

GCP-Kubernetes-Lab/Demo



Securing your container workloads in Kubernetes

<http://www.paloaltonetworks.com>

© 2018 Palo Alto Networks. Proprietary and Confidential

Table of Contents

About GCP Terraform Template.....	3
Support Policy.....	4
Instances Used	4
Prerequisites.....	4
GCP Project Setup	4
Task 1 – Create a Project	4
Task 2 – Enable the API	6
Terraform File setup.....	7
Task 1 - Download Terraform files	7
Task 2 - Gather Information and Update the Template File.....	10
Deploy the Terraform Template.....	15
Task 1 – Connect to GCP	15
Task 2 – Deploy the Terraform Template.....	16
Review what was deployed.....	18
Task 1 – Look around GCP console	18
Task 2 – Review the Kubernetes Cluster.....	21
Task 2 – Explore the cluster.....	21
Task 3 – Login into the firewall.....	25
Launch a two tiered application.....	31
Task 1 – Application Deployment YAML file	31
Task 2 – Launch the Application	36
Task 3 – Explore what was just deployed	36
Securing Inbound Traffic.....	41
Task 1 – Note the Internal Load Balancer’s IP Address	41
Task 2 – Update the Firewall’s NAT Policy	41
Task 3 – Connect to the Guestbook Frontend.....	42
Securing Outbound Traffic.....	44
Task 1 – Add Outbound Route.....	44
Task 2 – Add Kube-Apiserver route	46
Inspection of Inter-Pod traffic.....	48

Task 1 – Deploy a DaemonSet	49
Task 2 – Verify firewall logs to see inter-pod traffic	52
Conclusion	53

About GCP Terraform Template

GCP Terraform Templates are files that can deploy, configure, and launch GCP resources such as VPC networks, subnets, security groups, firewall rules, route tables, Kubernetes clusters, and more. These templates are used for ease of deployment and are key to any cloud deployment model.

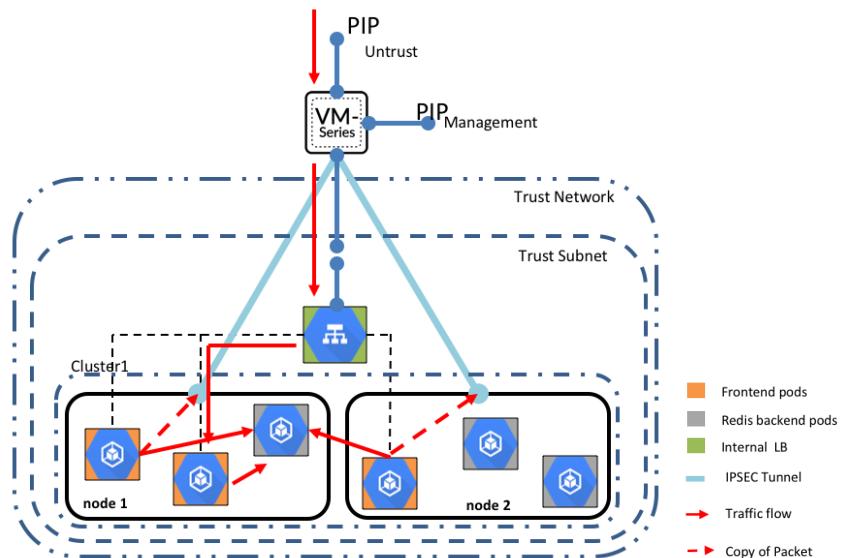
For more information on Templates refer to Google's documentation

<https://cloud.google.com/community/tutorials/managing-gcp-projects-with-terraform>

There are also many Terraform template s available here:

<https://github.com/GoogleCloudPlatform/terraform-google-examples>

This document will walk through deploying a Terraform template and a few Kubernetes(k8s) pods that host a two-tier web application. This template is designed to be deployed into an existing GCP project. After completing this guide, the following infrastructure will be instantiated:



This lab will also show one method to obtain east west traffic visibility between the pods which are hosting the two-tier application.

Support Policy

This template is released under an as-is, best effort, support policy. These scripts should be seen as community supported and Palo Alto Networks will contribute our expertise as and when possible. We do not provide technical support or help in using or troubleshooting the components of the project through our normal support options such as Palo Alto Networks support teams, or ASC (Authorized Support Centers) partners and backline support options. The underlying product used (the VM-Series firewall) by the scripts or templates are still supported, but the support is only for the product functionality and not for help in deploying or using the template or script itself.

Instances Used

When deploying this Terraform template the following machine types are used:

Instance	Machine Type	QTY
PayGo Bundle 1 - VM Series Firewall	n1-standard-4	1
Kubernetes Ubuntu Cluster Nodes	n1-standard-1	2
Internal Load Balancer		1

Note: There are GCP costs associated with each machine type launched, please refer to the Google instance pricing page <https://cloud.google.com/compute/pricing>

Prerequisites

Here are the prerequisites required to successfully launch this template:

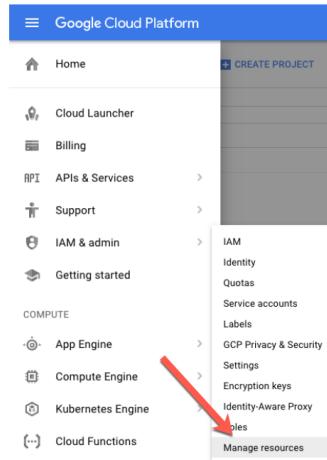
- Terraform application - Instructions on the installation can be found here: <https://www.terraform.io/intro/getting-started/install.html>
- GCP account- Account creation can be done here: <https://cloud.google.com/free/>
- Google Cloud SDK- GCP template installations in this guide are performed from the CLI. Install the SDK/CLI by selecting the relevant platform from the following link and following the installation instructions: <https://cloud.google.com/sdk/>

GCP Project Setup

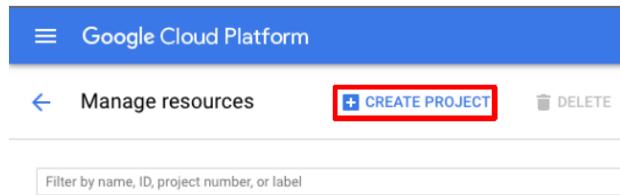
Task 1 – Create a Project

All GCP resources in this guide are deployed to a single project, which is an organizational boundary that separates users, resources, billing information, etc. It is similar to an AWS VPC or an Azure Resource Group. By default, GCP will create a project upon creation of an account.

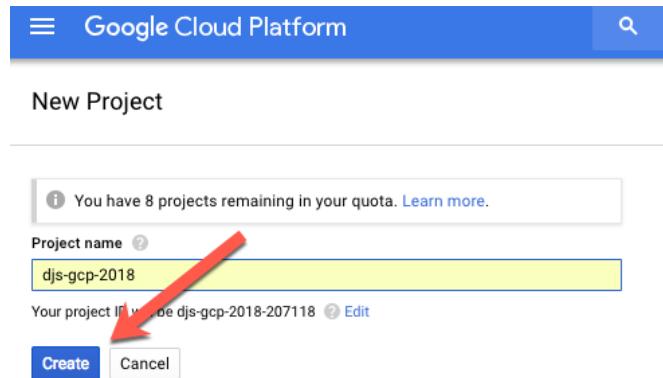
To create an additional dedicated project, use the drop-down on the left and select **IAM & admin > Manage Resources**:



Click **Create Project**:



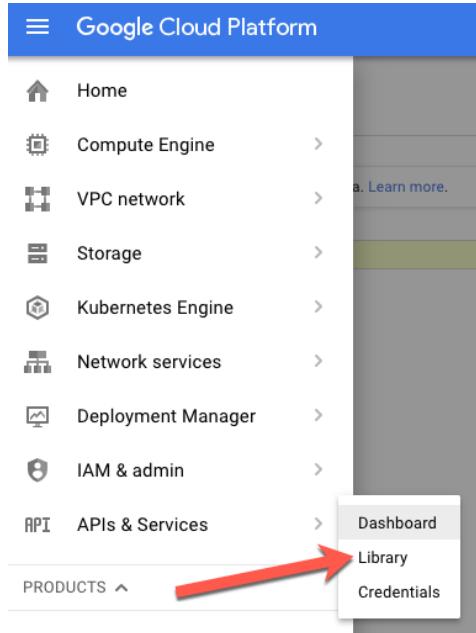
Specify a name for the project and click **Create**:



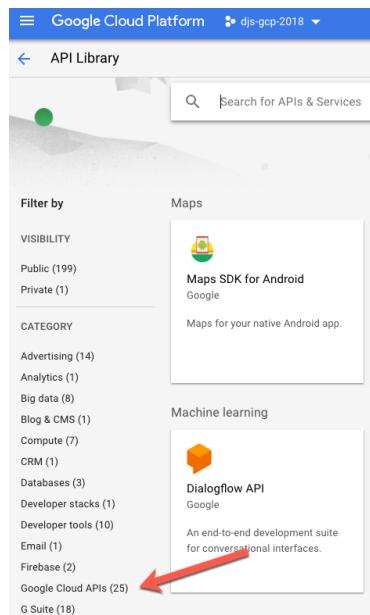
Note that project creation will take a few minutes.

Task 2 – Enable the API

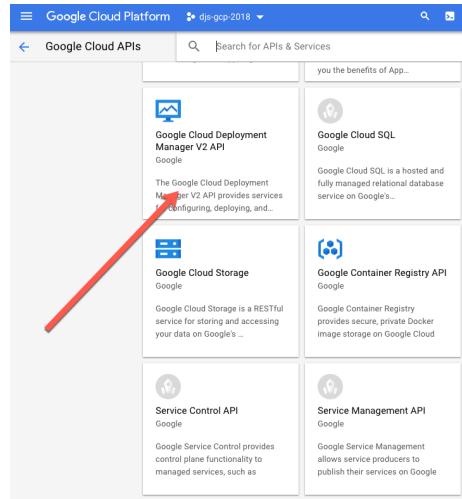
Deploying a template requires the Cloud Deployment Manager API be enabled on the project. Navigate to APIs & Services >Library:



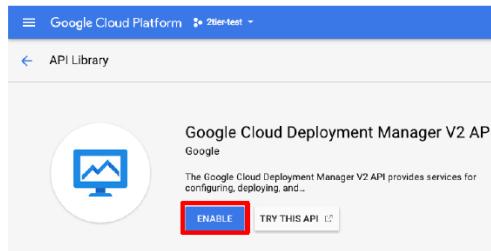
Select Google Cloud APIs on the Left-Hand-Side:



Select Google Cloud Deployment Manager V2 API:



Select **Enable**:



Enabling the API for the project will take a few minutes to complete.

Terraform File setup

Task 1 - Download Terraform files

Navigate to the following GitHub repository to get the terraform files:

<https://github.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/tree/master>

For this lab, the Main.tf and Variables.tf files will need to be downloaded to your local machine. Click on the Main.tf file:

PaloAltoNetworks / GCP-k8s-east-west-visibility

Code Issues 1 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit Add topics

6 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

djspears Delete GCP2018-Kubernetes-Guide.pdf Latest commit 3de70b0 2 hours ago

Main.tf (highlighted with a red arrow) Add files via upload 5 hours ago

README.md Update README.md 5 hours ago

Variables.tf Add files via upload 5 hours ago

alto-inspect.yaml Add files via upload 5 hours ago

guestbook-all-in-one.yaml Add files via upload 5 hours ago

README.md

Next Select Raw and save to a known local location:

PaloAltoNetworks / GCP-k8s-east-west-visibility

Code Issues 1 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master GCP-k8s-east-west-visibility / Main.tf Find file Copy path

djspears Add files via upload 5d16718 5 hours ago

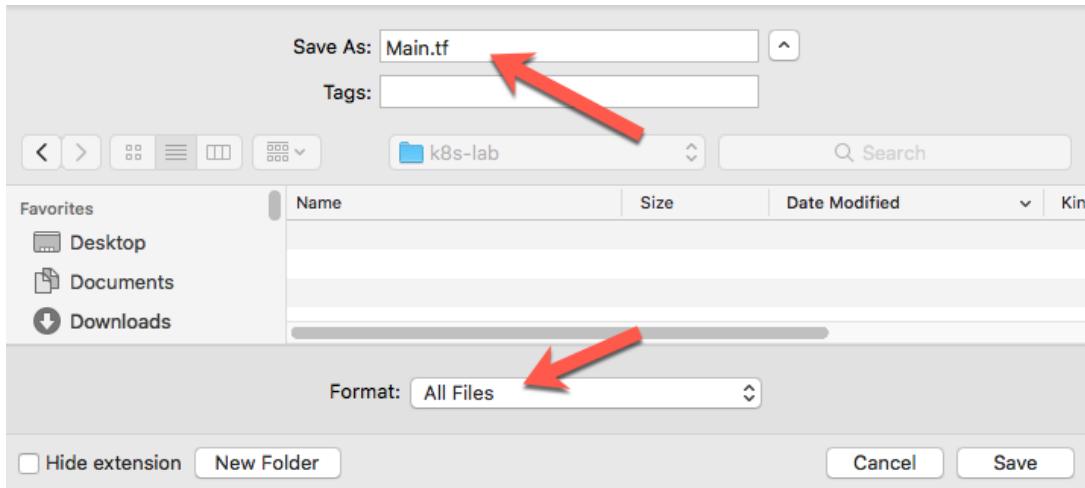
1 contributor

217 lines (181 sloc) | 5.46 KB

Raw Blame History

```
1 // Configure the Google Cloud provider
2 provider "google" {
3   credentials = "${file(var.credentials_file_path)}"
4   project     = "${var.my_gcp_project}"
5   region      = "${var.region}"
6 }
7
8 // Adding SSH Public Key Project Wide
9 resource "google_compute_project_metadata_item" "ssh-keys" {
10   key    = "ssh-keys"
11   value  = "${var.gce_ssh_user}:${var.gce_ssh_pub_key}"
```

Make sure the .tf file extension is added to the file. The following screenshot shows the mac save as dialogue box where the extension needs to be added and the Format needs to be changed to All Files:



Return to the repository and follow the previous steps to download the Variables.tf file to the same location as the Main.tf:

A screenshot of a GitHub repository page for 'PaloAltoNetworks / GCP-k8s-east-west-visibility'. The repository has 11 stars, 0 forks, and 1 issue. The current branch is 'master'. The file 'Main.tf' is shown in the code editor. The code content is as follows:

```
217 lines (181 sloc) | 5.46 KB
1 // Configure the Google Cloud provider
2 provider "google" {
3   credentials = "${file(var.credentials_file_path)}"
4   project     = "${var.my_gcp_project}"
5   region      = "${var.region}"
6 }
7
8 // Adding SSH Public Key Project Wide
9 resource "google_compute_project_metadata_item" "ssh-keys" {
10   key    = "ssh-keys"
11   value  = "${var.gce_ssh_user}:${var.gce_ssh_pub_key}"
```

An arrow points from the text 'Raw' to the 'Raw' button in the toolbar at the bottom of the code preview area. The toolbar also includes 'Blame' and 'History' buttons.

Task 2 - Gather Information and Update the Template File

The Terraform deployment files consist of a main file and a variables file. The Variables.tf file contains information that is easily modified and commonly changed for various situations. The variables in the Variables.tf file are used by the Main.tf file during deployment. Deploying this Terraform template in GCP does require modification of the template Variable.tf file to include deployment-specific information. The fields that must be updated are shown below:

```
// PROJECT Variables
variable "my_gcp_project" {
  default = "Your_Project_ID"
}

variable "region" {
  default = "us-central1"
}

variable "zone" {
  default = "us-central1-a"
}

variable "credentials_file_path" {
  description = "Path to the JSON file used to describe your account credentials"
  default      = "sample format: /GCP/singlefw-djs-gcp-2018/djs-gcp-2018-creds.json"
}

variable "gce_ssh_user" {
  description = " ssh user that is used in the public key"
  default     = "sample: davejspears"
}

variable "gce_ssh_pub_key" {
  description = " ssh key in the format: ssh-rsa <key> username "
  default     = "sample format: ssh-rsa LM4L63uRouhKaV7VyU/W2skcItmsdlyBIec/vqdHpj0icOZUo7XFxBc86sCFHB4IcKQ0B davejspears"
}
```

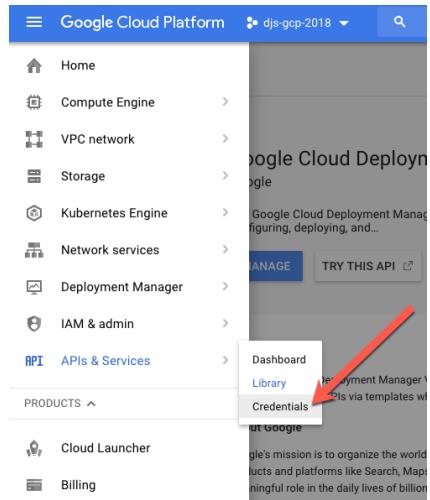


Project ID is the id that is associated with the GCP project that was created previously.

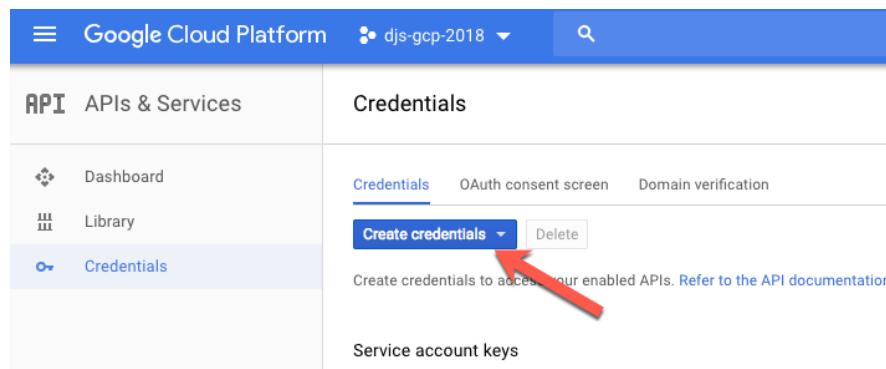
Credential file path is the path to the JSON file that has service credentials that will be used to deploy the template.

SSH User and SSH Pub Key are the SSH keys that can be used to access VM resources in the environment. In this lab the Firewall will be bootstrapped from an existing public GCP bucket. However, if that process fails, this SSH user can be used to access the firewall CLI.

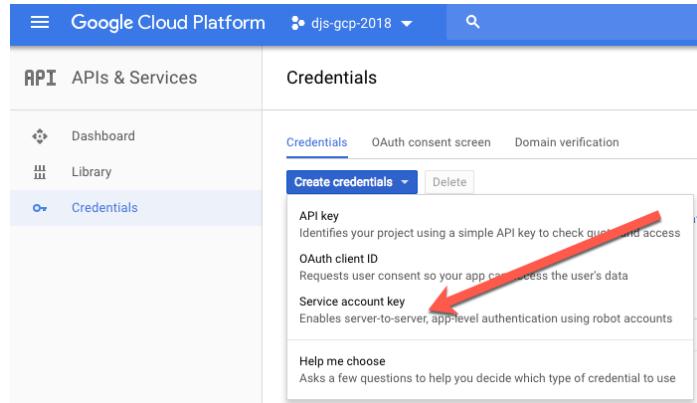
To create the credentials to access the APIs in JSON format. In GCP console go to (APIs & Services > Credentials:



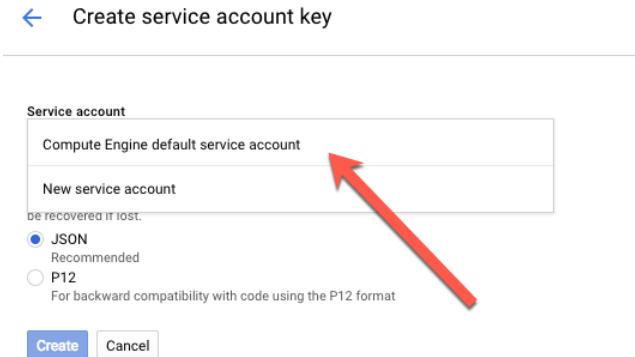
Click Create Credentials:



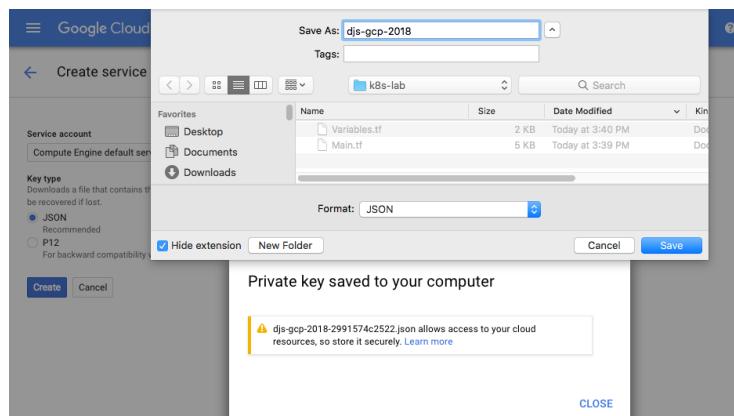
Then select Service Account Key:



Pick Compute Engine default service account and make sure the JSON format is ticked. Click Create:



download the file to your computer. It is easy to put the credential file in the same folder as the terraform Main.tf and Variables.tf files:



Use an editor of your choice to update the Variables.tf file with the appropriate path:

```
1 // PROJECT Variables
2 variable "my_gcp_project" {
3   default = "Your_Project_ID"
4 }
5
6 variable "region" {
7   default = "us-central1"
8 }
9
10 variable "zone" {
11   default = "us-central1-a"
12 }
13
14 variable "credentials_file_path" {
15   description = "Path to the JSON file used to describe your service account credentials"
16   default     = "/GCP/k8s-lab/djs-gcp-2018.json" ← Red arrow points here
17 }
```

If you do not already have an SSH key, the follow example shows how to create a SSH key on a Mac using the ssh-keygen -t rsa command:

```

1. Shell
SJOMAC3024G8WL:k8s-lab dspears$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/dspears/.ssh/id_rsa): /GCP/k8s-lab/djs-gcp-key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /GCP/k8s-lab/djs-gcp-key.
Your public key has been saved in /GCP/k8s-lab/djs-gcp-key.pub.
The key's randomart image is:
+---[RSA 2048]---+
| E.. |
| . +o . |
| .=.= |
| .B B o |
| .= B S |
| .o* + = +
| .++ = = o |
|ooo+ oo* . |
|=+= o+... |
+---[SHA256]---+
SJOMAC3024G8WL:k8s-lab dspears$ ls -la

```

In the previous example the keys were generated and stored in the same directory as the other lab files. The public and private keys can be seen using the ls -la command. I have also displayed the public key via the more command in the following screenshot:

```

SJOMAC3024G8WL:k8s-lab dspears$ ls -la
total 48
drwxr-xr-x@ 7 dspears wheel          224 Jun 13 16:21 .
drwxr-xr-x  28 dspears wheel          896 Jun 13 15:08 ..
-rw-r--r--@  1 dspears PALOALTONEWORK\Domain Users  5345 Jun 13 15:39 Main.tf
-rw-r--r--@  1 dspears PALOALTONEWORK\Domain Users 1560 Jun 13 16:11 Variables.tf
-rw-r--r--@  1 dspears PALOALTONEWORK\Domain Users 2328 Jun 13 16:08 djs-gcp-2018.json
-rw-----   1 dspears wheel          1679 Jun 13 16:21 djs-gcp-key
-rw-r--r--@  1 dspears wheel          404 Jun 13 16:21 djs-gcp-key.pub
SJOMAC3024G8WL:k8s-lab dspears$ more djs-gcp-key.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCB9RA/kjVZ7pUrh4oMHLJP1jDeMc3pk0UCL3aQ9jthYJm/uWiVjLZ3NEjiHTfQQITt
kbWQkfCcWhhnQ9E9vtLkkiutpqT3wfm1nUeSA0rcTSFGMCB0tqVJds/u/KIAxYpSSIZZ/GYk1qcT9oXHSF/mrfq1w+9aIX9Hsaa3rY
NIEJA42T/00LnH1LzMa268L9aEMEIxoKrx9d9BMHv/RjhdIwuWqMiq0+x/KvPTrPNhK6sq/5UbWZbnvvccjyS1xAF2a0rW9rwQ3vH6m
SQZjwVU0m41vqHqRTNJQKmEVyv8P5oaesU57ed0AtuLMpCoNpb9015m912h7PAaCnugeJ/ dspears@SJOMAC3024G8WL
dis-acp-kev.key (END)

```

Retrieve the username and key from the public key file. Then update the Variables.tf file:

```

// PROJECT Variables
variable "my_gcp_project" {
  default = "djs-gcp-2018"
}

variable "region" {
  default = "us-central1"
}

variable "zone" {
  default = "us-central1-a"
}

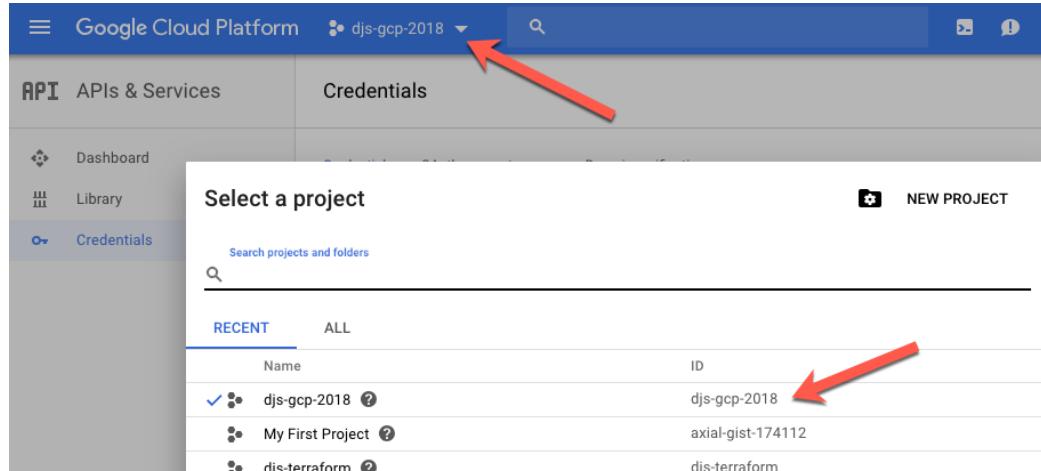
variable "credentials_file_path" {
  description = "Path to the JSON file used to describe your account credentials"
  default     = "/GCP/k8s-lab/djs-gcp-2018.json"
}

variable "gce_ssh_user" {
  description = "ssh user that is used in the public key"
  default     = "dspears@SJOMAC3024G8WL"
}

variable "gce_ssh_pub_key" {
  description = "ssh key in the format: ssh-rsa <key> username"
  default     = "ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCB9RA/kjVZ7pUrh4oMHLJP1jDeMc3pk0UCL3aQ9jthYJm/uWiVjLZ3NEjiHTfQQITt
kbWQkfCcWhhnQ9E9vtLkkiutpqT3wfm1nUeSA0rcTSFGMCB0tqVJds/u/KIAxYpSSIZZ/GYk1qcT9oXHSF/mrfq1w+9aIX9Hsaa3rY
NIEJA42T/00LnH1LzMa268L9aEMEIxoKrx9d9BMHv/RjhdIwuWqMiq0+x/KvPTrPNhK6sq/5UbWZbnvvccjyS1xAF2a0rW9rwQ3vH6m
SQZjwVU0m41vqHqRTNJQKmEVyv8P5oaesU57ed0AtuLMpCoNpb9015m912h7PAaCnugeJ/ dspears@SJOMAC3024G8WL
// VM-Series Firewall Variables

```

To get the GCP Project ID click the project selection drop down at the top of the GCP Console.
The Project ID is displayed in the project selection window:



Update the Variables.tf file with the GCP ID:

```
// PROJECT Variables
variable "my_gcp_project" {
  default = "djs-gcp-2018" ←
}

variable "region" {
  default = "us-central1"
}

variable "zone" {
  default = "us-central1-a"
}

variable "credentials_file_path" {
  description = "Path to the JSON file used to describe your account credentials"
  default     = "/GCP/k8s-lab/djs-gcp-2018.json"
}

variable "gce_ssh_user" {
  description = "ssh user that is used in the public key"
  default     = "dspears@SJOMAC3024G8WL"
}

variable "gce_ssh_pub_key" {
  description = "ssh key in the format: ssh-rsa <key> username "
  default     = "ssh-rsa AAAAB3NzaC1yc2EAAQABAAQc9RA/kjVZ7pUrh4oMHLJP1jDeMc3pk0UCL3aQ9jthYJm/uWiVjLZ3NEjiHTFQQITkbWQkFCcWlhQ9E9vtLkkiwutpqT3wfmlnUeSA0rcTSFGMCB0tqVJds/u/KIAxYpSS1ZZ/GYk1qcT9oXHSF/mrfq1w+9aI
// The rest of the variables do not need to be modified for the K8s Lab
// VM-Series Firewall Variables
```

Deploy the Terraform Template

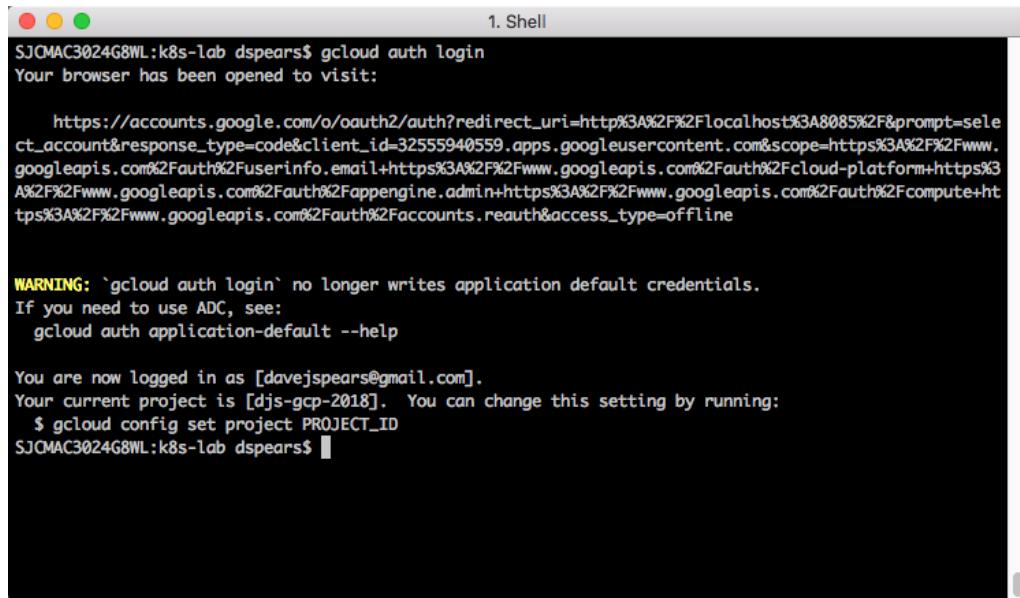
Task 1 – Connect to GCP

Open a terminal shell and navigate to the directory containing the Terraform template files.

Authenticate to the GCP environment from the command line with the command:

\$ **gcloud auth login**

- Copy/paste the link into a browser(if needed) and select the account to authenticate if a browser does not automatically launch:
- Review the requested permissions and click Allow:

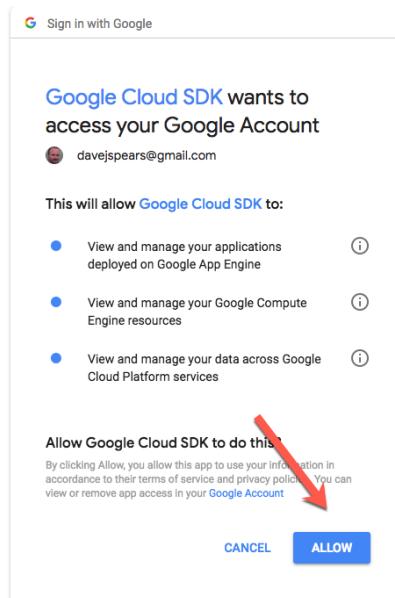


```
SJCMAC3024G8WL:k8s-lab dspears$ gcloud auth login
Your browser has been opened to visit:

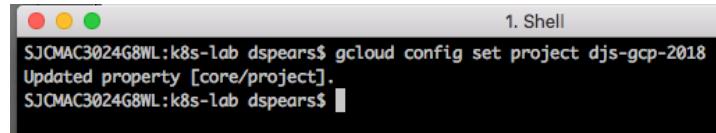
https://accounts.google.com/o/oauth2/auth?redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&prompt=select_account&response_type=code&client_id=32555940559.apps.googleusercontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&access_type=offline

WARNING: `gcloud auth login` no longer writes application default credentials.
If you need to use ADC, see:
  gcloud auth application-default --help

You are now logged in as [davejspears@gmail.com].
Your current project is [djs-gcp-2018]. You can change this setting by running:
  $ gcloud config set project PROJECT_ID
SJCMAC3024G8WL:k8s-lab dspears$
```



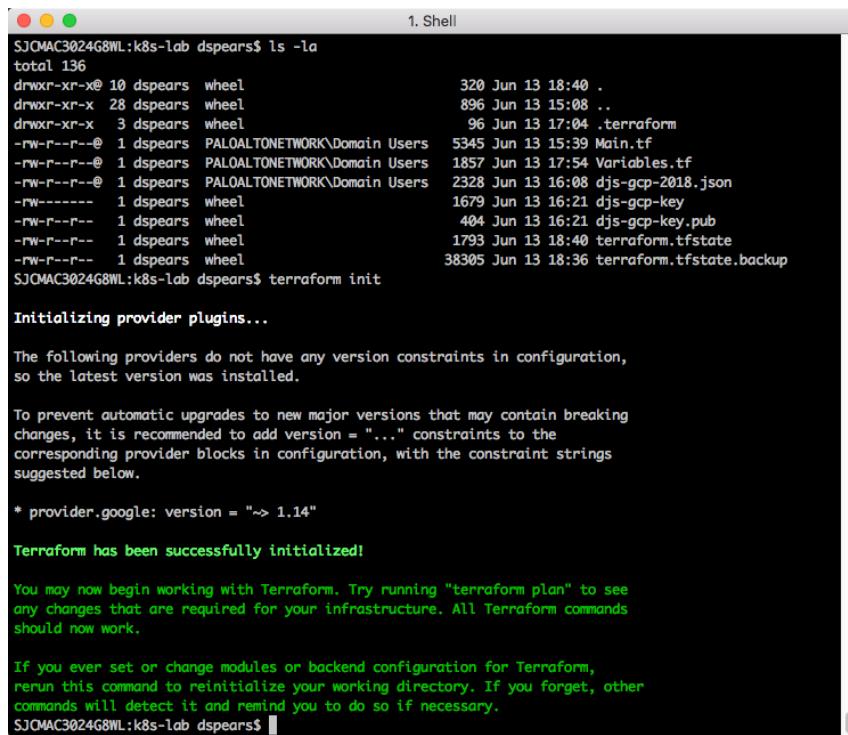
Note the Project ID. If it is not the project that was created for the lab, use the `$ gcloud config set project my_Project_id` command to change the project:



```
SJOMAC3024G8WL:k8s-lab dspears$ gcloud config set project djs-gcp-2018
Updated property [core/project].
SJOMAC3024G8WL:k8s-lab dspears$
```

Task 2 – Deploy the Terraform Template

Ensure you are in the directory with the Main.tf and Variables.tf files and execute the **terraform init** command which will initialize terraform:



```
SJOMAC3024G8WL:k8s-lab dspears$ ls -la
total 136
drwxr-xr-x@ 10 dspears  wheel          320 Jun 13 18:40 .
drwxr-xr-x  28 dspears  wheel          896 Jun 13 15:08 ..
drwxr-xr-x   3 dspears  wheel          96 Jun 13 17:04 .terraform
-rw-r--r--@  1 dspears  PALOALTONETWORK\Domain Users  5345 Jun 13 15:39 Main.tf
-rw-r--r--@  1 dspears  PALOALTONETWORK\Domain Users  1857 Jun 13 17:54 Variables.tf
-rw-r--r--@  1 dspears  PALOALTONETWORK\Domain Users  2328 Jun 13 16:08 djs-gcp-2018.json
-rw-----   1 dspears  wheel          1679 Jun 13 16:21 djs-gcp-key
-rw-r--r--   1 dspears  wheel          404 Jun 13 16:21 djs-gcp-key.pub
-rw-r--r--   1 dspears  wheel          1793 Jun 13 18:40 terraform.tfstate
-rw-r--r--   1 dspears  wheel          38305 Jun 13 18:36 terraform.tfstate.backup
SJOMAC3024G8WL:k8s-lab dspears$ terraform init

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "... " constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.google: version = "~> 1.14"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
SJOMAC3024G8WL:k8s-lab dspears$
```

Once the terraform init has completed run the **terraform plan** command. This will show what changes will be implemented with the terraform script. This will also identify if there are any errors detected with the terraform files:

```

1. Shell
network:
  "https://www.googleapis.com/compute/v
  <computed>
  "1"
  "32"
  <computed>
  "ubuntu"
  "2"
  "the-cluster"
  "default-pool"
  <computed>
  "n1-standard-1"
  "1"
  "https://www.googleapis.com/auth/monit
  "false"
  <computed>
  <computed>
  <computed>
  "false"
  <computed>
  <computed>
  "${google_compute_subnetwork.trust-su
  "us-central1-a"

Plan: 12 to add, 0 to change, 0 to destroy.

-----
Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

SJOMAC3024G8WL:k8s-lab dspears$ 

```

Now run the **terraform apply** command to deploy the template. At the action prompt enter **yes**.

```

SJOMAC3024G8WL:k8s-lab dspears$ terraform apply
google_compute_network.trust: Refreshing state... (ID: trust)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

+ google_compute_firewall.allow-inbound
  id:          <computed>
  allow.#:    "1"
  allow.186047796.ports.#: "2"
  allow.186047796.ports.0: "80"
  allow.186047796.ports.1: "22"
  allow.186047796.protocol: "tcp"
  destination_ranges.#: <computed>
  direction:   <computed>
  name:        "allow-inbound"
  network:    "${google_compute_network.untrust.se
  f_link}"
  priority:   "1000"
  project:    <computed>
  self_link:   <computed>
  source_ranges.#: "1"
  source_ranges.1080289494: "0.0.0.0/0"

node_config.0.image_type:          "ubuntu"
node_config.0.labels.#:           "2"
node_config.0.labels.cluster:     "the-cluster"
node_config.0.labels.pool:         "default-pool"
node_config.0.local_ssd_count:    <computed>
node_config.0.machine_type:       "n1-standard-1"
node_config.0.oauth_scopes.#:     "1"
node_config.0.oauth_scopes.1277378754: "https://www.googleapis.com/auth/monit
  "false"
  <computed>
  <computed>
  <computed>
  "false"
  <computed>
  <computed>
  "${google_compute_subnetwork.trust-su
  "us-central1-a"

Plan: 12 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```

It will take a few minutes to complete. If all goes well, Terraform will output; “Apply Complete!” and provide some additional output information about the resources deployed:

```
self_link:           "" => "<computed>"  
google_compute_route.trust: Still creating... (10s elapsed)  
google_compute_route.trust: Creation complete after 11s (ID: trust-route)  
  
Apply complete! Resources: 13 added, 0 changed, 0 destroyed.  
  
Outputs:  
  
k8s-cluster-name = [  
    cluster-1  
]  
pan-tf-name = [  
    firewall-1  
]  
pan-tf-trust-ip = [  
    10.5.2.4  
]  
SJC0MAC3024G8WL:k8-test dspears$
```

Review what was deployed

In this activity, you will:

Review the resources that have been launched

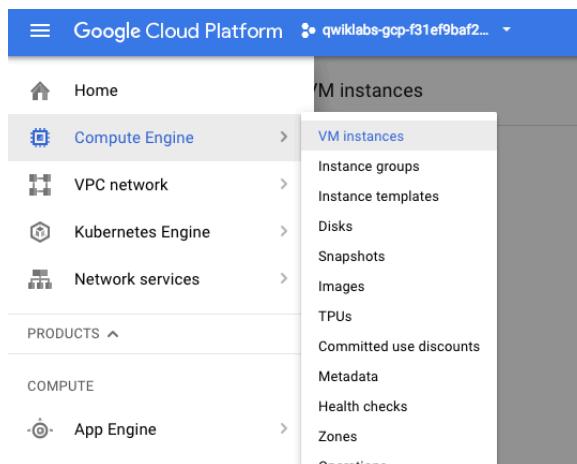
Log into the VM-Series firewall

Confirm bootstrap success

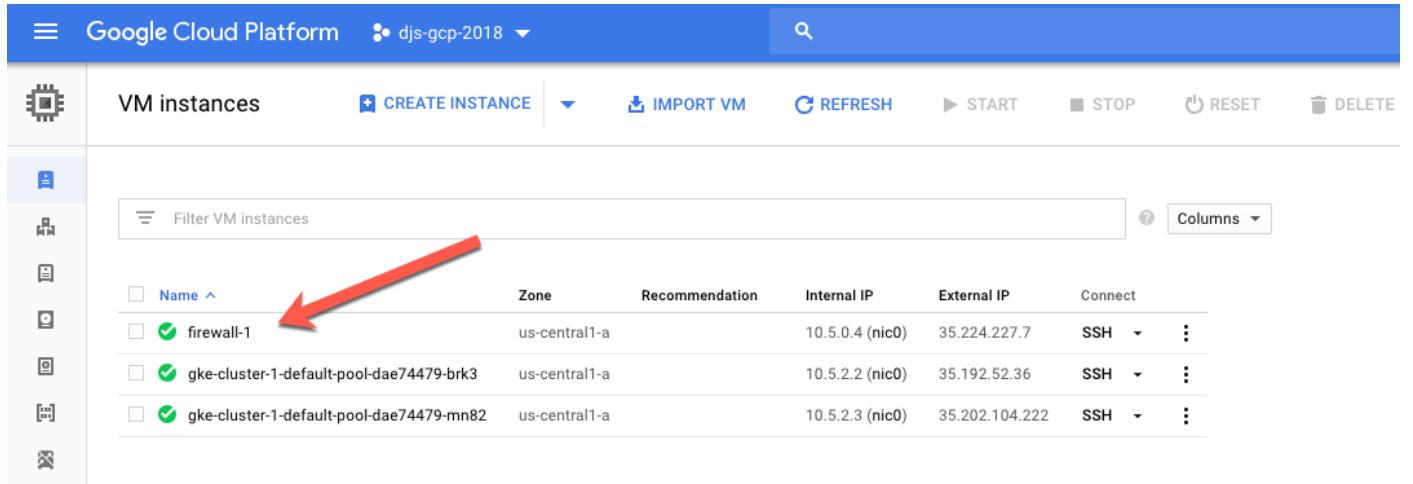
Inspect k8s cluster

Task 1 – Look around GCP console

Navigate to Compute Engine > VM Instances to see what VMs the terraform template deployed:

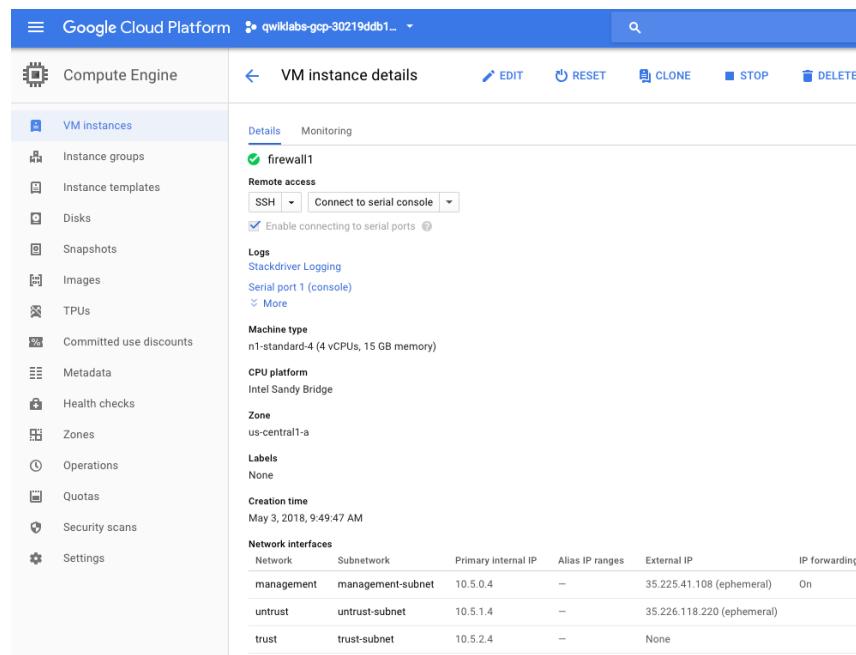


There should be 1 firewall and two k8s nodes displayed. Click on the firewall to open a detailed view of the deployed firewall:



VM instances

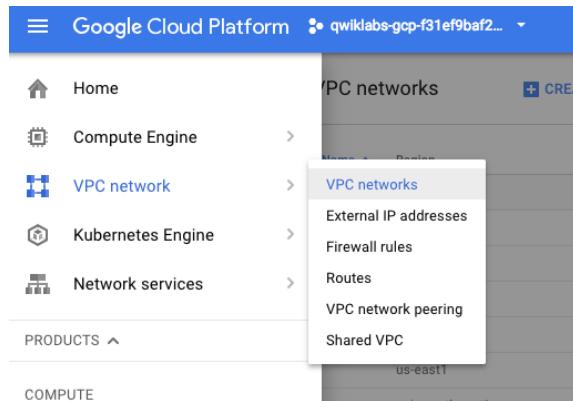
Name	Zone	Recommendation	Internal IP	External IP	Connect
firewall-1	us-central1-a		10.5.0.4 (nic0)	35.224.227.7	SSH
gke-cluster-1-default-pool-dae74479-brk3	us-central1-a		10.5.2.2 (nic0)	35.192.52.36	SSH
gke-cluster-1-default-pool-dae74479-mn82	us-central1-a		10.5.2.3 (nic0)	35.202.104.222	SSH



VM instance details

Network	Subnetwork	Primary Internal IP	Alias IP ranges	External IP	IP forwarding
management	management-subnet	10.5.0.4	—	35.225.41.108 (ephemeral)	On
untrust	untrust-subnet	10.5.1.4	—	35.226.118.220 (ephemeral)	
trust	trust-subnet	10.5.2.4	—	None	

Navigate to VPC Network > VPC Networks to see the different networks that have been created as part of the lab.



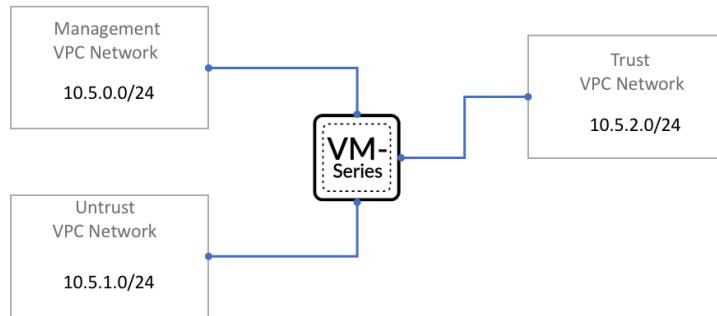
You should see 3 VPC networks: management, trust and untrust.

The screenshot shows the 'VPC networks' page in the Google Cloud Platform. On the left, there's a sidebar with 'VPC network' selected, followed by 'VPC networks' (which is also highlighted in blue). The main area displays a table of VPC networks:

	europe-west4	default	10.164.0.0/20	10.164.0.1
management	us-central1	management-subnet	Custom	1
trust	us-central1	trust-subnet	10.5.2.0/24	10.5.2.1
untrust	us-central1	untrust-subnet	Custom	1

A red arrow points to the 'management' row in the table.

The following diagram describes the network topology of what has been deployed:



Feel free to navigate through other parts of the Google Cloud Console such as VPC Networks > Routes, Firewall Rules. This will come in handy in activities later on.

Task 2 – Review the Kubernetes Cluster

Kubernetes is a portable, extensible, open-source orchestrator that is used to manage containerized workloads. Kubernetes has a large and rapidly growing ecosystem. The portability of Kubernetes allows for workloads to be migrated between various clouds (public or private). Further documentation is available at:

<https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

Navigate to Kubernetes Engine > Kubernetes clusters:

Kubernetes clusters						
<input type="checkbox"/>	Name	Location	Cluster size	Total cores	Total memory	Notifications
<input checked="" type="checkbox"/>	cluster-1	us-central1-a	2	2 vCPUs	7.50 GB	<button>Connect</button>

Task 2 – Explore the cluster

Let's explore some aspects of the Kubernetes cluster that were deployed as part of the terraform template. Click into the cluster to see some of the details of the cluster:

Kubernetes clusters						
<input type="checkbox"/>	Name	Location	Cluster size	Total cores	Total memory	Notifications
<input checked="" type="checkbox"/>	cluster-1	us-central1-a	2	2 vCPUs	7.50 GB	<button>Connect</button>

Below outlines some interesting info. As you can see the nodes reside in the same trust subnet as the firewall's trust interface:

Kubernetes Engine

Kubernetes clusters

cluster-1

	Details	Storage	Nodes
Cluster	Master version: 1.8.8-gke.0 Endpoint: 35.226.171.41 Client certificate: Enabled Kubernetes alpha features: Disabled Total size: 2 Master zone: us-central1-a Node zones: us-central1-a Network: trust Subnet: trust-subnet Alias IP ranges: Disabled Container address range: 10.8.0.0/14 Stackdriver Logging: Enabled Stackdriver Monitoring: Enabled Private cluster: Disabled Master authorized networks: Disabled Network policy: Disabled Legacy authorization: Disabled Maintenance window: Any time		
Labels	None		
Add-ons			
Permissions			

Kubernetes version and kube-apiserver

Subnet in which cluster is deployed

CIDR for all pods in the cluster

Navigate to the VM Instances to verify that the nodes are deployed. Note the Node internal IP addressing:

COMPUTE

- App Engine
- Compute Engine
- Kubernetes Engine
- Cloud Functions

VM instances

Name	Zone	Recommendation	Internal IP
firewall1	us-central1-a		10.5.0.4
gke-cluster-1-default-pool-e05e1286-m108	us-central1-a		10.5.2.3
gke-cluster-1-default-pool-e05e1286-tbs0	us-central1-a		10.5.2.2

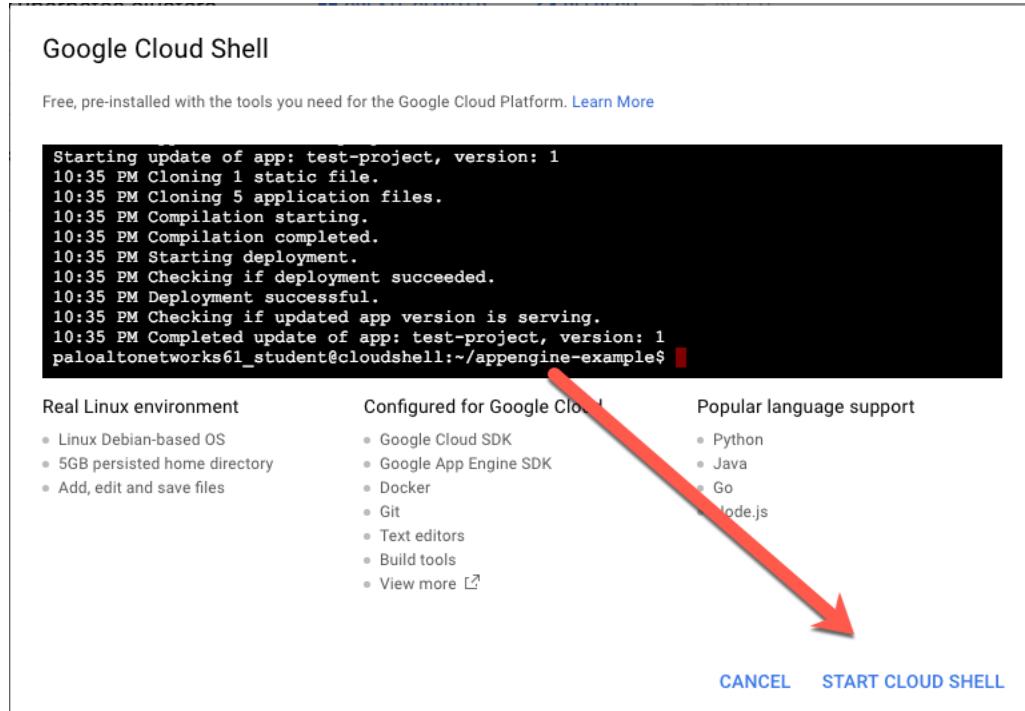
Head back to the cluster console and connect to the cluster:

The screenshot shows the GCP navigation bar on the left with sections for COMPUTE, STORAGE, and BIOTABLE. A dropdown menu is open under COMPUTE, showing 'Kubernetes Engine' selected. To its right is the 'Kubernetes clusters' page. This page has a header with 'CREATE CLUSTER', 'REFRESH', and 'DELETE' buttons. Below is a search bar and a table of clusters. One cluster, 'cluster-1', is listed with details: Location 'us-central1-a', Cluster size '2', Total cores '2 vCPUs', and Total memory '7.50 GB'. A red arrow points to the 'Connect' button at the bottom right of the cluster row.

And connect via the cloud shell:

The screenshot shows a 'Connect to the cluster' dialog box overlaid on the GCP interface. The dialog has two main sections: 'Command-line access' and 'Cloud Console dashboard'. In the 'Command-line access' section, there is a code snippet for configuring kubectl: '\$ gcloud container clusters get-credentials cluster-1 --zone us-central1-a --project quickstart-gcp-30219db141'. A red arrow points to the 'Run in Cloud Shell' button below it. In the 'Cloud Console dashboard' section, there is a 'Open workloads dashboard' button. At the bottom right of the dialog is an 'OK' button.

Clicking Run in Cloud Shell starts a google cloud shell. If this dialog box opens, click start cloud shell:



It may take a while to provision. Once provisioned press the Enter key to authenticate to the cluster. Afterwards the \$ prompt will be present:

A screenshot of a Cloud Shell terminal window. It shows the welcome message and the process of fetching cluster credentials:

```
Welcome to Cloud Shell! Type "help" to get started.
paloaltonetworks60_student@qwiklabs-gcp-dcbbd00379829f29:~$ gcloud container clusters get-credentials cluster-1 --zone us-central1-a --project qwiklabs-gcp-dcbbd00379829f29
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.
paloaltonetworks60_student@qwiklabs-gcp-dcbbd00379829f29:~$
```

Let us explore some pods and services that have deployed.

Run this command in the cloud shell:

```
$ kubectl get pods
```

A screenshot of a Cloud Shell terminal window. It shows the command being run and the resulting output:

```
Welcome to Cloud Shell! Type "help" to get started.
paloaltonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$ gcloud container clusters get-credentials cluster-1 --zone us-central1-a --project qwiklabs-gcp-30219
ddb14158a20
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.
paloaltonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$ kubectl get pods
No resources found.
paloaltonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$
```

Since we have not deployed any resources this is normal.

Now let us see what system pods have been deployed. Run this command in the shell:

```
$ kubectl get pods --all-namespaces -o wide
```

```
davejspears@cloudshell:~ (dje-gcp-2018)$ kubectl get pods --all-namespaces
NAMESPACE      NAME          READY   STATUS    RESTARTS   AGE     IP           NODE
kube-system    hepsster-v1.4.3-6644cc4b46-dwn8s   2/2     Running   0          17m    10.8.1.5   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kube-dns-778977457c-144rf      3/3     Running   0          18m    10.8.1.4   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kube-dns-778977457c-anvvg      3/3     Running   0          2m     10.8.1.6   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kube-dns-autoscaler-7db47cb9b7-j9n7n  1/1     Running   0          19m    10.8.1.3   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kube-proxy-gke-cluster-1-default-pool-2df01519-6sgf  1/1     Running   0          2m     10.5.2.2   gke-cluster-1-default-pool-2df01519-6sgf
kube-system    kube-proxy-gke-cluster-1-default-pool-2df01519-n3w8  1/1     Running   0          19m    10.5.2.3   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kubernetes-dashboard-6bb875b5bc-hls4m   1/1     Running   0          2m     10.8.1.7   gke-cluster-1-default-pool-2df01519-n3w8
davejspears@cloudshell:~ (dje-gcp-2018)$
```

Note: If the output does not show all the pods in a running state, wait and rerun the **kubectl get pods --all-namespaces -o wide** command until they do. An example of this is state is in the following screen-print:

```
davejspears@cloudshell:~ (dje-gcp-2018)$ kubectl get pods --all-namespaces
NAMESPACE      NAME          READY   STATUS    RESTARTS   AGE     IP           NODE
kube-system    hepsster-v1.4.3-6644cc4b46-dwn8s   0/2     Pending   0          11m    <none>      gke-cluster-1-default-pool-2df01519-n3w8
kube-system    hepsster-v1.4.3-7875d9f9f-mlv7g      2/2     Terminating   0          12m    10.8.0.4   gke-cluster-1-default-pool-2df01519-6sgf
kube-system    kube-dns-778977457c-2xwfq      3/3     Running   0          13m    10.8.0.3   gke-cluster-1-default-pool-2df01519-6sgf
kube-system    kube-dns-778977457c-144rf      0/3     ContainerCreating   0          12m    <none>      gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kube-dns-autoscaler-7db47cb9b7-j8n7n  1/1     Running   0          13m    10.8.1.3   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kube-proxy-gke-cluster-1-default-pool-2df01519-6sgf  1/1     Running   0          12m    10.5.2.2   gke-cluster-1-default-pool-2df01519-6sgf
kube-system    kube-proxy-gke-cluster-1-default-pool-2df01519-n3w8  1/1     Running   0          12m    10.5.2.3   gke-cluster-1-default-pool-2df01519-n3w8
kube-system    kubernetes-dashboard-6bb875b5bc-n7wj8   1/1     Running   0          13m    10.8.0.2   gke-cluster-1-default-pool-2df01519-6sgf
davejspears@cloudshell:~ (dje-gcp-2018)$
```

Now let us see what services have been deployed as part of the system:

Run the following in the shell:

```
$ kubectl get svc
```

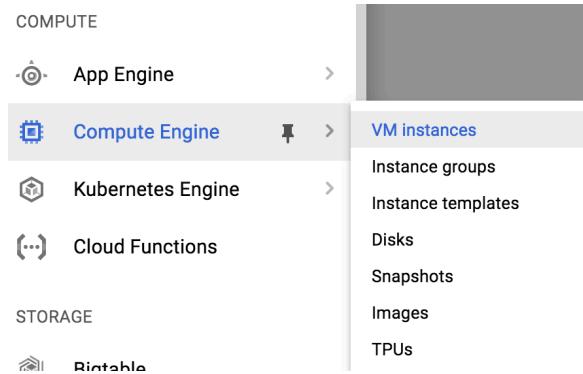
```
...labs-gcp-30219ddb14158a20:x + 
palolonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.11.240.1  <none>        443/TCP      7m
palolonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$
```

As you can see no services besides the system cluster have been deployed.

Task 3 – Log into the firewall

The VM-Series firewall deployed as part of the lab has been bootstrapped. Bootstrapping is a feature of the VM-Series firewall that allows you to load a pre-defined configuration into the firewall during boot-up. This ensures that the firewall is configured and ready at initial boot-up, thereby removing the need for manual configuration. The bootstrapping feature also enables automated deployment of the VM-Series.

Navigate to Compute Instances > VM Instances



Click on firewall instance name to get more information and identify the management interface IP.

This screenshot shows the 'VM instances' list page in the Compute Engine section of the Google Cloud Platform. A red arrow points to the 'Zone' column header, which is currently sorted by 'Name'. The table lists one instance named 'firewall1' located in 'us-central1-a' with an internal IP of '10.5.0.4' and an external IP of '35.225.41.108'.

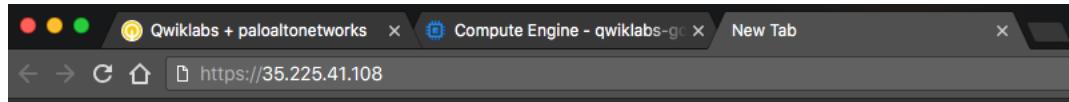
Name	Zone	Recommendation	Internal IP	External IP
firewall1	us-central1-a		10.5.0.4	35.225.41.108

Copy the External public IP of the management interface

This screenshot shows the 'VM instance details' page for the 'firewall1' instance. A red arrow points to the 'External IP' column in the 'Network interfaces' table. The table shows three interfaces: 'management' (IP 35.225.41.108), 'untrust' (IP 35.226.118.220), and 'trust' (IP None).

Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	IP forwarding
management	management-subnet	10.5.0.4	—	35.225.41.108 (ephemeral)	On
untrust	untrust-subnet	10.5.1.4	—	35.226.118.220 (ephemeral)	
trust	trust-subnet	10.5.2.4	—	None	

Open another browser tab and navigate to the firewall management interface:



If you get a security exception, please ignore for this lab and proceed to the firewall login page. We are using a self-signed certificate which causes the exception. Depending on how quickly you do this, you might see the following message. It is normal and part of the bootup process:

Server Rebooting

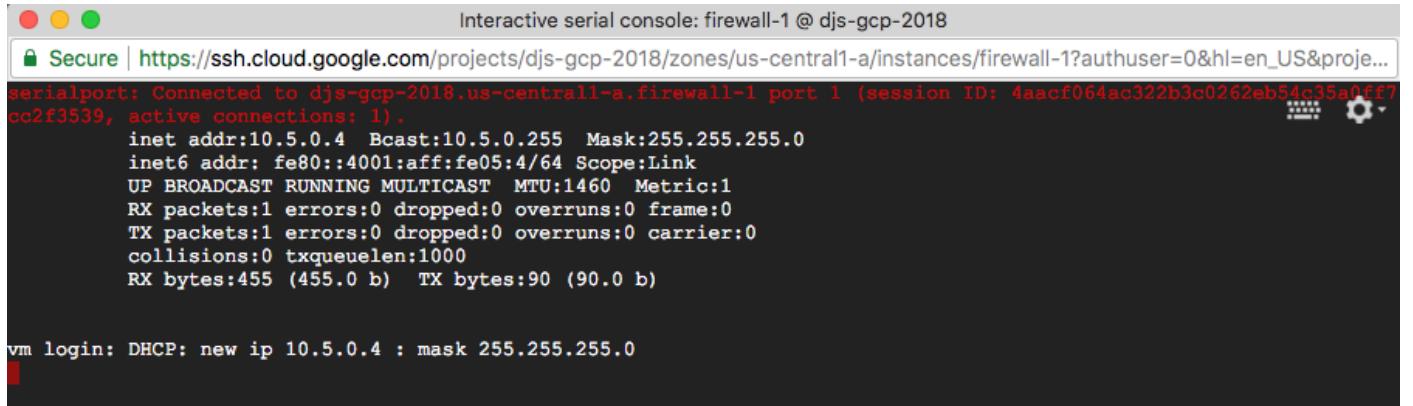
Please wait while the server reboots...



If the firewall does not seem available, it is possible to check the firewall via the GCP portal. You can click on the Connect to serial console to see if the firewall is up and running:

A screenshot of the Google Cloud Platform (GCP) VM instance details page for a machine named "firewall-1". The "Details" tab is selected. In the "Remote access" section, there is a dropdown menu set to "SSH" and a button labeled "Connect to serial console". A red arrow points to this "Connect to serial console" button. Other options in the dropdown include "Serial" and "Serial over LAN". Below the dropdown, there is a checked checkbox for "Enable connecting to serial ports". The page also displays information about the machine type (n1-standard-4), CPU platform (Intel Sandy Bridge), zone (us-central1-a), and labels (None).

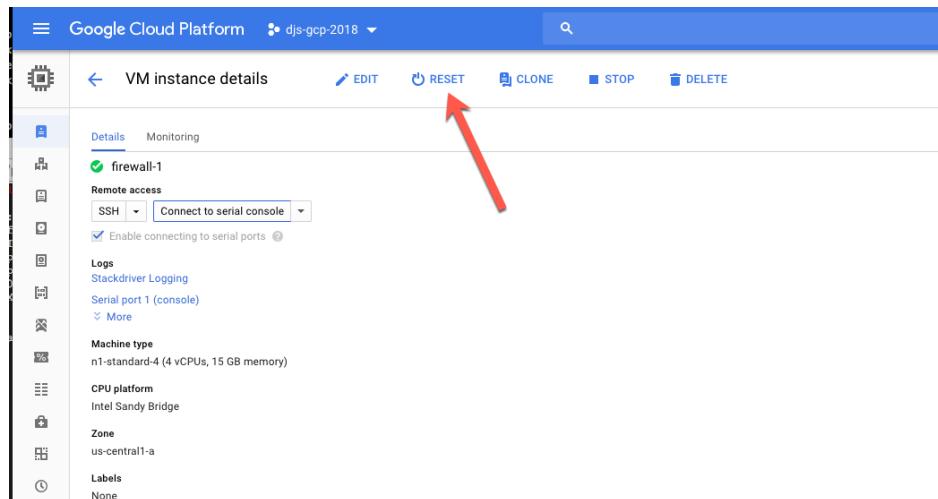
In some cases, the firewall may not be up and running yet. You can see in the prompt below the firewall is not all the way up.



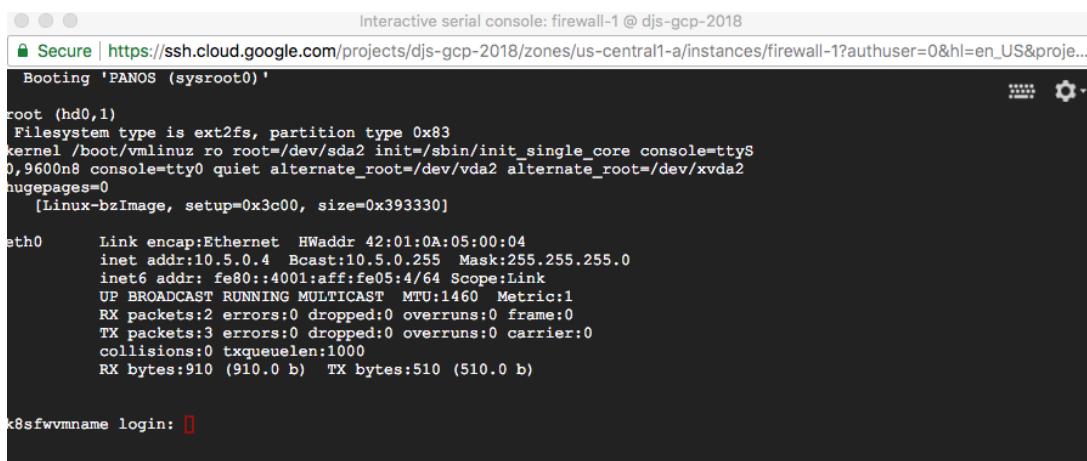
```
Interactive serial console: firewall-1 @ djs-gcp-2018
Secure | https://ssh.cloud.google.com/projects/djs-gcp-2018/zones/us-central1-a/instances/firewall-1?authuser=0&hl=en_US&proj...
serialport: Connected to djs-gcp-2018.us-central1-a.firewall-1 port 1 (session ID: 4aacf064ac322b3c0262eb54c35a0ff7cc2f3539, active connections: 1).
    inet addr:10.5.0.4 Bcast:10.5.0.255 Mask:255.255.255.0
    inet6 addr: fe80::4001:aff:fe05:4/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1460 Metric:1
        RX packets:1 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:455 (455.0 b) TX bytes:90 (90.0 b)

vm login: DHCP: new ip 10.5.0.4 : mask 255.255.255.0
```

You can wait for the VM to finish the bootup process or a quick reset of the VM seems to be a short fix:



After the reset the console shows the following prompt which is a good indication the firewall is booting up:



```
Interactive serial console: firewall-1 @ djs-gcp-2018
Secure | https://ssh.cloud.google.com/projects/djs-gcp-2018/zones/us-central1-a/instances/firewall-1?authuser=0&hl=en_US&proj...
Booting 'PANOS (sysroot0)'

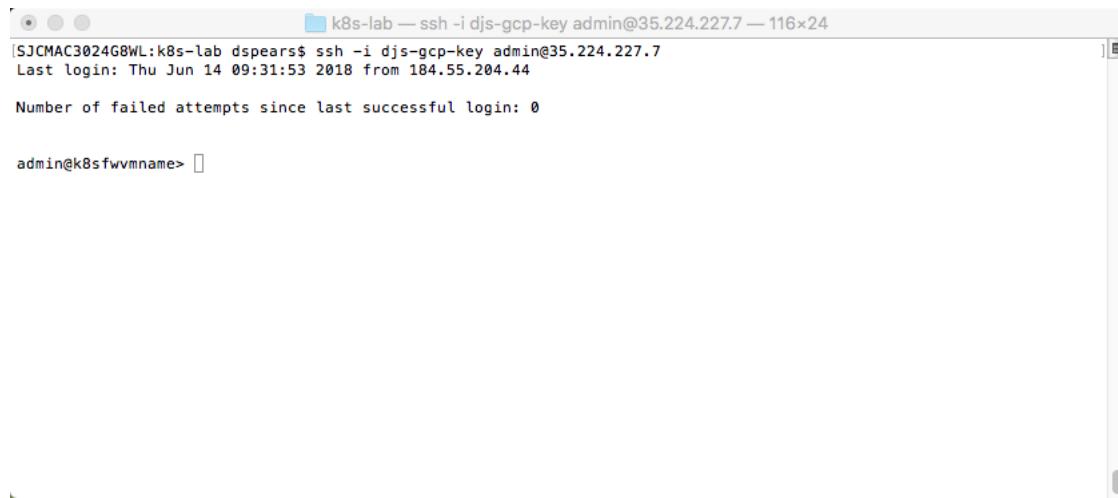
root (hd0,1)
Filesystem type is ext2fs, partition type 0x83
kernel /boot/vmlinuz ro root=/dev/sda2 init=/sbin/init_single_core console=ttyS0,9600n8 console=tty0 quiet alternate_root=/dev/vda2 alternate_root=/dev/xvda2 hugepages=0
[Linux-bzImage, setup=0x3c00, size=0x393330]

eth0      Link encap:Ethernet HWaddr 42:01:0A:05:00:04
          inet addr:10.5.0.4 Bcast:10.5.0.255 Mask:255.255.255.0
          inet6 addr: fe80::4001:aff:fe05:4/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1460 Metric:1
              RX packets:2 errors:0 dropped:0 overruns:0 frame:0
              TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:910 (910.0 b) TX bytes:510 (510.0 b)

k8sfsfwvmname login: □
```

Also, If you wish to SSH into the FW, the SSH key defined in the terraform template can be used. This syntax is:

```
ssh -i <private key file> admin@<ip address of firewall>
```



A screenshot of a terminal window titled "k8s-lab — ssh -i djs-gcp-key admin@35.224.227.7 — 116x24". The session output shows:

```
[SJCMAC3024G8WL:k8s-lab dspears$ ssh -i djs-gcp-key admin@35.224.227.7
Last login: Thu Jun 14 09:31:53 2018 from 184.55.204.44
Number of failed attempts since last successful login: 0

admin@k8sfwvmname> ]
```

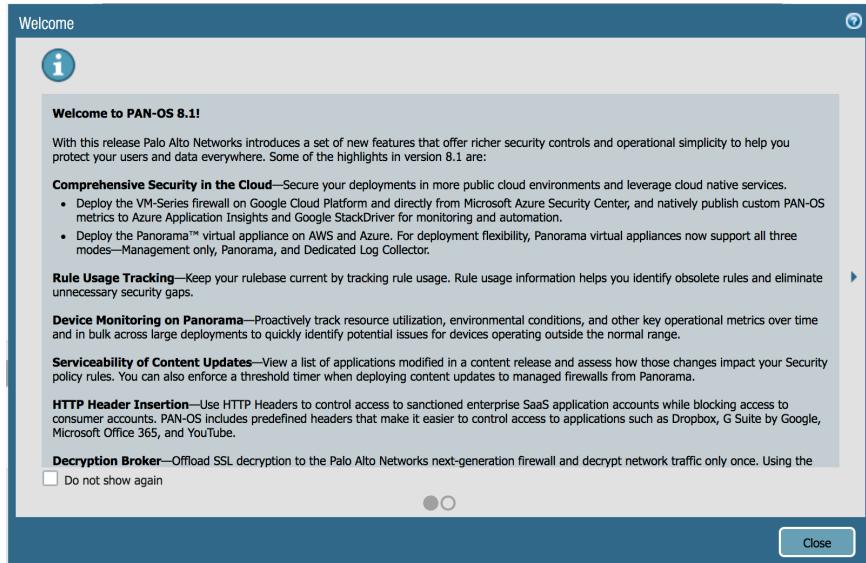
When presented with the login screen you should be able to login to the firewall using (Hint: It's a good idea to jot this password down or save it to a notepad as you will regularly need it):

username: **admin**

password: **Pal0Alt0@123**



Once logged in you will see a welcome screen, dismiss the welcome dialog box by clicking Close.



Click the Policies tab and you will notice a predefined security policy which was imported using the bootstrapping feature. There are also some predefined NAT policies:

The screenshot shows the Palo Alto Networks UI with the 'Policies' tab selected. On the left, a sidebar lists security features: NAT, QoS, Policy Based Forwarding, Decryption, Tunnel Inspection, Application Override, Authentication, and DoS Protection. The main area displays a table of policies. The table has columns for Name, Tags, Type, Zone, Address, User, HIP Profile, and Zone. The data is as follows:

	Name	Tags	Type	Zone	Address	User	HIP Profile	Zone
1	to-guestbook-sec-pol...	none	universal	any	any	any	any	any
2	default-deny-all	none	universal	any	any	any	any	any
3	intrazone-default	none	intrazone	any	any	any	any	(intrazone)
4	interzone-default	none	interzone	any	any	any	any	any

Click on the Dashboard tab, check to verify that the firewall has a serial number. The image defined in the terraform template is a PayGo bundle 1. This was used because a license will be required to view the logs later in the lab.

The screenshot shows the Palo Alto Networks Dashboard interface. The top navigation bar includes tabs for Dashboard, ACC, Monitor, and Policies. Below the navigation is a status bar with 'Layout: 3 Columns', 'Widgets', and 'Last updated: 10:43'. The main content area has two tabs: 'General Information' and 'System Resources'. The 'General Information' tab displays various device details, including:

Parameter	Value
Device Name	k8sfwvmname
MGT IP Address	10.5.0.4 (DHCP)
MGT Netmask	255.255.255.0
MGT Default Gateway	10.5.0.1
MGT IPv6 Address	unknown
MGT IPv6 Link Local Address	fe80::f001:aff:fe05:4/64
MGT IPv6 Default Gateway	
MGT MAC Address	42:01:0a:05:00:04
Model	PA-VM
Serial #	unknown
CPU ID	GCP-D7060200FFFB8B1F
UUID	E650FB87-E4A4-6529-E0A2-210F6BD425AB
VM License	none
VM Mode	GCE
Software Version	8.1.0
GlobalProtect Agent	0.0.0
Application Version	769-4439
URL Filtering Version	0000.00.00.000
GlobalProtect Clientless VPN Version	0
Time	Thu May 3 08:43:14 2018
Uptime	0 days, 0:52:02

The 'System Resources' tab shows management and data plane CPU usage and session counts.

Launch a two tiered application

In this activity, you will:

Optionally: Explore the application's manifest file

Launch a two-tier application within your cluster

Explore aspects of the application

In this activity we will start using Kubernetes specific terms such as Pods, Services, etc.

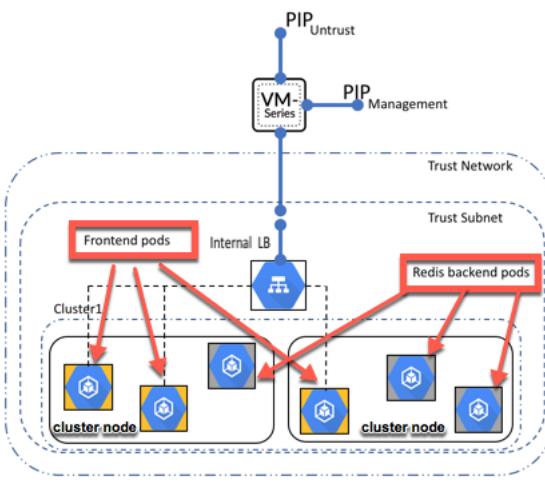
Here is a good primer: <https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/>

Task 1 – Application Deployment YAML file

Guestbooks have been used by businesses for many years as a way to connect with customers and obtain contact information for future events and promotions. Today, businesses such as popular retail stores, 5-star hotels and even small family-owned B & B's are turning to iPad guestbook apps to help them gather information and enhance the customer's "in-biz" experience. Acquiring email addresses and a social media following is a crucial part of any marketing plan. With much of the population using computers on a

daily basis, an email marketing plan is of the utmost importance. Using a guest book app in your store makes collecting email addresses a snap and offers enticing features with which the traditional paper and pen guestbook just can't compete. The guestbook application we will build and secure today could be used for Hotel website visits, shopping sites or any other business that wants to keep track of their customer and provide them with promotions or advertisements.

This lab will deploy the following simple Guestbook application on the cluster nodes created during the Terraform template deployment:



As you can see this is a two-tiered application with Pods that are dedicated to front-end web services and backend DB services.

If interested, the following section dives a bit deeper into the templates being used to create this application. This is a link to the application manifest. Optionally, click the link below and open it in a browser of your choice.

<https://github.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/blob/master/alto-inspect.yaml>

The manifest file declares various aspects of the application. For instance, it tells the orchestrator what type of resources you intend to deploy. In this case we will deploy a 2-tier simple redis application with a fronted and backend tier. The backend tier will consist of a redis-master and redis-slave for db redundancy. Front-end Service:

```
90  apiVersion: v1
91  kind: Service
92  metadata:
93    name: frontend
94  annotations:
95    cloud.google.com/load-balancer-type: "Internal"
96  labels:
97    app: guestbook
98    tier: frontend
99  spec:
100   # if your cluster supports it, uncomment the following to automatically create
101   # an external load-balanced IP for the frontend service.
102   type: LoadBalancer
103   ports:
104     - port: 80
105   selector:
106     app: guestbook
107     tier: frontend
108   ---
109  apiVersion: extensions/v1beta1
110  kind: Deployment
111  metadata:
112    name: frontend
113  spec:
114    replicas: 3
115    template:
116      metadata:
117        labels:
118          app: guestbook
119          tier: frontend
120    spec:
121      containers:
122        - name: php-redis
123          image: gcr.io/google-samples/gb-frontend:v4
124          resources:
125            requests:
126              cpu: 100m
127              memory: 100Mi
128          env:
129            - name: GET_HOSTS_FROM
130              value: dns
131            # If your cluster config does not include a dns service, then to
132            # instead access environment variables to find service host
133            # info, comment out the 'value: dns' line above, and uncomment the
134            # line below:
135            # value: env
136      ports:
137        - containerPort: 80
```

This tells the orchestrator that the service will be exposed via an internal load balancer

Redis-backend-master :

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: redis-master
5    labels:
6      app: redis
7      tier: backend
8      role: master
9  spec:
10   #type: LoadBalancer
11   ports:
12     - port: 6379
13       targetPort: 6379
14   selector:
15     app: redis
16     tier: backend
17     role: master
18   ---
19  apiVersion: extensions/v1beta1
20  kind: Deployment
21  metadata:
22    name: redis-master
23  spec:
24    replicas: 1
25    template:
26      metadata:
27        labels:
28          app: redis
29          role: master
30          tier: backend
31    spec:
32      containers:
33        - name: master
34          image: gcr.io/google_containers/redis:e2e # or just image: redis
35          resources:
36            requests:
37              cpu: 100m
38              memory: 100Mi
39          ports:
40            - containerPort: 6379
```

Redis-backend-slave:

```
42  apiVersion: v1
43  kind: Service
44  metadata:
45    name: redis-slave
46    labels:
47      app: redis
48      tier: backend
49      role: slave
50  spec:
51    #type: LoadBalancer
52    ports:
53      - port: 6379
54    selector:
55      app: redis
56      tier: backend
57      role: slave
58  ---
59  apiVersion: extensions/v1beta1
60  kind: Deployment
61  metadata:
62    name: redis-slave
63  spec:
64    replicas: 2
65    template:
66      metadata:
67        labels:
68          app: redis
69          role: slave
70          tier: backend
71      spec:
72        containers:
73          - name: slave
74            image: gcr.io/google_samples/gb-redisslave:v1
75        resources:
76          requests:
77            cpu: 100m
78            memory: 100Mi
79        env:
80          - name: GET_HOSTS_FROM
81            value: dns
82            # If your cluster config does not include a dns service, then to
83            # instead access an environment variable to find the master
84            # service's host, comment out the 'value: dns' line above, and
85            # uncomment the line below:
86            # value: env
87        ports:
88          - containerPort: 6379
89  ---
```

Even though we have two tiers in our application, only one (the frontend service) is exposed to the outside world via a load balancer. The annotation listed above tells GCP and Kubernetes that the load balancer would be of type: Internal.

More on this later.

Task 2 – Launch the Application

Back in your cloud shell type the following command to deploy the application pods:

```
$ kubectl apply -f https://raw.githubusercontent.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/master/guestbook-all-in-one.yaml
```

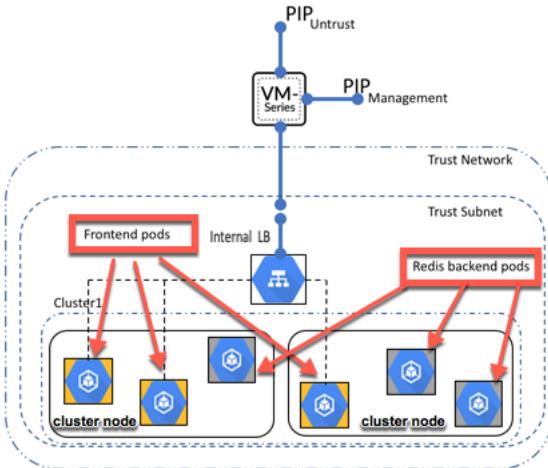


```
paloaltonetworks63_student@gwikelabs-gcp-30219ddb14158a20:~$ kubectl create -f https://raw.githubusercontent.com/PaloAltoNetworks/ignite2018-how16/master/guestbook-all-in-one.yaml
service "redis-master" created
deployment "redis-master" created
service "redis-slave" created
deployment "redis-slave" created
service "frontend" created
deployment "frontend" created
paloaltonetworks63_student@gwikelabs-gcp-30219ddb14158a20:~$
```

You should see the services and deployments being created.

Task 3 – Explore what was just deployed

As shown previously, this is what has been built:



Let's validate this by list the new pods in your cluster. In your cloud shell:

```
$ kubectl get pods -o wide
```

You may see the status as Pending or ContainerCreating. This is usually a normal situation:

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
frontend-685d7ff496-4kvqd	0/1	Pending	0	45s	<none>	<none>
frontend-685d7ff496-6z1mz	0/1	Pending	0	45s	<none>	<none>
frontend-685d7ff496-vxdrr	0/1	Pending	0	45s	<none>	<none>
redis-master-57cc594f67-qp6g5	0/1	Pending	0	52s	<none>	<none>
redis-slave-84845b8fd8-npz2r	0/1	Pending	0	49s	<none>	<none>
redis-slave-84845b8fd8-v5g8l	0/1	Pending	0	49s	<none>	<none>

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
frontend-685d7ff496-h2vwk	0/1	ContainerCreating	0	37s	<none>	gke-cluster-1-default-pool-b04e7f9e-ghbs
frontend-685d7ff496-tbbns	0/1	ContainerCreating	0	37s	<none>	gke-cluster-1-default-pool-b04e7f9e-ghbs
frontend-685d7ff496-z9q49	0/1	ContainerCreating	0	37s	<none>	gke-cluster-1-default-pool-b04e7f9e-s61t
redis-master-57cc594f67-7x4ws	0/1	ContainerCreating	0	37s	<none>	gke-cluster-1-default-pool-b04e7f9e-s61t
redis-slave-84845b8fd8-qdkr5	1/1	Running	0	37s	10.8.1.8	gke-cluster-1-default-pool-b04e7f9e-ghbs
redis-slave-84845b8fd8-rk4xx	0/1	ContainerCreating	0	37s	<none>	gke-cluster-1-default-pool-b04e7f9e-s61t

By executing the **kubectl get pods -o wide** again, you start seeing that the Ready and Status of pods change as they start up. Verify that all the pods get to created and running.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
frontend-685d7ff496-c8fd7	1/1	Running	0	1m	10.8.1.9	gke-cluster-1-default-pool-168614d7-5b66
frontend-685d7ff496-njtw8	1/1	Running	0	1m	10.8.1.8	gke-cluster-1-default-pool-168614d7-5b66
frontend-685d7ff496-vfmrh	1/1	Running	0	1m	10.8.0.10	gke-cluster-1-default-pool-168614d7-bmlv
redis-master-57cc594f67-7g6hn	1/1	Running	0	1m	10.8.0.8	gke-cluster-1-default-pool-168614d7-bmlv
redis-slave-84845b8fd8-8n2fb	1/1	Running	0	1m	10.8.1.7	gke-cluster-1-default-pool-168614d7-5b66
redis-slave-84845b8fd8-qhadt	1/1	Running	0	1m	10.8.0.9	gke-cluster-1-default-pool-168614d7-bmlv

NOTE: In most cases the containers will move to a running status within 5min. If they stay in the Pending state for more than 5min, it is often quicker to delete the containers and redeploy them. To do this enter the following command:

```
$ kubectl delete -f https://raw.githubusercontent.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/master/guestbook-all-in-one.yaml
```

and then redeploy the containers:

```
$ kubectl apply -f https://raw.githubusercontent.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/master/guestbook-all-in-one.yaml
```

davejspears@cloudshell:~ (djs-gcp-2018)\$	kubectl delete -f https://raw.githubusercontent.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/master/guestbook-all-in-one.yaml
	service "redis-master" deleted
	deployment "redis-master" deleted
	service "redis-slave" deleted
	deployment "redis-slave" deleted
	service "frontend" deleted
	deployment "frontend" deleted
davejspears@cloudshell:~ (djs-gcp-2018)\$	kubectl apply -f https://raw.githubusercontent.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/master/guestbook-all-in-one.yaml
	service "redis-master" created
	deployment "redis-master" created
	service "redis-slave" created
	deployment "redis-slave" created
	service "frontend" created
	deployment "frontend" created
davejspears@cloudshell:~ (djs-gcp-2018)\$	

Next let's look at the load balancing service for the front-end pod. Execute the following command in the shell:

```
$ kubectl get svc
```

If you notice the "external-ip" of the frontend service's load balancer you will see that

- a) Initially this may show as pending:

```
paloaltonetworks99_student@qwiklabs-gcp-e6ef6351e932f0e2:~$ kubectl get svc
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP    PORT(S)        AGE
frontend   LoadBalancer 10.11.251.50 <pending>     80:31644/TCP  54s
kubernetes ClusterIP  10.11.240.1  <none>        43/TCP       13m
redis-master ClusterIP 10.11.253.152 <none>        6379/TCP    55s
redis-slave  ClusterIP 10.11.243.162 <none>        6379/TCP    55s
paloaltonetworks99_student@qwiklabs-gcp-e6ef6351e932f0e2:~$
```

Repeating the "kubectl get svc" command will eventually show a private ip address for the internal load balancer:

```
..._labs-gcp-30219ddb14158a20 x + ..._x
paloaltonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$ kubectl get svc
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP    PORT(S)        AGE
frontend   LoadBalancer 10.11.249.87  10.5.2.5     80:30895/TCP  1m
kubernetes ClusterIP  10.11.240.1  <none>        43/TCP       9m
redis-master ClusterIP 10.11.244.152 <none>        6379/TCP    1m
redis-slave  ClusterIP 10.11.251.147 <none>        6379/TCP    1m
paloaltonetworks63_student@qwiklabs-gcp-30219ddb14158a20:~$
```

- b) The private ip address is from the trust subnet's CIDR block

The screenshot shows the Google Cloud Platform VPC network interface. On the left, there is a sidebar with icons for VPC network, VPC networks, External IP addresses, and a search bar. The main area displays a table for VPC networks. It shows two entries: 'trust' and 'untrust'. The 'trust' entry has a 'CLUSTER-IP' of 10.11.249.87 and an 'EXTERNAL-IP' of 10.5.2.5. The 'untrust' entry has a 'CLUSTER-IP' of 10.11.251.147 and an 'EXTERNAL-IP' of 6379/TCP. The table also includes columns for 'PORT(S)', 'AGE', and 'Subnets'.

Name	Type	Cluster-IP	External-IP	Port(s)	Age
trust	LoadBalancer	10.11.249.87	10.5.2.5	80:30895/TCP	1m
untrust	LoadBalancer	10.11.251.147	<none>	6379/TCP	1m

To see the new Kubernetes resources in the GCP console navigate to Kubernetes Engine > Workloads:

The screenshot shows the Google Cloud Platform Kubernetes Engine interface. On the left, there is a sidebar with icons for Home, Compute Engine, VPC network, Kubernetes Engine, Cloud Functions, Storage, and Bigtable. The main area displays a table for workloads. It shows three entries: 'frontend', 'redis-master', and 'redis-slave'. The table includes columns for 'Name', 'Status', 'Type', 'Pods', 'Namespace', and 'Cluster'. A dropdown menu is open over the 'Workloads' section of the sidebar, showing options like 'Workloads', 'Services', 'Configuration', 'Storage', and 'Cloud Launcher'.

Name	Status	Type	Pods	Namespace	Cluster
frontend	OK	Deployment	3/3	default	cluster-1
redis-master	OK	Deployment	1/1	default	cluster-1
redis-slave	OK	Deployment	2/2	default	cluster-1

Here you can see the workloads that were just deployed. You can click on any of these to get more information:

The screenshot shows the Google Cloud Platform interface for the Kubernetes Engine. On the left, there's a sidebar with icons for Kubernetes clusters, Workloads (which is selected), Services, Configuration, and Storage. The main area is titled "Workloads" with "REFRESH" and "DEPLOY" buttons. A sub-section explains what workloads are: "Workloads are deployable units of computing that can be created and managed in a cluster." Below this is a table with the following data:

Name	Status	Type	Pods	Namespace	Cluster
frontend	OK	Deployment	3/3	default	cluster-1
redis-master	OK	Deployment	1/1	default	cluster-1
redis-slave	OK	Deployment	2/2	default	cluster-1

The Internal Load Balancer can be seen by navigating to Network Services > Load Balancing

This screenshot shows the Google Cloud Platform Network Services section. The sidebar includes Home, Compute Engine, VPC network, and Kubernetes Engine (selected). Under PRODUCTS, there are links for SQL and Spanner. Under NETWORKING, there are links for VPC network, Network services (which is selected), Hybrid Connectivity, and Network Security. A dropdown menu for "Load balancing" is open, showing "Cloud DNS". A terminal window shows the output of the command "kubectl get svc":

```
158a20:~$ kubectl get svc
NAME           PORT(S)        AGE
Cloud DNS      80:30895/TCP   1m
Cloud CDN      80.2.5.443/TCP  9m
Cloud Launcher  6379/TCP     1m
Cloud SQL      5432/TCP     1m
```

Click on the load balancer to get more details:

Google Cloud Platform

Network services

Load balancing

CREATE LOAD BALANCER

REFRESH

DELETE

Load balancers

Cloud DNS

Cloud CDN

Name

a278e3d014eef11e893ba42010a80016

Protocol

TCP (Internal)

Backends

1 regional backend service (1 instance group)

To edit load balancing resources like forwarding rules and target proxies, go to the [advanced menu](#).

This also shows that the internal load balancer is on the trust network and has an internal IP address that will be used later in the firewall configuration:

Google Cloud Platform

Network services

Load balancing

Load balancer details

EDIT

DELETE

a278e3d014eef11e893ba42010a80016

Frontend

Protocol

TCP

Subnetwork

trust-subnet (10.5.2.0/24)

IP:Ports

10.5.2.5:80

Backend

Region: us-central1 Network: trust Endpoint protocol: TCP Session affinity: None Health check: k8s-624d9abdeb1869c8-node

Advanced configurations

Instance group

k8s-ig-624d9abdeb1869c8

Zone

us-central1-a

Healthy

2 / 2

Autoscaling

Off

Cloud Launcher

Securing Inbound Traffic

In this activity, you will:

Secure traffic that is inbound to your frontend service

Validate that traffic is visible in the Firewall logs

Task 1 – Note the Internal Load Balancer’s IP Address

From the previous activity: make a note of the internal load balancer’s ip address. (very probable it is 10.5.2.5)

Task 2 – Update the Firewall’s NAT Policy

Login in to the VM-Series firewall using the management interface’s ip address.

Hint: Check Activity 1, Task 2 if you don’t have ip address jotted down

Click the Policies Tab and navigate to NAT on the left. Click on the to-guestbook-nat-rule to edit it:

The screenshot shows the Palo Alto Networks Management Interface. The top navigation bar includes tabs for Dashboard, ACC, Monitor, Policies (which is selected), Objects, Network, and Dev. On the left, a sidebar lists security features: Security, NAT (selected), QoS, Policy Based Forwarding, Decryption, Tunnel Inspection, Application Override, Authentication, and DoS Protection. The main pane displays a table of NAT policies. The first row, labeled '1', contains the rule 'to-guestbook-nat-rule'. A red arrow points to this row. The second row, labeled '2', contains the rule 'outbound'. The columns in the table are Name, Tags, Source Zone, Destination Zone, and Destination Interface. The 'to-guestbook-nat-rule' row shows 'none' in the Tags column, 'untrust' in both Source and Destination Zone columns, and 'ethernet1/1' in the Destination Interface column.

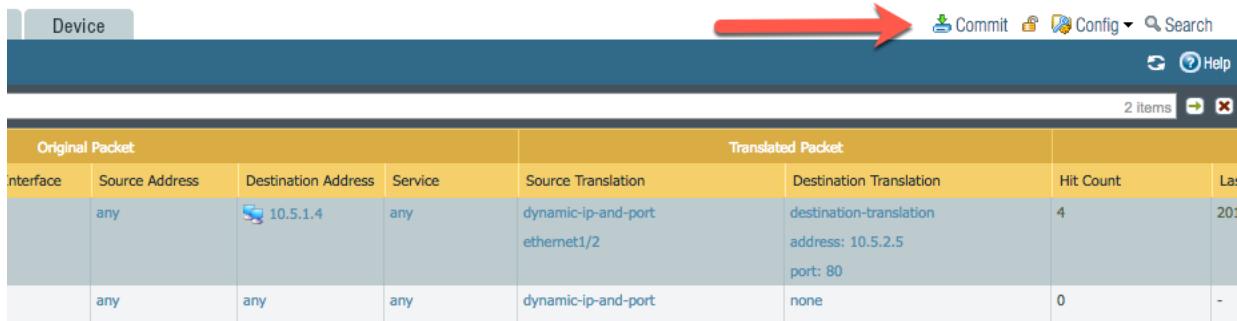
	Name	Tags	Source Zone	Destination Zone	Destination Interface
1	to-guestbook-nat-rule	none	untrust	untrust	ethernet1/1
2	outbound	none	any	untrust	ethernet1/1

Click on the Translated Packet tab:

The screenshot shows the 'NAT Policy Rule' dialog box. The tabs at the top are General, Original Packet, and Translated Packet (which is selected). The 'Source Address Translation' section has 'Dynamic IP And Port' selected as the Translation Type, 'Interface Address' selected as the Address Type, 'ethernet1/2' selected as the Interface, and 'None' selected as the IP Address. The 'Destination Address Translation' section has 'Static IP' selected as the Translation Type, '10.5.2.5' entered in the Translated Address field, and '80' entered in the Translated Port field. At the bottom right are OK and Cancel buttons.

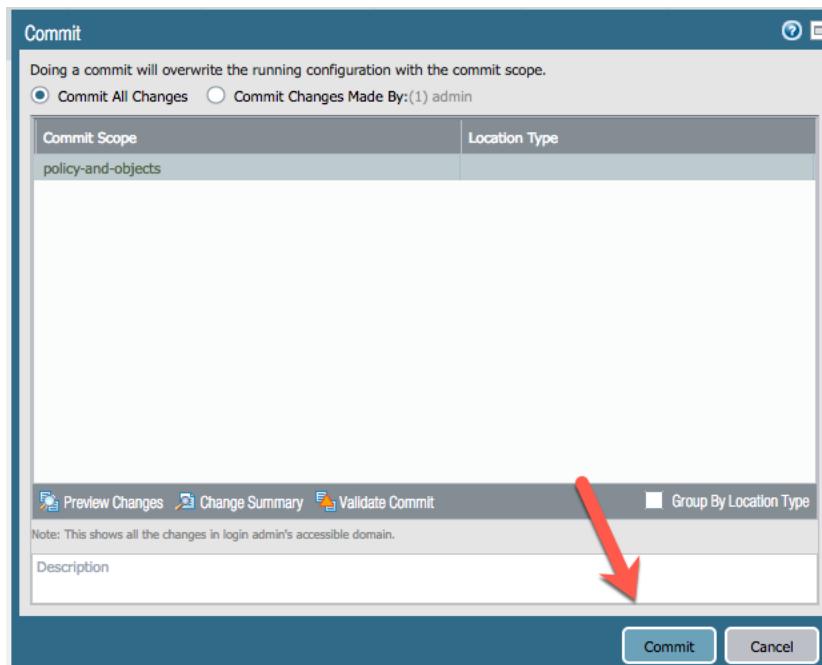
In the Translated Address field, enter the load balancer’s ip address from Task 1 and click OK. It might not need to be changed.

If a change was needed, click the commit link on the top right



Original Packet								Translated Packet	
Interface	Source Address	Destination Address	Service	Source Translation	Destination Translation	Hit Count	Last Hit		
	any	10.5.1.4	any	dynamic-ip-and-port ethernet1/2	destination-translation address: 10.5.2.5 port: 80	4	2019-07-10 10:45:20		
	any	any	any	dynamic-ip-and-port	none	0	-		

When the next dialog box opens, click the commit button



Task 3 – Connect to the Guestbook Frontend

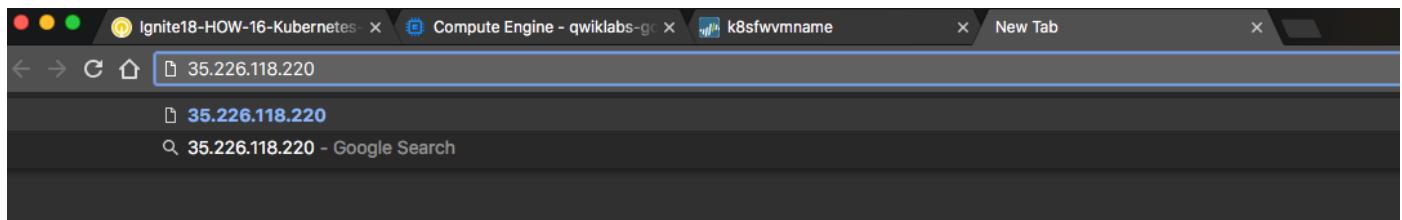
The VM-Series is now protecting your Kubernetes workload. In order to connect to the guestbook's frontend service, you will connect to the untrust ip-address of the firewall. Navigate to the Compute Engine> VM Instances screen.

Then click on the firewall and copy the External IP address of the untrust interface:

The screenshot shows the Google Cloud Platform Compute Engine interface. On the left, there's a sidebar with options: App Engine, Compute Engine (selected), Kubernetes Engine, and Cloud Functions. Under Compute Engine, a dropdown menu is open with 'VM instances' selected. The main area shows 'VM instance details' for an instance named 'firewall1'. The 'Network interfaces' table lists three interfaces: 'management' (internal IP 10.5.0.4, external IP 35.225.41.108), 'untrust' (internal IP 10.5.1.4, external IP 35.226.118.220), and 'trust' (internal IP 10.5.2.4, external IP None). A red arrow points to the 'External IP' column for the 'untrust' interface.

Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	IP forwarding
management	management-subnet	10.5.0.4	—	35.225.41.108 (ephemeral)	On
untrust	untrust-subnet	10.5.1.4	—	35.226.118.220 (ephemeral)	
trust	trust-subnet	10.5.2.4	—	None	

In your browser navigate to the untrust-ip address of the firewall



And you should be greeted with the frontend of your guestbook application:

A screenshot of a web browser window. The address bar shows the URL 'Not Secure | 35.226.118.220'. Below the address bar, the browser displays the text 'Guestbook'. The main content area shows a form with a text input field containing 'Hello World' and a blue 'Submit' button.

Type a message in the dialog box and click Submit. The message should be echoed on the website.

Guestbook

Messages

Hello World

Submit

Open the firewall interface and navigate to the monitor tab. I have added the NAT Dest IP Column so it is possible to see the address translation to the internal load balancer IP address.

The screenshot shows the Palo Alto Networks Firewall interface. The top navigation bar includes Dashboard, ACC, Monitor (which is selected), Policies, Objects, Network, and Device. On the left, there's a sidebar with 'Logs' expanded, showing categories like Traffic, Threat, URL Filtering, WildFire Submissions, Data Filtering, HIP Match, User-ID, Tunnel Inspection, Configuration, System, and Alarms. The main pane displays a table of logs. The columns are: Receive Time, Type, From Zone, To Zone, Source, NAT Dest IP, Destination, To Port, Application, Action, Rule, Session End Reason, and Bytes. The data in the table is as follows:

Receive Time	Type	From Zone	To Zone	Source	NAT Dest IP	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
05/03 09:44:38	end	untrust	web	184.55.204.44	10.5.2.5	10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy	aged-out	5.0k
05/03 09:43:57	end	untrust	web	184.55.204.44	10.5.2.5	10.5.1.4	80	incomplete	allow	to-guestbook-sec-policy	tcp-fin	848
05/03 09:43:41	start	untrust	web	184.55.204.44	10.5.2.5	10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy	n/a	679
05/03 09:42:32	start	untrust	web	184.55.204.44	10.5.2.5	10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy	n/a	751

Securing Outbound Traffic

In this activity, you will:

Secure outbound traffic from the cluster

Validate traffic is in the Firewall logs

Task 1 – Add Outbound Route

To secure any traffic that is originating from within the cluster we need to add a couple of routes in the GCP console. Go to VPC Network > Routes

The screenshot shows the GCP VPC Network Routes page under the Networking section. The left sidebar lists 'VPC network', 'Network services', and 'Hybrid Connectivity'. The right panel shows a list of routes. One route is highlighted with a gray background: 'Routes'.

You will see a few default routes added by Kubernetes during the initial template launch:

The screenshot shows the Google Cloud Platform VPC network Routes page. On the left, there's a sidebar with 'VPC network' selected. The main area has tabs for 'Routes' (selected), '+ CREATE ROUTE', 'REFRESH', and 'DELETE'. A table lists several default routes:

	default-route-ae75e0c3534835f3	0.0.0.0/0	1000	None	Default internet gateway	trust
	default-route-c28522ea8b1a14f8	10.5.0.0/24	1000	None	Virtual network	management
	default-route-da50f4fa1ac6d95a	10.132.0.0/20	1000	None	Virtual network	default
	default-route-e6bcc1325d731489	10.148.0.0/20	1000	None	Virtual network	default
	default-route-e6d1ad29c60cfdfc	10.146.0.0/20	1000	None	Virtual network	default
	default-route-e76604a7e4330fd1	10.158.0.0/20	1000	None	Virtual network	default
	default-route-f13e6e84b623a1bc	0.0.0.0/0	1000	None	Default internet gateway	management
	default-route-f59dc1e5fb74d0d3	0.0.0.0/0	1000	None	Default internet gateway	untrust
	default-route-f83453670643dc95	10.138.0.0/20	1000	None	Virtual network	default
	gke-cluster-1-f2e29940-00b35403-4eee-11e8-93ba-42010a80016f	10.8.0.0/24	1000	None	gke-cluster-1-default-pool-b04e7f9e-s61t (Zone us-central1-a)	trust
	gke-cluster-1-f2e29940-00cace88-4eee-11e8-93ba-42010a80016f	10.8.1.0/24	1000	None	gke-cluster-1-default-pool-b04e7f9e-ghbs (Zone us-central1-a)	trust

Click on Create Route:

The screenshot shows the same VPC network Routes page as before, but with a red arrow pointing to the '+ CREATE ROUTE' button.

Create a route with the following Parameters:

The screenshot shows the 'Create a route' dialog box. On the left, there's a sidebar with 'VPC network' selected. The main area has a back arrow and the title 'Create a route'. The form fields are:

- Name: secure-outbound
- Description (Optional):
- Network: trust
- Destination IP range: 0.0.0.0/0
- Priority: 100
- Instance tags (Optional):
- Next hop: Specify an instance
- Next hop instance: firewall

At the bottom are 'Create' and 'Cancel' buttons, and a note: 'Equivalent REST or command line'.

Name: secure-outbound
Network: trust
Destination IP Range: 0.0.0.0/0
Priority: 100
Next hop: Specify an instance
Next Hop Instance: firewall1

And then click Create

Task 2 – Add Kube-Apiserver route

Navigate to Kubernetes Engine > Kubernetes clusters and click on the cluster.

The screenshot shows the GCP console interface. On the left, there's a sidebar with sections for COMPUTE, STORAGE, and BIOTABLE. Under COMPUTE, 'Kubernetes Engine' is selected, and its sub-menu 'Kubernetes clusters' is highlighted. The main content area shows a table titled 'Kubernetes clusters' with one row: 'cluster-1'. A red box highlights the 'cluster-1' row. Below the table, there are tabs for 'Details', 'Storage', and 'Nodes', with 'Details' being the active tab. The 'Details' tab displays various cluster configurations, including 'Master version' (1.8.8-gke.0), 'Endpoint' (35.226.171.41), 'Client certificate' (Enabled), 'Kubernetes alpha features' (Disabled), 'Total size' (2), and 'Master zone' (us-central1-a). A red arrow points to the 'Endpoint' value.

When the cluster details page opens, identify the kube-apiserver ip address. Labeled as Endpoint in the page. Copy this address:

This screenshot shows the 'Kubernetes clusters' details page for 'cluster-1'. The 'Endpoint' field is highlighted with a red box and a red arrow pointing to it. The value '35.226.171.41' is displayed next to the label 'Endpoint'. Other cluster details shown include Master version (1.8.8-gke.0), Client certificate (Enabled), Kubernetes alpha features (Disabled), Total size (2), and Master zone (us-central1-a).

Navigate back to VPC Networks > Routes and create another route that will allow the kube-apiserver to perform health checks without the firewall getting in the way:

The screenshot shows the Google Cloud Platform interface for managing VPC networks. On the left, there's a navigation menu under 'NETWORKING' with options like 'VPC network', 'Network services', and 'Hybrid Connectivity'. Under 'VPC network', there are sub-options: 'VPC networks', 'External IP addresses', 'Firewall rules', 'Routes' (which is selected and highlighted in grey), 'VPC network peering', and 'Shared VPC'. The main content area is titled 'Routes' and shows a table of existing routes. The table has columns for 'Route ID', 'Destination IP range', and 'Next hop'. A red arrow points to the '+ CREATE ROUTE' button at the top right of the table.

Create a route with the following parameters:

The screenshot shows the 'Create a route' dialog box. On the left, there's a sidebar with 'VPC network' and 'Routes' selected. The main form fields are: 'Name' (apiserver-bypass), 'Description (Optional)', 'Network' (trust), 'Destination IP range' (35.226.171.41/32), 'Priority' (100), 'Instance tags (Optional)', 'Next hop' (Default internet gateway). At the bottom are 'Create' and 'Cancel' buttons. A red arrow points to the 'Create' button.

Name: apiserver-bypass

Network: trust

Destination IP Range: [Insert kube-apiserver IP] /32

Priority: 100

Next hop: Default internet gateway

And then click Create.

Navigate back to the firewall monitor tab and you can now see outbound traffic as well from the cluster nodes.

Receive Time	Type	From Zone	To Zone	Source	NAT Source IP	Destination	To Port	Action	Rule	Session End Reason	Bytes
05/03 10:57:41	start	web	untrust	10.5.2.2	10.5.1.4	74.125.70.95	443	google-base	allow	interzone-default	n/a
05/03 10:57:41	start	web	untrust	10.5.2.2	10.5.1.4	74.125.70.95	443	ssl	allow	interzone-default	n/a
05/03 10:57:40	end	web	untrust	10.5.2.2	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:39	end	web	untrust	10.8.1.4	10.5.1.4	10.11.240.10	53	dns	allow	interzone-default	aged-out
05/03 10:57:39	end	web	untrust	10.5.2.3	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:36	end	web	untrust	10.5.2.2	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:35	end	web	untrust	10.5.2.2	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:35	end	web	untrust	10.5.2.2	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:34	end	web	untrust	10.5.2.3	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:34	end	web	untrust	10.8.1.4	10.5.1.4	10.11.240.10	53	dns	allow	interzone-default	aged-out
05/03 10:57:34	end	web	untrust	10.8.1.4	10.5.1.4	10.11.240.10	53	dns	allow	interzone-default	aged-out
05/03 10:57:32	start	web	untrust	10.5.2.3	10.5.1.4	74.125.124.95	443	google-base	allow	interzone-default	n/a
05/03 10:57:32	start	web	untrust	10.5.2.3	10.5.1.4	74.125.124.95	443	ssl	allow	interzone-default	n/a
05/03 10:57:31	start	web	untrust	10.5.2.2	10.5.1.4	74.125.70.95	443	google-base	allow	interzone-default	n/a
05/03 10:57:31	start	web	untrust	10.5.2.2	10.5.1.4	74.125.70.95	443	ssl	allow	interzone-default	n/a
05/03 10:57:29	end	web	untrust	10.8.1.4	10.5.1.4	10.11.240.10	53	dns	allow	interzone-default	aged-out
05/03 10:57:29	end	web	untrust	10.8.1.4	10.5.1.4	10.11.240.10	53	dns	allow	interzone-default	aged-out
05/03 10:57:28	end	web	untrust	10.5.2.2	10.5.1.4	74.125.70.95	443	google-base	allow	interzone-default	tcp-fin
05/03 10:57:28	end	web	untrust	10.5.2.3	10.5.1.4	108.177.112.95	443	google-base	allow	interzone-default	tcp-fin

These source addresses are the instance addresses of the Kubernetes cluster servers:

Name	Zone	Recommendation	Internal IP
firewall1	us-central1-a		10.5.0.4
gke-cluster-1-default-pool-e515ebaf-dhz6	us-central1-a		10.5.2.3
gke-cluster-1-default-pool-e515ebaf-fzdg	us-central1-a		10.5.2.2

Inspection of Inter-Pod traffic

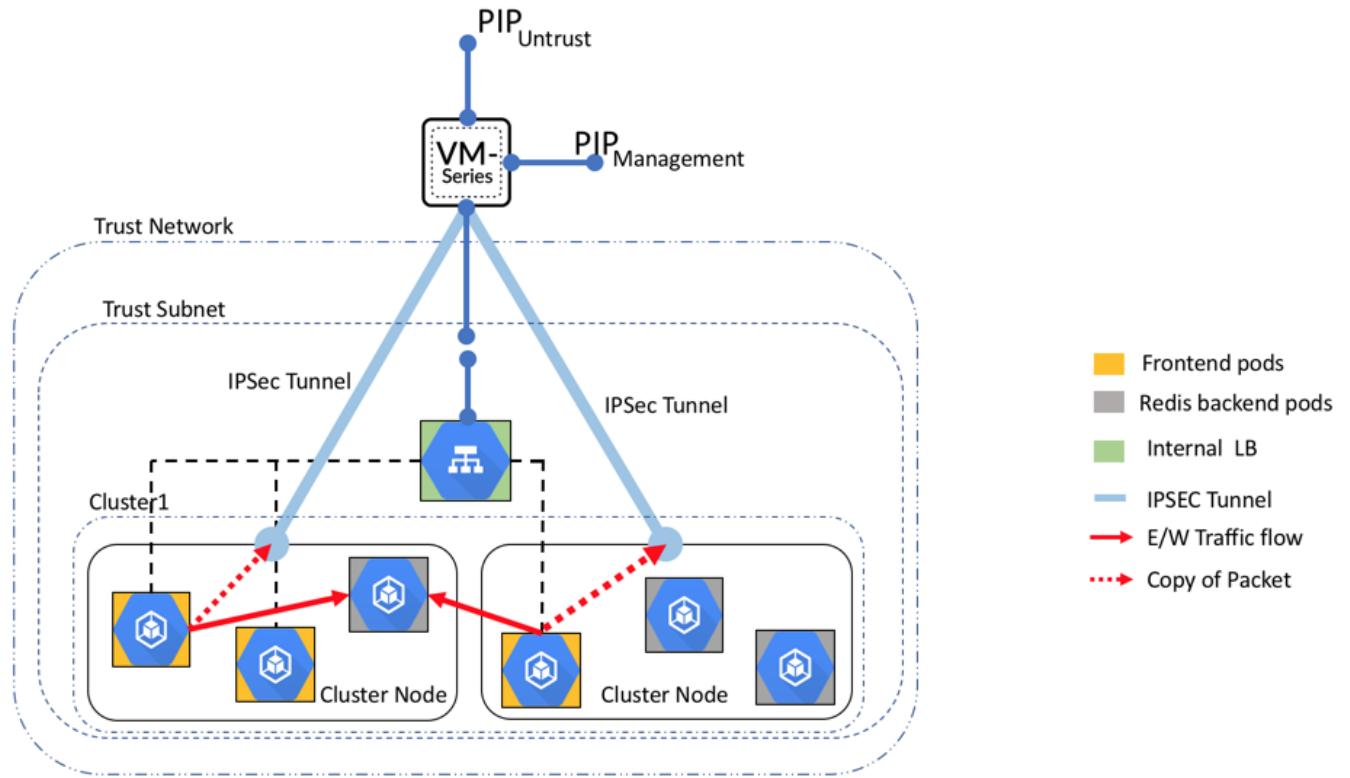
In this activity, you will:

Configure the k8s cluster to be able to inspect Pod-to-Pod traffic by the firewall

Validate that traffic is visible in the Firewall logs

One of the key challenges when deploying microservices is the lack of visibility of traffic that is flowing between services (or pods) within the cluster.

In this activity you will configure your Kubernetes nodes to redirect a copy of the packet to the VM-Series, which will provide visibility into the traffic between pods and services within your cluster. Task 1 will deploy a DaemonSet which will send a copy of the intra pod traffic between the frontend and backend pods over an IPSec tunnel to the PANW firewall for inspection. The following drawing provides a visual of this concept:



Task 1 – Deploy a DaemonSet

A DaemonSet is a Kubernetes resource that ensures that a copy of a pod is run on all (or some) of your nodes

<https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

In your cloud Shell, run the following command:

```
$ kubectl apply -f https://raw.githubusercontent.com/PaloAltoNetworks/GCP-k8s-east-west-visibility/master/alto-inspect.yaml
```

NOTE: The following link can be used as the WWCE is a private repo.

```
$ kubectl apply -f https://raw.githubusercontent.com/djspears/GCP-K8s/master/alto-inspect.yaml
```

```
paloaltonetworks122_student@qwiklabs-gcp-250f11d5b7b9ef75:~$ kubectl apply -f https://raw.githubusercontent.com/PaloAltoNetworks/ignite2018-howt16/master/alto-inspect.yaml
daemonset "alto-inspect" created
paloaltonetworks122_student@qwiklabs-gcp-250f11d5b7b9ef75:~$
```

You should see “daemonset alto-inspect” created in the shell. To verify that the pods were launched you can list all pods in all namespaces:

```
$ kubectl get pods --all-namespaces -o wide
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
default	frontend-685d7ff496-gfk9d	1/1	Running	0	3m	10.28.0.9	gke-cluster-1-default-pool-6bd5f20-nndj
default	frontend-685d7ff496-kpf9g	1/1	Running	0	3m	10.28.0.10	gke-cluster-1-default-pool-6bd5f20-nndj
default	frontend-685d7ff496-n81zn	1/1	Running	0	3m	10.28.1.9	gke-cluster-1-default-pool-6bd5f20-x3hg
default	redis-master-57cc594f67-gbvzq	1/1	Running	0	4m	10.28.1.7	gke-cluster-1-default-pool-6bd5f20-x3hg
default	redis-slave-84845b8fd8-7frjt	1/1	Running	0	4m	10.28.1.8	gke-cluster-1-default-pool-6bd5f20-x3hg
default	redis-slave-84845b8fd8-fztmf	1/1	Running	0	4m	10.28.0.8	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	alto-inspect-g2z4f	1/1	Running	0	3m	10.5.2.3	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	alto-inspect-xqcnt	1/1	Running	0	3m	10.5.2.2	gke-cluster-1-default-pool-6bd5f20-x3hg
kube-system	event-exporter-v0.1.7-5c4d5550cf-6z24j	2/2	Running	0	1m	10.28.0.4	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	fluentd-gcp-v2.0.9-kq42h	2/2	Running	0	13m	10.28.0.2	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	fluentd-gcp-v2.0.9-x57pb	2/2	Running	0	13m	10.28.1.2	gke-cluster-1-default-pool-6bd5f20-x3hg
kube-system	heapster-v1.4.3-699fc4bd5b-2d9sq	3/3	Running	0	12m	10.28.0.7	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	kube-dns-778977457c-qd79d	3/3	Running	0	14m	10.28.1.4	gke-cluster-1-default-pool-6bd5f20-x3hg
kube-system	kube-dns-778977457c-w9cv7	3/3	Running	0	12m	10.28.0.6	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	kube-dns-autoscaler-7db47cb9b7-sqjmv	1/1	Running	0	14m	10.28.0.5	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	kube-proxy-gke-cluster-1-default-pool-6bd5f20-nndj	1/1	Running	0	13m	10.5.2.3	gke-cluster-1-default-pool-6bd5f20-nndj
kube-system	kube-proxy-gke-cluster-1-default-pool-6bd5f20-x3hg	1/1	Running	0	13m	10.5.2.2	gke-cluster-1-default-pool-6bd5f20-x3hg
kube-system	kubernetes-dashboard-768854d6dc-cm88g	1/1	Running	0	14m	10.28.1.5	gke-cluster-1-default-pool-6bd5f20-x3hg
kube-system	17-default-backend-6497bcd4d-zvg78	1/1	Running	0	14m	10.28.1.3	gke-cluster-1-default-pool-6bd5f20-x3hg

By deploying the alto-inspect DaemonSet, we have now established an IPSec tunnel between each of the nodes in the cluster to the trust interface of the firewall.

In order to ensure that the tunnels are up, log into the firewall and Navigate to Network > IPSec Tunnels. If the status is green, the tunnels are up.



Verify that the Peer Address matches the private ip addresses of your Kubernetes nodes

The screenshot shows two main sections. On the left, the 'IKE Gateways' table lists two entries: 'test-ike-gw' (Peer Address 10.5.2.3) and 'test-ike-gw2' (Peer Address 10.5.2.2). A red box highlights the 'Peer Address' column for both entries. On the right, the 'VM instances' table lists two entries: 'gke-cluster-1-default-pool-6bd55f20-nndj' (Internal IP 10.5.2.3) and 'gke-cluster-1-default-pool-6bd55f20-x3hg' (Internal IP 10.5.2.2). A red arrow points from the highlighted 'Peer Address' column in the IKE Gateways table to the 'Internal IP' column in the VM instances table.

Name	Peer Address	Interface	Local IP	ID	Type
test-ike-gw	10.5.2.3	ethernet1/2			
test-ike-gw2	10.5.2.2	ethernet1/2			

Name	Zone	Recommendation	Internal IP
gke-cluster-1-default-pool-6bd55f20-nndj	us-west1-a		10.5.2.3
gke-cluster-1-default-pool-6bd55f20-x3hg	us-west1-a		10.5.2.2

If not, in the firewall, go to Network> IKE Gateways and modify the test-ike-gw and test-ike-gw2 peer addresses to match the internal ip addresses of the nodes. It doesn't matter which gateway gets which peer ip address.

The screenshot shows the 'IKE Gateway' configuration dialog. The 'General' tab is selected. The 'Name' field is set to 'test-ike-gw'. The 'Peer IP Address Type' is set to 'IP' with the value '10.5.2.3'. The 'Authentication' method is set to 'Pre-Shared Key'. The 'Pre-shared Key' field contains '*****'. The 'Confirm Pre-shared Key' field also contains '*****'. The 'OK' button is visible at the bottom right.

Name	Peer Address	Interface	IP	ID	Type
test-ike-gw	10.5.2.3	ethernet1/2			
test-ike-gw2	10.5.2.2	ethernet1/2			

IF any changes were needed, be sure to Commit the changes.

Task 2 – Verify firewall logs to see inter-pod traffic

Go to guestbook and submit another message. You should see the traffic between the pods.

Guestbook

Another Message|

Submit

Hello World!

Another Message

The screenshot shows the Palo Alto Networks Firewall interface. The top navigation bar includes tabs for Dashboard, ACC, Monitor (which is selected), Policies, Objects, Network, and Device. On the left, a sidebar menu under 'Logs' shows options like Traffic, Threat, URL Filtering, WildFire Submissions, Data Filtering, HIP Match, User-ID, Tunnel Inspection, Configuration, System, Alarms, Authentication, and Unified. The main area displays a table of logs. A red box highlights the first two rows of the table, which show traffic from 'web' zone to 'web' zone, source '10.8.1.9', destination '10.8.0.8', port '6379', and application 'redis'. The table has columns for Receive Time, Type, From Zone, To Zone, Source, Source User, Destination, To Port, Application, Action, Rule, and Session Reason.

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Sess Reas
05/13 14:59:53	start	web	web	10.8.1.9		10.8.0.8	6379	redis	allow	intrazone-default	n/a
05/13 14:59:53	start	web	web	10.8.1.9		10.8.1.6	53	dns	allow	intrazone-default	n/a
05/13 14:59:53	start	untrust	web	12.130.117.201		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy	n/a
05/13 14:59:53	start	web	web	10.8.1.2		169.254.169.254	53	dns	allow	intrazone-default	n/a
05/13 14:59:52	end	web	web	10.8.0.9		10.8.1.6	53	dns	allow	intrazone-default	aged
05/13 14:59:52	start	web	web	10.8.0.2		64.233.182.95	443	google-base	allow	intrazone-default	n/a
05/13 14:59:52	start	web	web	10.8.0.2		64.233.182.95	443	ssl	allow	intrazone-default	n/a
05/13 14:59:52	start	web	web	10.8.0.2		169.254.169.254	53	dns	allow	intrazone-default	n/a

If there is a lot of traffic that is making it difficult to see the intra pod traffic, it is possible to filter by application:

The screenshot shows the Palo Alto Networks Firewall interface with a search filter '(app eq redis)' applied. The table of logs now only displays entries related to the redis application. A red arrow points to the 'Type' column header. The table has columns for Receive Time, Type, From Zone, To Zone, Source, Source User, Destination, To Port, Application, and Action.

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action
05/13 15:00:07	end	web	web	10.8.1.9		10.8.0.8	6379	redis	allow
05/13 15:00:06	end	web	web	10.8.0.9		10.8.0.8	6379	redis	allow
05/13 15:00:02	end	web	web	10.8.1.9		10.8.0.8	6379	redis	allow
05/13 14:59:53	start	web	web	10.8.1.9		10.8.0.8	6379	redis	allow
05/13 14:59:52	start	web	web	10.8.0.9		10.8.0.8	6379	redis	allow
05/13 14:59:48	start	web	web	10.8.1.9		10.8.0.8	6379	redis	allow

Take note of the Source and Destination addresses of the redis application traffic.

Now navigate back to the cluster command shell and run the following command:

```
$ kubectl get pods -o wide
```

Notice that the Source Addresses from the redis traffic in the monitor tab corresponds to the frontend pods:

The screenshot shows a terminal window with the title "...labs-gcp-250f11d5b7b9ef75". The command "kubectl get pods -o wide" is run, displaying a table of pod information. The table has columns: NAME, READY, STATUS, RESTARTS, AGE, IP, and NODE. Several rows are highlighted with a red border. The highlighted rows are: "frontend-685d7ff496-2mlks", "frontend-685d7ff496-7j58l", and "frontend-685d7ff496-rf4rh". These correspond to the three Redis slave pods mentioned in the monitor tab. Other visible rows include "redis-master-57cc594f67-x79gc", "redis-slave-84845b8fd8-8zmsc", and "redis-slave-84845b8fd8-q5gfp". The last row is "paloaltonetworks122_student@qwiklabs-gcp-250f11d5b7b9ef75:~\$".

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
frontend-685d7ff496-2mlks	1/1	Running	0	20m	10.8.0.10	gke-cluster-1-default-pool-bla68f2d-d71x
frontend-685d7ff496-7j58l	1/1	Running	0	20m	10.8.0.9	gke-cluster-1-default-pool-bla68f2d-d71x
frontend-685d7ff496-rf4rh	1/1	Running	0	20m	10.8.1.9	gke-cluster-1-default-pool-bla68f2d-3bh9
redis-master-57cc594f67-x79gc	1/1	Running	0	20m	10.8.0.8	gke-cluster-1-default-pool-bla68f2d-d71x
redis-slave-84845b8fd8-8zmsc	1/1	Running	0	20m	10.8.0.7	gke-cluster-1-default-pool-bla68f2d-d71x
redis-slave-84845b8fd8-q5gfp	1/1	Running	0	20m	10.8.1.8	gke-cluster-1-default-pool-bla68f2d-3bh9
paloaltonetworks122_student@qwiklabs-gcp-250f11d5b7b9ef75:~\$						

And the destination address corresponds to the redis-backend:

The screenshot shows a terminal window with the title "...labs-gcp-250f11d5b7b9ef75". The command "kubectl get pods -o wide" is run, displaying a table of pod information. The table has columns: NAME, READY, STATUS, RESTARTS, AGE, IP, and NODE. Several rows are highlighted with a red border. The highlighted rows are: "frontend-685d7ff496-2mlks", "frontend-685d7ff496-7j58l", and "frontend-685d7ff496-rf4rh". These correspond to the three Redis slave pods mentioned in the monitor tab. Other visible rows include "redis-master-57cc594f67-x79gc", "redis-slave-84845b8fd8-8zmsc", and "redis-slave-84845b8fd8-q5gfp". The last row is "paloaltonetworks122_student@qwiklabs-gcp-250f11d5b7b9ef75:~\$".

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
frontend-685d7ff496-2mlks	1/1	Running	0	20m	10.8.0.10	gke-cluster-1-default-pool-bla68f2d-d71x
frontend-685d7ff496-7j58l	1/1	Running	0	20m	10.8.0.9	gke-cluster-1-default-pool-bla68f2d-d71x
frontend-685d7ff496-rf4rh	1/1	Running	0	20m	10.8.1.9	gke-cluster-1-default-pool-bla68f2d-3bh9
redis-master-57cc594f67-x79gc	1/1	Running	0	20m	10.8.0.8	gke-cluster-1-default-pool-bla68f2d-d71x
redis-slave-84845b8fd8-8zmsc	1/1	Running	0	20m	10.8.0.7	gke-cluster-1-default-pool-bla68f2d-d71x
redis-slave-84845b8fd8-q5gfp	1/1	Running	0	20m	10.8.1.8	gke-cluster-1-default-pool-bla68f2d-3bh9
paloaltonetworks122_student@qwiklabs-gcp-250f11d5b7b9ef75:~\$						

End of Activity

Conclusion

Congratulations! You have now successfully integrated the PANW VM series firewall to gain visibility into North/South traffic entering and leaving the Kubernetes cluster and E/W traffic between a Guestbook Application Frontend and Backend pods.