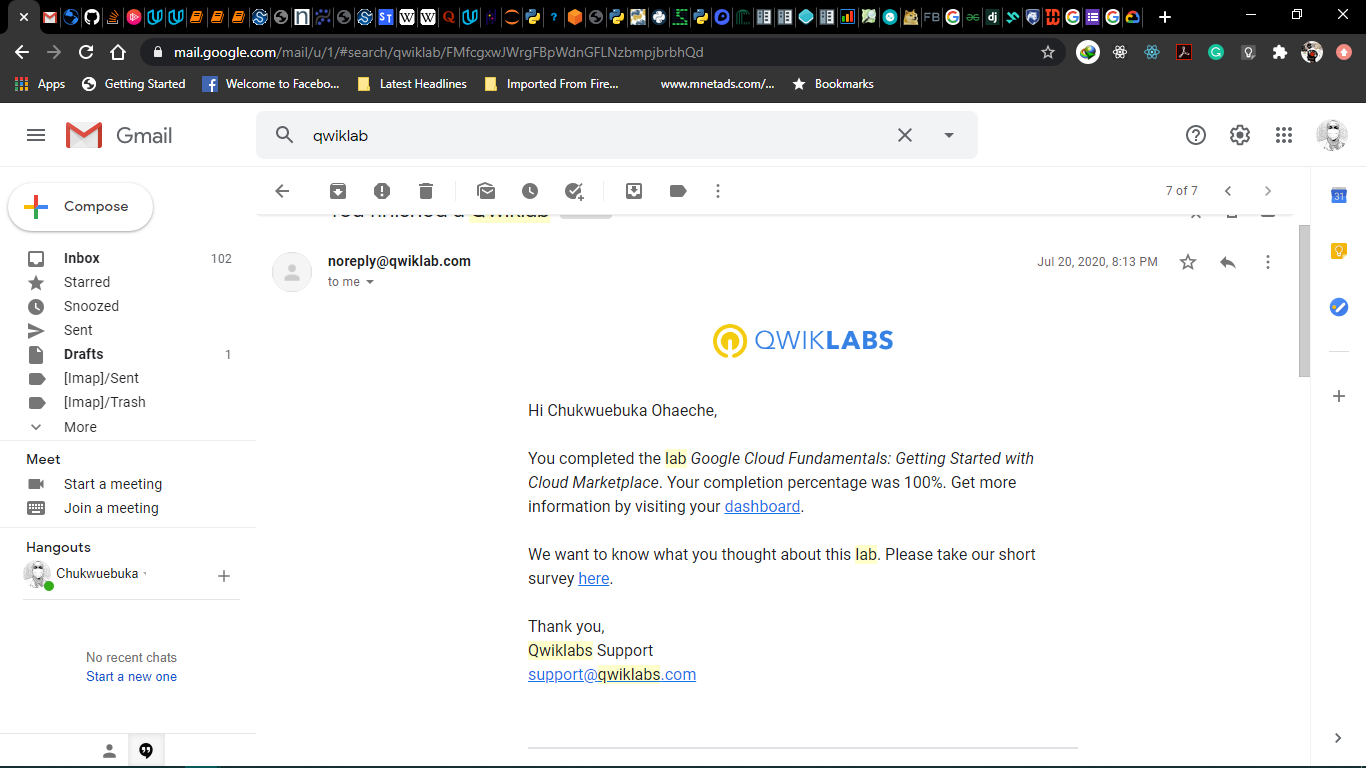
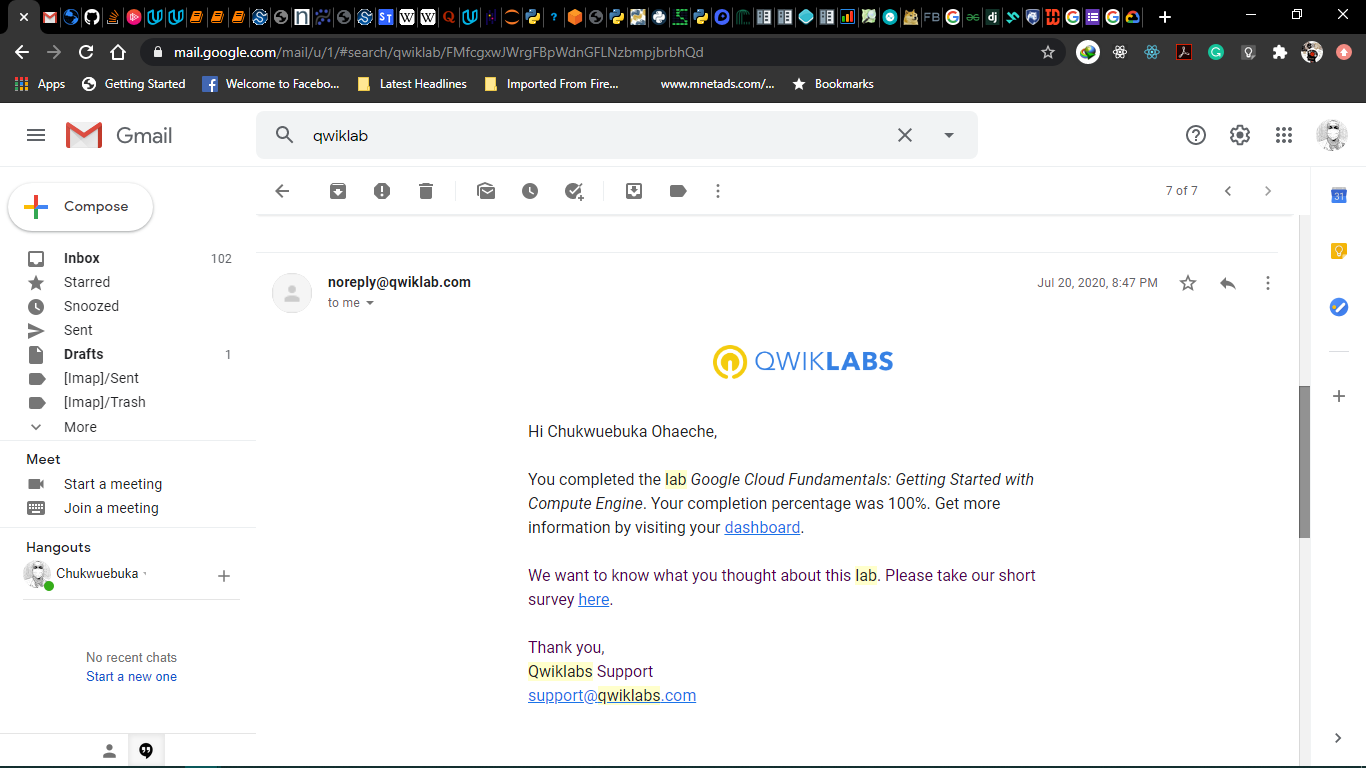
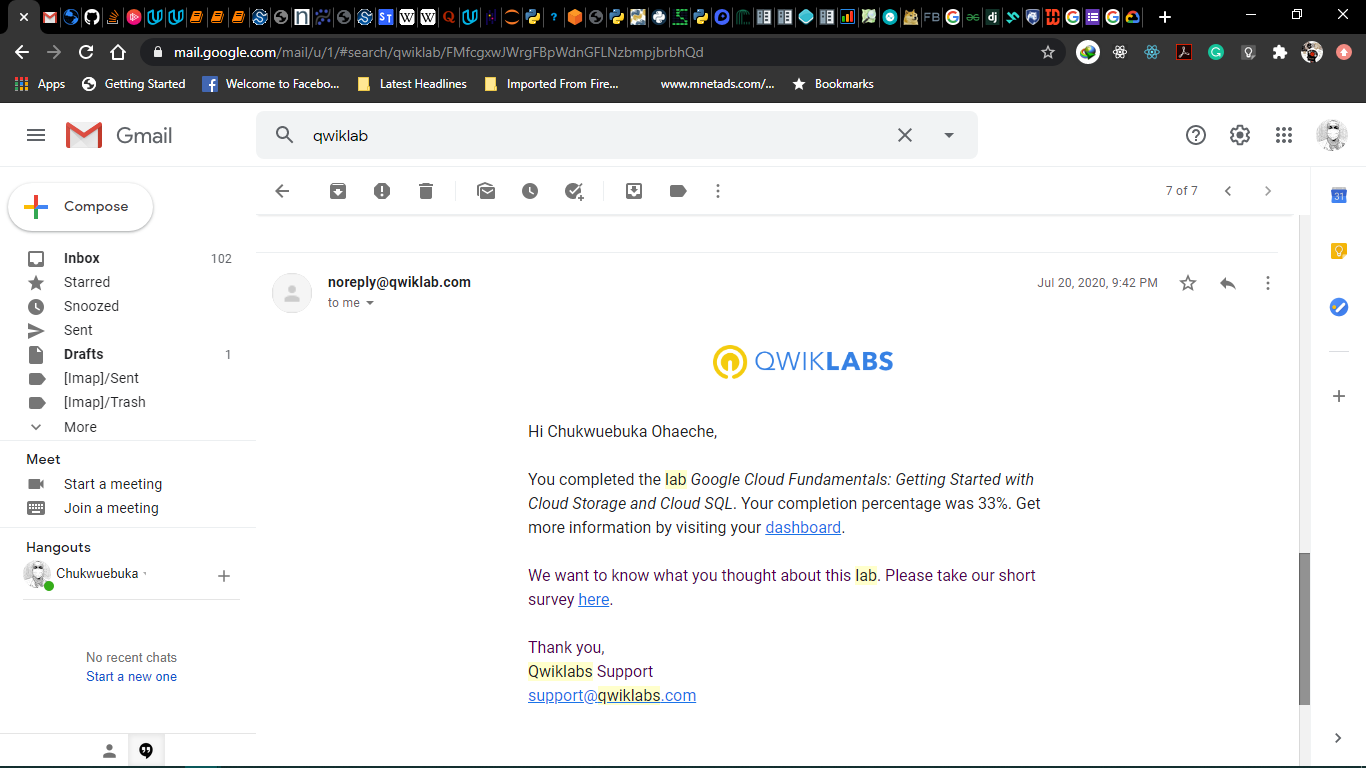
**Google Cloud Fundamentals: Getting Started with Cloud Marketplace**



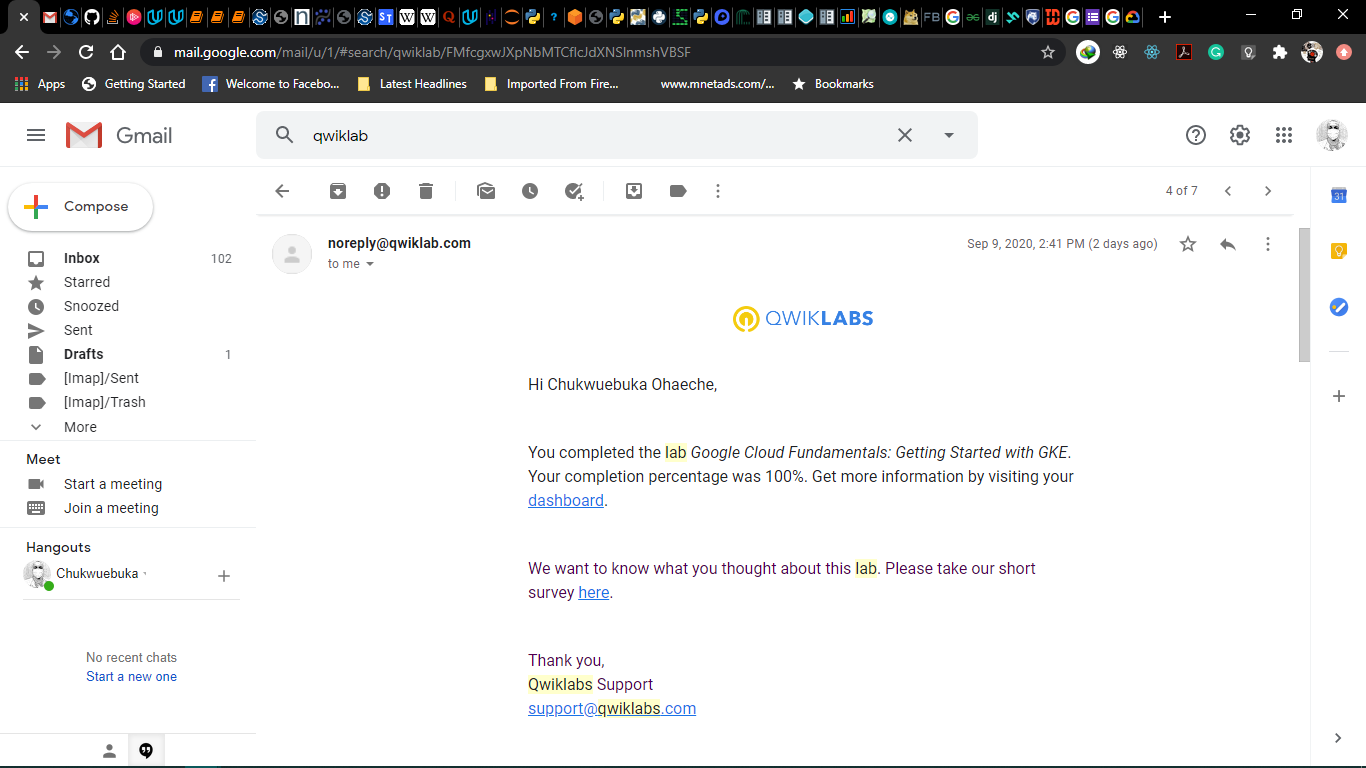
**Google Cloud Fundamentals: Getting Started with Compute Engine**



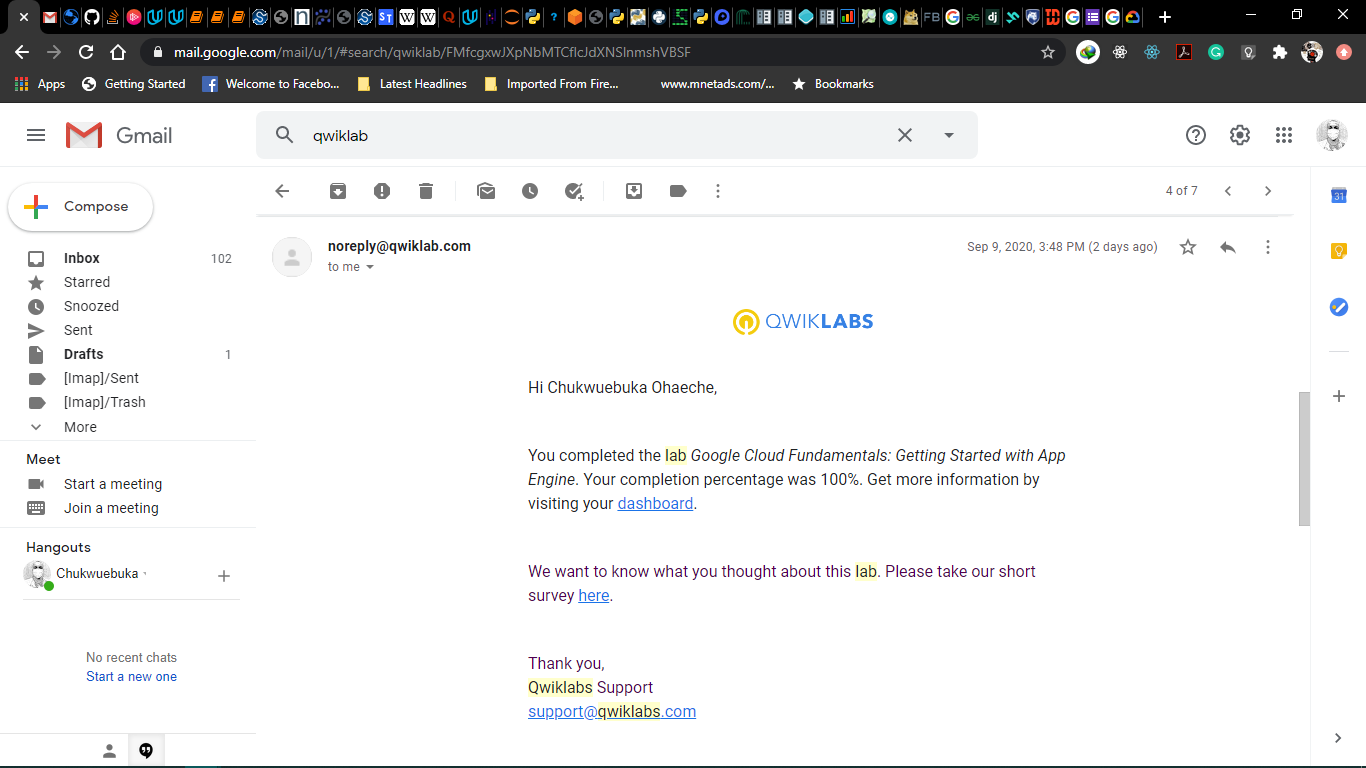
**Google Cloud Fundamentals: Getting Started with Cloud Storage and Cloud SQL**



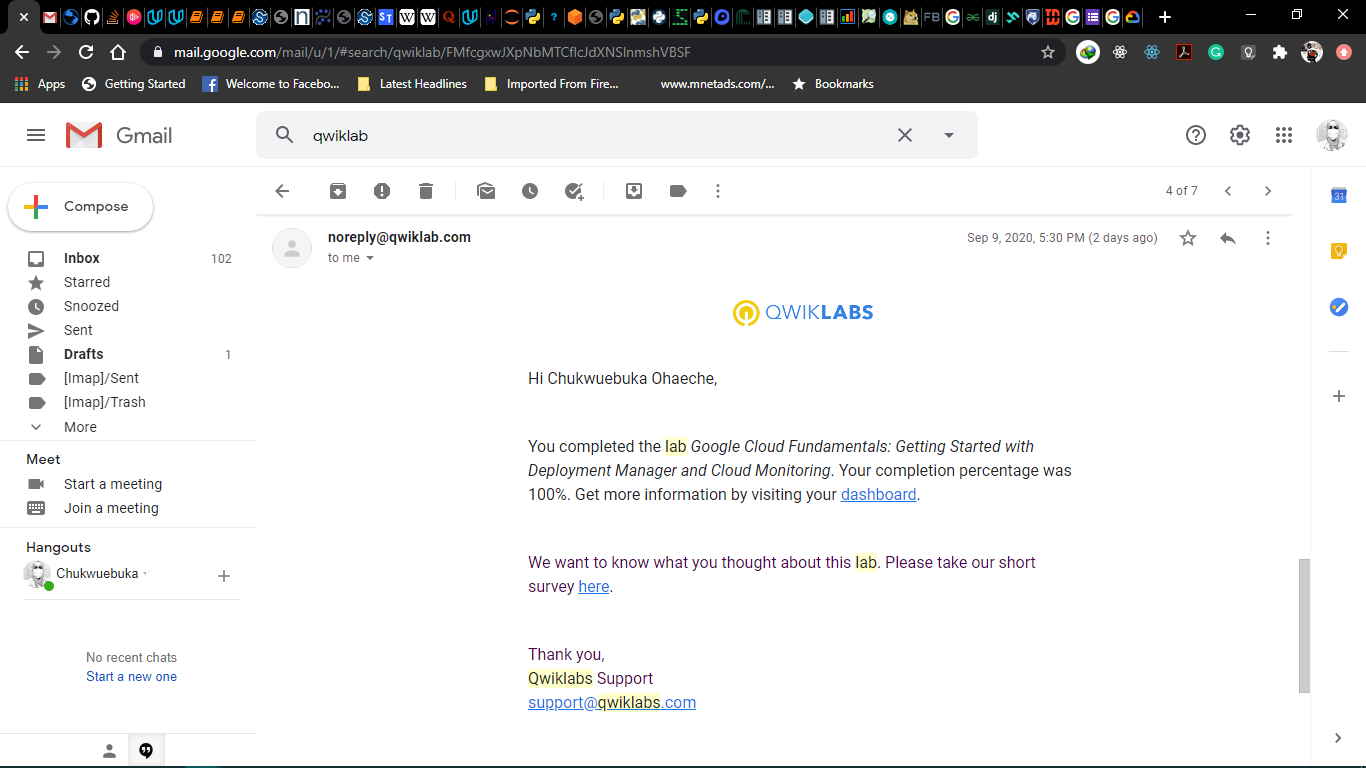
**Google Cloud Fundamentals: Getting Started with GKE**



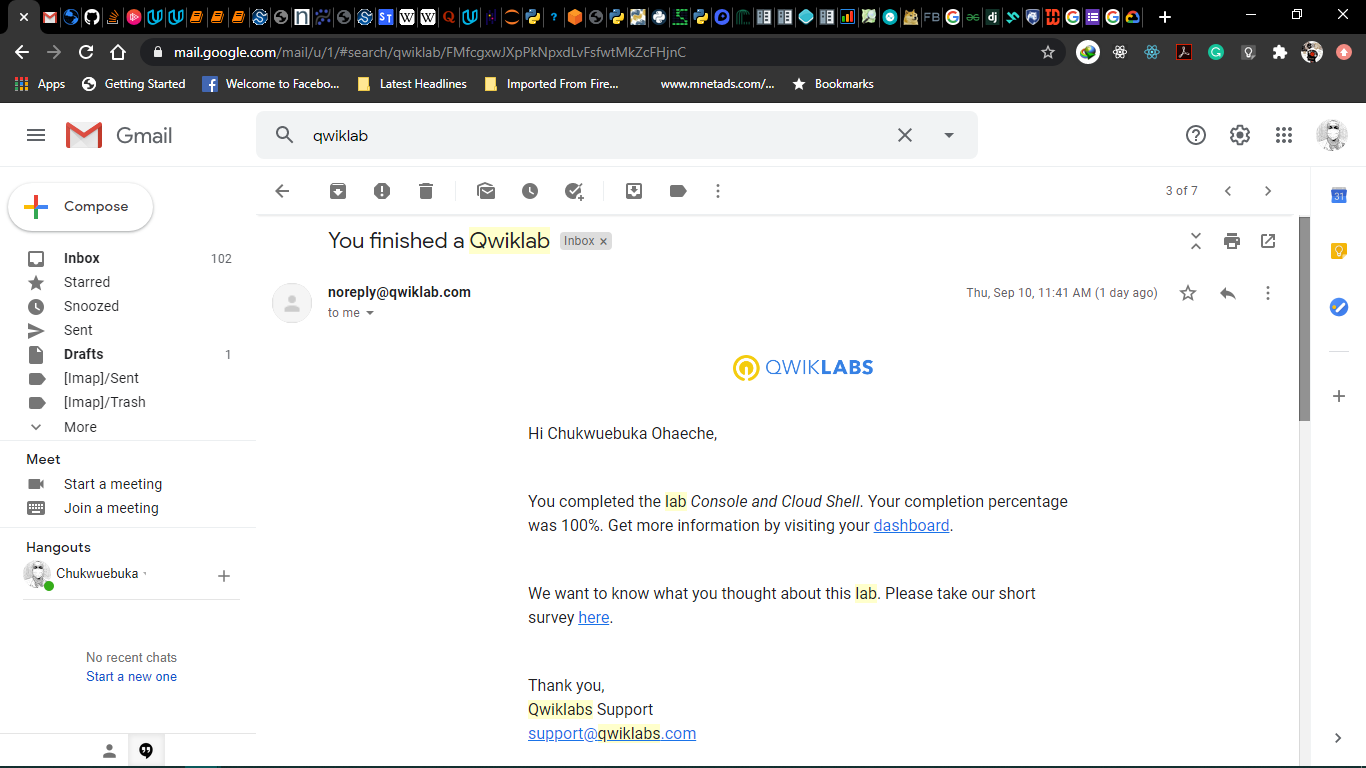
**Google Cloud Fundamentals: Getting Started with App Engine**



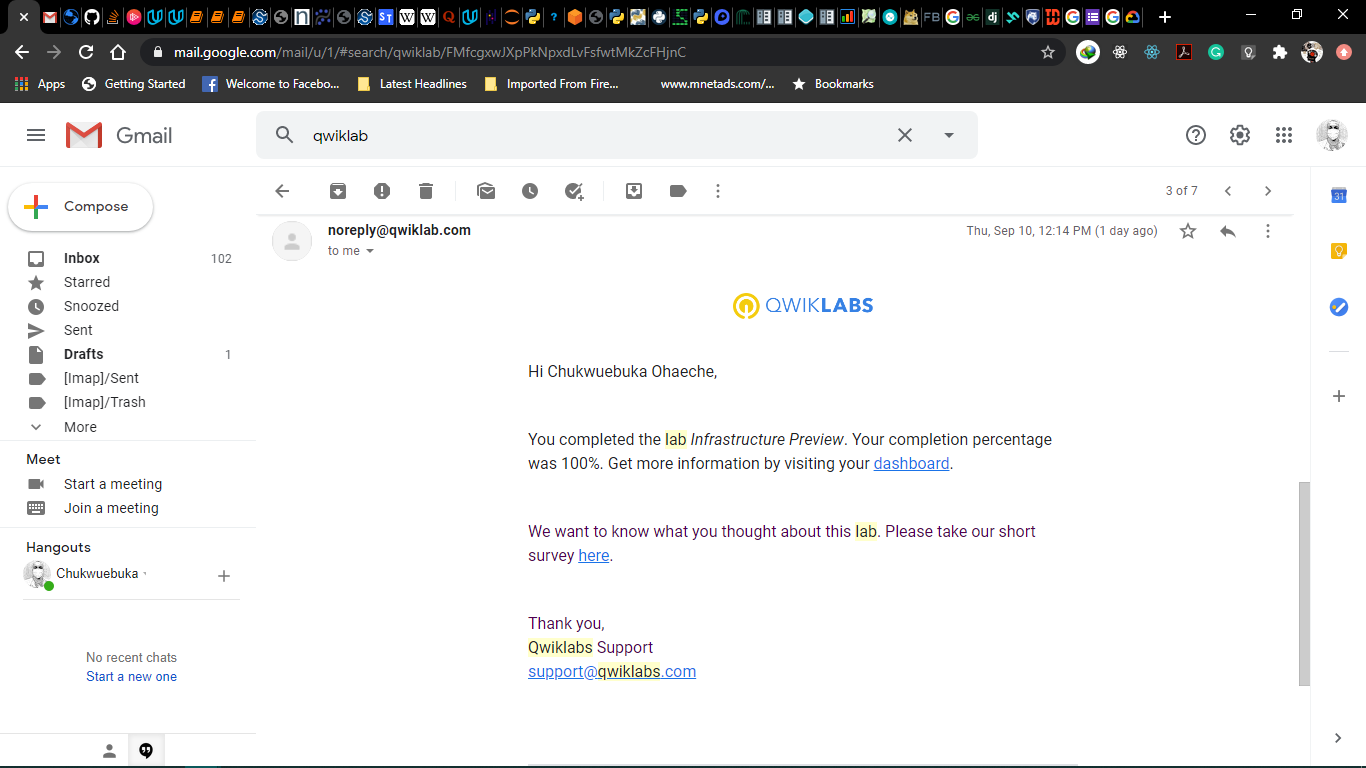
**Google Cloud Fundamentals: Getting Started with Deployment Manager and Cloud Monitoring**



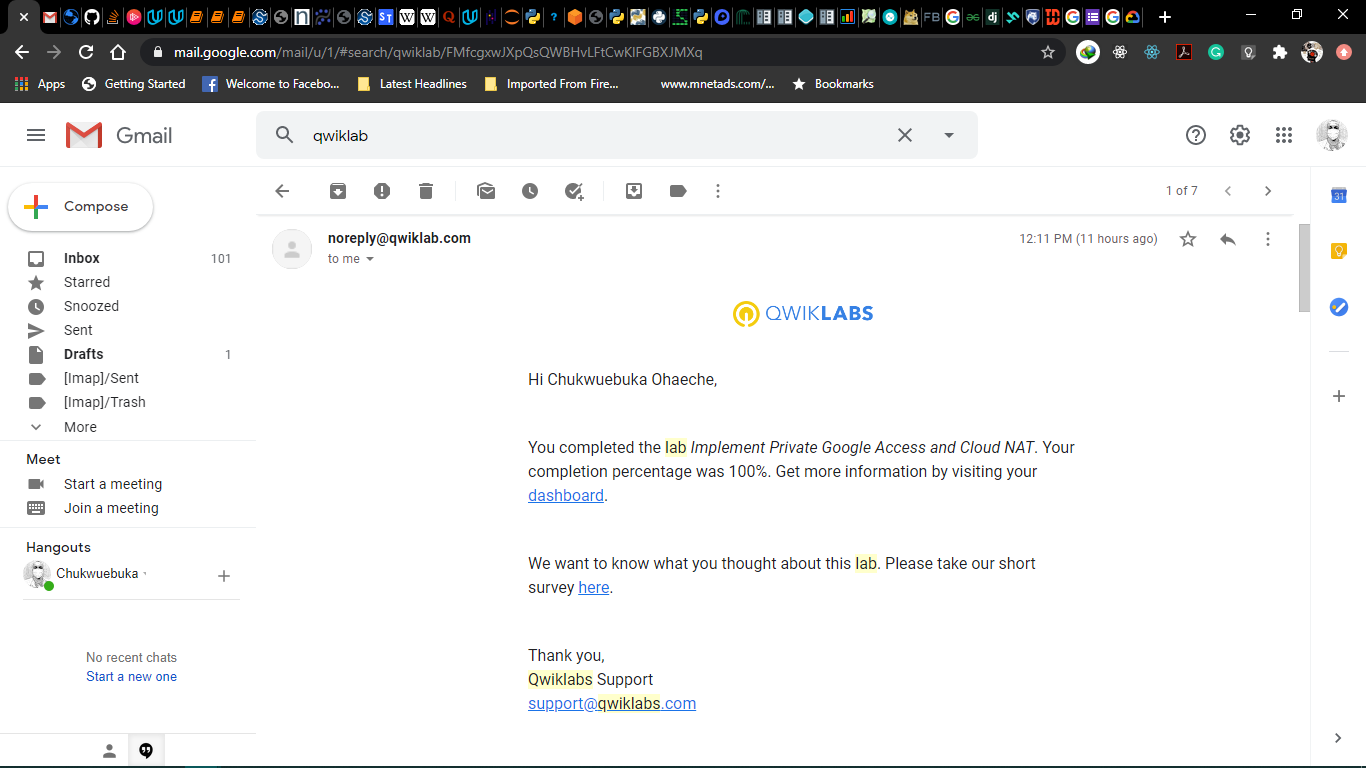
**Console and Cloud Shell**



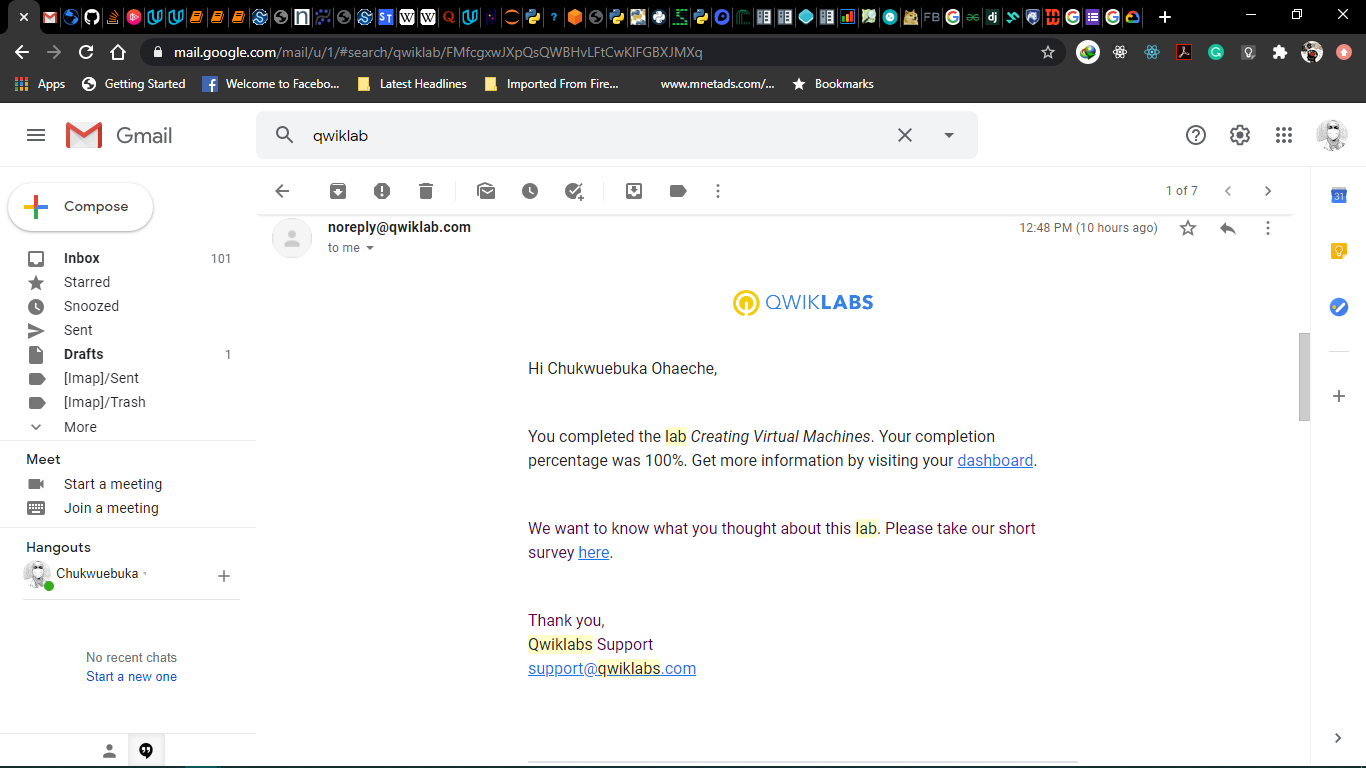
**Infrastructure Preview**



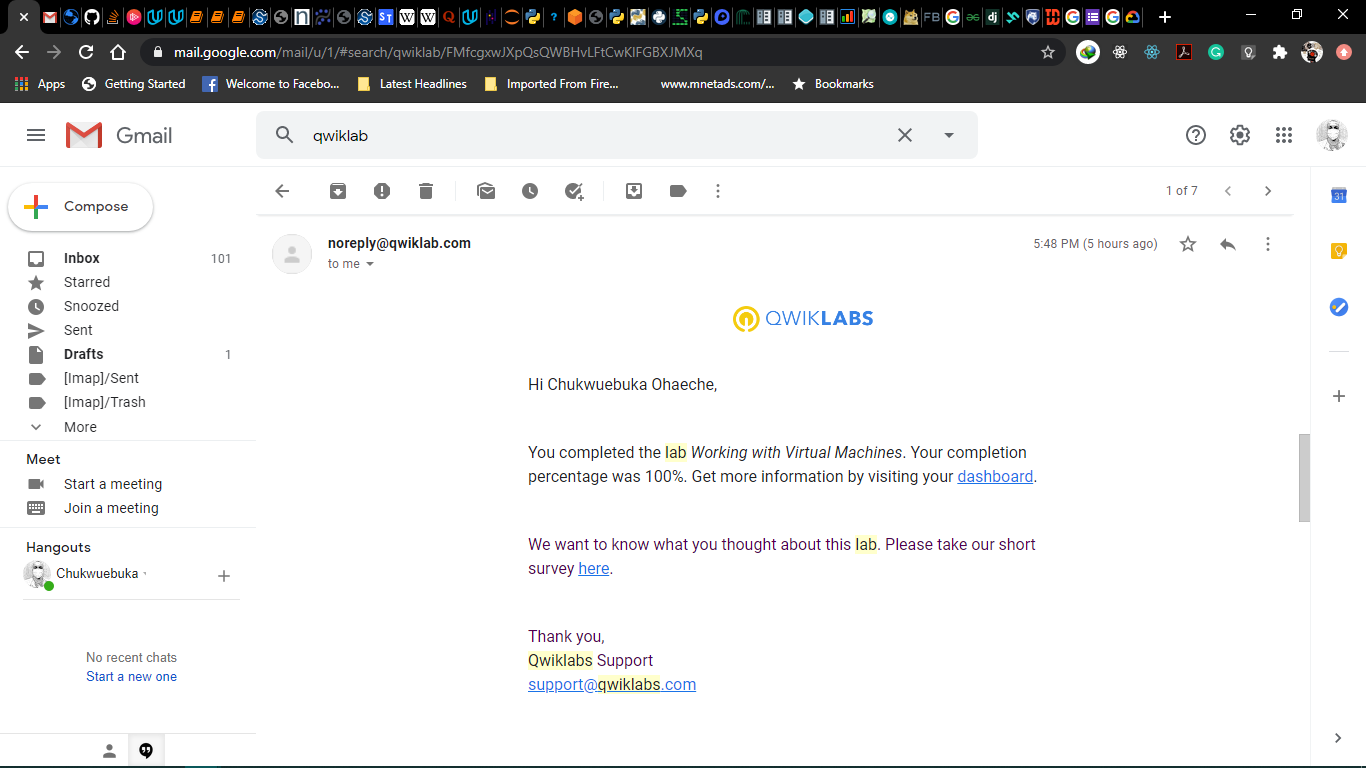
**Implement Private Google Access and Cloud NAT**



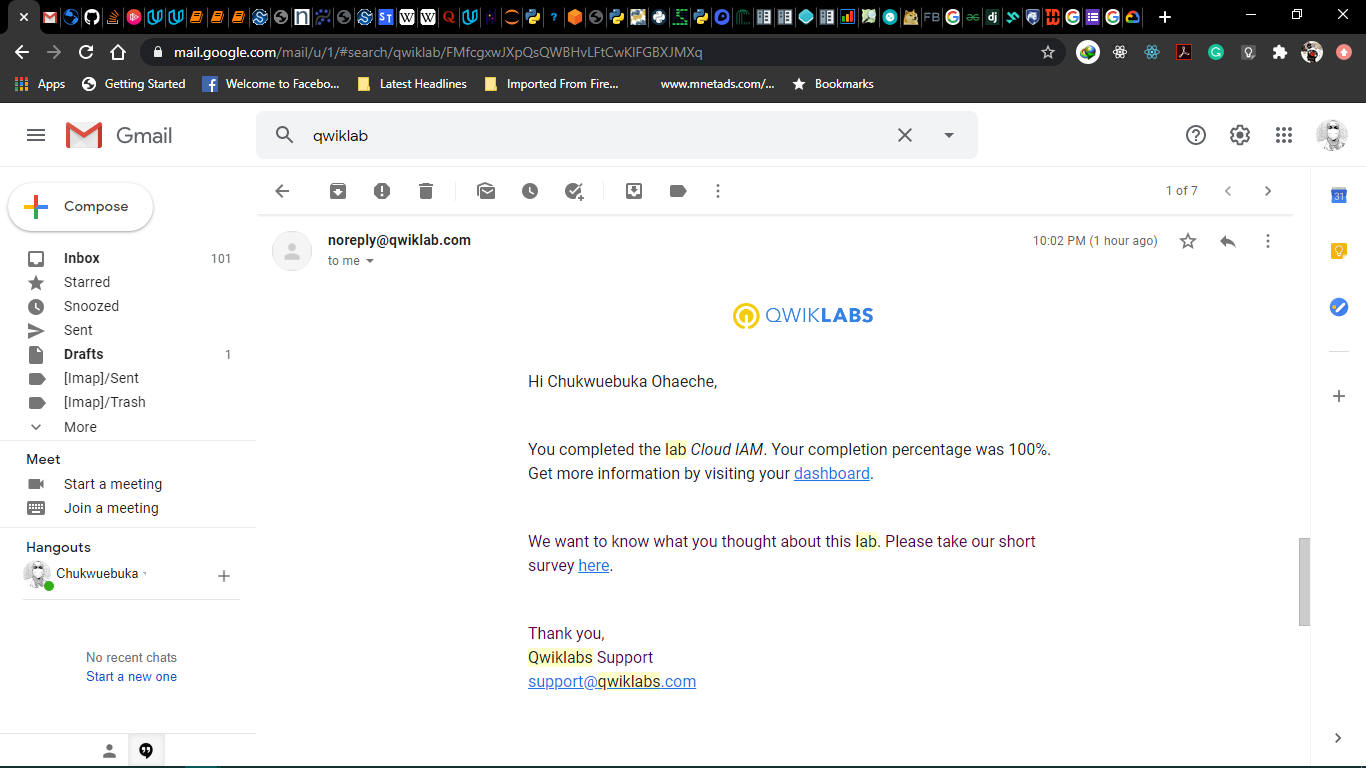
**Creating Virtual Machines**



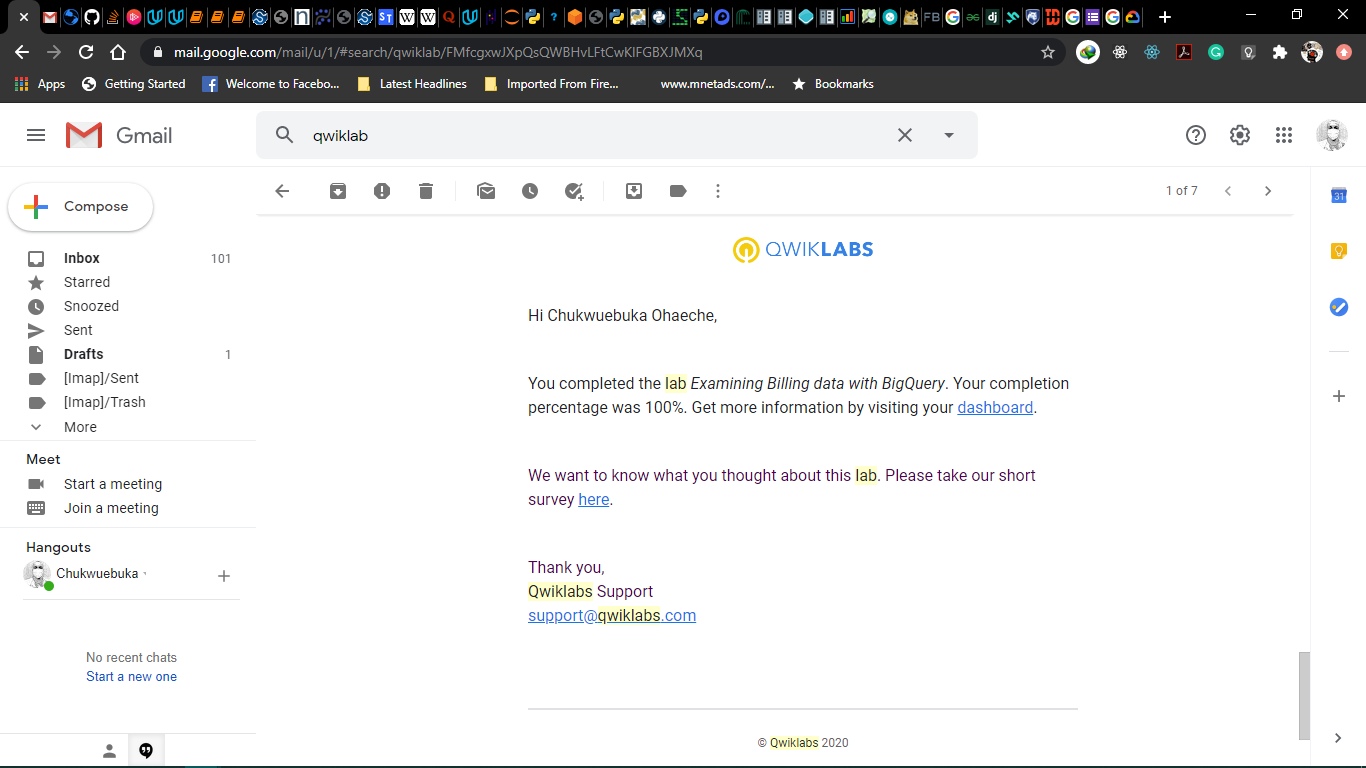
**Working with Virtual Machines**



**Cloud IAM**



**Examining Billing data with BigQuery**



**GCP Fundamentals: Getting Started with Kubernetes Engine**

In GCP console, on the top right toolbar, click the Open Cloud Shell button.

For convenience, place the zone that Qwiklabs assigned you to into an environment variable called MY\_ZONE

*export MY\_ZONE=us-central1-a*

Start a Kubernetes cluster managed by Kubernetes Engine. Name the cluster webfrontend and configure it to run 2 nodes:

*gcloud container clusters create webfrontend --zone $MY\_ZONE --num-nodes 2*

After the cluster is created, check your installed version of Kubernetes using the kubectl version command:

*kubectl version*

View your running nodes in the GCP Console.

*gcloud compute instances list*

From your Cloud Shell prompt, launch a single instance of the nginx container. (Nginx is a popular web server.)

*kubectl create deploy nginx --image=nginx:1.17.10*

View the pod running the nginx container:

*kubectl get pods*

Expose the nginx container to the Internet:

*kubectl expose deployment nginx --port 80 --type LoadBalancer*

View the new service:

*kubectl get services*

Scale up the number of pods running on your service:

*kubectl scale deployment nginx --replicas 3*

Confirm that Kubernetes has updated the number of pods:

*kubectl get pods*

Confirm that your external IP address has not changed:

*kubectl get services*

**GCP Fundamentals: Getting Started with App Engine**

In GCP console, on the top right toolbar, click the Open Cloud Shell button.

gcloud is the command-line tool for Google Cloud Platform. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

*gcloud auth list*

You can list the project ID with this command:

*gcloud config list project*

Initialize your App Engine app with your project and choose its region:

*gcloud app create --project=$DEVSHELL\_PROJECT\_ID*

When prompted, select the region where you want your App Engine application located.

Choose 14 for us-central1

Clone the source code repository for a sample application in the hello\_world directory:

*git clone* [*https://github.com/GoogleCloudPlatform/python-docs-samples*](https://github.com/GoogleCloudPlatform/python-docs-samples)

Navigate to the source directory:

*cd python-docs-samples/appengine/standard\_python3/hello\_world*

Execute the following command to download and update the packages list.

*sudo apt-get update*

Set up a virtual environment in which you will run your application

*sudo apt-get install virtualenv*

If prompted [Y/n], press Y and then Enter.

*virtualenv -p python3 venv*

Activate the virtual environment.

*source venv/bin/activate*

Install dependencies.

*pip install -r requirements.txt*

Run the application:

*python main.py*

In Cloud Shell, click Web preview (Web Preview) > Preview on port 8080 to preview the application.

To access the Web preview icon, you may need to collapse the Navigation menu.

To end the test, return to Cloud Shell and press Ctrl+C to abort the deployed service.

Verify that the app is not deployed.

*gcloud app browse*

To deploy your application to the App Engine Standard environment:

Navigate to the source directory:

*cd ~/python-docs-samples/appengine/standard\_python3/hello\_world*

Deploy your Hello World application.

*gcloud app deploy*

If prompted "Do you want to continue (Y/n)?", press Y and then Enter.

This app deploy command uses the app.yaml file to identify project configuration.

Launch your browser to view the app at http://YOUR\_PROJECT\_ID.appspot.com

*gcloud app browse*

In the Cloud Console, on the Navigation menu (Navigation menu), click App Engine > Settings.

Click Disable application.

*gcloud app instances delete i1 --service=s1 --version=v1*

If you refresh the browser window you used to view to the application site, you'll get a 404 error.

**GCP Fundamentals: Getting Started with Deployment Manager and Stackdriver**

In GCP console, on the top right toolbar, click the Open Cloud Shell button (Activate Cloud Shell). Click Continue.

For your convenience, place the zone that Qwiklabs assigned you to into an environment variable called MY\_ZONE. At the Cloud Shell prompt, type this partial command:

*export MY\_ZONE=us-central1-a*

At the Cloud Shell prompt, download an editable Deployment Manager template:

*gsutil cp gs://cloud-training/gcpfcoreinfra/mydeploy.yaml mydeploy.yaml*

In the Cloud Shell, use the sed command to replace the PROJECT\_ID placeholder string with your Google Cloud Platform project ID using this command:

*sed -i -e "s/PROJECT\_ID/$DEVSHELL\_PROJECT\_ID/" mydeploy.yaml*

In the Cloud Shell, use the sed command to replace the ZONE placeholder string with your Google Cloud Platform zone using this command:

*sed -i -e "s/ZONE/$MY\_ZONE/" mydeploy.yaml*

View the mydeploy.yaml file, with your modifications, with this command:

*cat mydeploy.yaml*

Build a deployment from the template:

*gcloud deployment-manager deployments create my-first-depl --config mydeploy.yaml*

When the deployment operation is complete, the gcloud command displays a list of the resources named in the template and their current state.

*gcloud compute instances list*

*gcloud compute instances describe test-instance*

Return to your Cloud Shell prompt. Launch the nano text editor to edit the mydeploy.yaml file:

*nano mydeploy.yaml*

Find the line that sets the value of the startup script, value: "apt-get update", and edit it so that it looks like this:

*value: "apt-get update; apt-get install nginx-light -y"*

Do not disturb the spaces at the beginning of the line. The YAML templating language relies on indented lines as part of its syntax. As you edit the file, be sure that the v in the word value in this new line is immediately below the k in the word key on the line above it.

Press Ctrl+O and then press Enter to save your edited file.

Press Ctrl+X to exit the nano text editor.

Return to your Cloud Shell prompt. Enter this command to cause Deployment Manager to update your deployment to install the new startup script:

*gcloud deployment-manager deployments update my-first-depl --config mydeploy.yaml*

Wait for the gcloud command to display a message confirming that the update operation was completed successfully.

*gcloud compute instances list*

*gcloud compute instances describe test-instance*

To open a command prompt on the my-vm instance,

*gcloud compute ssh example-instance --zone=us-central1-a*

In the ssh session on my-vm, execute this command to create a CPU load:

dd if=/dev/urandom | gzip -9 >> /dev/null &

This Linux pipeline forces the CPU to work on compressing a continuous stream of random data.

In the Google Cloud Platform Console, click on Navigation menu > Monitoring.

*gcloud monitoring dashboards describe MY-DASHBOARD*

Using your VM's open SSH window and the code shown on the Agents page, install both the Monitoring and Logging agents on your project's VM.

*gcloud compute ssh --project PROJECT\_ID --zone ZONE VM\_NAME*

Add the agent's package repository:

*curl -sSO https://dl.google.com/cloudagents/add-monitoring-agent-repo.sh*

*sudo bash add-monitoring-agent-repo.sh*

*sudo apt-get update*

Install the agent:

List the available versions of the agent in order to select which version to install:

*sudo apt-cache madison stackdriver-agent*

To install the latest version of the agent, run:

*sudo apt-get install stackdriver-agent*

Start the agent service

*sudo service stackdriver-agent start*

You can delete the installation script after it runs successfully.

To verify that the agent is working as expected, run:

*sudo service stackdriver-agent status*

Open a terminal connection to your VM instance using SSH or a similar tool.

*gcloud compute ssh example-instance --zone=us-central1-a*

Change to a directory you have write access to, for example your home directory.

Add the agent's package repository:

*curl -sSO https://dl.google.com/cloudagents/add-logging-agent-repo.sh*

*sudo bash add-logging-agent-repo.sh*

*sudo apt-get update*

Install the agent:

List the available versions of the agent in order to select which version to install:

*sudo apt-cache madison google-fluentd*

To install the latest version of the agent, run:

*sudo apt-get install google-fluentd*

Install the Configuration files.

For unstructured logging, run:

*sudo apt-get install -y google-fluentd-catch-all-config*

For structured logging, run:

*sudo apt-get install -y google-fluentd-catch-all-config-structured*

For information about structured logging, refer to the structured logging guide.

Start the agent service

*sudo service google-fluentd start*

In the Metric pane of Metrics Explorer, select the resource type GCE VM instance and the metric CPU usage.

In the resulting graph, notice that CPU usage increased sharply a few minutes ago.

Terminate your workload generator. Return to your ssh session on my-vm and enter this command:

*kill %1*