud shell. To access cloud shell you can use the following link (assumes you already have a cloud platform account setup and configured): https://console.cloud.google.com/?cloudshell=true. The benefit of using cloud shell is that Node.JS, Docker, and the Google Cloud SDK are already configured and available for us to utilize so no setup is needed to immediately start using them!

The only thing we need to do to setup our environment is install Kubernetes. To do this we use the following command:

```
gcloud components install kubectl
```

Create your Node.js Application

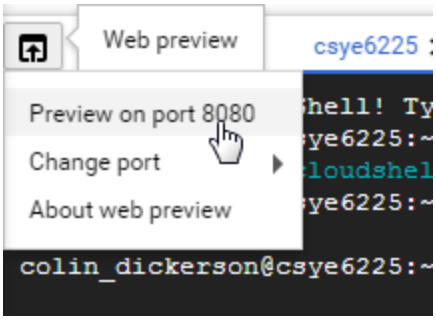
Next we'll create a Node.js application to run on our VM. We need to create a folder called "hellonode" then inside that create a file named "server.js". This can easily be done using the mkdir command and nano.

The server.js file needs to contain the following code:

```
var http = require('http');
var handleRequest = function(request, response) {
  console.log('Received request for URL: ' + request.url);
  response.writeHead(200);
  response.end('Hello World!');
};
var www = http.createServer(handleRequest);
www.listen(8080);
```

We then need to run this script using the command "node server.js". The command will run in a loop until we force it to close.

Once it's running we can view the result using the "Web Preview" command in the cloud shell environment.



This will open a webpage and we see it says "Hello World" telling us that the server.js file is running correctly. We'll also see a log entry in the shell window for the request. Finally use the command "Cntrl+C" to exist the server.js program.

This was a really quick example to show us how we can create a simple Node.JS program. In the next steps we'll examine how to containerize it.

Create a Docker Container Image

Next we'll create a docker setup file. Use the command "nano Dockerfile" and past the following text into the file:

```
FROM node:4.4
EXPOSE 8080
COPY server.js .
CMD node server.js
```

Next we need to build the docker image using the following command. Note my project ID is "csye6225" replace this value with your own project id.

```
docker build -t gcr.io/csye6225/hello-node:v1 .
```

This builds our docker image using the basic Dockerfile configuration we previously created.

Next we'll try the image out using docker with the following command:

```
docker run -d -p 8080:8080 --name hello_tutorial gcr.io/csye6225/hello-
node:v1
```

If we next run the command:

```
curl http://localhost:8080
```

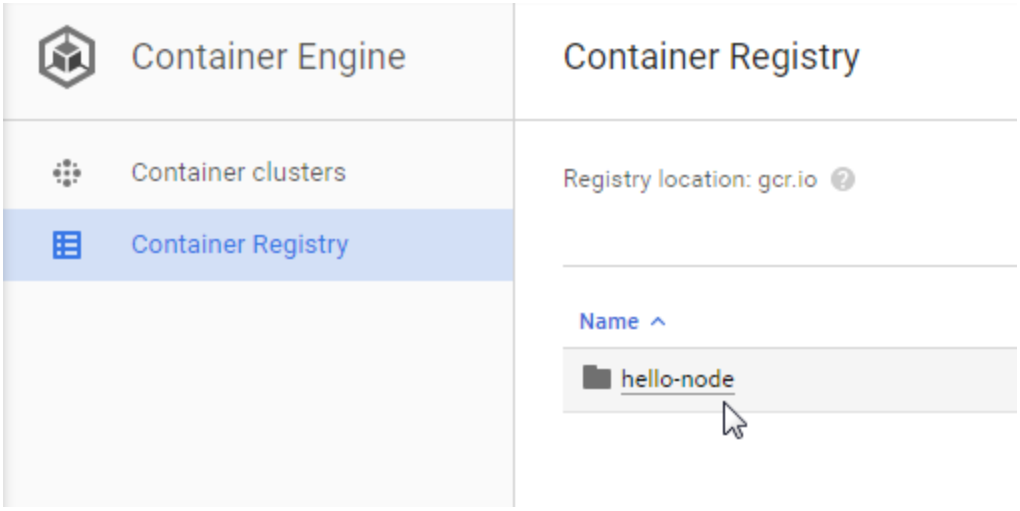
we get the “Hello World” response, indicating the docker container is up and running! We’ll now stop docker using the command.

```
docker stop hello_tutorial
```

Next, now that we’ve verified our docker image works as intended we can push it to the google container engine using the following command:

```
gcloud docker -- push gcr.io/csye6225/hello-node:v1
```

Once the push completes (takes a while) we can see the container listed in our GCP Container Repository



Observations

Following this process highlighted to me how simple creating containers can be, and how easily they can be scripted and pushed out to the Cloud Platform. Containers offer a better solution over VMs because they are more lightweight and can be really pre-configured.

Reading further down the tutorial document it discusses how with a single command you can easily scale the number of containers to adapt to higher load. This allows for

quicker reaction and the ability to rapidly scale to larger loads. Additionally it highlights how the software can be used to seamlessly upgrade containers to a new version.

These containers are a great improvement over traditional VMs as they allow for less overhead, greater flexibility and can help reduce the overall cost of a cloud based system