Terraform IaC Lab Documentation

Palo Alto Networks

Aug 13, 2021

overview

Version: 1.0.0

CHAPTER 1

Welcome

Welcome to the Terraform Infrastructure as Code Lab!

In this lab we will be deploying a multi-tiered web application to the cloud. To accomplish this we'll first need to define the infrastructure supporting our application as well as how we're going to secure the application. Rather than manually defining the infrastructure elements within the cloud provider portal, we're going to define our infrastructure in code using Terraform's declarative language and stateful build capabilities.

A key element of our infrastructure design will be the Palo Alto Networks VM-Series firewall. Using the Terraform provider for PAN-OS, we will define the configuration of the VM-Series firewall, but we'll be using Panorama to manage that configuration. The PAN-OS bootstrapping feature will be used to initialize the VM-Series firewall and point it to Panorama. Once it registers with Panorama, it will receive its runtime configuration.

Lastly, we will ensure that the firewall is able to respond effectively to changes made to the application infrastructure.

CHAPTER 2

Objective

The objective of this workshop is to deploy and secure a WordPress content management system in Google Cloud Platform. This web application will be supported by an Apache web server and a MariaDB database server residing in two separate subnets.

As part of our infrastructure deployment, a VM-Series virtual firewall will be inserted between the untrusted public subnet, the web subnet, and the database subnet. Our goal is to define, deploy, and configure all cloud infrastructure elements including the VM-series firewall using Terraform exclusively.

CHAPTER 3

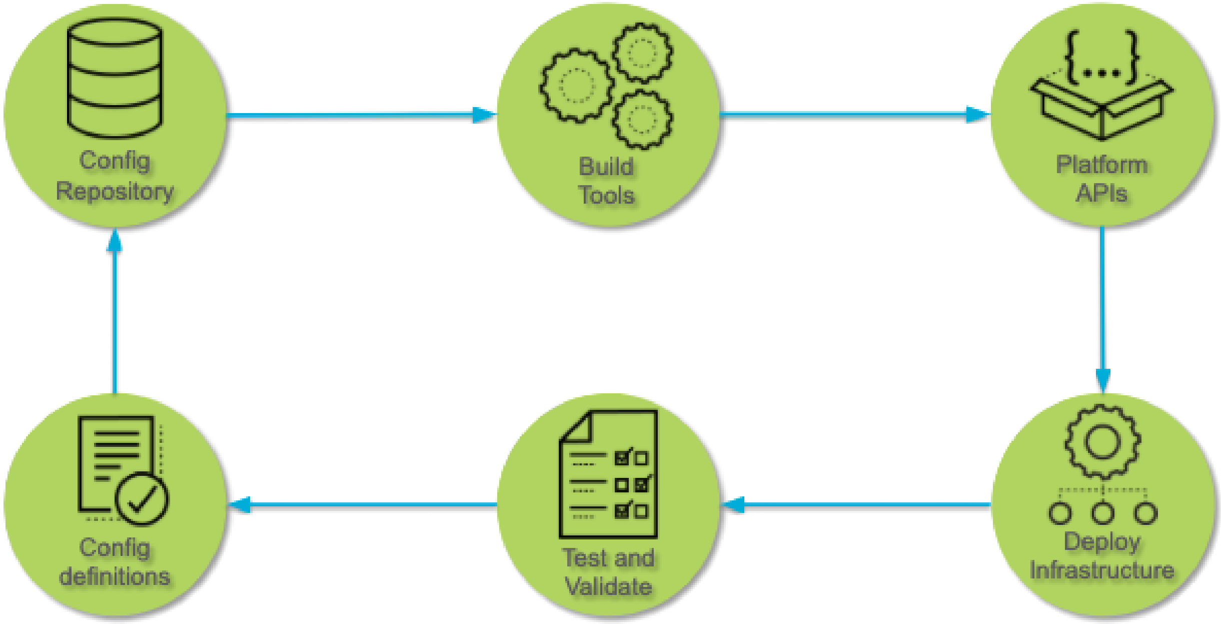
Learning Outcomes

- Learn best practices for managing infrastructure as code environments
- Learn how to use Terraform to define and deploy cloud infrastructure
- Learn about bootstrapping best practices for VM-Series firewalls in the cloud
- Learn about best practices for automating the management of VM-series firewalls

3.1 Lab Introduction

3.1.1 Infrastructure as Code

This training lab provides hands-on exposure to Infrastructure as Code (IaC) concepts and practices. IaC simply stated is the practice of defining, utilizing, and maintaining infrastructure definitions as one would application source code.



Config Definitions Infrastructure deployments and their configurations can be defined in various languages. Terraform provides an infrastructure definition language called HCL that is multi-cloud in that it supports many different public and private cloud environments. Most IaC languages are declarative in nature. This means they define the desired state of the infrastructure rather than the steps necessary to achieve that state.

Config Repository A version control system (VCS) is a key element in managing application source code. Since infrastructure definitions are just another form of source code, a VCS is a key element of IaC environemts as well. It supports revision control, code checkout/checkin, team collaboration, code review, and API hooks for automated build tools.

Build Tools Build tools act on the infrastructure definitions to build and configure the infrastructure components. Often, this build process is incorporated into a larger Continuous Delivery (CD) pipeline with automated tests and gate factors that govern the build process.

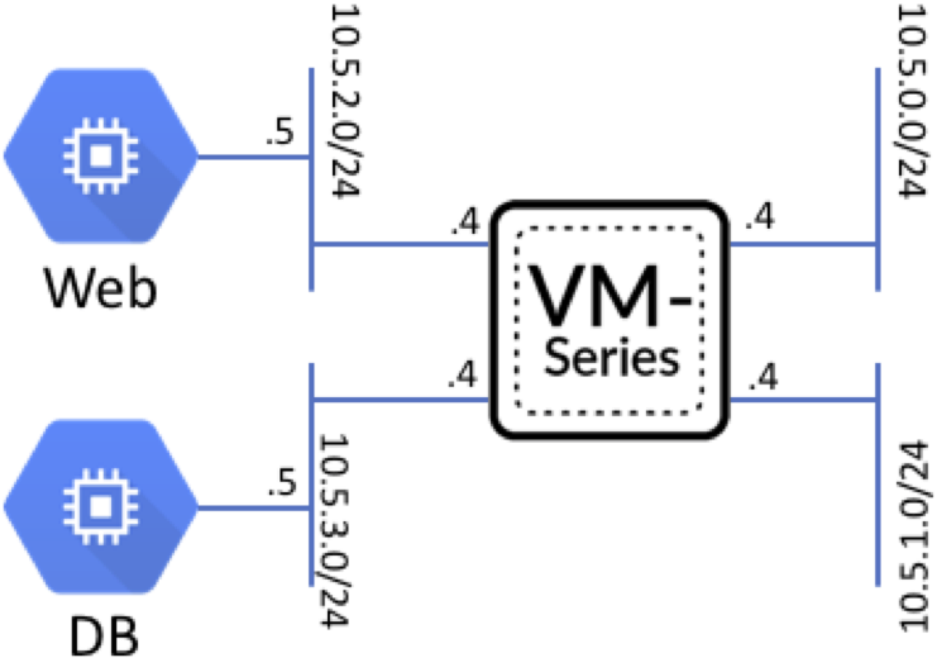
Platform APIs In order to automate the deployment and configuration of infrastructure elements an API must be accessible to the build tool being used. Cloud providers such as AWS, GCP, and Azure all have various APIs supporting different infrastructure components and services. Third-party virtual appliances such as the VM-Series virtual firewall also has an API that may be used for configuration purposes. However, the use of a common infrastructure definition language and build tools all but eliminates the need to interact directly with any of these APIs.

Deploy Infrastructure The build tool is responsible for processing the infrastructure definition and leveraging the backend APIs and orchestrating the deployment the infrastructure components. Terraform accomplishes this by first assessing the state of each infrastructure resource. If nothing is instantiated, Terraform will take the necessary steps to deploy the resource and configure it based on the parameters that have been provided.

Test and Validate A set of tests may be performed to ensure that the deployed infrastructure aligns with the definitions provided in the infrastructure code. Since Terraform maintains state files detailing the live environment, it is easy to run a `terraform plan` command to identify and resolve and disparties.

3.1.2 Lab Topology

The following diagram details the infrastructure topology associated with the web application we will be deploying. The VM-Series firewall will be used to inspect and protect “North-South” traffic between the `untrust` and `web` zones as well as "East-West" traffic between the web and database zones.



Subnet	Address	Interface
Management	10.5.0.0/24	Management
Untrust	10.5.1.0/24	ethernet1/1
Web	10.5.2.0/24	ethernet1/2
Database	10.5.3.0/24	ethernet1/3

3.1.3 Lab Components

Qwiklabs This lab is launched using Qwiklabs, which is an online learning platform that deploys and provides access to cloud-based lab environments. Qwiklabs will establish a set of temporary set of credentials in the cloud provider in order to deploy and access the cloud infrastructure and services.

Google Cloud Platform (GCP) Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube.

Panorama Panorama is the centralized management and reporting platform for PAN-OS devices including firewall appliances, VM-Series virtual firewalls, WildFire appliances, and log collectors. It serves as the configuration "source of truth" for a PAN-OS infrastructure deployment and supports the same XML and REST APIs as other PAN-OS devices.

Hashicorp Terraform Each cloud provider offers a mechanism that allow you to define a set of infrastructure element or services and orchestrate their instantiation. However, these tools and templates are specific to each cloud provider. We will be using Terraform to perform this function as it provides a common set of capabilities and a template formats acroos all cloud providers.

3.2 Requirements

The following are requirements of this training workshop:

- A laptop with Internet connectivity (SSH and HTTPS access is required).
- A standards-compliant web browser (Google Chrome is recommended).
- An SSH client (e.g., OpenSSH, PuTTY, SecureCRT, etc).
- An understanding of Linux operating system basics.
- Proficiency with a Linux text editor such as vim or nano.
- A basic understanding of cloud computing concepts.

3.3 Setup

In this activity you will:

- Launch the lab
- Log into the Google Cloud Platform console
- Launch a Cloud Shell instance
- Clone the lab software repository

This lab can be used in two different ways, through an organized class using Qwiklabs, or at your own pace using your personal GCP account. The two ways differ only in the setup method - every thing after logging into the GCP console is the same.

> **Warning:** If you use your own personal GCP account, you are responsible for all incurred charges and deleting the created resources after completing the lab.

Follow the instructions for the method you are using for this lab:

- *Setup - Qwiklabs Class*
- *Setup - Personal Account*

3.3.1 Setup - Qwiklabs Class

> **Warning:** Before you start it is recommended that you launch a private instance of your web browser. This will prevent the use of cached Google credentials when logging into the GCP console. This will help ensure you do not incur any personal charges.

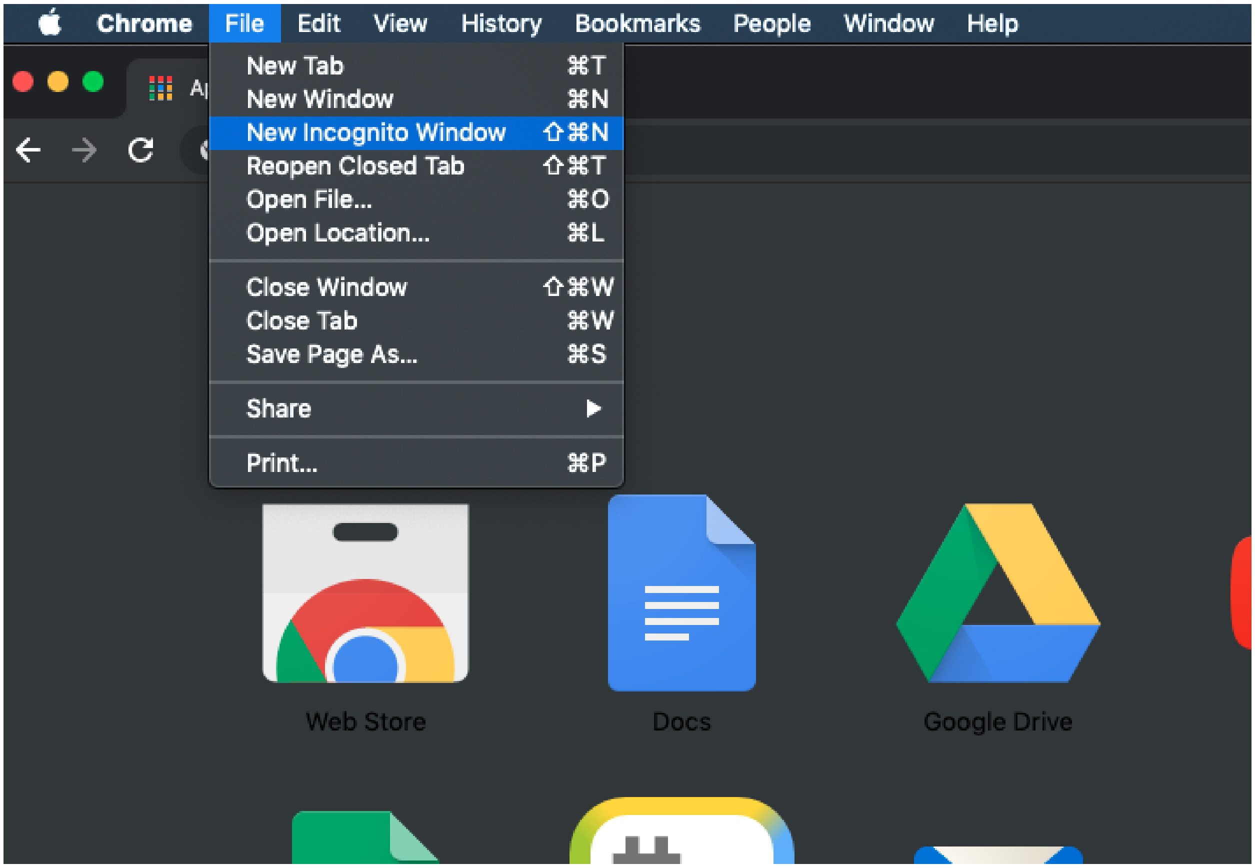


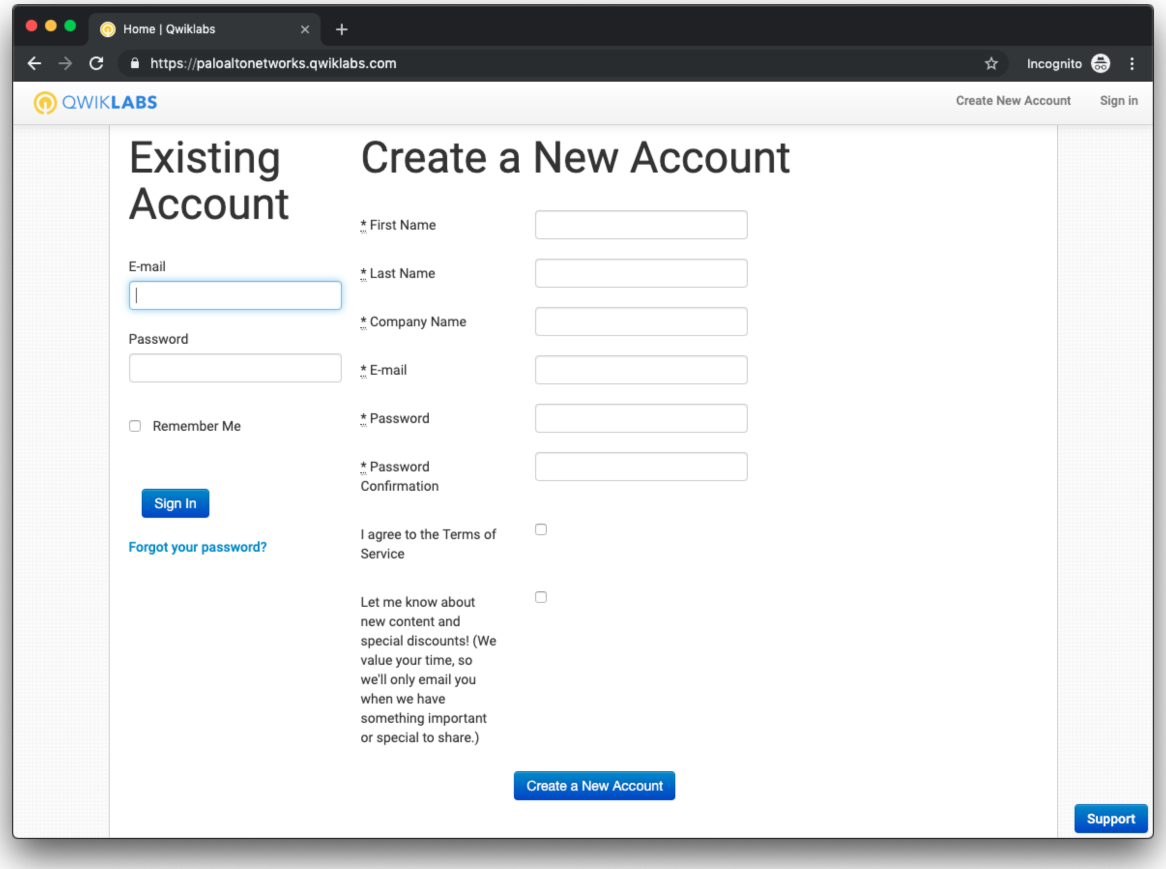
Fig. 1: Chrome Incognito mode

Log into the Qwiklabs portal

Navigate to the Qwiklabs URL in your web browser.

`https://paloaltonetworks.qwiklabs.com`

Log in with your Qwiklabs credentials (sign up if you are new to Qwiklabs). You must use your corporate email address for the username.



□ I agree to the Terms of Service

□ Let me know about new content and special discounts! (We value your time, so we'll only email you when we have something important or special to share.)

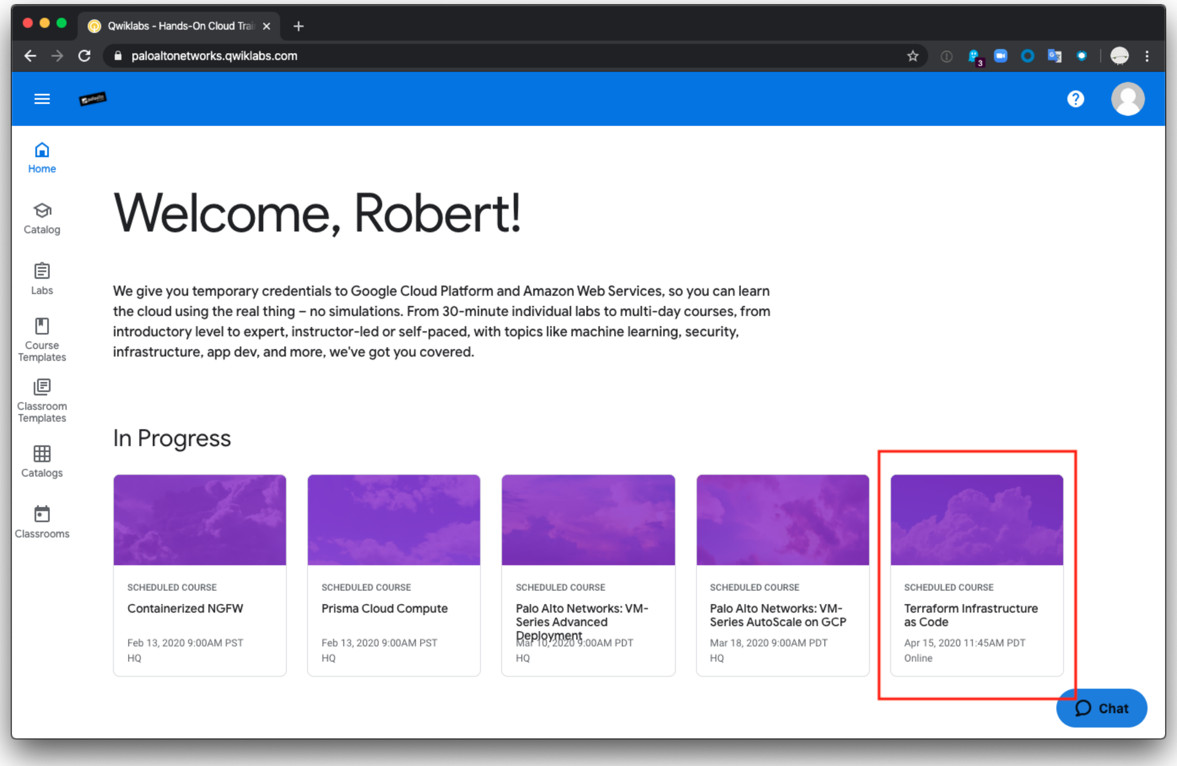
Launch the lab environment

Confirm that the course containing the phrase **Terraform Infrastructure as Code** is listed under In Progress on the welcome screen. Click on the this course in order to add it to your My Learning inventory.

You will will be directed to a screen containing the available labs. Click on the **Terraform Infrastructure as Code Lab**.

You will be presented with a lab evironment screen, you will need to click the **Start Lab** button. Qwiklabs will then provision a set of account credentials for GCP.

The lab environment will take a few minutes to provision. Once it is completed, a set of GCP credentials will be added to the lefthand panel.



Welcome, Robert!

We give you temporary credentials to Google Cloud Platform and Amazon Web Services, so you can learn the cloud using the real thing – no simulations. From 30-minute individual labs to multi-day courses, from introductory level to expert, instructor-led or self-paced, with topics like machine learning, security, infrastructure, app dev, and more, we've got you covered.

In Progress

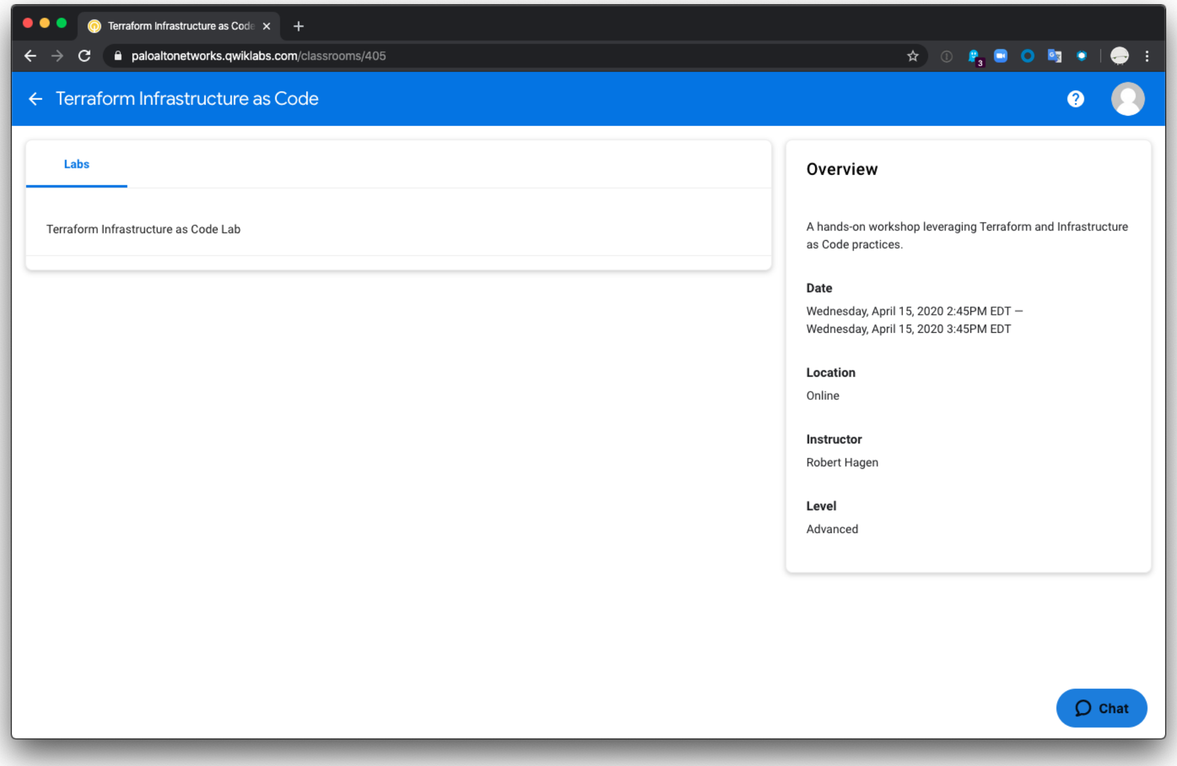
SCHEDULED COURSE
Containerized NGFW
Feb 13, 2020 9:00AM PST
HQ

SCHEDULED COURSE
Prisma Cloud Compute
Feb 13, 2020 9:00AM PST
HQ

SCHEDULED COURSE
Palo Alto Networks: VM-Series Advanced Deployment
Mar 10, 2020 9:00AM PDT
HQ

SCHEDULED COURSE
Palo Alto Networks: VM-Series AutoScale on GCP
Mar 18, 2020 9:00AM PDT
HQ

SCHEDULED COURSE
Terraform Infrastructure as Code
Apr 15, 2020 11:45AM PDT
Online


Terraform Infrastructure as Code
paloaltonetworks.qwiklabs.com/classrooms/405
Labs
Chat

Terraform Infrastructure as Code

Terraform Infrastructure as Code Lab

Overview

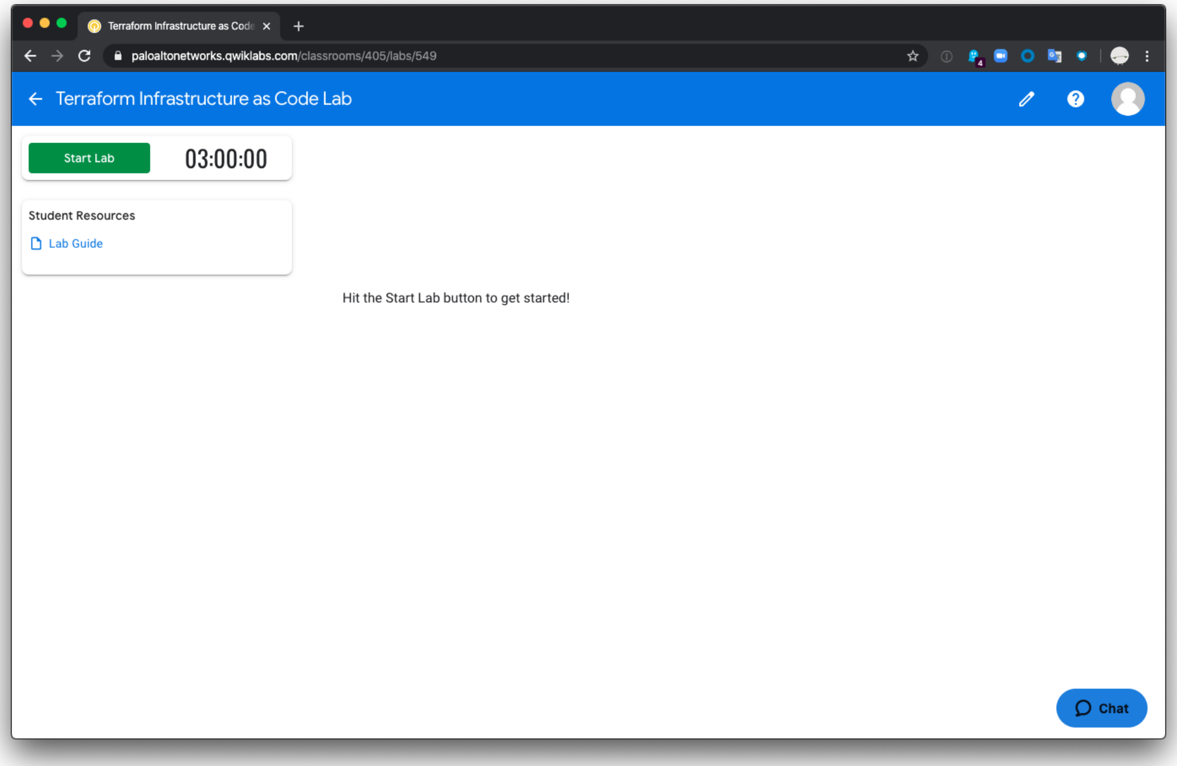
A hands-on workshop leveraging Terraform and Infrastructure as Code practices.

Date
Wednesday, April 15, 2020 2:45PM EDT – Wednesday, April 15, 2020 3:45PM EDT

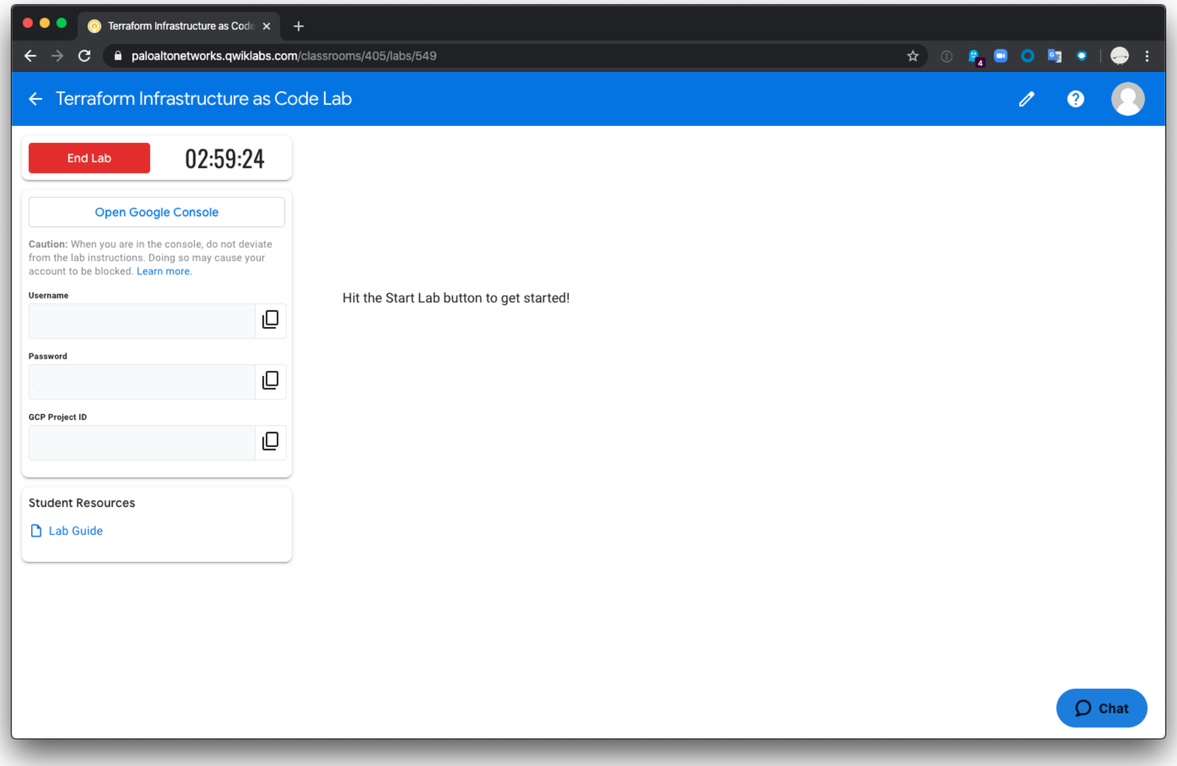
Location
Online

Instructor
Robert Hagen

Level
Advanced


Terraform Infrastructure as Code
paloaltonetworks.qwiklabs.com/classrooms/405/labs/549
Terraform Infrastructure as Code Lab
Start Lab
03:00:00
Student Resources
Lab Guide
Chat

Hit the Start Lab button to get started!


Terraform Infrastructure as Code
paloaltonetworks.qwiklabs.com/classrooms/405/labs/549
Terraform Infrastructure as Code Lab
End Lab
02:59:24
Open Google Console
Username
Password
GCP Project ID
Student Resources
Lab Guide
Chat

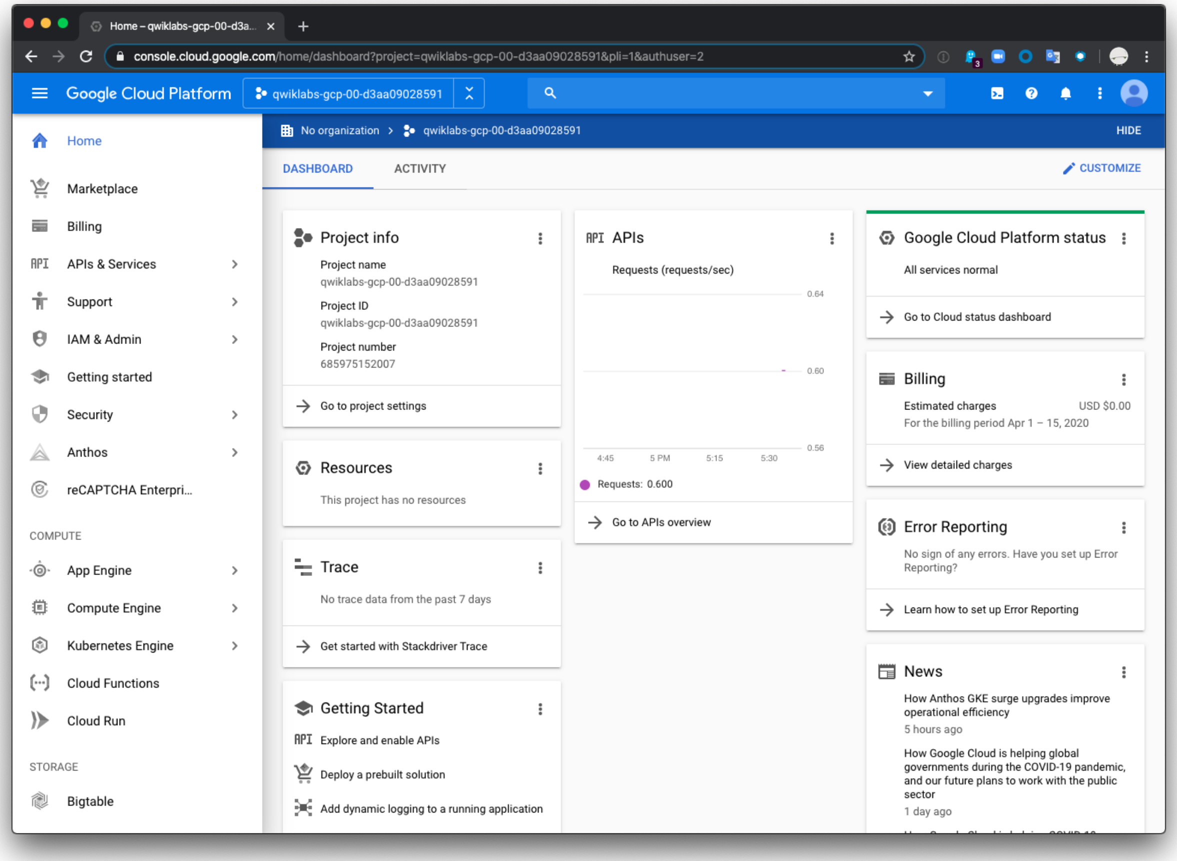
Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Hit the Start Lab button to get started!

Log into Google Cloud

Click on the **Open Google Console** to launch a new browser tab and log into GCP. Be sure to use the credentials provided by Qwiklabs rather than your personal Google account. This will ensure that you are not charged for any infrastructure that you deploy.

Once you've authenticated, accept the user agreements and account recovery defaults. You will then be presented with the Google Cloud dashboard.



Project info

Project name
qwiklabs-gcp-00-d3aa09028591

Project ID
qwiklabs-gcp-00-d3aa09028591

Project number
685975152007

→ Go to project settings

Resources

This project has no resources

Trace

No trace data from the past 7 days

→ Get started with Stackdriver Trace

Getting Started

Explore and enable APIs

Deploy a prebuilt solution

Add dynamic logging to a running application

APIs

Requests (requests/sec)

Requests: 0.600

→ Go to APIs overview

Google Cloud Platform status

All services normal

→ Go to Cloud status dashboard

Billing

Estimated charges USD $0.00

For the billing period Apr 1 – 15, 2020

→ View detailed charges

Error Reporting

No sign of any errors. Have you set up Error Reporting?

→ Learn how to set up Error Reporting

News

How Anthos GKE surge upgrades improve operational efficiency
5 hours ago

How Google Cloud is helping global governments during the COVID-19 pandemic, and our future plans to work with the public sector
1 day ago

Warning: Continue on with the lab in the *Launch the Cloud Shell* section.

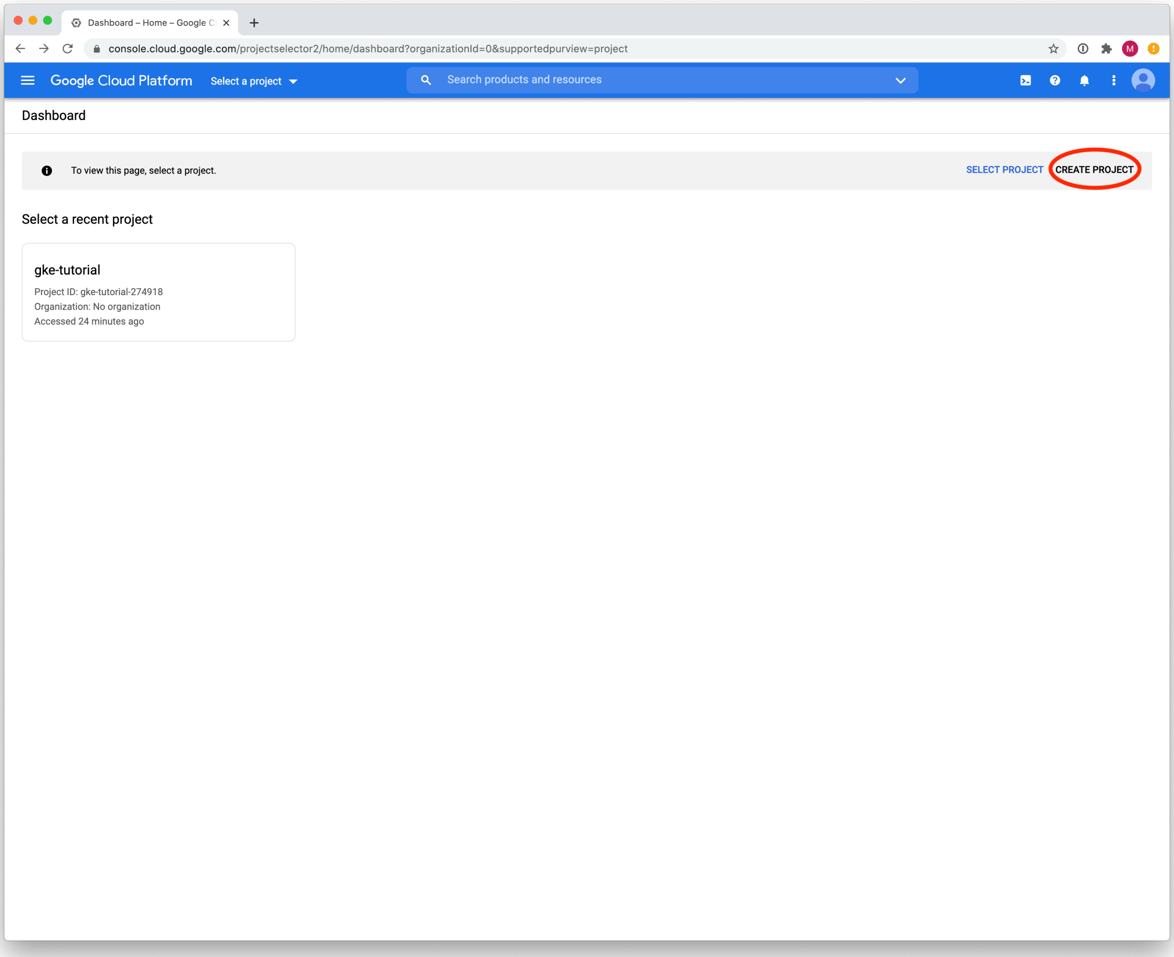
3.3.2 Setup - Personal Account

Create a GCP project

Log into the GCP console, and click **Create Project** to create a new project.

Give your project a unique name, and click **Create**.

Make sure your project is selected at the top of the console.



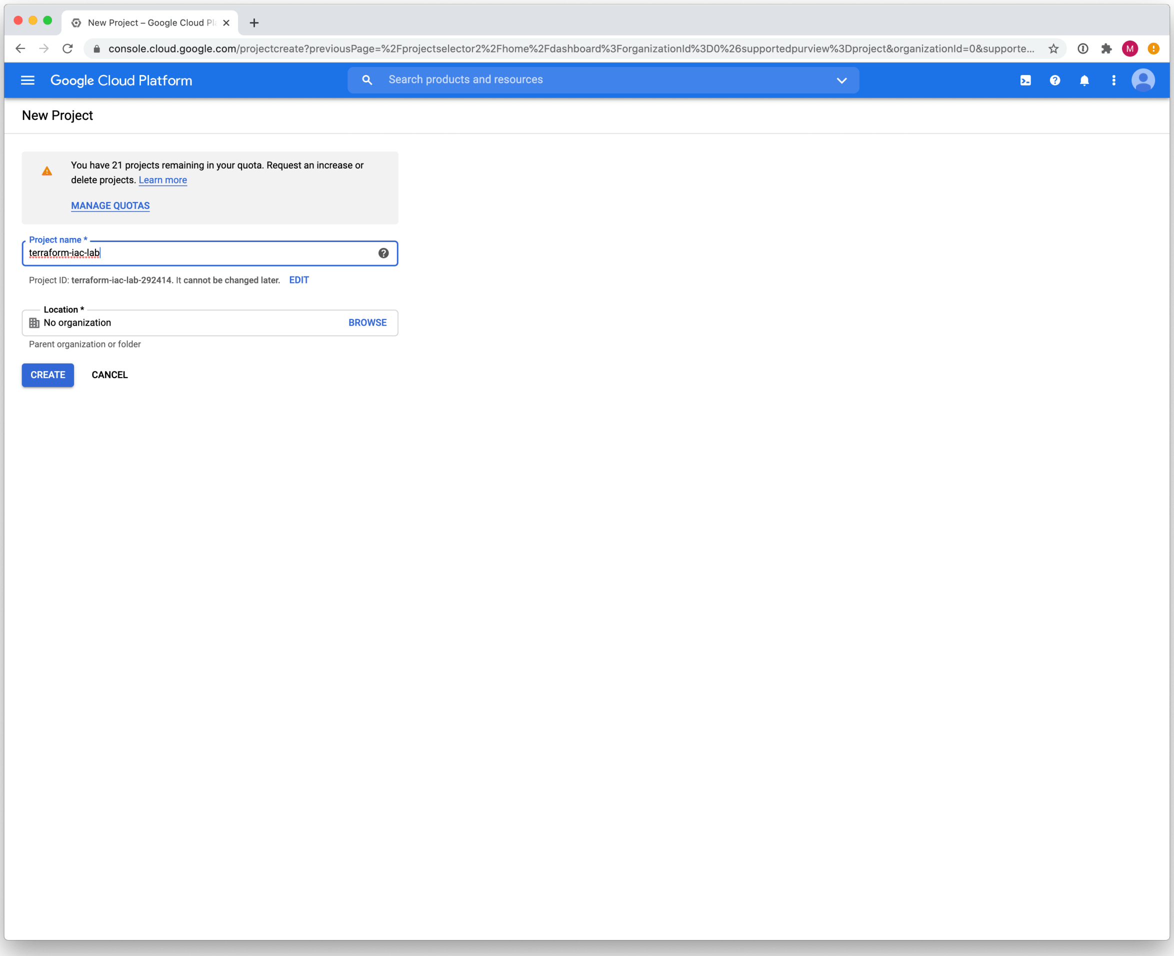
Dashboard – Home – Google C
console.cloud.google.com/projectselector2/home/dashboard?organizationId=0&supportedpurview=project
Google Cloud Platform
Select a project
Search products and resources
Dashboard

To view this page, select a project.

SELECT PROJECT CREATE PROJECT

Select a recent project

gke-tutorial
Project ID: gke-tutorial-274918
Organization: No organization
Accessed 24 minutes ago



New Project – Google Cloud Pl
console.cloud.google.com/projectcreate?previousPage=%2Fprojectselector2%2Fhome%2Fdashboard%3ForganizationId%3D0%26supportedpurview%3Dproject&organizationId=0&supporte...
Google Cloud Platform
Search products and resources

New Project

You have 21 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

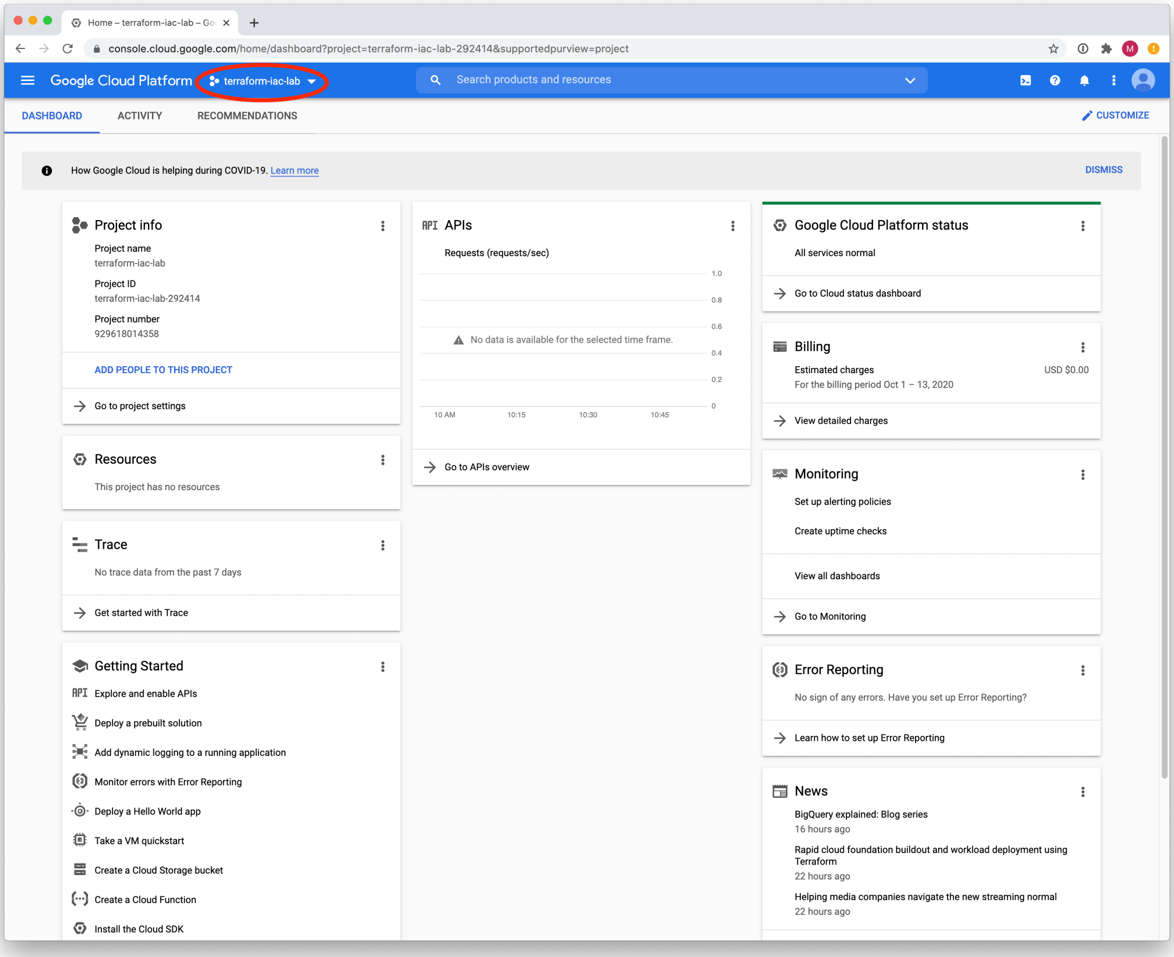
Project name *
terraform-iac-lab

Project ID: terraform-iac-lab-292414. It cannot be changed later. EDIT

Location *
No organization BROWSE

Parent organization or folder

CREATE CANCEL



Home – terraform-iac-lab – Go
console.cloud.google.com/home/dashboard?project=terraform-iac-lab-292414&supportedpurview=project
Google Cloud Platform
terraform-iac-lab
Search products and resources
DASHBOARD
ACTIVITY
RECOMMENDATIONS
CUSTOMIZE
DISMISS
1.0
0.8
0.6
0.4
0.2
0
10 AM
10:15
10:30
10:45

How Google Cloud is helping during COVID-19. Learn more

Project info

Project name
terraform-iac-lab

Project ID
terraform-iac-lab-292414

Project number
929618014358

ADD PEOPLE TO THIS PROJECT

→ Go to project settings

Resources

This project has no resources

Trace

No trace data from the past 7 days

→ Get started with Trace

Getting Started

- Explore and enable APIs
- Deploy a prebuilt solution
- Add dynamic logging to a running application
- Monitor errors with Error Reporting
- Deploy a Hello World app
- Take a VM quickstart
- Create a Cloud Storage bucket
- Create a Cloud Function
- Install the Cloud SDK

APIs

Requests (requests/sec)

No data is available for the selected time frame.

→ Go to APIs overview

Google Cloud Platform status

All services normal

→ Go to Cloud status dashboard

Billing

Estimated charges USD $0.00
For the billing period Oct 1 – 13, 2020

→ View detailed charges

Monitoring

Set up alerting policies

Create uptime checks

View all dashboards

→ Go to Monitoring

Error Reporting

No sign of any errors. Have you set up Error Reporting?

→ Learn how to set up Error Reporting

News

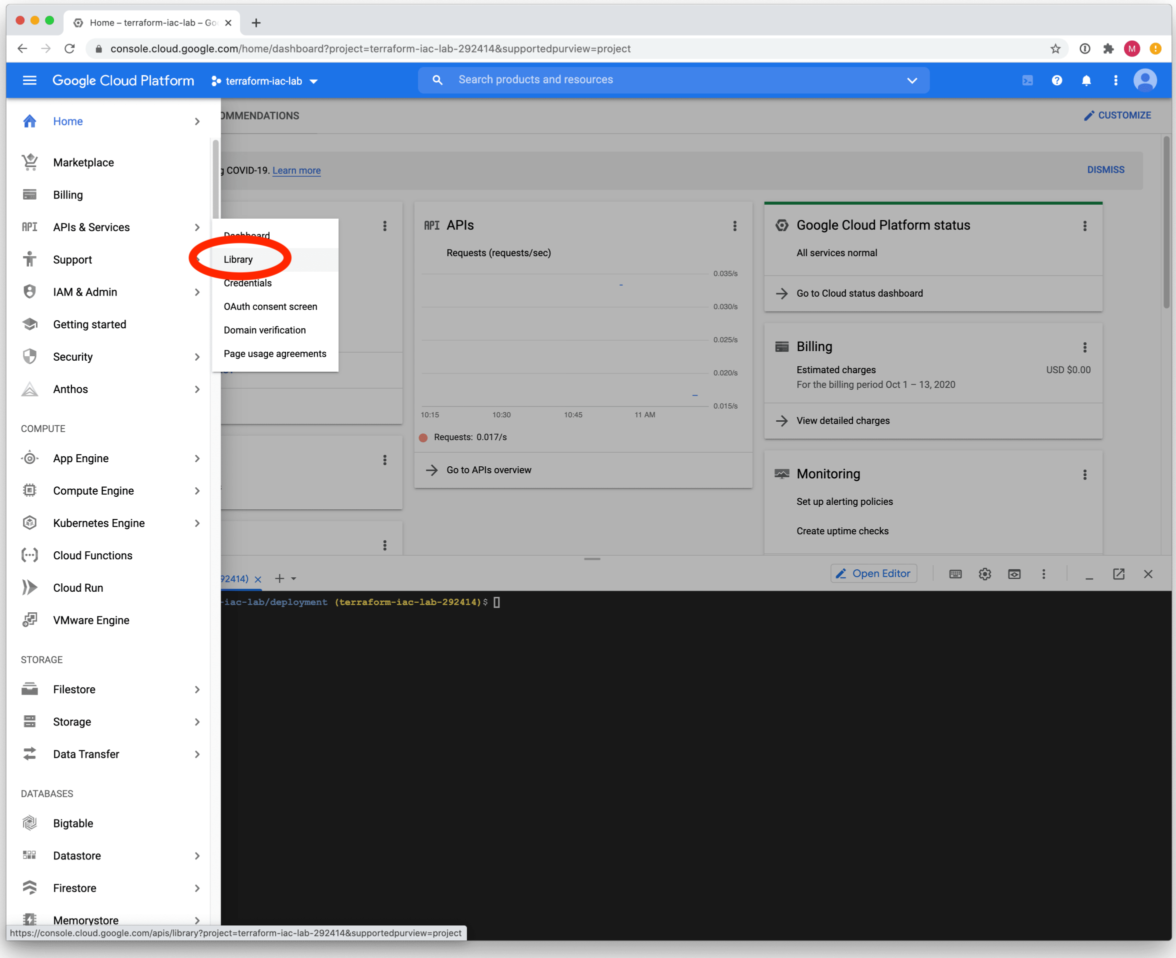
BigQuery explained: Blog series
16 hours ago

Rapid cloud foundation buildout and workload deployment using Terraform
22 hours ago

Helping media companies navigate the new streaming normal
22 hours ago

Enable Compute Engine API

Now, enable the Compute Engine API for this project. From the menu, select **APIs & Services > Library**.



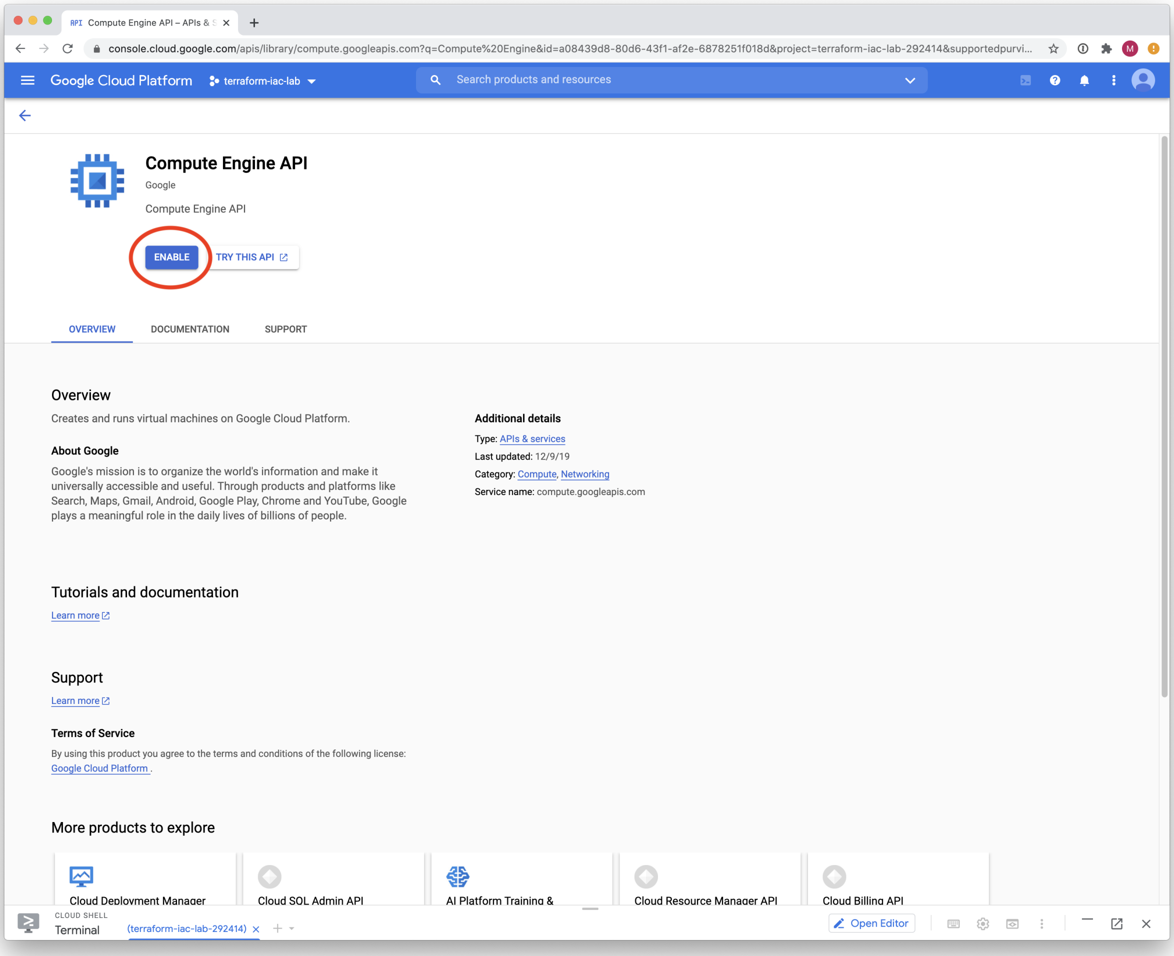
Search for *Compute Engine API*, then click on it.

Finally, click **Enable** to enable the API. It will take a few minutes to fully activate.

> **Warning:** Continue on with the lab in the *Launch the Cloud Shell* section.

3.3.3 Launch the Cloud Shell

In the upper right corner of the dashboard, click on the shell prompt icon labeled **Activate Cloud Shell** to launch a Cloud Shell instance. This is an on-demand Linux shell that contains many of the tools that will be used in this lab. The Cloud Shell instance is where you will be working throughout the lab.



Compute Engine API – APIs &
console.cloud.google.com/apis/library/compute.googleapis.com?q=Compute%20Engine&id=a08439d8-80d6-43f1-af2e-6878251f018d&project=terraform-iac-lab-292414&supportedpurvi...
Google Cloud Platform
terraform-iac-lab
Search products and resources

Compute Engine API

Google

Compute Engine API

ENABLE
TRY THIS API
OVERVIEW
DOCUMENTATION
SUPPORT

Overview

Creates and runs virtual machines on Google Cloud Platform.

About Google

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

Additional details

Type: APIs & services

Last updated: 12/9/19

Category: Compute, Networking

Service name: compute.googleapis.com

Tutorials and documentation

Learn more

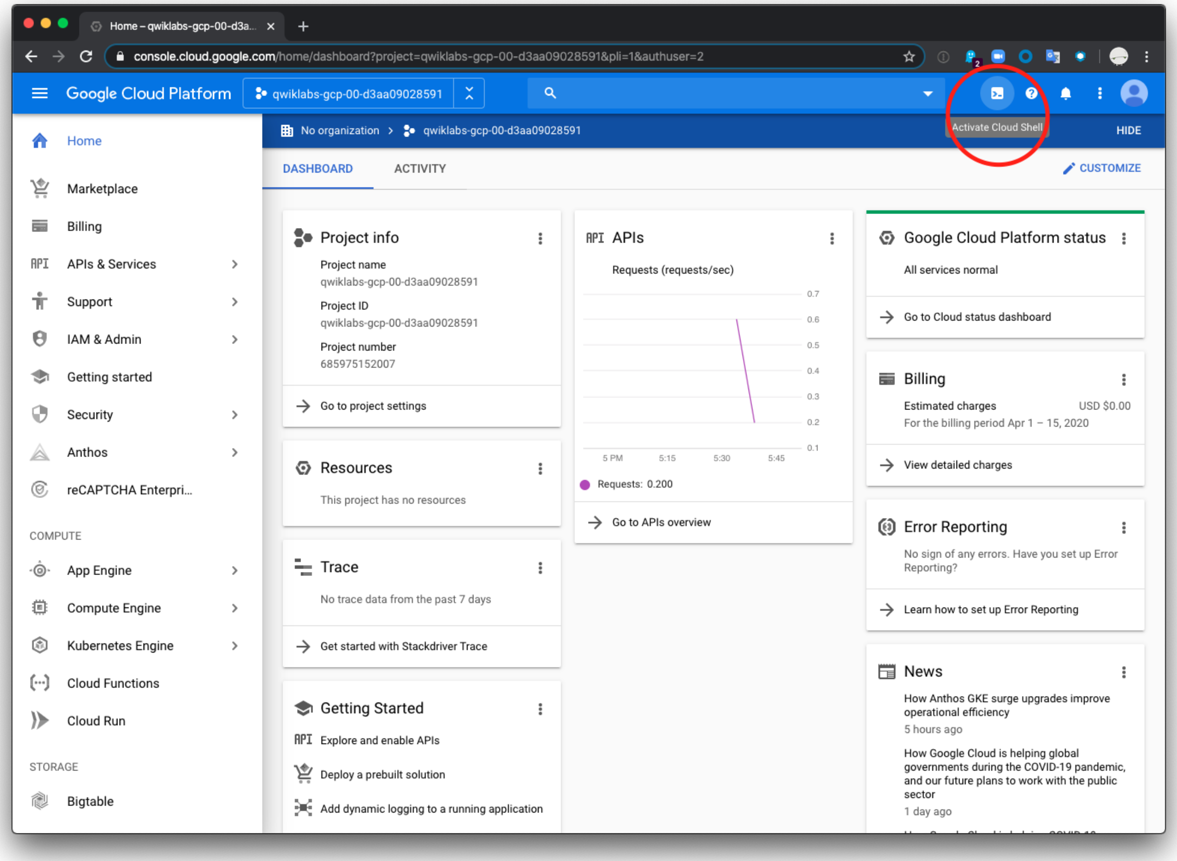
Support

Learn more

Terms of Service

By using this product you agree to the terms and conditions of the following license: Google Cloud Platform .

More products to explore


Home – qwiklabs-gcp-00-d3a...
console.cloud.google.com/home/dashboard?project=qwiklabs-gcp-00-d3aa09028591&pli=1&authuser=2
Google Cloud Platform
qwiklabs-gcp-00-d3aa09028591
Activate Cloud Shell
No organization
qwiklabs-gcp-00-d3aa09028591
HIDE
Home
Marketplace
Billing
APIs & Services
Support
IAM & Admin
Getting started
Security
Anthos
reCAPTCHA Enterpri...
COMPUTE
App Engine
Compute Engine
Kubernetes Engine
Cloud Functions
Cloud Run
STORAGE
Bigtable
DASHBOARD
ACTIVITY
CUSTOMIZE
Project info
Project name
qwiklabs-gcp-00-d3aa09028591
Project ID
qwiklabs-gcp-00-d3aa09028591
Project number
685975152007
Go to project settings
Resources
This project has no resources
Trace
No trace data from the past 7 days
Get started with Stackdriver Trace
Getting Started
Explore and enable APIs
Deploy a prebuilt solution
Add dynamic logging to a running application
APIs
Requests (requests/sec)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
5 PM
5:15
5:30
5:45
Requests: 0.200
Go to APIs overview
Google Cloud Platform status
All services normal
Go to Cloud status dashboard
Billing
Estimated charges
USD $0.00
For the billing period Apr 1 – 15, 2020
View detailed charges
Error Reporting
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting
News
How Anthos GKE surge upgrades improve operational efficiency
5 hours ago
How Google Cloud is helping global governments during the COVID-19 pandemic, and our future plans to work with the public sector
1 day ago

3.3.4 Clone the lab software repository

Once you are presented with the Cloud Shell prompt you will need to clone the GitHub repository used in this lab. This repository (or *repo*) contains the files needed to deploy the network and compute infrastructure we'll be working with.

```
$ git clone https://github.com/PaloAltoNetworks/terraform-iac-lab.git
```

3.4 Terraform Background

3.4.1 Terraform At a Glance

- Company: HashiCorp
- Integration First Available: January 2018
- Configuration: HCL (HashiCorp Configuration Language)
- PAN-OS Terraform Provider
- GitHub Repo
- Implementation Language: golang

3.4.2 Configuration Overview

Many Files, One Configuration

Terraform allows you to split your configuration into as many files as you wish. Any Terraform file in the current working directory will be loaded and concatenated with the others when you tell Terraform to apply your desired configuration.

State Management

Terraform saves details about a deployment it has done to a file referred to as a "state file". While this state file can be stored locally from the host on which a deployment has been executed, it is best practice to store the state file remotely so that collaborators can manage infrastructure from a common state file. Terraform Enterprise and Terraform Cloud both provide centralized state management, but it can also be done using cloud storage capabilities.

If changes are made to infrastructure that was deployed by Terraform, the state file may differ from what's actually deployed. This is actually not a big deal, as many of Terraform's commands do a Read operation to check the actual state against what's saved in the state file. Any changes that are found are then saved to the state file automatically.

Example Terraform Configuration

Here's an example of a Terraform configuration file. We will discuss the parts of this config below.

3.4.3 Terminology

Plan

A Terraform **plan** is the sum of all Terraform configuration files in a given directory. These files are generally written in *HCL*.

Provider

A **provider** can loosely thought of to be a product (such as the Palo Alto Networks firewall) or a service (such as AWS, Azure, or GCP). The provider understands the underlying API to the product or service, making individual parts of those things available as *resources*.

Most providers require some kind of configuration in order to use. For the `panos` provider, this is the authentication credentials of the firewall or Panorama that you want to configure.

Providers are configured in a provider configuration block (e.g. `- provider "panos" {...}`, and a plan can make use of any number of providers, all working together.

Resource

A **resource** is an individual component that a provider supports create/read/update/delete operations for.

For the Palo Alto Networks firewall, this would be something like an ethernet interface, service object, or an interface management profile.

Data Source

A **data source** is like a resource, but read-only.

For example, the `panos` provider has a data source that gives you access to the results of `show system info`.

Attribute

An **attribute** is a single parameter that exists in either a resource or a data source. Individual attributes are specific to the resource itself, as to what type it is, if it's required or optional, has a default value, or if changing it would require the whole resource to be recreated or not.

Attributes can have a few different types:

- *String*: `"foo"`, `"bar"`
- *Number*: `7`, `"42"` (quoting numbers is fine in HCL)
- *List*: `["item1", "item2"]`
- *Boolean**: `true`, `false`
- *Map*: `{"key": "value"}` (some maps may have more complex values)

Variables

Terraform plans can have *variables* to allow for more flexibility. These variables come in two flavors: user variables and attribute variables. Whenever you want to use variables (or any other Terraform interpolation), you'll be enclosing it in curly braces with a leading dollar sign: `"${...}"`

User variables are variables that are defined in the Terraform plan file with the `variable` keyword. These can be any of the types of values that attributes can be (default is string), and can also be configured to have default values. When using a user variable in your plan files, they are referenced with `var` as a prefix: `var.hostname`. Terraform looks for local variable values in the file `terraform.tfvars`.

Attribute variables are variables that reference other resources or data sources within the same plan. Specifying a resource attribute using an attribute variable creates an implicit dependency, covered below.

Dependencies

There are two ways to tell Terraform that resource "A" needs to be created before resource "B": the universal *depends_on* resource parameter or an attribute variable. The first way, using *depends_on*, is performed by adding the universal parameter "depends_on" within the dependent resource. The second way, using attribute variables, is performed by referencing a resource or data source attribute as a variable: `panos_management_profile.ssh.name`

Modules

Terraform can group resources together in reusable pieces called *modules*. Modules often have input variables to allow for customization, and output variable so that the resources they create can be accessed. This lab uses modules to group together elements for the base networking components, the firewall, and the created instances.

For example, the firewall configuration is located in `deployment/modules/firewall`. Calling this module creates the firewall instance, the network interfaces, and various other resources.

It can be used in another Terraform plan like this:

This calls the firewall module, and passes in values for the variables it requires.

3.4.4 Common Commands

The Terraform binary has many different CLI arguments that it supports. We'll discuss only a few of them here:

```
$ terraform init
```

`terraform init` initializes the current directory based off of the local plan files, downloading any missing provider binaries or modules.

```
$ terraform plan
```

`terraform plan` refreshes provider/resource states and reports what changes need to take place.

```
$ terraform apply
```

`terraform apply` refreshes provider/resource states and makes any needed changes to the resources.

```
$ terraform destroy
```

`terraform destroy` refreshes provider/resource states and removes all resources that Terraform created.

3.5 Panorama Configuration

In this activity you will:

- Initialize the Terraform provider
- Create the terraform.tfvars file
- Learn about the provided modules
- Assemble configuration/main.tf

For this portion of the lab, you will be using the Palo Alto Networks PAN-OS Terraform provider.

First, change to the Terraform configuration directory.

```
$ cd ~/terraform-iac-lab/configuration
```

3.5.1 Why Panorama?

In this lab we will be leveraging a Panorama instance to configure the VM-Series firewall we'll be deploying. There are other options for configuring the VM-Series firewall such as using Terraform to configure the firewall directly or, even simpler, bootstrapping the firewall with a full *bootstrap.xml* file. However, we can make our boostrap package simpler and reusable across multiple VM-Series instances by leveraging Panorama.

In this scenario we can provide VM-Series instance with the address of its Panorama console, the Device Group and Template Stack it will be a member of, and an authorization key to register the instance securely with Panorama. Once it registers with Panorama, the VM-Series instance will be mapped into the appropriate Device Group and Template Stack and the configurations contained therein will be pushed to the firewall.

The following is an example of an `init-cfg.txt` file used in the bootstrap process.

```
type=dhcp-client
ip-address=
default-gateway=
netmask=
ipv6-address=
ipv6-default-gateway=
hostname=Ca-FW-DC1
vm-auth-key=755036225328715
panorama-server=10.5.107.20
panorama-server-2=10.5.107.21
tplname=FINANCE_TG4
dgname=finance_dg
dns-primary=10.5.6.6
dns-secondary=10.5.6.7
op-command-modes=jumbo-frame, mgmt-interface-swap**
op-cmd-dpdk-pkt-io=***
dhcp-send-hostname=yes
dhcp-send-client-id=yes
dhcp-accept-server-hostname=yes
dhcp-accept-server-domain=yes
```

3.5.2 Provider Initialization

Your first task is to set up the communications between the provider and the provided Panorama instance. There's several ways this can be done. The IP address, username, and password (or API key) can be set as variables in

Terraform, and can be typed in manually each time the Terraform plan is run, or specified on the command line using the `-var` command line option to `terraform plan` and `terraform apply`. You can also reference a JSON file in the provider configuration which can contain the configuration.

Another way you can accomplish this is by using environment variables. Use the following commands to add the appropriate environment variables, substituting in the values provided by the instructor:

```
$ export PANOS_HOSTNAME="<YOUR PANORAMA MGMT IP GOES HERE>"
$ export PANOS_USERNAME="<YOUR STUDENT NAME>"
$ export PANOS_PASSWORD="Ignite2020!"
```

Note: Replace the text `<YOUR PANORAMA MGMT IP GOES HERE>` and `<YOUR STUDENT NAME>` with the values provided to you by the instructor. If you're doing this lab on your own, you'll need your own Panorama instance.

Now, you should see the variables exported in your shell, which you can verify using the `env | grep PANOS` command:

```
PANOS_HOSTNAME=panorama.panlabs.org
PANOS_USERNAME=studentXX
PANOS_PASSWORD=Ignite2020!
```

With these values defined, we can now initialize the Terraform panos provider with the following command.

```
$ terraform init
```

The provider is now ready to communicate with our Panorama instance.

3.5.3 Create configuration/terraform.tfvars

Our Terraform plan in this directory will create a device group, template, and template stack on our shared Panorama. So we don't overwrite the configuration of other students in the class, create a file called `terraform.tfvars` and define values for the device group, template name, and template stack name:

```
device_group    = "studentXX-dg"
template        = "studentXX-template"
stack           = "studentXX-stack"
```

Replace the strings `studentXX-dg`, `studentXX-template`, and `studentXX-stack` with the values provided by the instructor.

3.5.4 Learn about the provided modules

You have been provided with two Terraform modules in the `configuration/modules` directory that will build out our Panorama configuration. Here's a snippet of the contents of main.tf in the `configuration/modules/network` directory:

Terraform will use this configuration to build out the contents of the template and template stack specified by the `template` and `stack` variables.

The `network` module also specifies some outputs that can be fed to other modules in the configuration:

The module to populate the device group works in a similar fashion.

3.5.5 Assemble configuration/main.tf

Add the following to `configuration/main.tf` to build out the template and template stack on our Panorama instance:

Now run `terraform init` (you need to run `init` each time you add a new module) and `terraform plan`. You will see the Terraform provider determine what changes need to be made, and output all the changes that will be made to the configuration. If you run `terraform apply`, those changes will be added to the candidate configuration, but not committed (*why?*).

Add the next section to `configuration/main.tf` to build out the device group:

This module has variables for the names of zones and interfaces to avoid hard coding values. Our networking module outputs those names from what it creates, so we can chain these two modules together.

You can run `terraform init`, `terraform plan`, and `terraform apply` to populate the device group on Panorama.

Since Terraform is unable to commit configuration to PAN-OS on it's own, we have provided a Golang helper program to commit your user's changes to Panorama. You use a null resource provisioner in your main.tf to have Terraform run the program for you.

Add the following section to `configuration/main.tf` to issue the commit:

Your completed `configuration/main.tf` should look like this:

Now, run `terraform init` and `terraform apply` to finalize the changes. Log in to the Panorama web UI and verify that your changes have been committed. You're now ready to deploy the environment and have your firewall bootstrap from this configuration.

3.6 Lab Deployment

In this activity you will:

- Create a service account credential file
- Create an SSH key-pair
- Create the terraform.tfvars file
- Add the bootstrap module
- Add the firewall module
- Deploy the infrastructure

3.6.1 Introduction

The lab infrastructure will also be deployed in GCP using Terraform. A Terraform plan has been provided that will initialize the GCP provider, and call most of the modules responsible for creating the network, compute, and storage resources needed. You will add the modules for creating the bootstrap configuration as well as the VM-Series firewall.

First, change to the Terraform deployment directory:

```
$ cd ~/terraform-iac-lab/deployment
```

3.6.2 Create a service account credential file

Terraform will need to authenticate to GCP to perform the deployment. We *could* authenticate to GCP using the username presented in the Qwiklabs panel when the lab was started. However, the Compute Engine default service account is typically used because it is certain to have all the neccesary permissions.

List the email address of the Compute Engine default service account.

```
$ gcloud iam service-accounts list
```

Use the following `gcloud` command to download the credentials for the **Compute Engine default service account** using its associated email address (displayed in the output of the previous command).

```
$ gcloud iam service-accounts keys create ~/gcp_compute_key.json --iam-account <EMAIL_
↪ADDRESS>
```

Verify the JSON credentials file was successfully created.

```
$ cat ~/gcp_compute_key.json
```

3.6.3 Create an SSH key-pair

All Compute Engine instances are required to have an SSH key-pair defined when the instance is created. This is done to ensure secure access to the instance will be available once it is created.

Create an SSH key-pair with an empty passphrase and save them in the `~/.ssh` directory.

```
$ ssh-keygen -t rsa -b 1024 -N '' -f ~/.ssh/lab_ssh_key
```

Note: GCP has the ability to manage all of its own SSH keys and propagate them automatically to projects and instances. However, the VM-Series is only able to make use of a single SSH key. Rather than leverage GCP's SSH key management process, we've created our own SSH key and configured Compute Engine to use our key exclusively.

3.6.4 Create deployment/terraform.tfvars

In this directory you will find the three main files associated with a Terraform plan: `main.tf`, `variables.tf`, and `outputs.tf`. View the contents of these files to see what they contain and how they're structured.

```
$ more main.tf
$ more variables.tf
$ more outputs.tf
```

The file `main.tf` defines the providers that will be used and the resources that will be created (more on that shortly). Since it is poor practice to hard code values into the plan, the file `variables.tf` will be used to declare the variables that will be used in the plan (but not necessarily their values). The `outputs.tf` file will define the values to display that result from applying the plan.

Create a file called `terraform.tfvars` in the current directory that contains the following variables and their values. You will need to add a number of things:

GCP configuration:

project: The GCP project ID to use.

region: The GCP region we are using.

zone: The GCP zone we are using.

Authentication information

credentials_file: Path to the JSON credentials file.

public_key_file: Path to the SSH public key file.

Firewall information:

fw_name: The name for the firewall.

Panorama bootstrap information:

panorama: The hostname/IP address of Panorama.

tplname: The template stack created in the previous section (replace XX with your student number).

dgname: The device group created in the previous section (replace XX with your student number).

vm_auth_key: The VM auth key for Panorama.

Your file should look similar to the following, with the appropriate values replaced:

Note: If you're running this lab on your own, replace these values appropriately. See the Panorama documentation for generating the VM auth key.

3.6.5 Add the bootstrap module

Add the following module definition to `deployment/main.tf`:

This uses a module that has been published to the Terraform module registry for public use. (If you'd like to review the code, it's on the PaloAltoNetworks GitHub page.) This will create the Google Storage bucket for holding a PAN-OS bootstrap configuration, as well as the required files.

3.6.6 Add the firewall module

Now we need to add another module definition to `deployment/main.tf` to specify the firewall configuration:

This module creates the VM-Series instance. Notice how the outputs from the *bootstrap* and *vpc* modules are used as inputs to this one.

3.6.7 Deploy the infrastructure

Your completed `deployment/main.tf` file should look like this:

Now, you're ready to deploy the infrastructure. Run the following commands:

```
$ terraform init
$ terraform plan
$ terraform apply
```

As we saw before, `terraform init` will install all required providers and modules, `terraform plan` will show all the infrastructure that will be created, and `terraform apply` will create the infrastructure.

At a high level, the completed Terraform configuration will:

1. **Run the `bootstrap` module**
    1. Create a GCP storage bucket for the firewall bootstrap package
    2. Apply a policy to the bucket allowing read access to `allUsers`
    3. Create the `/config/init-cfg.txt`, `/config/bootstrap.xml`, `/software`, `/content`, and `/license` objects in the bootstrap bucket
2. **Run the `vpc` module**
    1. Create the VPC
    2. Create the Internet gateway
    3. Create the `management`, `untrust`, `web`, and `database` subnets
    4. Create the security groups for each subnet
    5. Create the default route for the `web` and `database` subnets
3. **Run the `firewall` module**
    1. Create the VM-Series firewall instance
    2. Create the VM-Series firewall interfaces
    3. Create the public IPs for the `management` and `untrust` interfaces
4. **Run the `web` module**
    1. Create the web server instance
    2. Create the web server interface
5. **Run the `database` module**
    1. Create the database server instance
    2. Create the database server interface

The deployment process should finish in a few minutes and you will be presented with the public IP addresses of the VM-Series firewall management and untrust interfaces. However, the VM-Series firewall can take up to *ten minutes* to complete the initial bootstrap process.

Once the firewall has completed the bootstrap process, it should be listed in Panorama as a managed device in your device group under Panorama > Managed Devices > Summary.

3.7 Validation Testing

In this activity you will:

- Access the Apache web server
- Access the WordPress application
- Post a blog article
- Verify firewall rule matches

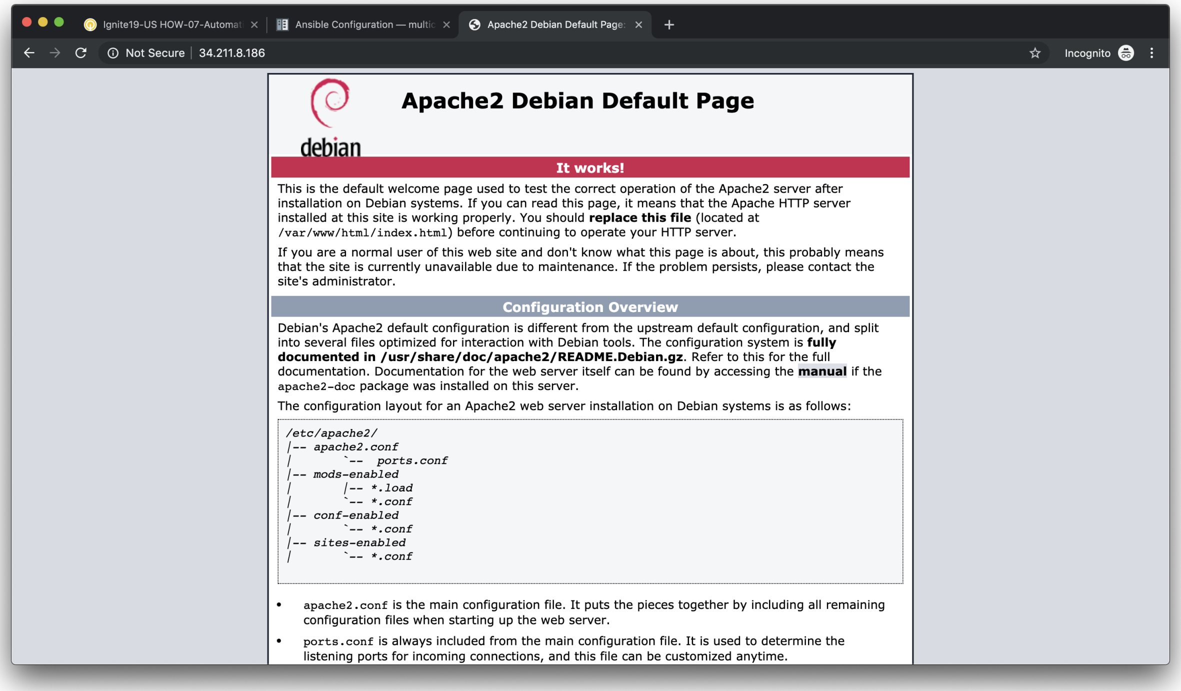
The previous two activities had you deploy and configure the infrastructure supporting our WordPress application. Now it's time to see if everything works as planned. If so, you should be able to access the application, post a blog article, and verify that the appropriate firewall rules are being hit. If not, you will need to troubleshoot your configs and make the necessary corrections.

3.7.1 Access the Apache web server

The web server is using the firewall's untrust interface address in a destination NAT rule. Run the following commands to determine the IP address of this interface.

```
$ cd ~/terraform-iac-lab/deployment
$ terraform output
```

Open a new tab in your web browser and go to `http://<web-server-ip-address>`. You should see the Apache default home page.


Ignite19-US HOW-07-Automat
Ansible Configuration — multic
Apache2 Debian Default Page:
Not Secure
34.211.8.186
Incognito
debian

Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.

3.7.2 Access the WordPress application

Append `/wordpress` to the end of the web server URL and the WordPress installation page should be displayed.

Fill in values of your choosing for the **Site Name**, **Username**, and **Your Email**. These are only for testing and do not need to be real values.

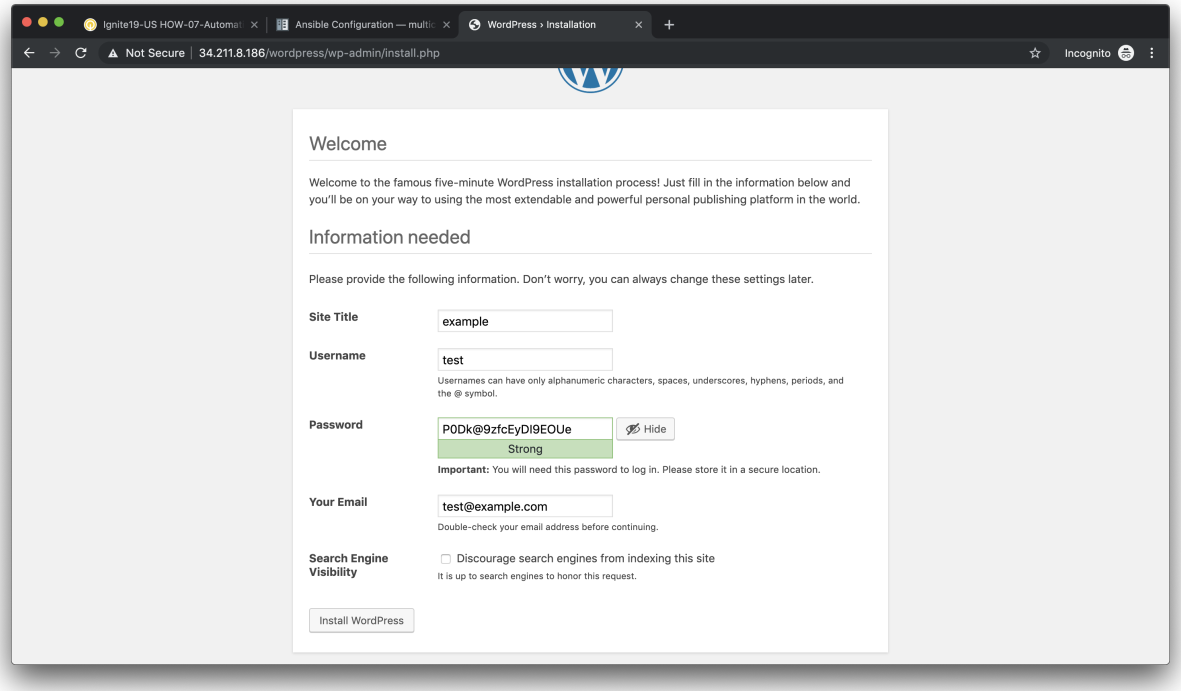
Note: Make sure you copy the password that is provided to your clipboard. Otherwise you may not be able to log in once WordPress is installed.

Click **Install WordPress** when you are done.

On the following page, click on **Log In** to log into the WordPress administrator dashboard.

Log into WordPress using the username and password you created.

You will then be presented with the WordPress administrator dashboard.


Ignite19-US HOW-07-Automat
Ansible Configuration — multic
WordPress › Installation
Not Secure
34.211.8.186/wordpress/wp-admin/install.php
Incognito

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title: example

Username: test

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password: P0Dk@9zfcEyDI9EOUe

Strong

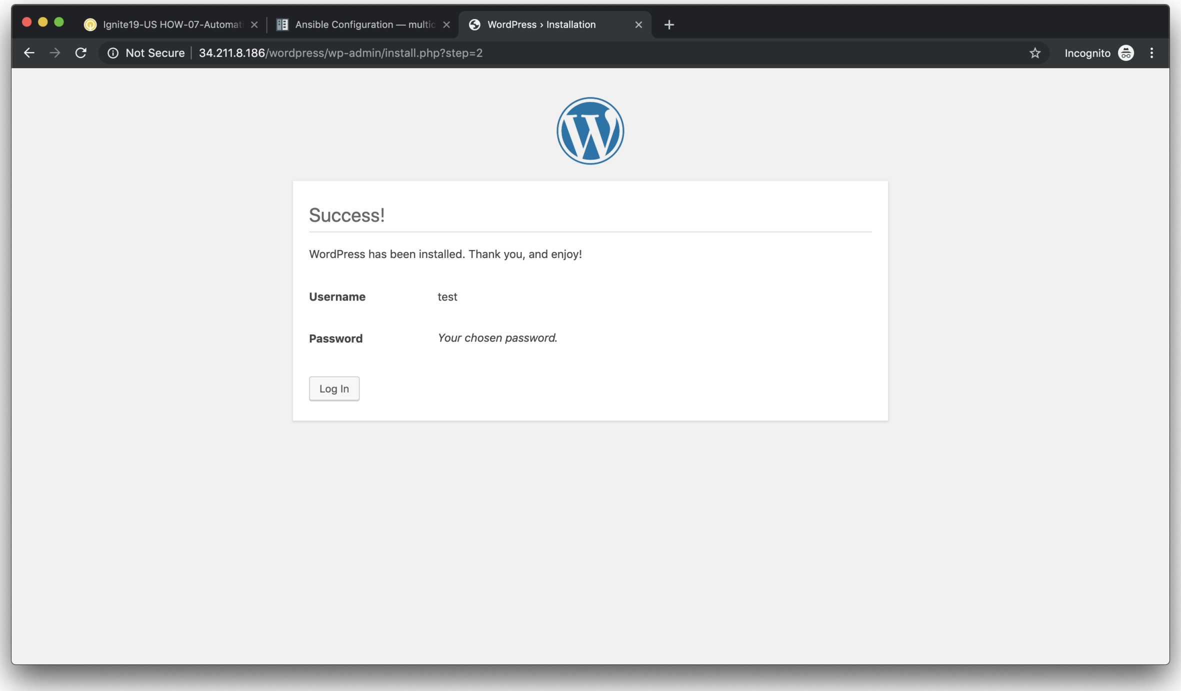
Important: You will need this password to log in. Please store it in a secure location.

Your Email: test@example.com

Double-check your email address before continuing.

Search Engine Visibility: □ Discourage search engines from indexing this site

It is up to search engines to honor this request.

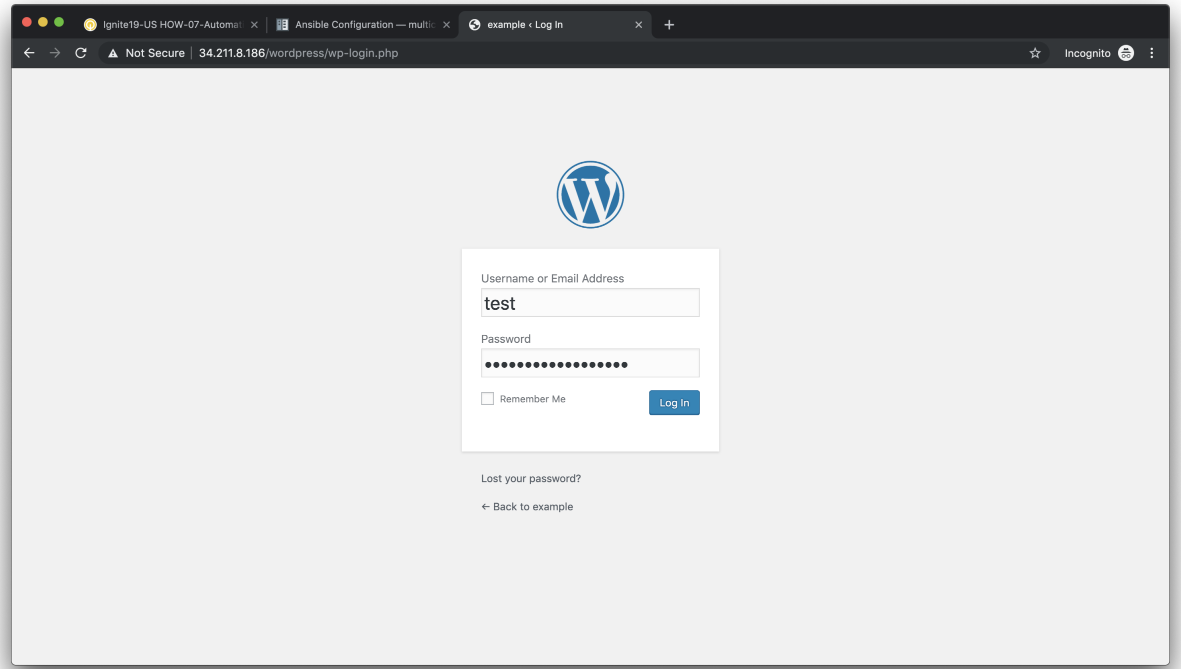

Ignite19-US HOW-07-Automat
Ansible Configuration — multic
WordPress › Installation
Not Secure
34.211.8.186/wordpress/wp-admin/install.php?step=2
Incognito

Success!

WordPress has been installed. Thank you, and enjoy!

Username: test

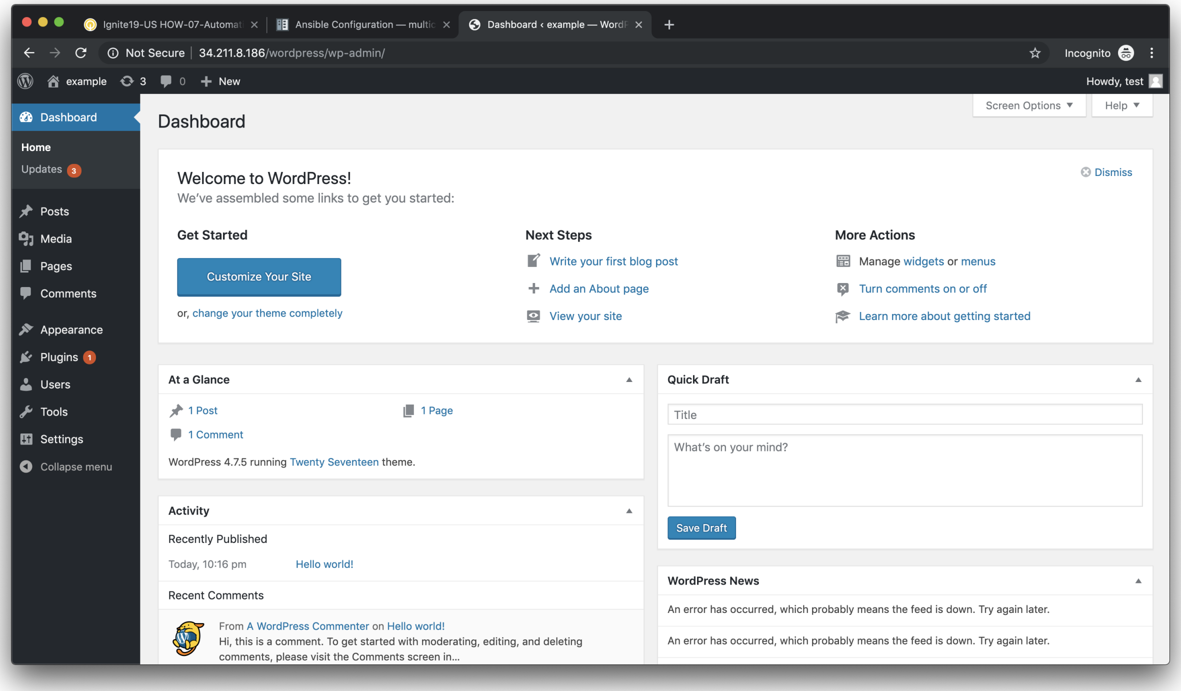
Password: *Your chosen password.*



34.211.8.186/wordpress/wp-login.php

Username or Email Address: test

Password: ••••••••••••••••••



34.211.8.186/wordpress/wp-admin/

Dashboard

Welcome to WordPress!
We've assembled some links to get you started:

Get Started
or, change your theme completely

Next Steps
Write your first blog post
Add an About page
View your site

More Actions
Manage widgets or menus
Turn comments on or off
Learn more about getting started

1 Post
1 Page
1 Comment
WordPress 4.7.5 running Twenty Seventeen theme.

Recently Published
Today, 10:16 pm Hello world!

Recent Comments
From A WordPress Commenter on Hello world!
Hi, this is a comment. To get started with moderating, editing, and deleting comments, please visit the Comments screen in…

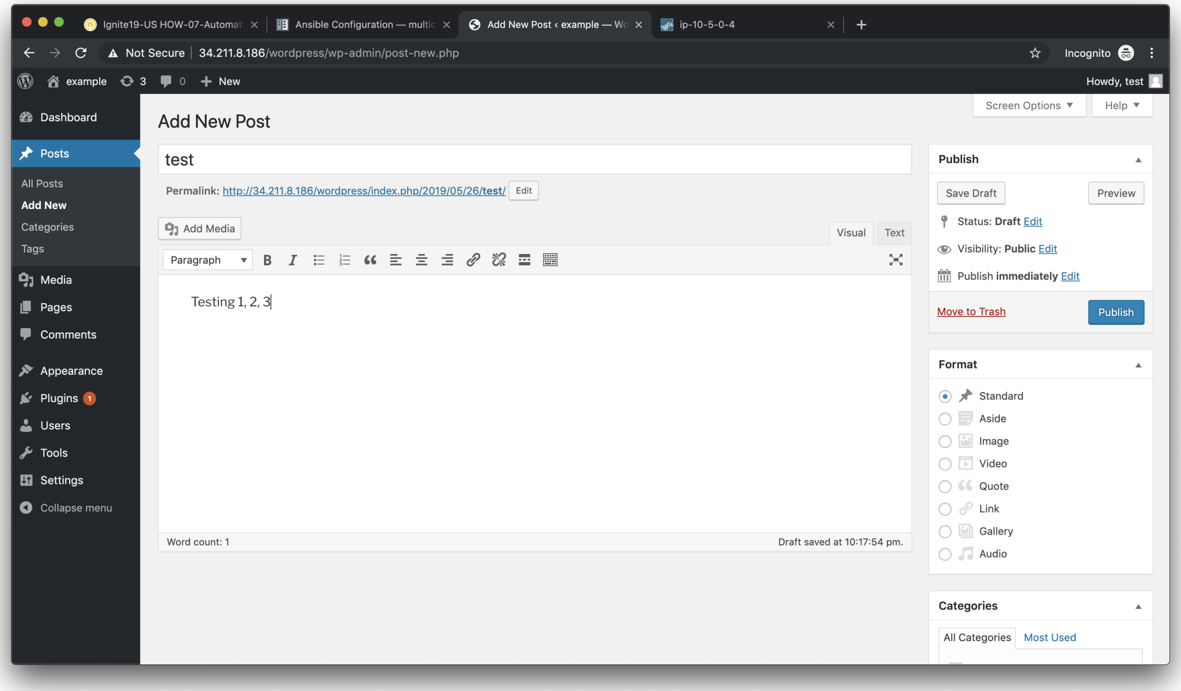
An error has occurred, which probably means the feed is down. Try again later.

An error has occurred, which probably means the feed is down. Try again later.

3.7.3 Post a blog article

Now that you've successfully logged into the WordPress administrator dashboard, let's post an update to the blog.

Click on **Write your first blog post** under the **Next Steps** section. You will be presented with the **Add New Post** editor.



34.211.8.186/wordpress/wp-admin/post-new.php

Add New Post

test

Permalink: http://34.211.8.186/wordpress/index.php/2019/05/26/test/

Testing 1, 2, 3

Word count: 1

Draft saved at 10:17:54 pm.

Status: Draft

Visibility: Public

Publish immediately

Enter a title for your post and some sample content. Then click on **Publish** to post the update.

You can then click on **Preview** to see the published blog update.

3.7.4 Verify firewall rule matches

Now that we've confirmed the WordPress application is working properly, let's see what is happening with our firewall rules.

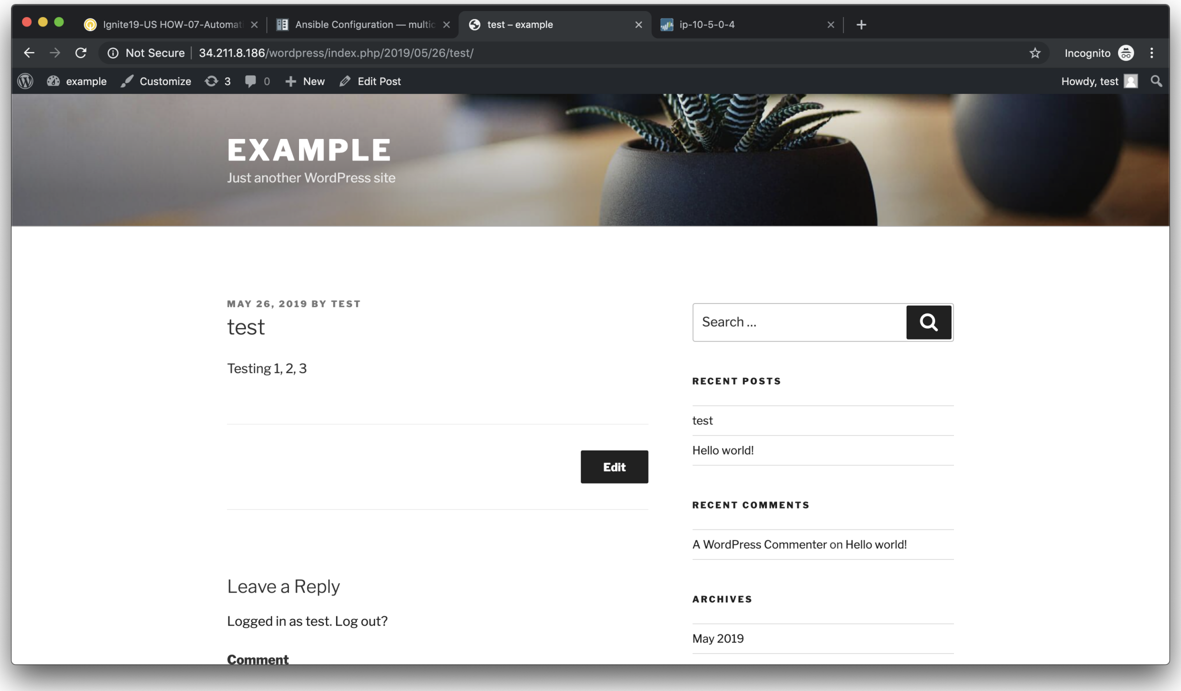
Log into the firewall administrator web interface at `https://<firewall-management-ip>` using these credentials:

- Username: `admin`
- Password: `Ignite2020!`

Navigate to **Policies > Security**, and scroll to the right. You will see details on the security rules that are being hit.

Scroll back to the left, find the security rule entitled *Allow web inbound*. Then click on the drop-down menu icon to the right of the rule name and select *Log Viewer*.

You will see all of the logs associated with inbound web traffic. Notice the applications identified are *web-browsing* and *blog-posting*.


Ignite19-US HOW-07-Automat
Ansible Configuration — multic
test – example
ip-10-5-0-4
Not Secure | 34.211.8.186/wordpress/index.php/2019/05/26/test/
Incognito
example
Customize
3
0
New
Edit Post
Howdy, test

EXAMPLE

Just another WordPress site

MAY 26, 2019 BY TEST

test

Testing 1, 2, 3

Leave a Reply

Logged in as test. Log out?

Comment

RECENT POSTS

test

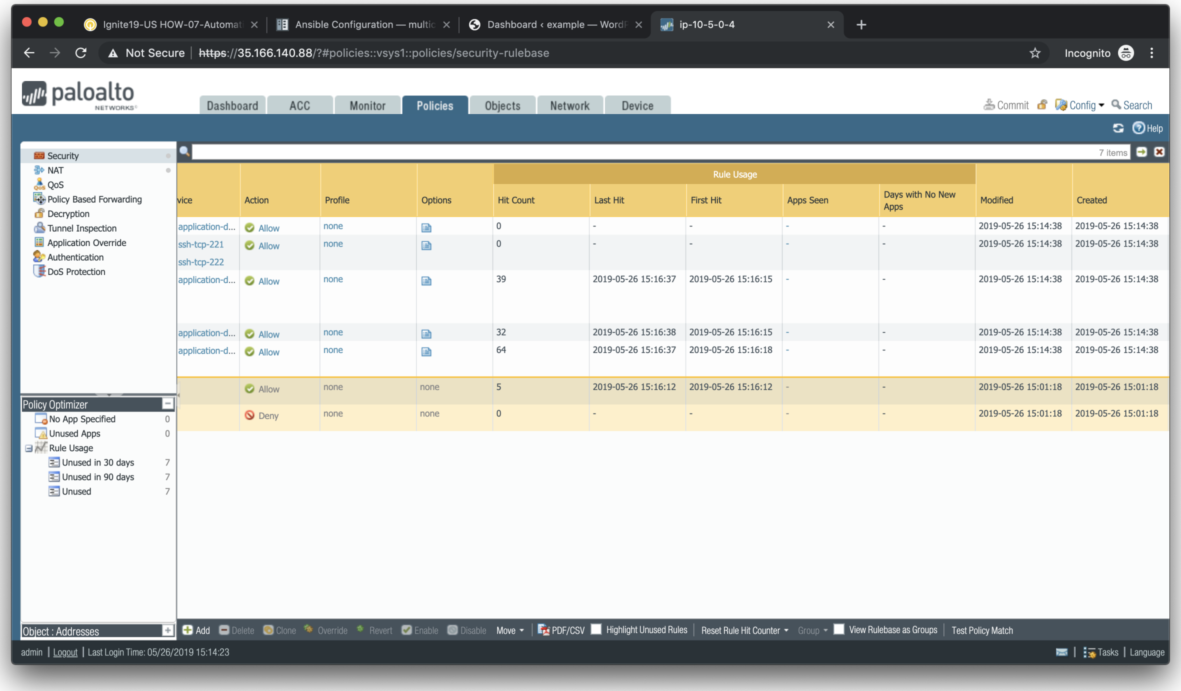
Hello world!

RECENT COMMENTS

A WordPress Commenter on Hello world!

ARCHIVES

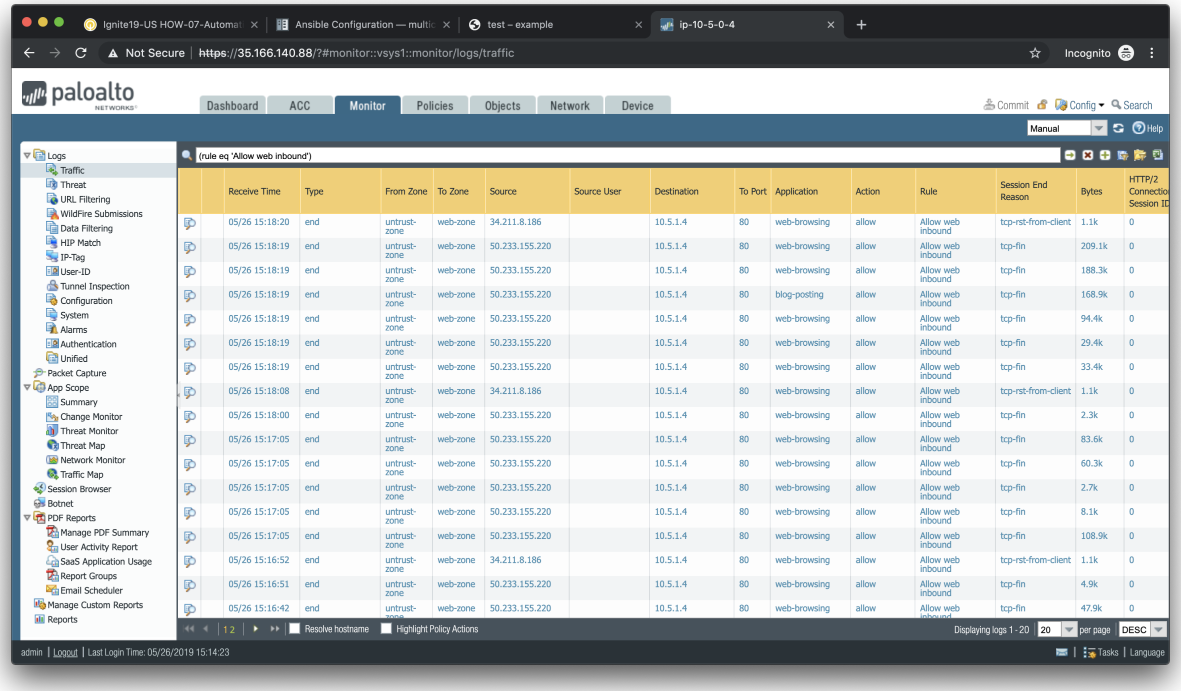
May 2019



				Rule Usage						
vice	Action	Profile	Options	Hit Count	Last Hit	First Hit	Apps Seen	Days with No New Apps	Modified	Created
application-d...	Allow	none	▤	0	-	-	-	-	2019-05-26 15:14:38	2019-05-26 15:14:38
ssh-tcp-221 ssh-tcp-222	Allow	none	▤	0	-	-	-	-	2019-05-26 15:14:38	2019-05-26 15:14:38
application-d...	Allow	none	▤	39	2019-05-26 15:16:37	2019-05-26 15:16:15	-	-	2019-05-26 15:14:38	2019-05-26 15:14:38
application-d...	Allow	none	▤	32	2019-05-26 15:16:38	2019-05-26 15:16:15	-	-	2019-05-26 15:14:38	2019-05-26 15:14:38
application-d...	Allow	none	▤	64	2019-05-26 15:16:37	2019-05-26 15:16:18	-	-	2019-05-26 15:14:38	2019-05-26 15:14:38
	Allow	none	none	5	2019-05-26 15:16:12	2019-05-26 15:16:12	-	-	2019-05-26 15:01:18	2019-05-26 15:01:18
	Deny	none	none	0	-	-	-	-	2019-05-26 15:01:18	2019-05-26 15:01:18

Policy Optimizer	
No App Specified	0
Unused Apps	0
Rule Usage	
Unused in 30 days	7
Unused in 90 days	7
Unused	7

Object : Addresses



(rule eq 'Allow web inbound')

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes	HTTP/2 Connection Session ID
05/26 15:18:20	end	untrust-zone	web-zone	34.211.8.186		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-rst-from-client	1.1k	0
05/26 15:18:19	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	209.1k	0
05/26 15:18:19	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	188.3k	0
05/26 15:18:19	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	blog-posting	allow	Allow web inbound	tcp-fin	168.9k	0
05/26 15:18:19	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	94.4k	0
05/26 15:18:19	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	29.4k	0
05/26 15:18:19	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	33.4k	0
05/26 15:18:08	end	untrust-zone	web-zone	34.211.8.186		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-rst-from-client	1.1k	0
05/26 15:18:00	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	2.3k	0
05/26 15:17:05	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	83.6k	0
05/26 15:17:05	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	60.3k	0
05/26 15:17:05	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	2.7k	0
05/26 15:17:05	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	8.1k	0
05/26 15:17:05	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	108.9k	0
05/26 15:16:52	end	untrust-zone	web-zone	34.211.8.186		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-rst-from-client	1.1k	0
05/26 15:16:51	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	4.9k	0
05/26 15:16:42	end	untrust-zone	web-zone	50.233.155.220		10.5.1.4	80	web-browsing	allow	Allow web inbound	tcp-fin	47.9k	0

Displaying logs 1 - 20 | 20 per page

Note: You may find source IPs other than your own as the web server is open to the public and will likely be discovered by web crawlers and other discovery tools aimed at public cloud providers.

Navigate back to **Policies > Security** and click on the **Log Viewer** for the *Allow web to db* rule.

You will see all of the MySQL (actually MariaDB) database traffic between the WordPress web server and the database backend.

3.8 Summary

Congratulations! You have completed the hands-on Infrastructure as Code lab.

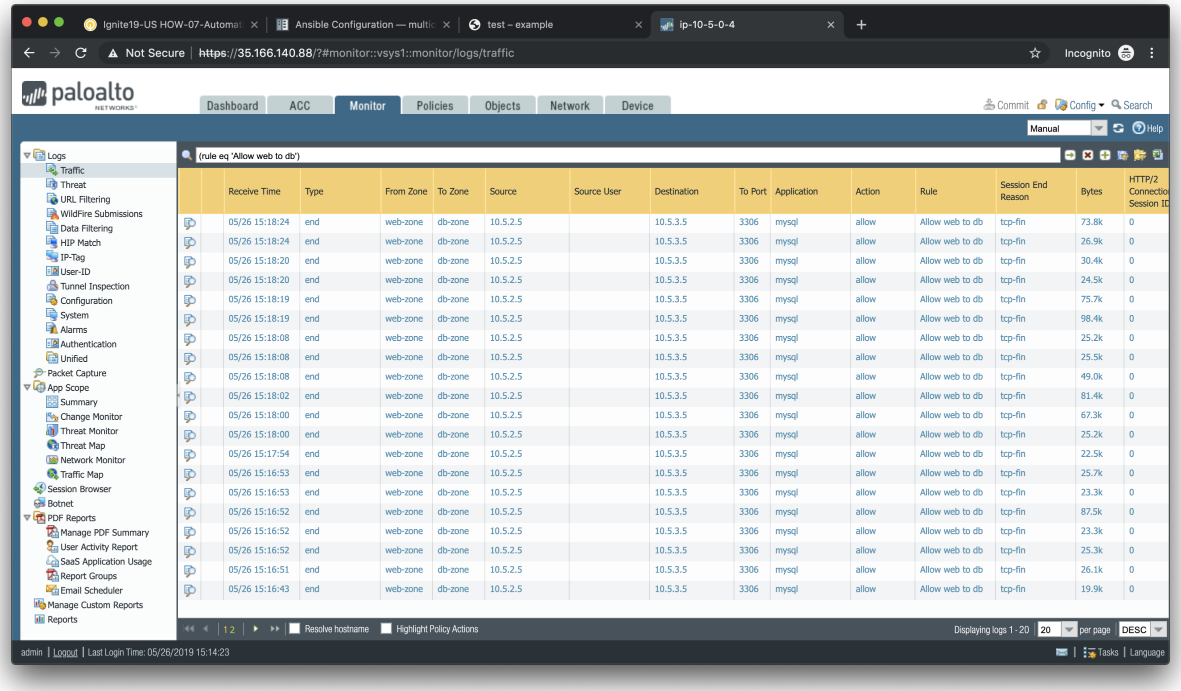
3.8.1 What We've Accomplished

We've completed the following activities:

- **Configure:** We used Terraform to prepare Panorama for bootstrapping by configuring a device group, template, and template stack.
- **Deployment:** We built out our application in GCP using Terraform, and used the bootstrap capability of VM-Series to automatically retrieve the desired configuration from Panorama.

3.9 Cleaning Up

In this activity you will:



(rule eq 'Allow web to db')

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes	HTTP/2 Connectio Session ID
05/26 15:18:24	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	73.8k	0
05/26 15:18:24	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	26.9k	0
05/26 15:18:20	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	30.4k	0
05/26 15:18:20	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	24.5k	0
05/26 15:18:19	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	75.7k	0
05/26 15:18:19	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	98.4k	0
05/26 15:18:08	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	25.2k	0
05/26 15:18:08	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	25.5k	0
05/26 15:18:08	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	49.0k	0
05/26 15:18:02	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	81.4k	0
05/26 15:18:00	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	67.3k	0
05/26 15:18:00	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	25.2k	0
05/26 15:17:54	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	22.5k	0
05/26 15:16:53	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	25.7k	0
05/26 15:16:53	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	23.3k	0
05/26 15:16:52	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	87.5k	0
05/26 15:16:52	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	23.3k	0
05/26 15:16:52	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	25.3k	0
05/26 15:16:51	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	26.1k	0
05/26 15:16:43	end	web-zone	db-zone	10.5.2.5		10.5.3.5	3306	mysql	allow	Allow web to db	tcp-fin	19.9k	0

Displaying logs 1 - 20 | 20 per page

- Destroy the lab deployment

3.9.1 Destroy the lab deployment

When deploying infrastructure in the public cloud it is important to tear it down when you are done, otherwise you will end up paying for services that are no longer needed. We'll need to go back to the deployment directory and use Terraform to destroy the infrastructure we deployed at the start of the lab.

Change into the `deployment` directory, and tell Terraform to destroy all of the resources it has created by running `terraform destroy`.

```
$ cd ~/terraform-iac-lab/deployment
$ terraform destroy
```

> **Warning:** If you are doing the lab on your own, this is a critical step to stop being charged by GCP.
>
> Qwiklabs will take care of destroying everything when the lab expires, but destroying cloud resources when they are no longer needed is a good habit to get in to.

3.10 Further Reading

3.10.1 Terraform

- Terraform Documentation
- Terraform panos Provider

- Terraform: Up & Running

3.11 Terraform and Commits

One thing to know when working with Terraform is that it does not have support for committing your configuration. To commit your configuration, you can use the following Golang code.

```
package main

import (
    "encoding/json"
    "flag"
    "log"
    "os"

    "github.com/PaloAltoNetworks/pango"
)

type Credentials struct {
    Hostname string `json:"hostname"`
    Username string `json:"username"`
    Password string `json:"password"`
    ApiKey string `json:"api_key"`
    Protocol string `json:"protocol"`
    Port uint `json:"port"`
    Timeout int `json:"timeout"`
}

func getCredentials(configFile, hostname, username, password, apiKey string)␣
↪(Credentials) {
    var (
        config Credentials
        val string
        ok bool
    )

    // Auth from the config file.
    if configFile != "" {
        fd, err := os.Open(configFile)
        if err != nil {
            log.Fatalf("ERROR: %s", err)
        }
        defer fd.Close()

        dec := json.NewDecoder(fd)
        err = dec.Decode(&config)
        if err != nil {
            log.Fatalf("ERROR: %s", err)
        }
    }

    // Auth from env variables.
    if val, ok = os.LookupEnv("PANOS_HOSTNAME"); ok {
        config.Hostname = val
    }
    if val, ok = os.LookupEnv("PANOS_USERNAME"); ok {
```

(continues on next page)

(continued from previous page)

```
        config.Username = val
    }
    if val, ok = os.LookupEnv("PANOS_PASSWORD"); ok {
        config.Password = val
    }
    if val, ok = os.LookupEnv("PANOS_API_KEY"); ok {
        config.ApiKey = val
    }

    // Auth from CLI args.
    if hostname != "" {
        config.Hostname = hostname
    }
    if username != "" {
        config.Username = username
    }
    if password != "" {
        config.Password = password
    }
    if apiKey != "" {
        config.ApiKey = apiKey
    }

    if config.Hostname == "" {
        log.Fatalf("ERROR: No hostname specified")
    } else if config.Username == "" && config.ApiKey == "" {
        log.Fatalf("ERROR: No username specified")
    } else if config.Password == "" && config.ApiKey == "" {
        log.Fatalf("ERROR: No password specified")
    }

    return config
}

func main() {
    var (
        err error
        configFile, hostname, username, password, apiKey string
        job uint
    )

    log.SetFlags(log.Ldate | log.Ltime | log.Lmicroseconds)

    flag.StringVar(&configFile, "config", "", "JSON config file with panos connection␣
↪info")
    flag.StringVar(&hostname, "host", "", "PAN-OS hostname")
    flag.StringVar(&username, "user", "", "PAN-OS username")
    flag.StringVar(&password, "pass", "", "PAN-OS password")
    flag.StringVar(&apiKey, "key", "", "PAN-OS API key")
    flag.Parse()

    config := getCredentials(configFile, hostname, username, password, apiKey)

    fw := &pango.Firewall{Client: pango.Client{
        Hostname: config.Hostname,
        Username: config.Username,
        Password: config.Password,
```

(continues on next page)

(continued from previous page)

```
        ApiKey: config.ApiKey,
        Protocol: config.Protocol,
        Port: config.Port,
        Timeout: config.Timeout,
        Logging: pango.LogOp | pango.LogAction,
    }}
    if err = fw.Initialize(); err != nil {
        log.Fatalf("Failed: %s", err)
    }

    job, err = fw.Commit(flag.Arg(0), true, true, false, true)
    if err != nil {
        log.Fatalf("Error in commit: %s", err)
    } else if job == 0 {
        log.Printf("No commit needed")
    } else {
        log.Printf("Committed config successfully")
    }
}
```

This code reads the hostname, username, and password from the environment variables we set earlier.

You will need to do the following to compile and run this code:

1. Open a text editor, add the code above to it and save the file as `commit.go`.
2. Install the Go libraries for PAN-OS.

```
$ go get github.com/PaloAltoNetworks/pango
```

3. Compile the source code.

```
$ go build commit.go
```

4. Run the executable (using your existing environment variables).

```
$ ./commit <optional commit comment>
```