

Ignite19-US-HOW-14-Protecting-Kubernetes

Hands-On-Workshop

Protecting your container workloads in Kubernetes

PanHandler Version



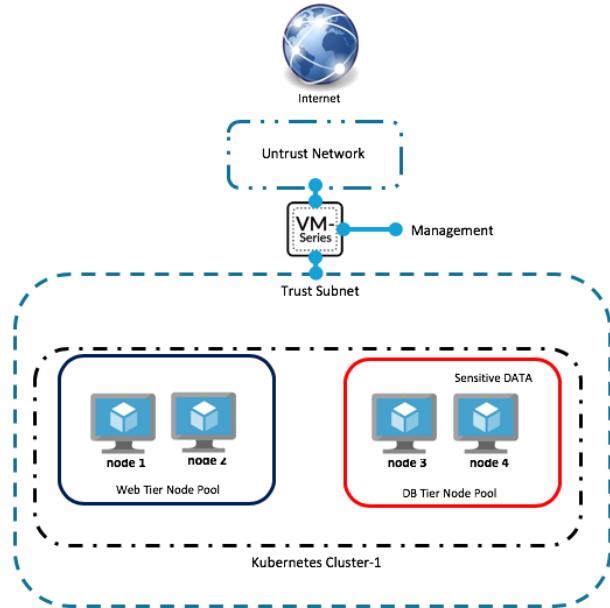
Table of Contents

Lab Overview	4
About GCP Terraform Templates	5
Support Policy	6
Activity 1 – GCP Setup	12
Task 1 – Create a Project	12
Task 2 – Enable the Needed APIs	14
Activity 2 – PanHandler Setup	16
Task 1 – Gather Information and update the Skillet Provisioning page	16
Activity 3 – Provision the Skillet	24
Task 1 – Deploy the Skillet	24
Activity 4 – Review what was deployed	28
Task 1 – Understand what has been initially deployed	28
Task 2 – Look around GCP console	29
Task 3 – Login into the firewall	33
Activity 5 – Container Image Scanning for Vulnerabilities	37
Task 1 – Connect to a Cloud Shell	38
Task 2 – Build and Scan the Application Container Image	41
Activity 6 – Kubernetes App Manifest Scanning for Security Misconfigurations	45
Task 1 – Scan the Application Manifest for Security Best Practices	46
Task 2 – Update the Manifest to Fix the Policy Violations	48
Activity 7 – Launch a two tiered application	50
Task 1 – Inspect the Guestbook Manifest file	50
Task 2 – Launch the Application	54
Task 3 – Explore what was just deployed	54
Activity 8 – Securing Inbound Traffic	59
Task 1 – Note the Internal Load Balancer’s IP Address	59
Task 2 – Update the Firewall’s NAT Policy	59
Task 3 – Connect to the Guestbook Frontend	61
Activity 9 – Securing Outbound Traffic	64
Task 1 – Add Kube-API server route	64

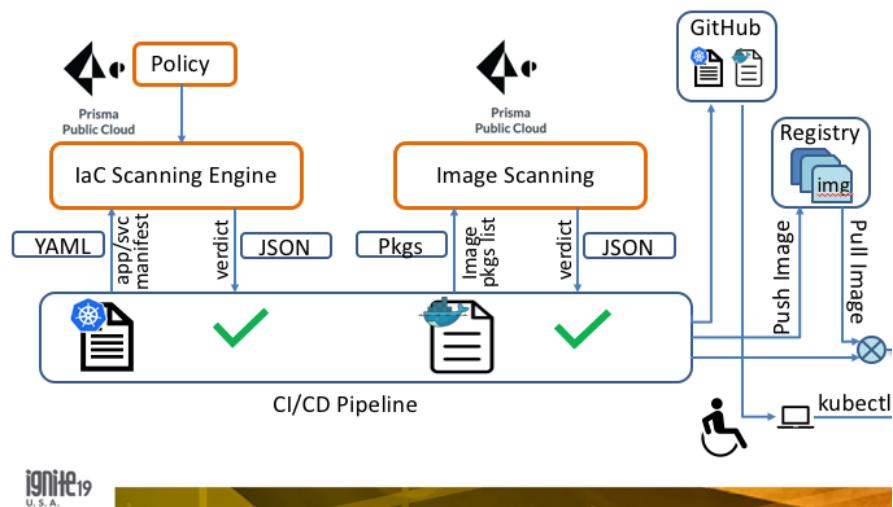
Task 2 – Add Outbound Route	65
Task 3 – Test Outbound Pod Traffic	68
Activity 10 – Investigate Inter Node-Pool traffic	71
Task 1 – View Node Pool Setup	72
Task 2 – View GCP Routing Rules	75
Task 3 – View the VM-Series Firewall Routing	76
Task 4 – Validate North/South and East/West traffic in the firewall logs	77
Conclusion	79

Lab Overview

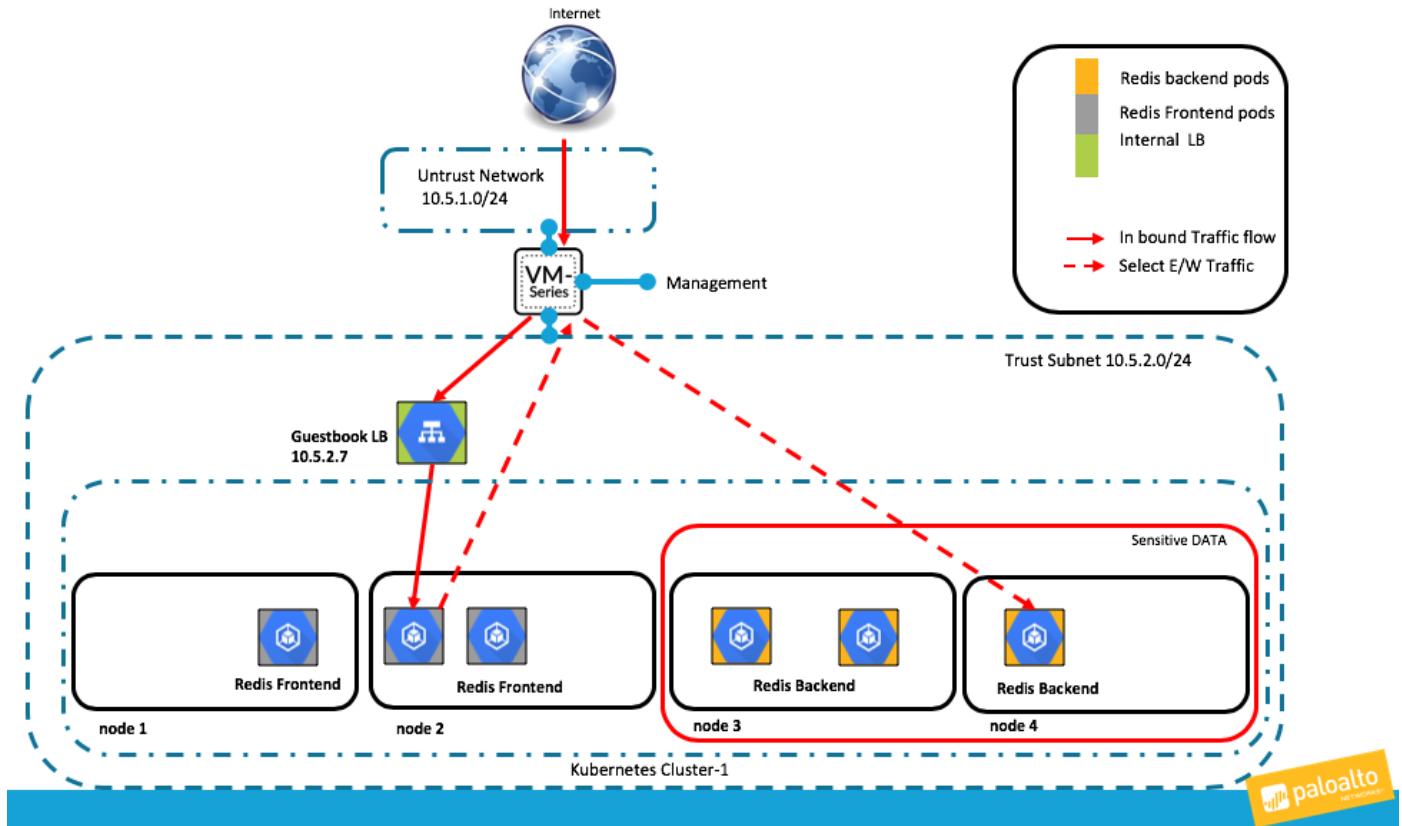
This lab will walk through deploying a Terraform template that deploys a Kubernetes(k8s) cluster, Palo Alto Networks VM-Series firewall into an existing GCP project. After successfully deploying the terraform template, the following infrastructure will be instantiated:



Then the lab will then guide through the use of the Prisma Public Cloud API Scanning of a container image and K8s Manifest file.



Finally the lab will deploy the manifest file to the k8s cluster and walk through visibility of both North/South and select East/West traffic flows. The following diagram shows the deployed pods with the traffic flows:



About GCP Terraform Templates

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>

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	4
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/>
- **Docker:** Docker will need to be installed to run Panhandler. Here are some links to help with the install:
For Mac:<https://docs.docker.com/docker-for-mac/install/>
For Windows:<https://docs.docker.com/docker-for-windows/install/>

- **Panhandler:** Once Docker is installed and running, you will need to pull the panhandler container image from docker hub and build a Panhandler container. From a Mac terminal window or a PC powershell window run the following commands:

```
docker pull paloaltonetworks/panhandler
```

```
Last login: Thu Jun 27 07:30:20 on ttys003
SJOMACCO5JHD4:~ dspears$ docker pull paloaltonetworks/panhandler
Using default tag: latest
latest: Pulling from paloaltonetworks/panhandler
c87736221ed0: Already exists
c3f51b0d0765: Already exists
b5a0acc6b737: Already exists
14428c09e432: Already exists
b47477238ca8: Pull complete
f1226e832a4a: Pull complete
446bd56924c3: Pull complete
29e8a1174576: Downloading 187.6MB/207.8MB
93c4936622fc: Download complete
fd49e257f2b4: Download complete
070bc254bc2b: Download complete
```

After panhandler image is downloaded from Docker Hub, start Panhandler using the following commands:

Mac command:

```
docker run -t -p 8888:80 -v $HOME:/root/ paloaltonetworks/panhandler &
```

Windows command:

```
docker run -t -p 8888:80 -v /c/Users/%USERNAME%:/root/ paloaltonetworks/panhandler
```

```
SJOMACCO5JHD4:~ dspears$ docker run -t -p 8888:80 -v $HOME:/root/ paloaltonetworks/panhandler &
[1] 4264
SJOMACCO5JHD4:~ dspears$ Adding package panhandler to celery
Adding package panhandler to celery
Adding app config for app: panhandler
{'name': 'panhandler', 'label': 'Panhandler', 'repositories': [{ 'name': 'Iron-Skillet-v81', 'url': 'https://github.com/PaloAltoNetworks/iron-skillet-directory': 'Iron-Skillet v81'}, { 'name': 'Default Skillets', 'url': 'https://github.com/PaloAltoNetworks/Skillets.git', 'branch': 'master'}, 'application_data': { 'recommended_repos_link': 'https://raw.githubusercontent.com/nembery/panhandler-content-links/master/links.yaml', 'menu': 'Panhandler', 'menu_option': 'Welcome', 'attributes': { 'template_name': 'panhandler/welcome.html'}, 'context': { 'title': 'Panhandle', 'documentation_link': 'https://panhandler.readthedocs.io/en/latest/', 'next': 'collections'}}, { 'name': 'import', 'class': 'ImportRepositoryView', 'menu': 'Panhandler', 'menu_option': 'Import', 'attributes': { 'snippet': 'import_repo', 'header': 'Import Repository', 'title': 'Enter a valid git url and desired branch here', 'ext': 'This view imports a new git repository. Once imported, any skillets defined in that repository will be available in the Skillet Collection.\n', 'help_link_title': 'Import Skillets Documentation', 'help_link': 'https://panhandler.readthedocs.io/en/master/importing_skillets.html', 'list_skillet_collections_view': 'Skillet Collections View', 'menu': 'Panhandler', 'menu_option': 'Skillet Collections', 'name': 'snippets_by_type', 'class': 'ListSnippetsByTypeView'}}, { 'name': 'list_snippets_by_type', 'class': 'ListSnippetsByTypeView', 'menu': 'Panhandler', 'menu_option': 'List Snippets by Type', 'attributes': { 'template_name': 'panhandler/list_snippets_by_type.html'}}]
```

At this point you should have a container running on your machine with Panhandler. To access Panhandler, open a browser and navigate to the following URL:

<http://localhost:8888>

You should see the following login page.

The screenshot shows a simple login interface. At the top, a dark header bar says "Login". Below it, a white form area has the title "Enter Authentication". It contains two text input fields: one for "Username" and one for "Password", both of which are currently empty. At the bottom of the form is a dark blue "Login" button.

Default login username and password are:

Username: paloalto

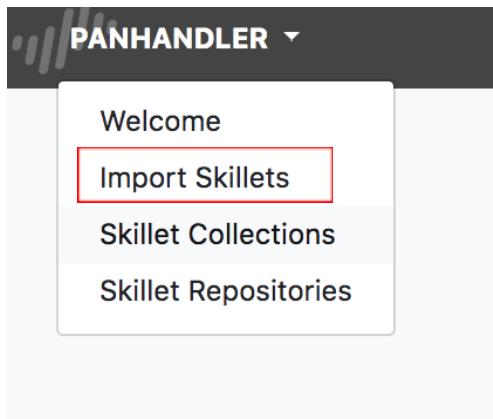
Password: panhandler

Once you are logged in, you will get a welcome screen:

The screenshot shows the Panhandler landing page. The title "Panhandler" is prominently displayed. Below the title is a short description: "Panhandler is a tool for sharing PAN-OS configurations and configuration sets. These configs are stored and shared via Git repositories. In addition to PAN-OS, Panhandler can also consume other types of repositories, such as Terraform templates, or generic network device configuration templates." Underneath this, there is a section titled "Latest Updates:" with two bullet points: "2019-01-25: MSSP templates updates brings updated configuration for internet gateway for Gold, Silver, and bronze configuration sets." and "2019-01-01: Iron Skillet Day One best practice configs added!". At the bottom of the page are two buttons: a blue "Learn more" button and a dark grey "Get Started" button.

Deployment:

In the upper left hand corner, you will see the Panhandler drop down. Click on the drop down and navigate to the “Import Skillets” option.



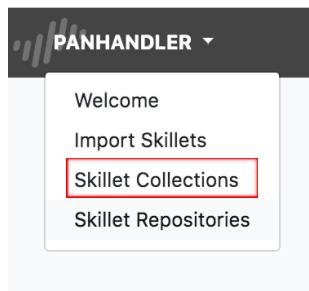
To import the GCP k8s skillet, add the GitHub Repository for the Security framework. Once the Repository Name and Git Repository URL are added, select submit:

GitHub URL: <https://github.com/wwce/terraform.git>

A screenshot of a "Import Repository" form. The form has a light blue header with the title "Import Repository" and a help/gear icon. The main area contains instructions: "Enter a valid git url and desired branch below". It has three input fields: "Repository Name" (value: "WWCE-Terraform"), "Git Repository HTTPS URL" (value: "https://github.com/wwce/terraform", highlighted with a red rectangle), and "Branch" (value: "master"). A "Submit" button is at the bottom right.

NOTE: It is possible that this has already been added to your repository through another guide as there are multiple Skillets in this Repository.

Once the repository has been added/updated, you are now ready to deploy the demo framework. From the upper left hand corner Panhandler drop down list, select “Skillet Collections”



You will see the screen below. Press Go under the GCP K8s Prisma API Skillet:

A screenshot of the 'Skillet Collections' page in the Panhandler application. The page displays a grid of collections. A red arrow points to the 'Go' button under the 'GCP K8s Prisma API' collection. The collections listed are: AWS Jenkins Security Framework, Azure Jenkins Security Framework, Configure, Deploy, Example Skillets, GCP Jenkins Security Framework, GCP K8s Prisma API, IronSkillet, and Public Cloud. Each collection shows the number of skillets it contains and a 'Go' button.

And on the Next Screen that opens, click “Go”

The screenshot shows the PANHANDLER interface with the title "Skillet Collection: GCP K8s Prisma API". At the top, there are "Sort" and "Filter" buttons, and a search bar on the right labeled "Search". Below the title, there is a horizontal navigation bar with tabs: Name (which is selected), Type, PAN-OS, Panorama, Panorama-GPCS, REST, Template, Terraform, Workflow, and Python. A large card titled "GCP 4-node k8s cluster with VM-Series Firewall" is displayed. The card contains a detailed description of the skillet: "This skillet deploys a 4-node GCP k8s cluster with a VM-Series Firewall for both N/S and E/W Inspection. This is the base deployment used in the Ignite 19 k8s HOW lab. There is also a guide that walks through deploying a 2 tier container". At the bottom of the card is a blue "Go" button. A red arrow points to this "Go" button.

The following screen should open. Activity 2 will walk through getting the information needed to complete the fields:

The screenshot shows a "Provision" dialog box for a "GCP 4-node k8s cluster with VM-Series Firewall". The dialog has several input fields:

- GCP Container Ver: 1.11.10-gke.4
- GCP Project: djs-gcp-2018
- GCP Region: us-central1
- GCP Zone: us-central1-a
- Path to the JSON file used to describe your account credentials: djs-gcp-2018-creds.json

At the bottom right of the dialog is a blue "Submit" button. A red arrow points to this "Submit" button.

Activity 1 – GCP Setup

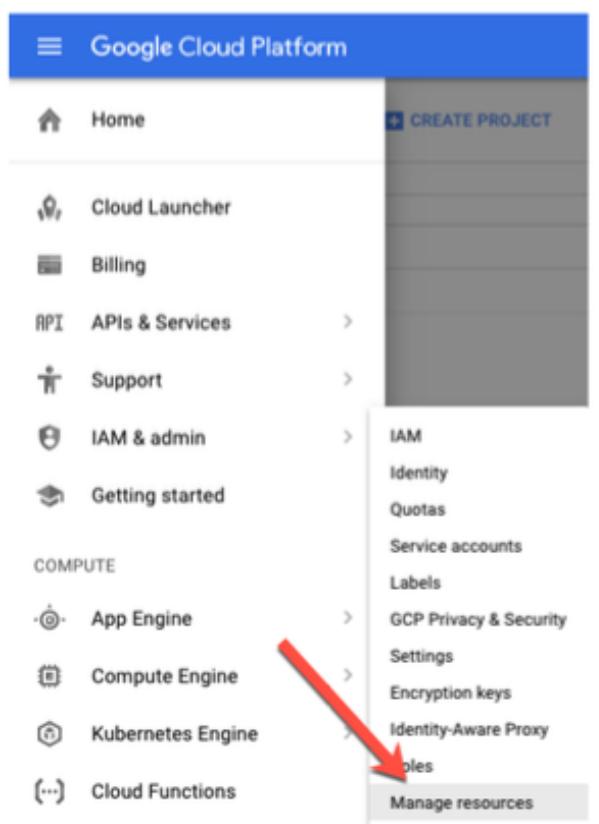
In this activity, you will:

- *Create a project*
- *Enable the APIs needed for this lab*

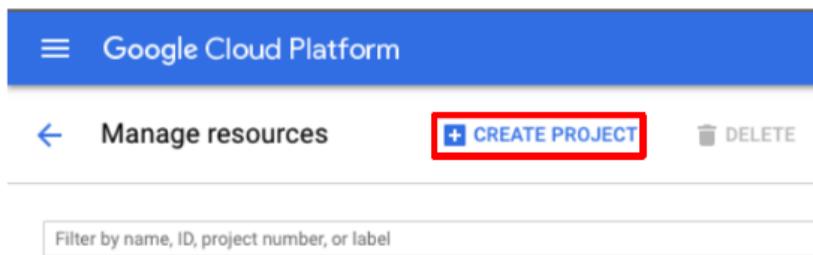
Task 1 – Create a Project

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.

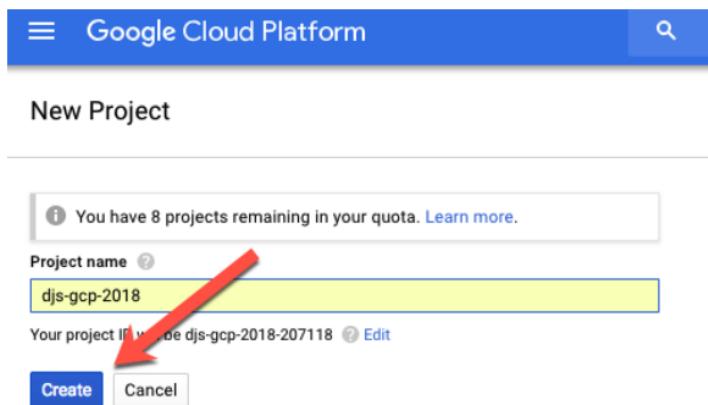
- To create an additional dedicated project, use the drop-down on the left and select **IAM & admin > Manage Resources**:



→ Click **Create Project**:



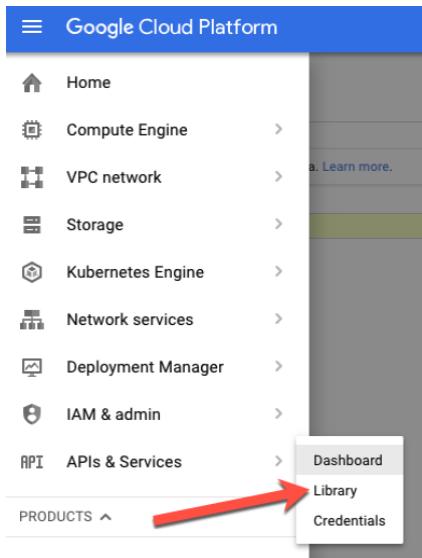
→ Specify a name for the project and click **Create**:



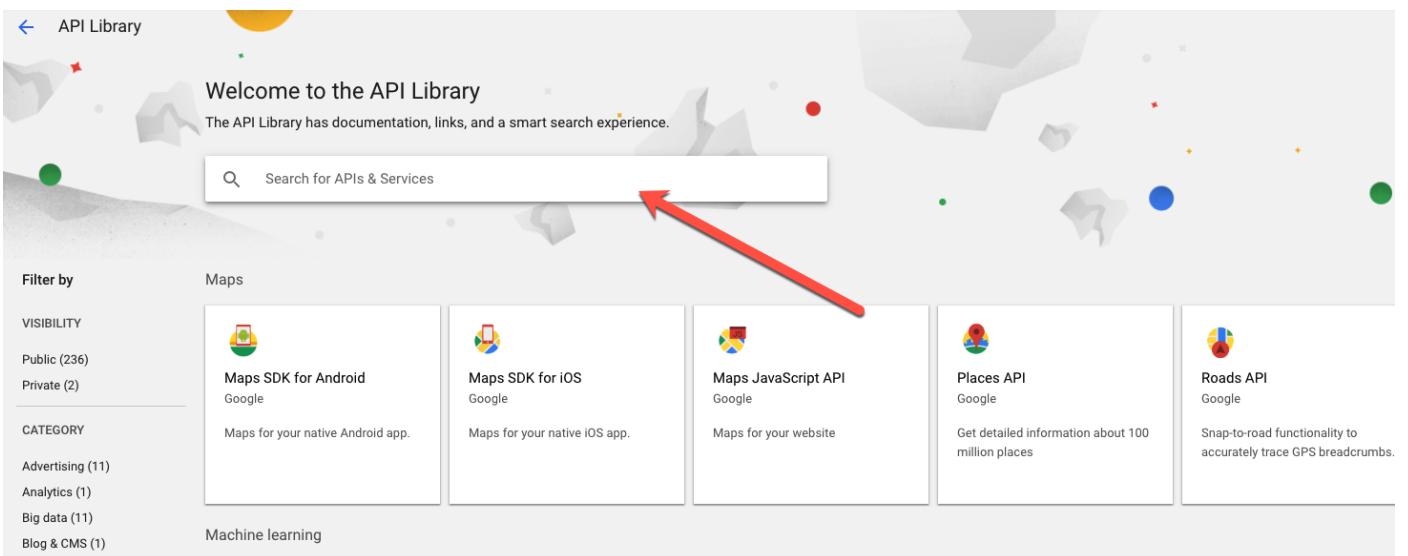
Note that project creation will take a few minutes.

Task 2 – Enable the Needed APIs

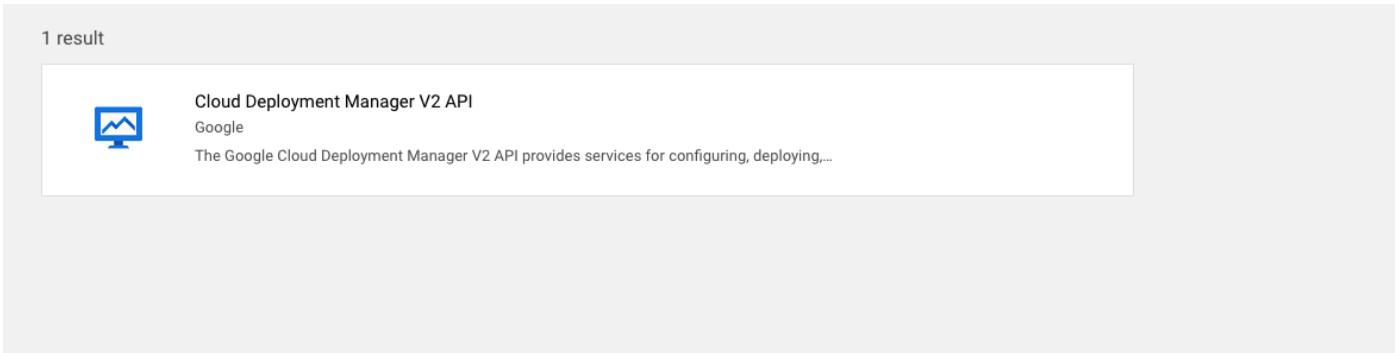
Deploying a template requires the Cloud Deployment Manager API be enabled on the project.
Navigate to **APIs & Services > Library**:



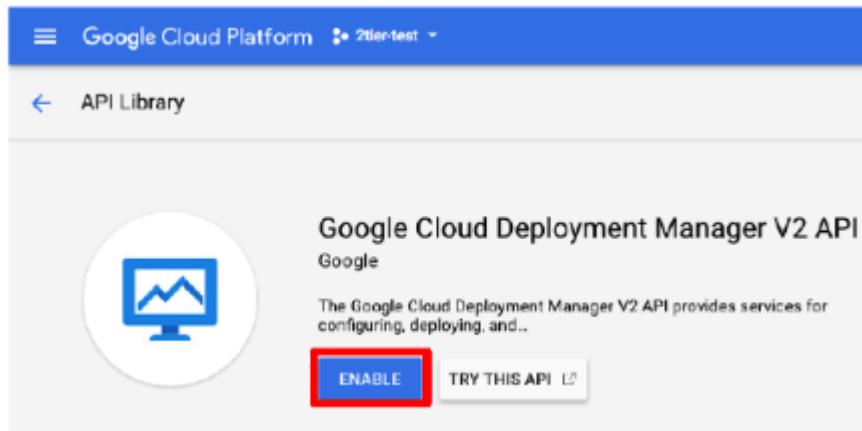
→ In the Search area at the top, type Google Cloud Deployment Manager V2 API :



→ Click on Cloud Deployment Manager V2 API

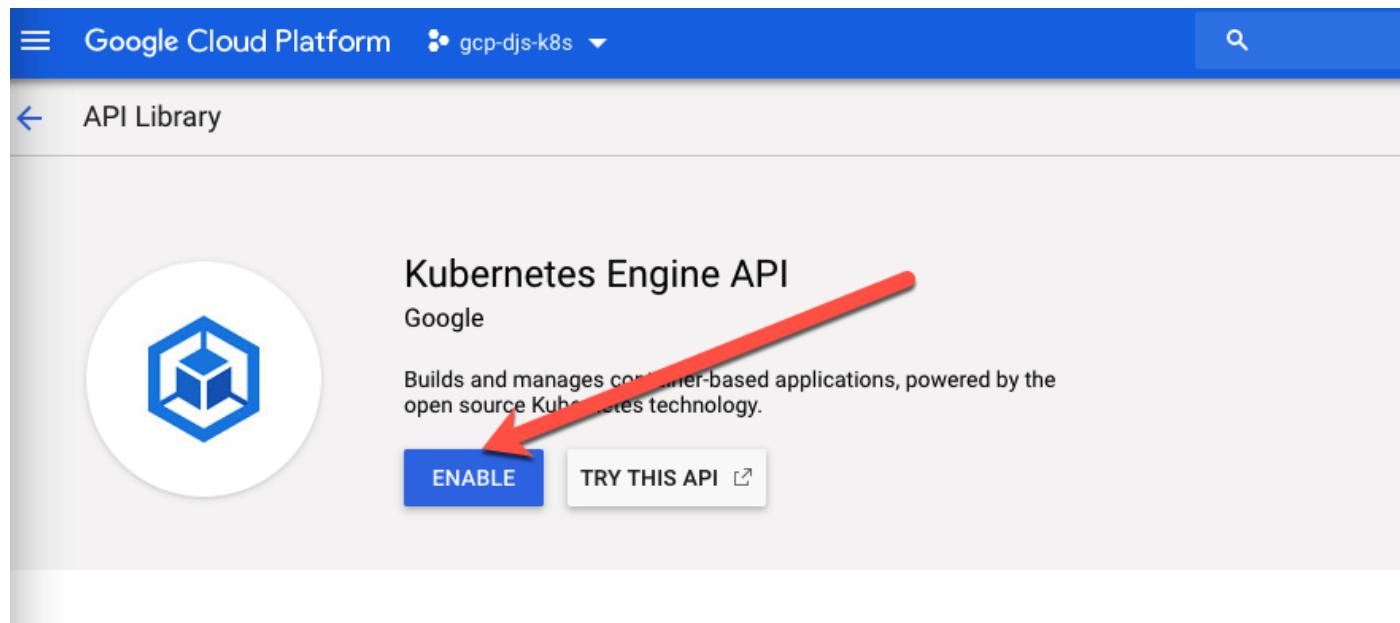


→ Select **Enable**:



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

Repeat these steps for the Kubernetes Engine API.



End of Activity 1

Activity 2 – PanHandler Setup

In this activity, you will:

- *Identify the variable parameters needed for PanHandler*

Task 1 – Gather Information and update the Skillet Provisioning page

Deploying this Skillet does require some information to be inputted into the PanHandler web page. The fields that must be updated are shown below:

PANHANDLER

paloalto ▾

Provision

GCP 4-node k8s cluster with VM-Series Firewall

GCP Container Ver:
1.11.10-gke.4

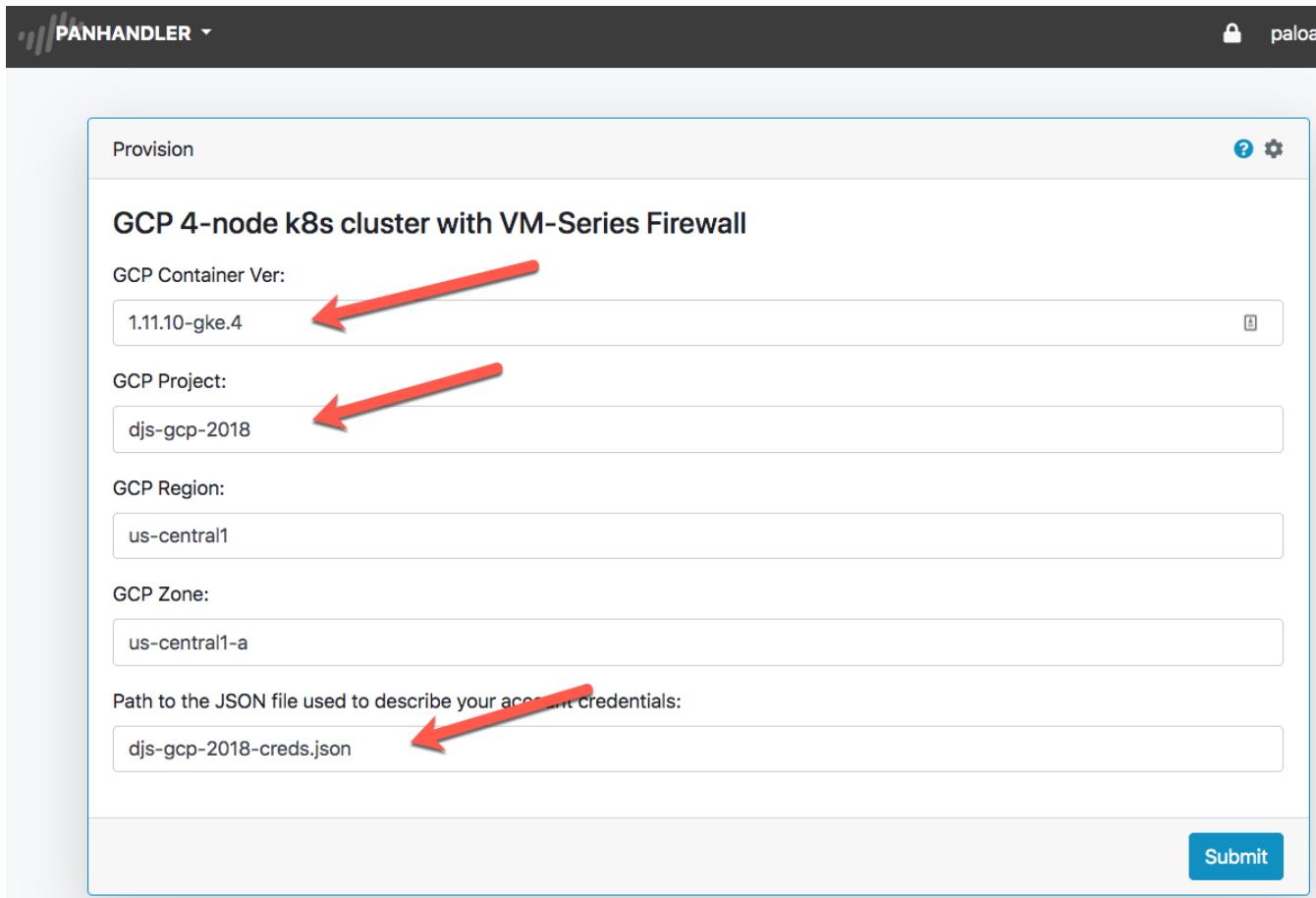
GCP Project:
djs-gcp-2018

GCP Region:
us-central1

GCP Zone:
us-central1-a

Path to the JSON file used to describe your account credentials:
djs-gcp-2018-creds.json

Submit



A description of the fields that need to be updated are:

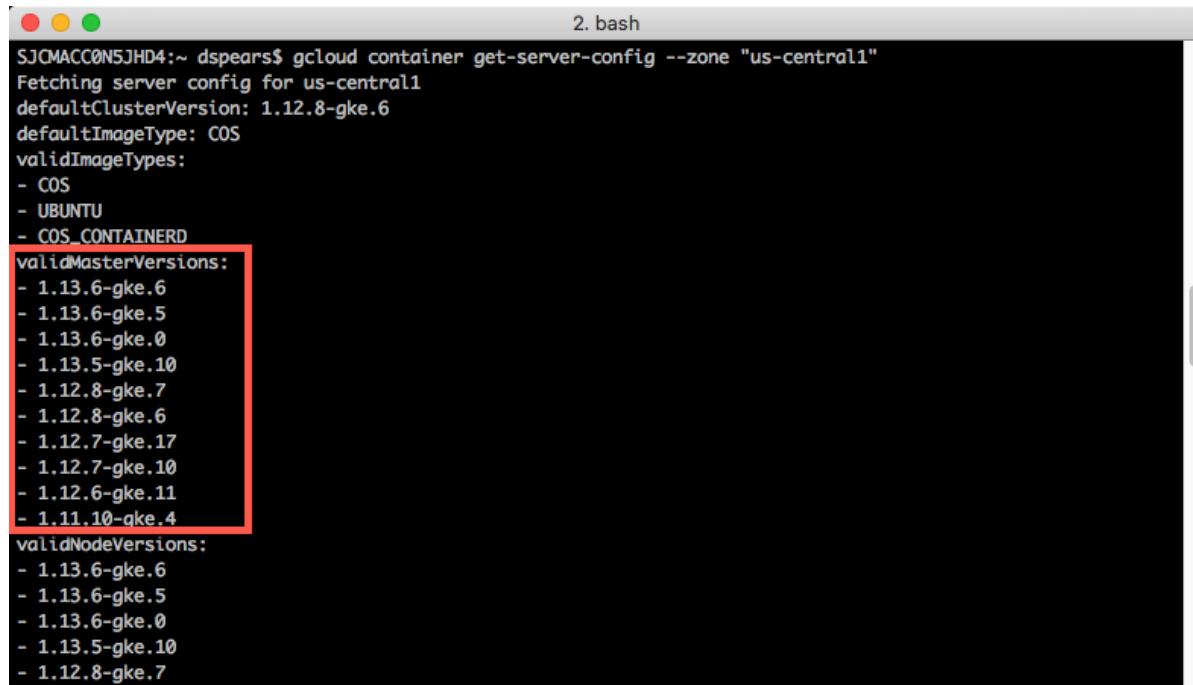
GCP Container-ver This is the K8s Master version. Google updates versions often so it is likely that the version is no longer available.

GCP Project 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.

- To validate the master k8s version run the following command:

```
gcloud container get-server-config --zone "us-central1"
```



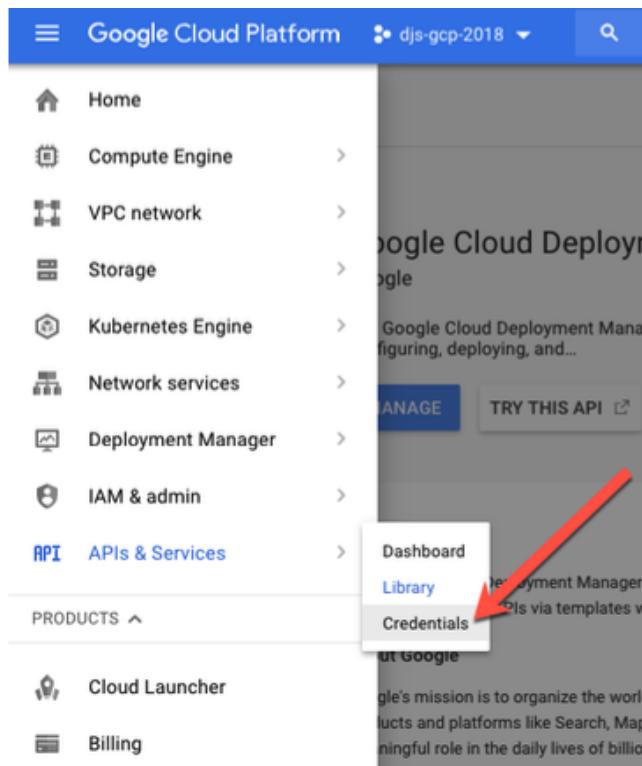
A terminal window titled '2. bash' showing the output of the 'gcloud container get-server-config --zone "us-central1"' command. The output includes server config details like defaultClusterVersion (1.12.8-gke.6), defaultImageType (COS), validImageTypes (COS, UBUNTU, COS_CONTAINERD), validMasterVersions (1.13.6-gke.6, 1.13.6-gke.5, 1.13.6-gke.0, 1.13.5-gke.10, 1.12.8-gke.7, 1.12.8-gke.6, 1.12.7-gke.17, 1.12.7-gke.10, 1.12.6-gke.11, 1.11.10-gke.4), and validNodeVersions (1.13.6-gke.6, 1.13.6-gke.5, 1.13.6-gke.0, 1.13.5-gke.10, 1.12.8-gke.7). A red box highlights the 'validMasterVersions' section.

```
SJCMACCON5JHD4:~ dspears$ gcloud container get-server-config --zone "us-central1"
Fetching server config for us-central1
defaultClusterVersion: 1.12.8-gke.6
defaultImageType: COS
validImageTypes:
- COS
- UBUNTU
- COS_CONTAINERD
validMasterVersions:
- 1.13.6-gke.6
- 1.13.6-gke.5
- 1.13.6-gke.0
- 1.13.5-gke.10
- 1.12.8-gke.7
- 1.12.8-gke.6
- 1.12.7-gke.17
- 1.12.7-gke.10
- 1.12.6-gke.11
- 1.11.10-gke.4
validNodeVersions:
- 1.13.6-gke.6
- 1.13.6-gke.5
- 1.13.6-gke.0
- 1.13.5-gke.10
- 1.12.8-gke.7
```

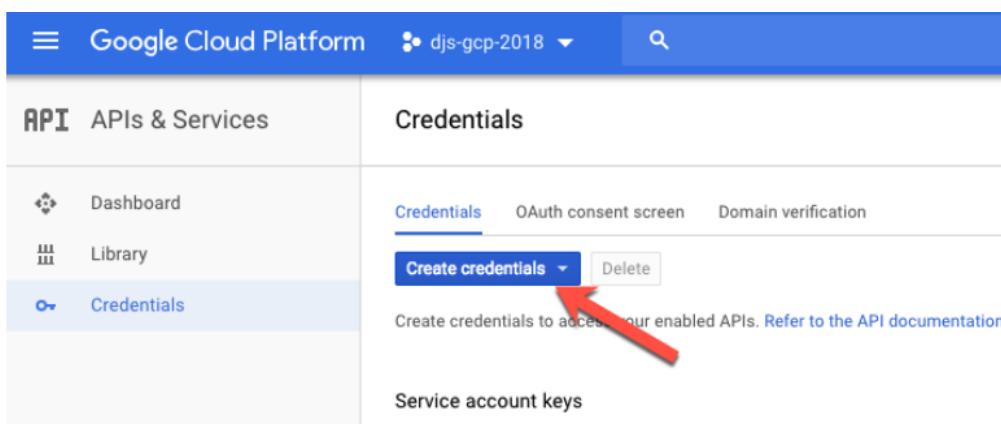
- Make sure a valid Master Version listed is in the GCP Container Ver. I would recommend using an older version as they seem to be a little more stable.

```
variable "container-ver" {
  default = "1.11.10-gke.4"
}
```

→ 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:

The screenshot shows the Google Cloud Platform interface. On the left, there's a sidebar with 'API APIs & Services' and 'Credentials' selected. The main area is titled 'Credentials' with tabs for 'Credentials', 'OAuth consent screen', and 'Domain verification'. A sub-menu 'Create credentials' is open, showing options: 'API key', 'OAuth client ID', 'Service account key', and 'Help me choose'. The 'Service account key' option is highlighted with a red arrow. It is described as enabling server-to-server, app-level authentication using robot accounts.

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

[← Create service account key](#)

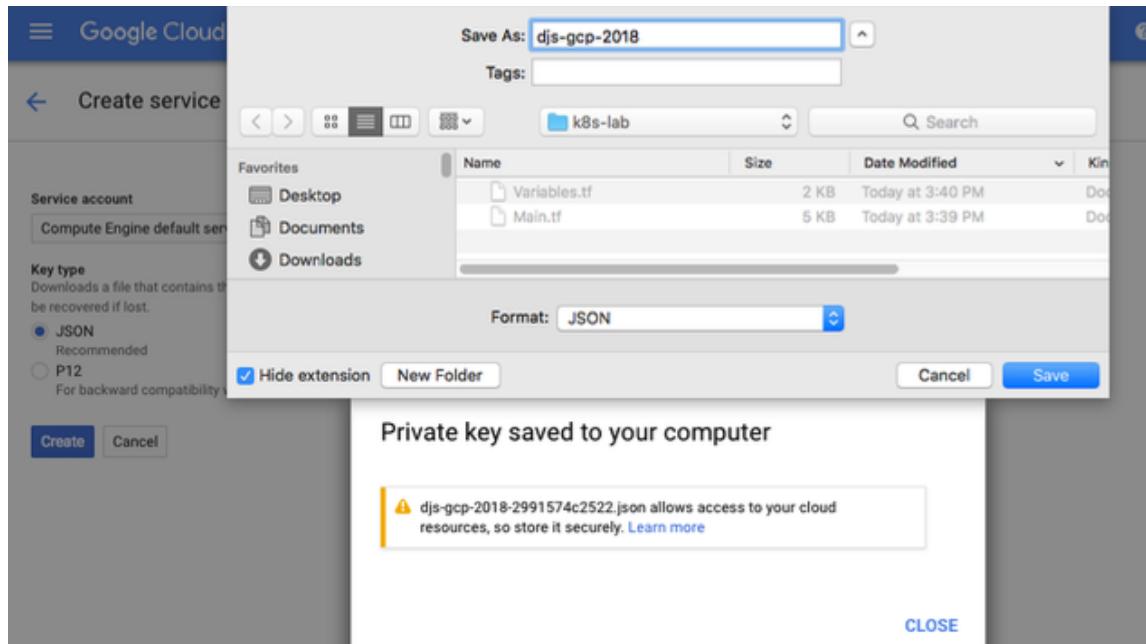
The screenshot shows a 'Create service account key' dialog. It has a 'Service account' section with 'Compute Engine default service account' selected. Below it is a 'New service account' input field and a note about recovering lost keys. Under 'Format', 'JSON' is selected (marked with a blue dot) and 'Recommended' is noted. 'P12' is also listed as an option. At the bottom are 'Create' and 'Cancel' buttons, with 'Create' being highlighted.

NOTE: If this option does not appear, you are likely using a brand new account with free credits.

- To get this option to appear go to Compute Engine > VM Instances and wait for it to finish “getting ready”. Return to APIs & Services > Credentials and the Compute Engine default Service account should be available.

The screenshot shows the Google Cloud Platform interface. The top navigation bar includes the 'Google Cloud Platform' logo, the project name 'My First Project', a search bar, and a dropdown menu. Below the navigation bar, the main content area has a header 'Compute Engine' and 'VM instances'. On the left, a sidebar lists various Compute Engine services: VM instances (selected), Instance groups, Instance templates, Sole tenant nodes, Disks, Snapshots, Images, TPUs, Committed use discounts, Metadata, and Health checks. A red arrow points from the sidebar towards the center of the screen. In the center, there is a message: 'Compute Engine is getting ready. This may take a minute or more.' followed by a link to 'Compute Engine documentation'. To the right of the message is a box titled 'Compute Engine VM instances' containing a brief description of what Compute Engine is and three buttons: 'Create', 'Import', and 'Take the quickstart'.

- 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:



Remember that the container was started with a -v which mounted the home directory to root. As a result the path for the credentials file will be /root/ < json path within the home directory. In this case the path on my Mac to the file is:

/Users/dspears/GCP/k8s-lab/djs-gcp-2018.json

That path will be the following within the container:

/root/GCP/k8s-lab/djs-gcp-2018.json

- Update the Path field as needed:

The screenshot shows the 'Provision' configuration screen in the PANHANDLER interface. It includes fields for GCP Container Ver., Project, Region, Zone, and a 'Path to the JSON file used to describe your account credentials' field containing the path to the downloaded JSON file. A red arrow points to this path field.

- 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:

The screenshot shows the GCP console interface. In the top left, it says "Google Cloud Platform". Below that, there's a dropdown menu showing "djs-gcp-2018". A red arrow points from this dropdown to the "Select a project" dialog box. The dialog box has a title "Select a project" and a search bar. It shows two tabs: "RECENT" (which is selected) and "ALL". There are three projects listed:

Name	ID
<input checked="" type="checkbox"/> djs-gcp-2018	djs-gcp-2018
<input type="checkbox"/> My First Project	axial-gist-174112
<input type="checkbox"/> dis-terraform	dis-terraform

A second red arrow points to the "ID" column for the selected project "djs-gcp-2018".

- Update the Variables.tf file with the GCP ID:

The screenshot shows the PANHANDLER provisioning interface. At the top, it says "PANHANDLER" and "paloalto". The main form is titled "Provision" and has the following fields:

- GCP Container Ver: 1.11.10-gke.4
- GCP Project:
- GCP Region: us-central1
- GCP Zone: us-central1-a
- Path to the JSON file used to describe your account credentials: /root/GCP/k8s-lab/djs-gcp-2018.json

At the bottom right of the form is a blue "Submit" button.

End of Activity 2

Activity 3 – Provision the Skillet

In this activity, you will:

- *Initialize and deploy the skillet*

Task 1 – Deploy the Skillet

Once the information is put into the PanHandler page. Launch the skillet by clicking Submit:

PANHANDLER ▾ palalto ▾

Provision ? ⚙

GCP 4-node k8s cluster with VM-Series Firewall

GCP Container Ver:

GCP Project:

GCP Region:

GCP Zone:

Path to the JSON file used to describe your account credentials:

↓

Submit

and then Choose the Validate, Init, and Apply action and click Submit:

PANHANDLER

Terraform Template

Choose the action to perform

Template Name:

Validate, Init, and Apply

Submit

The first action is a Terraform Init. Once that is done press Continue:

PANHANDLER

Terraform Template

Completed: Executing Task: Terraform Init

Initializing provider plugins...

- Checking for available provider plugins on https://releases.hashicorp.com...

- Downloading plugin for provider "google" (2.10.0)...

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 = "~> 2.10"

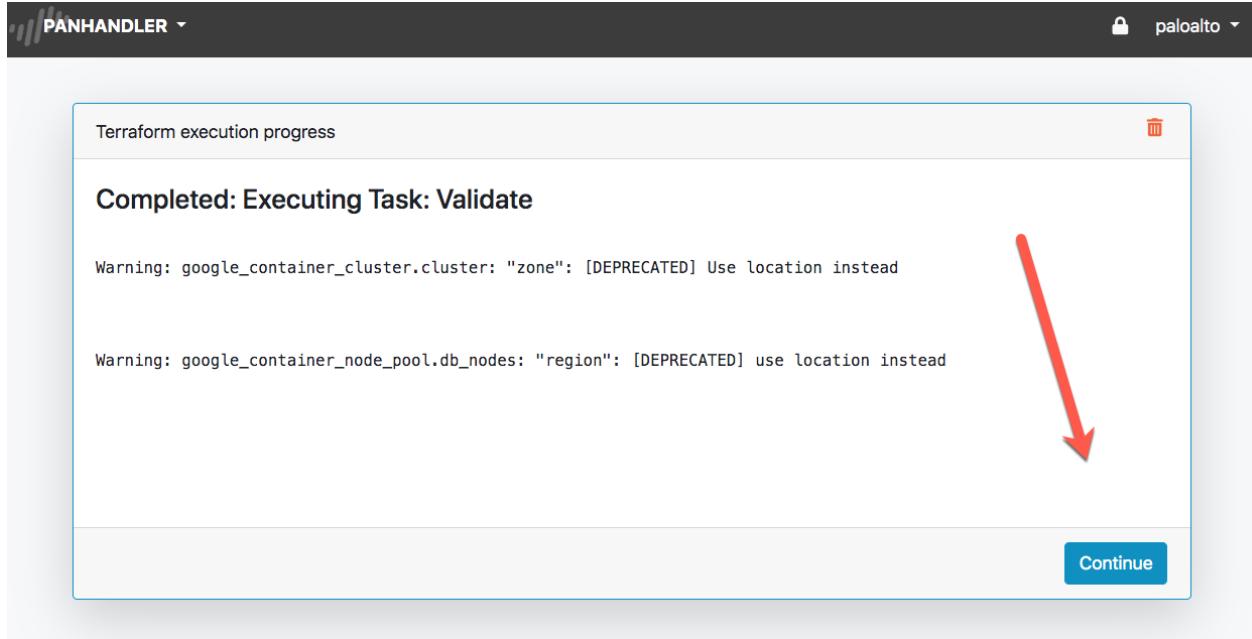
Terraform has been successfully initialized!

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.

Continue

The next action to complete is the validation. Once this is done press Continue:



Terraform execution progress

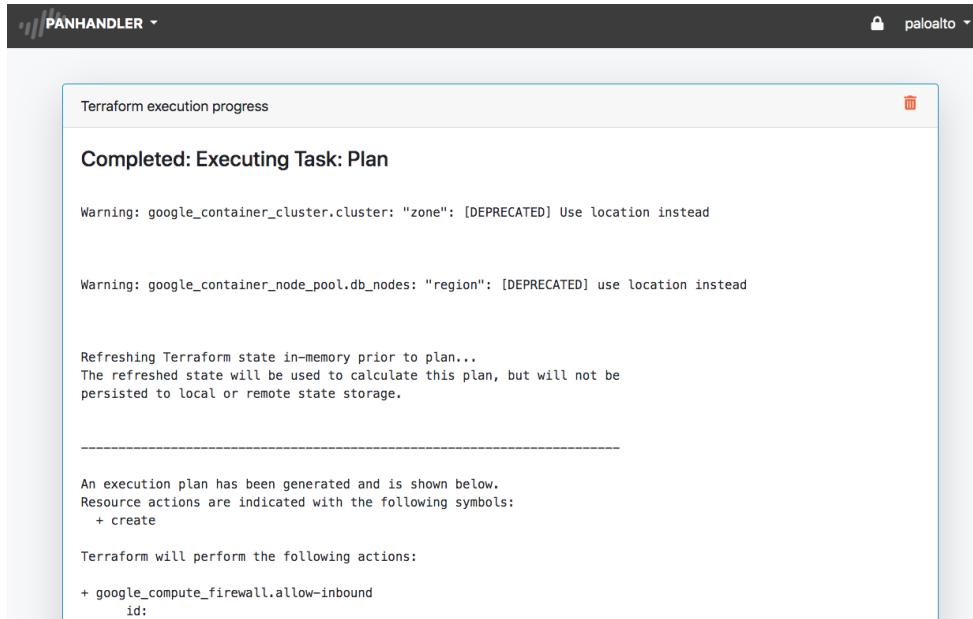
Completed: Executing Task: Validate

```
Warning: google_container_cluster.cluster: "zone": [DEPRECATED] Use location instead

Warning: google_container_node_pool.db_nodes: "region": [DEPRECATED] use location instead
```

Continue

Next the execution should start:



Terraform execution progress

Completed: Executing Task: Plan

```
Warning: google_container_cluster.cluster: "zone": [DEPRECATED] Use location instead

Warning: google_container_node_pool.db_nodes: "region": [DEPRECATED] use location instead

Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

-----
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:
```

This will continue for a while and once done will display the following screen:

The screenshot shows the PANHANDLER interface with the title "Terraform execution progress". Below it, the message "Completed: Executing Task: Apply" is displayed. The log output shows the creation of various Google Compute Engine resources:

```
google_compute_project_metadata_item.ssh-keys: Creating...
key:    "" => "ssh-keys"
project: "" => ""
value:   "" => "dspears@SJCMAC3024G8WL:ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQC3bjwN/LY87F0ZH/uuRXS5ku30XkxsFI"
google_compute_network.management: Creating...
auto_create_subnetworks:      "" => "false"
delete_default_routes_on_create: "" => "false"
gateway_ipv4:                 "" => ""
name:                          "" => "management"
project:                      "" => ""
routing_mode:                 "" => ""
self_link:                     "" => ""
google_compute_network.untrust: Creating...
  auto_create_subnetworks:      "" => "false"
```

Click Continue to return to the Skillet Menu:

The screenshot shows the PANHANDLER interface after a Terraform apply operation has completed. The log output indicates successful creation of resources and the completion of the apply process:

```
next_hop_gateway: <-- default internet gateway
next_hop_network: "" => ""
priority:        "" => "100"
project:         "" => ""
self_link:       "" => ""
google_compute_route.trust: Creating...
dest_range:     "" => "0.0.0.0/0"
name:           "" => "trust-route"
network:        "" => "https://www.googleapis.com/compute/v1/projects/djs-gcp-2018/global/networks/trust-route"
next_hop_instance: "" => "firewall-1"
next_hop_instance_zone: "" => "us-central1-a"
next_hop_network: "" => ""
priority:        "" => "100"
project:         "" => ""
self_link:       "" => ""
google_compute_route.k8mgmt: Still creating... (10s elapsed)
google_compute_route.trust: Still creating... (10s elapsed)
google_compute_route.k8mgmt: Creation complete after 15s (ID: cluster-endpoint-route)
google_compute_route.trust: Creation complete after 16s (ID: trust-route)

Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:
k8s-cluster-endpoint = [
  35.184.115.68
]
k8s-cluster-name = [
  cluster-1
]
k8s-cluster_ipv4_cidr = [
  10.16.0.0/14
]
pan-tf-name = [
  firewall-1
]
```

A large red arrow points from the bottom of the log area down towards the "Continue" button at the bottom right of the screen.

End of Activity 3

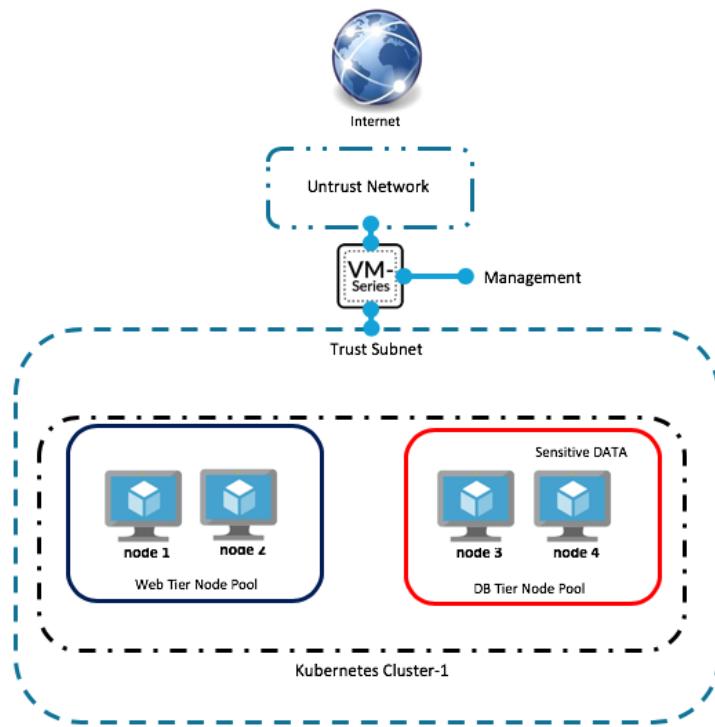
Activity 4 – Review what was deployed

In this activity, you will:

- **Review the resources that have been launched**
- **Log into the VM-Series firewall**
- **Confirm bootstrap success and firewall licensing**

Task 1 – Understand what has been initially deployed

During the lab environment creation a number of things have been deployed automatically. The following diagram shows the initial lab deployment:

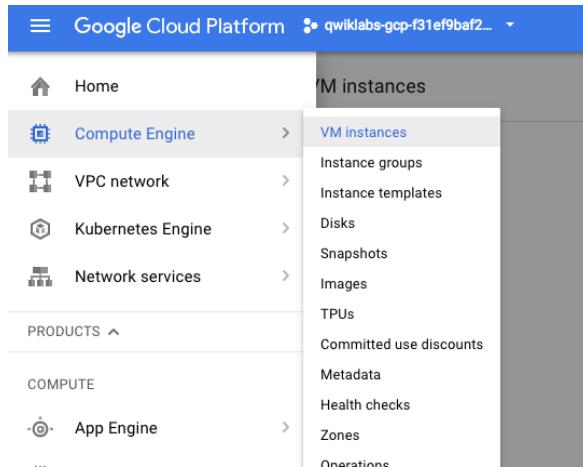


Some things to Note:

- A VM-Series Firewall has been bootstrapped with an initial configuration
- A K8s cluster has been created with 4 nodes in two separate node pools
- The VM-Series will be used for both North/South and East/West Inspection.

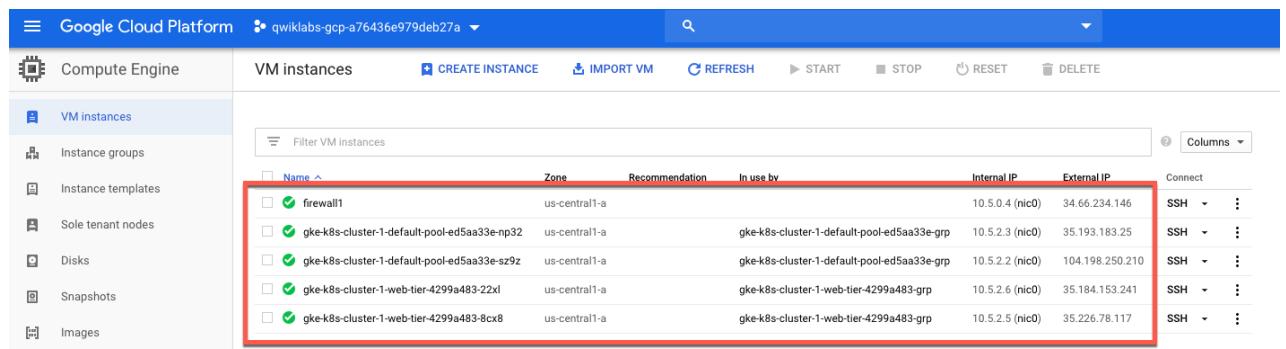
Task 2 – Look around GCP console

- Navigate to **Compute Engine > VM Instances** to see that VM-Series firewall has already been launched when the lab was started.



The screenshot shows the Google Cloud Platform navigation bar at the top with the project name "qwiklabs-gcp-f31ef9baf2...". Below it is a sidebar with categories like Home, Compute Engine, VPC network, Kubernetes Engine, Network services, and others under PRODUCTS and COMPUTE. A dropdown menu for "Compute Engine" is open, showing options: VM instances (which is highlighted), Instance groups, Instance templates, Disks, Snapshots, Images, TPUs, Committed use discounts, Metadata, Health checks, Zones, and Operations. The "VM instances" option is the current selection.

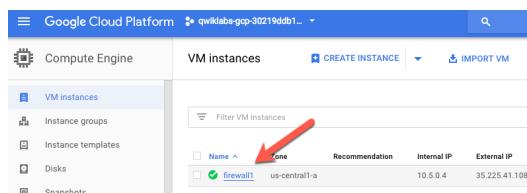
You should see a firewall and 4 nodes that were deployed as part of the k8's cluster:



The screenshot shows the "VM instances" page in the Compute Engine section of the GCP console. The left sidebar lists VM instances, Instance groups, Instance templates, Sole tenant nodes, Disks, Snapshots, and Images. The main area displays a table of VM instances with columns: Name, Zone, Recommendation, In use by, Internal IP, External IP, and Connect. A red box highlights the first row, which corresponds to the "firewall1" instance. The table data is as follows:

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
firewall1	us-central1-a			10.5.0.4 (nic0)	34.66.234.146	SSH
gke-k8s-cluster-1-default-pool-ed5aa33e-np32	us-central1-a		gke-k8s-cluster-1-default-pool-ed5aa33e-grp	10.5.2.3 (nic0)	35.193.183.25	SSH
gke-k8s-cluster-1-default-pool-ed5aa33e-sz9z	us-central1-a		gke-k8s-cluster-1-default-pool-ed5aa33e-grp	10.5.2.2 (nic0)	104.198.250.210	SSH
gke-k8s-cluster-1-web-tier-4299a483-22xl	us-central1-a		gke-k8s-cluster-1-web-tier-4299a483-grp	10.5.2.6 (nic0)	35.184.153.241	SSH
gke-k8s-cluster-1-web-tier-4299a483-8cx8	us-central1-a		gke-k8s-cluster-1-web-tier-4299a483-grp	10.5.2.5 (nic0)	35.226.78.117	SSH

- Clicking on the **firewall1** opens a detailed view of the deployed firewall:



The screenshot shows the same "VM instances" page as before, but now the "firewall1" instance is selected, indicated by a red arrow pointing to its "Zone" column. The rest of the interface is identical to the previous screenshot.

Note the **External IP Addresses** of the firewall on the firewall VM Instance details screen. These will be used later in the lab to connect the firewall and test application functionality:

The screenshot shows the 'VM instance details' page for a VM named 'firewall1'. The 'External IP' column is highlighted with a red box. The table data is as follows:

Name	Network	Subnetwork	Primary Internal IP	Alias IP ranges	External IP	Network Tier	IP forwarding	Network details
nic0	management	management-subnet	10.50.4	—	34.66.234.146 (ephemeral)	Premium	On	View details
nic1	untrust	untrust-subnet	10.5.1.4	—	35.225.61.208 (ephemeral)	Premium	On	View details
nic2	trust	trust-subnet	10.52.4	—	None	Premium	On	View details

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

The screenshot shows the Google Cloud Platform navigation menu. The 'VPC network' option under 'Compute Engine' is highlighted with a red box. A dropdown menu for 'VPC networks' is open, listing options: External IP addresses, Firewall rules, Routes, VPC network peering, and Shared VPC.

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

The screenshot shows the Google Cloud Platform VPC networks interface. On the left, there's a sidebar with options like VPC networks, External IP addresses, Firewall rules, Routes, and VPC network peering. The main area is titled "VPC networks" and shows a table of networks. There are four rows in the table:

	europe-west4	default	10.164.0.0/20	10.164.0.1
management	us-central1	management-subnet	Custom	1
trust	us-central1	trust-subnet	Custom	1
untrust	us-central1	untrust-subnet	10.5.1.0/24	10.5.1.1

A red arrow points to the "management" row.

- Navigate to **Kubernetes Engine > Clusters** and validate that the K8s cluster was successfully built and is running:

The screenshot shows the Google Cloud Platform Kubernetes Engine clusters interface. On the left, there's a sidebar with options like Clusters, Workloads, Services, Applications, Configuration, and Storage. The main area is titled "Kubernetes clusters" and shows a table of clusters. There are two rows in the table:

	Name	Location	Cluster size	Total cores	Total memory	Master version	Expiration time	Notifications
<input type="checkbox"/>	k8s-cluster-1	us-central1-a	4	4 vCPUs	15.00 GB	1.11.8-gke.6	May 25, 2019	

A red arrow points to the first cluster row.

→ Clicking on the cluster will open a more detailed view of the k8s cluster that was deployed:

k8s-cluster-1

Cluster

- Master version: 1.11.8-gke.6
- Endpoint: 35.225.47.120
- Client certificate: Enabled
- Binary Authorization: Disabled
- Kubernetes alpha features: Enabled
- Expiration time: May 25, 2019, 9:35:03 AM
- Total size: 4
- Master zone: us-central1-a
- Node zones: us-central1-a
- Network: trust
- Subnet: trust-subnet
- VPC-native (alias IP): Disabled
- Pod address range: 10.16.0.0/14
- Intranode visibility: Disabled
- Stackdriver Logging: Enabled
- Stackdriver Monitoring: Enabled
- Private cluster: Disabled
- Master authorized networks: Disabled
- Network policy: Disabled
- Legacy authorization: Disabled
- Maintenance window: Any time
- Cloud TPU: Disabled
- Application-layer Secrets Encryption: Disabled
- Node auto-provisioning: Disabled
- Vertical Pod Autoscaling: Disabled

Labels: None

Add-ons

Permissions

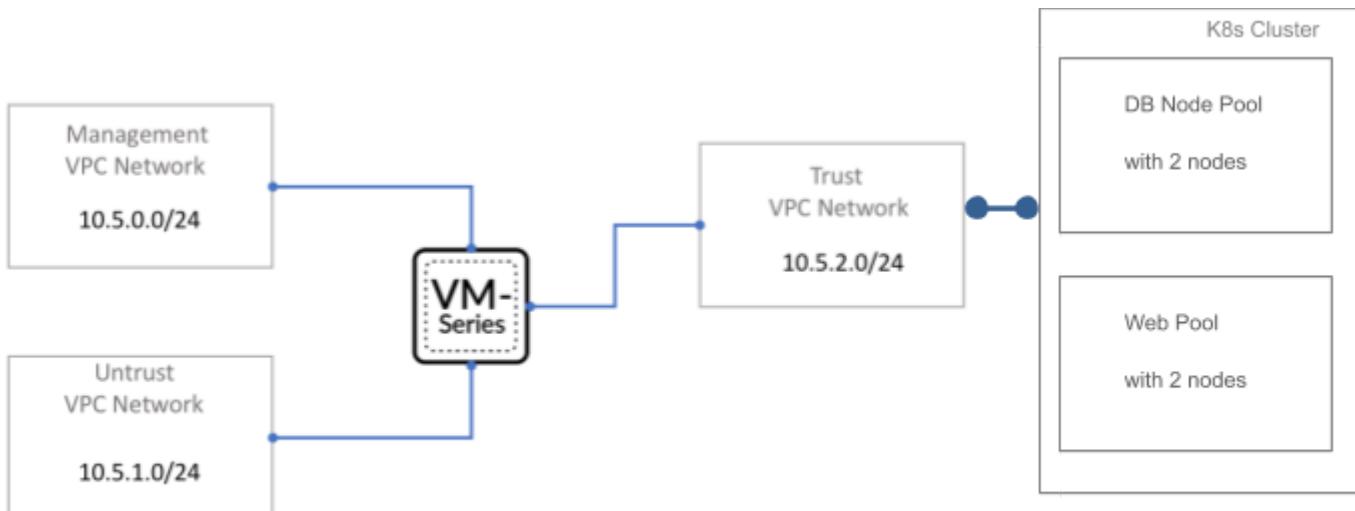
Node Pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit on the top bar. [Learn more](#)

default-pool (2 nodes, version 1.11.8-gke.6)
web-tier (2 nodes, version 1.11.8-gke.6)

Note: the **number of nodes (Total size)**, **node networks**, and **Pod IP address ranges** in the K8s Cluster details.

The following diagram describes the topology that has been deployed:

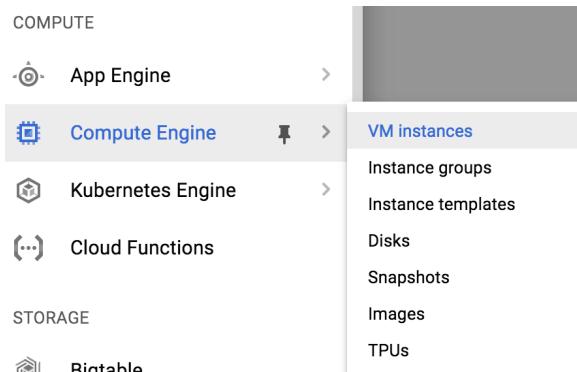


Feel free to navigate through other parts of the Google Cloud Console such as **VPC Networks > Routes, Firewall Rules**. This will come in handy in activities later on.

Task 3 – Login 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 predefined 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 Engine > VM Instances**



- Click on **firewall1** instance name to get more information and identify the management interface IP.

A screenshot of the Google Cloud Platform 'VM instances' page. The left sidebar shows 'Compute Engine' selected, with 'VM instances' also selected. The main area displays a table of VM instances. A red arrow points to the 'Zone' column for the 'firewall1' instance, which is listed as 'us-central1-a'. The table includes columns for 'Name', 'Zone', 'Recommendation', 'Internal IP', and 'External IP'. The 'firewall1' instance has an internal IP of 10.5.0.4 and an external IP of 35.225.41.108.

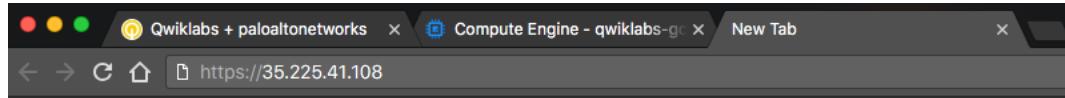
Name	Zone	Recommendation	Internal IP	External IP
firewall1	us-central1-a		10.5.0.4	35.225.41.108

→ Copy the **External public IP** of the management interface

The screenshot shows the Google Cloud Platform Compute Engine VM instance details for an instance named 'firewall1'. The 'Details' tab is selected. In the 'Network interfaces' section, there is a table with the following data:

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. 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):

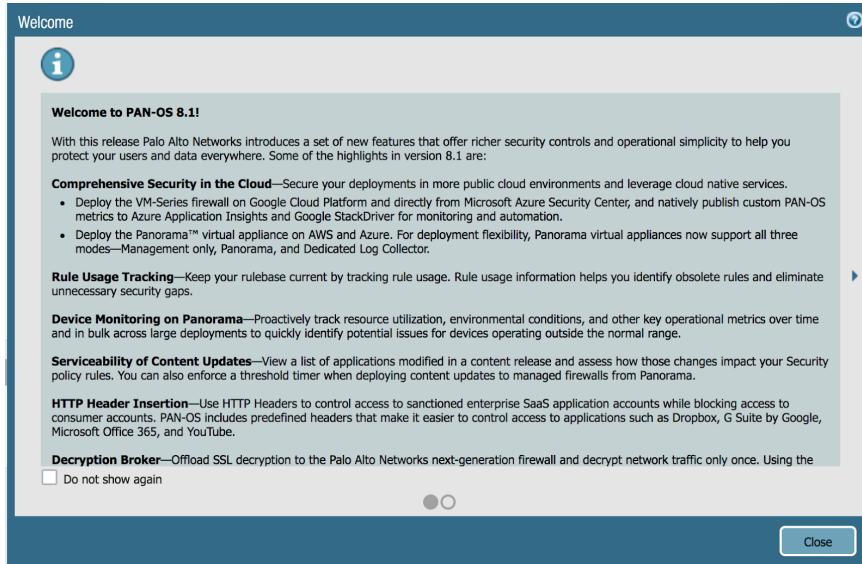
username: **admin**

password: **Pal0Alt0@123**

Note: Those are zeros not capital O's.



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 various policy categories. The main area displays a table of security policies:

Name	Type	Source			Destination			Application	Service	Action
		Zone	Address	User	Zone	Address				
1 to-guestbook-sec-pol...	universal	[untrust]	any	any	[web]	any	any	service-http	Allow	
2 to-wordpress-sec-pol...	universal	[untrust]	any	any	[web]	any	any	TCP8888	Allow	
3 EW	intrazone	[web]	any	any	any	(intrazone)	any	dns	any	
4 Outbound-deny	universal	[web]	any	any	[untrust]	any	any	ssh	any	
5 Outbound	universal	[web]	any	any	[untrust]	any	any	any	Deny	
6 default-deny-all	universal	[any]	any	any	[any]	any	any	any	Deny	
7 intrazone-default	intrazone	[any]	any	any	(intrazone)	any	any	any	Allow	
8 interzone-default	interzone	[any]	any	any	[any]	any	any	any	Allow	

→ Click on the **Dashboard** tab, check to verify that the firewall has a serial number.

The screenshot shows the Palo Alto Networks Dashboard interface. The top navigation bar includes tabs for Dashboard, ACC, Monitor, and Policies, with the Dashboard tab selected. Below the navigation is a status bar showing 'Layout: 3 Columns' and 'Widgets' with a last update of '10:43'. The main content area is divided into two sections: 'General Information' and 'System Resources'. The 'General Information' section displays various device details, including the 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::4001:aff:fe05:4/64), MGT IPv6 Default Gateway (42:01:0a:05:00:04), Model (PA-VM), Serial # (unknown, highlighted with a red arrow), CPU ID (GCP-D7060200FFFFB8B1F), UUID (E650FB87-E1A4-6529-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 (0000.00.00.00), GlobalProtect Clientless VPN Version (0), Time (Thu May 3 08:43:14 2018), and Uptime (0 days, 0:52:02). The 'System Resources' section shows Management CPU at 0%, Data Plane CPU at 0%, and a Session Count of 2 / 1248.

We are using the PayAsYouGo image from GCP Cloud Launcher. A license will be required to view the logs later in the lab.

End of Activity 4

Activity 5 – Container Image Scanning for Vulnerabilities

In this activity, you will:

- Scan your container images for security vulnerabilities using Prisma Public Cloud (formerly Redlock) free public APIs
 - Scan publicly available container images for security vulnerabilities
 - Patch the images
 - Push the patched image to your container registry
-

What are container images?

A container image is a lightweight, standalone, executable packaging of software which includes everything needed to run an application: code, runtime, libraries and local configuration.

Container images are made up of different layers. Every container image has a base layer (parent layer) which is usually an Operating System. The subsequent layers are built on top of it which might include language runtimes, libraries, and code (file, executables), etc.

Each of the layers are immutable and built on top of the previous layer. These layers are independent of each other. For example OpenSSL can be installed on many different base images (OS). Most of the layers are reusable such as base layer, libraries, language runtimes, which are pulled from internal or external shared repositories such as DockerHub, GitHub, npm, etc.

How are containers built?

Dockerfile:

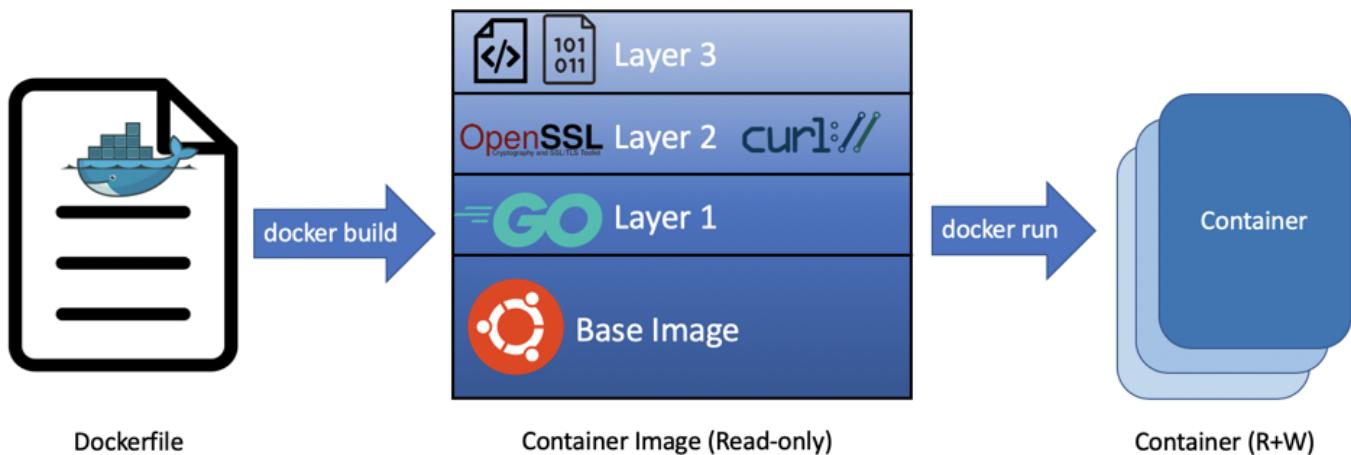
Dockerfile is the manifest with build instructions on how to build a specific container image.

Image:

Container images are read-only templates from which containers are launched. Each image contains a series of layers as explained above.

Container:

A container is a running and mutable (Read + Write) form of the image.



What does “scanning” my image mean?

Prisma Public Cloud (formerly Redlock) Image Scanning service provides free public API to scan your container images. When you “scan” an image, you are getting a list of all the vulnerabilities from all the packages and base OS installed in the image across all the layers. Each layer could contain multiple packages. The scan result will give you all the known vulnerabilities grouped by severity or package.

Task 1 – Connect to a Cloud Shell

NOTE: This is going to be done through the Kubernetes cluster so kubectl commands can be run later.

→ Click on **Kubernetes Engine > Clusters**

Setting	Value
Client certificate	Enabled
Binary Authorization	Disabled
Clusters	Enabled Jun 21, 2019, 2
Workloads	4
Services	us-central1-a
Applications	trust
Configuration	trust-subnet
Storage	Disabled 10.16.0.0/14
Intranode visibility	Disabled
Stackdriver Logging	Enabled
Stackdriver Monitoring	Enabled
Private cluster	Disabled
Master authorized networks	Disabled

→ Click on the **Connect** button next to your cluster

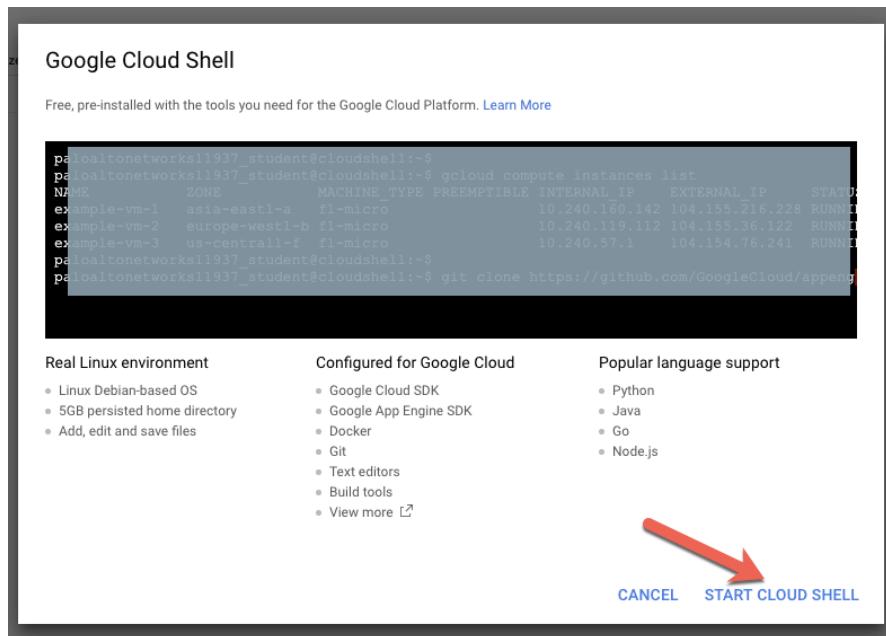
The screenshot shows the Google Cloud Platform interface for managing Kubernetes clusters. On the left, there's a sidebar with icons for Clusters, Workloads, Services, Applications, Configuration, and Storage. The main area is titled 'Kubernetes clusters' and includes buttons for 'CREATE CLUSTER', 'DEPLOY', 'REFRESH', and 'DELETE'. Below these are sections for 'Kubernetes alpha clusters' and 'Kubernetes beta clusters'. A table lists a single cluster: 'k8s-cluster-1' located in 'us-central1-a' with a size of 4 vCPUs, 15.00 GB memory, and version 1.11.0-gke.6. The 'Connect' button is highlighted with a red arrow.

→ Hit **Run in Cloud Shell** in the popup window

This screenshot shows a modal dialog titled 'Connect to the cluster'. It provides instructions for connecting via command-line or dashboard. The 'Command-line access' section contains a command line for 'gcloud container clusters get-credentials' and a 'Run in Cloud Shell' button, which is also highlighted with a red arrow. The 'Cloud Console dashboard' section has a 'Open Workloads dashboard' button. At the bottom right of the dialog is an 'OK' button.

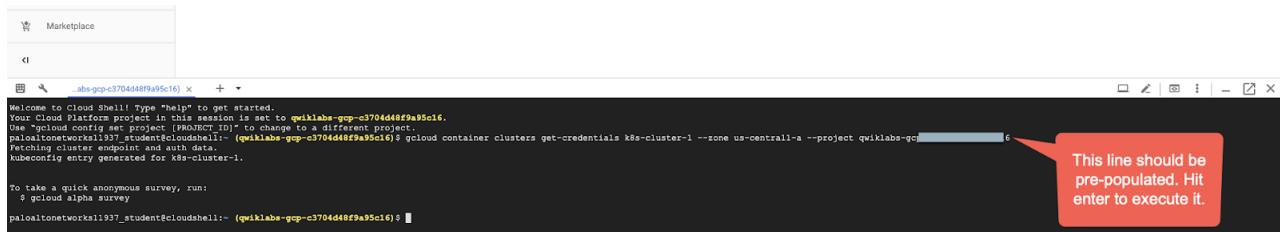
Note: If you're doing this for the first time, you will also see the following popup window.

→ Hit **START CLOUD SHELL** in that window



→ Hit enter to execute the pre-populated command in the Cloud Shell to connect to the Kubernetes cluster

```
gcloud container clusters get-credentials k8s-cluster-1 --zone us-central1-a --project <project-id>
```



→ Open the Cloud Shell in a new tab for convenience (DO NOT CLOSE THE NEW TAB)



Task 2 – Build and Scan the Application Container Image

In this activity we will start by building our app container image, then we will scan it for security vulnerabilities.

We will build the frontend service for our Guestbook app. The Development team wrote the code, and now we are packaging the code in a container.

This is the Dockerfile, which describes what we are including in this image (base OS/image, code and code dependencies/libraries):

```
1 # Copyright 2016 The Kubernetes Authors.
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 #     http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 FROM php:5-apache
16
17 RUN apt-get update
18 RUN pear channel-discover pear.nrk.io
19 RUN pear install nrk/Predis
20
21 # If the container's stdio is connected to systemd-journald,
22 # /proc/self/fd/{1,2} are Unix sockets and apache will not be able to open()
23 # them. Use "cat" to write directly to the already opened fds without opening
24 # them again.
25 RUN sed -i 's#errorLog /proc/self/fd/2#errorLog "|$/bin/cat 1>&2#" /etc/apache2/apache2.conf
26 RUN sed -i 's#customLog /proc/self/fd/1 combined#customLog "|$/bin/cat" combined#' /etc/apache2/apache2.conf
27
28 # Add the application code to the image
29 ADD guestbook.php /var/www/html/guestbook.php
30 ADD controllers.js /var/www/html/controllers.js
31 ADD index.html /var/www/html/index.html
```

In this Dockerfile...

- We are using PHP:5-apache **base image** for our frontend app container (line 15)
https://hub.docker.com/_/php
- Installing and updating **dependencies** (line 17-19)
- Adding the frontend **app code**, PHP, JavaScript, and HTML to the image (line 29-31)

Now, let's build and scan this image.

- Download the lab repo in Cloud Shell by running below cmd:

```
git clone https://github.com/PaloAltoNetworks/ignite2019-how14.git
```

```
paloaltonetworks11946_student@cloudshell:~ (qwiklabs-gcp-161afff642864f2c)$ git clone https://github.com/PaloAltoNetworks/ignite2019-how14.git
Cloning into 'ignite2019-how14'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 23 (delta 5), reused 20 (delta 5), pack-reused 0
Unpacking objects: 100% (23/23), done.
paloaltonetworks11946_student@cloudshell:~ (qwiklabs-gcp-161afff642864f2c)$ ls
ignite2019-how14 README-cloudshell.txt
```

- Access the directory with the Dockerfile by running the following command:

```
cd ignite2019-how14/code
```

```
paloaltonetworks11946_student@cloudshell:~ (qwiklabs-gcp-161afff642864f2c)$ cd ignite2019-how14/code/
paloaltonetworks11946_student@cloudshell:~/ignite2019-how14/code (qwiklabs-gcp-161afff642864f2c)$ ls
controllers.js Dockerfile Dockerfile.withScanning guestbook.php index.html Makefile
```

- Edit the Dockerfile to add Prisma Public Cloud image scanning API call

- Open the Dockerfile in your favorite editor

nano Dockerfile

- Go to the end of the file and append the following two lines at the end

ARG rl_args

```
RUN SCAN_CMD=$(eval "curl https://vscanapidoc.redlock.io/scan.sh 2>/dev/null") && echo
"$SCAN_CMD" | sh
```

- Save and exit

Press **Ctrl + o** to save and then **Ctrl + x** to exit

After making the changes Dockerfile should look like this:

```
paloaltonetworks11946_student@cloudshell:~/ignite2019-how14/code (qwiklabs-gcp-161afff642864f2c)$ tail Dockerfile
# them. Use "cat" to write directly to the already opened fds without opening
# them again.
RUN sed -i 's#ErrorLog /proc/self/fd/2#ErrorLog "|$/bin/cat 1>\&2"#' /etc/apache2/apache2.conf
RUN sed -i 's#CustomLog /proc/self/fd/1 combined#CustomLog "|/bin/cat" combined#' /etc/apache2/apache2.conf

ADD guestbook.php /var/www/html/guestbook.php
ADD controllers.js /var/www/html/controllers.js
ADD index.html /var/www/html/index.html
ARG rl_args
RUN SCAN_CMD=$(eval "curl https://vscanapidoc.redlock.io/scan.sh 2>/dev/null") && echo "$SCAN_CMD" | sh
```

What we're doing here by adding the r1_args and SCAN_CMD is, we are listing all the packages installed in this image and getting the list of all the vulnerabilities associated with those packages from the **Prisma Public Cloud free public image scanning API**. The **Prisma Public Cloud** Infrastructure as Code Scanner will provide a pass/fail for the build based on the list of vulnerabilities we get back.

ARG r1_args is for passing the build arguments to configure when to pass/fail the build and how to group/see the scan result. See <https://vscanapidoc.redlock.io/> for more information.

Note: For your convenience, we have placed the final Dockerfile as `Dockerfile.withScanning` in the `ignite2019-how14/code` folder.

→ You can copy that one using the following command:

cp Dockerfile.withScanning Dockerfile

→ Build the Docker image using the following command. This will make the actual API call during the build and display the scan result.

docker build -t gb-frontend:v4 . -f ./Dockerfile

```
gpatel@cloudshell:~/examples/guestbook/php-redis (cps-containers-dev)$ docker build -t gb-frontend:v4 -f ./Dockerfile .
Sending build context to Docker daemon 11.26kB
Step 1/10 : FROM php:5-apache
--> 24c79195c1e
Step 2/10 : RUN apt-get update
--> Using cache
--> 786a33637fffc
Step 3/10 : RUN pear channel-discover pear.nrk.io
--> Using cache
--> f48f92067a15
Step 4/10 : RUN pear install nrk/Predis
--> Using cache
--> lc00a07d5aad
Step 5/10 : RUN sed -i 's#${ErrorLog} /proc/self/fd/2#${ErrorLog} "|$bin/cat 1>\&2"##' /etc/apache2/apache2.conf
--> Using cache
--> 1dbbe1114be
Step 6/10 : RUN sed -i 's#${CustomLog} /proc/self/fd/1 combined#${CustomLog} "|$bin/cat" combined##' /etc/apache2/apache2.conf
--> Using cache
--> 7d910a6b7a9c
Step 7/10 : ADD guestbook.php /var/www/html/guestbook.php
--> 4604211d0230
Step 8/10 : ADD controllers.js /var/www/html/controllers.js
--> cfe1bec97983
Step 9/10 : ADD index.html /var/www/html/index.html
--> 53be33fd4ee8
Step 10/10 : RUN SCAN_CMD=$(eval "curl https://vscanapidoc.redlock.io/scan.sh 2>/dev/null") && echo "$SCAN_CMD" | sh
--> Running in 9bdf91997513

{
  "Report": {
    "Summary": {
      "high_cve_count": 35, ←
      "medium_cve_count": 234,
      "low_cve_count": 97,
      "unknown_cve_count": 6,
      "total_cve_count": 372,
      "total_packages_count": 100, ←
      "failure_reason": "threshold_exceeded"
    }
  }
}
The command '/bin/sh -c SCAN_CMD=$(eval "curl https://vscanapidoc.redlock.io/scan.sh 2>/dev/null") && echo "$SCAN_CMD" | sh' returned a non-zero code: 1
gpatel@cloudshell:~/examples/guestbook/php-redis (cps-containers-dev)$
```

→ Next, analyze the completed results and take note of the following:

- Notice the docker build failing with a non-zero exit code
- It fails because the vulnerability scan result received from the Prisma Public Cloud image scan API endpoint indicate more than one packages have known vulnerabilities
- Notice that the final image would have had 38 high severity CVEs, 248 medium and 102 low severity CVEs, totaling 394 CVEs.

Note: Your results may be different as new CVEs are being identified.

- The number of packages analyzed are 100
- Failure reason is the number of CVEs exceeded the threshold (by default 1)

→ Next, get the list of CVEs grouped by the packages by passing the

--build-arg rl_args="report=detail;group_by=package"

argument to the docker build command

```
docker build -t gb-frontend:v4 . -f ./Dockerfile --build-arg  
rl_args="report=detail;group_by=package"
```

```
gpateel@cloudshell:~/examples/guestbook/php-redis (cps-containers-dev)$ docker build -t gb-frontend:v4 -f ./Dockerfile --build-arg rl_args="report=detail;group_by=package" .
Sending build context to Docker daemon 11.26kB
Step 1/11 : FROM php:5.6-apache
--> 5e7c79195c6d
Step 2/11 : RUN apt-get update
--> Using cache
--> b515aef831346
Step 3/11 : RUN pear channel-discover pear.nrk.io
--> Using cache
--> 7b0b55fbcd
Step 4/11 : RUN pear install nrk/Predis
--> Using cache
--> 7ac8167b4d6a
Step 5/11 : RUN sed -i '$#ErrorLog /proc/self/fd/2##$# /etc/apache2/apache2.conf
--> Using cache
--> d95ea97c42f9
Step 6/11 : RUN sed -i '$#CustomLog /proc/self/fd/1 combined#CustomLog |/bin/cat" combined#' /etc/apache2/apache2.conf
--> Using cache
--> c26622f90e
Step 7/11 : ADD guestbook.php /var/www/html/guestbook.php
--> Using cache
--> d95ea97c42f9
Step 8/11 : ADD controllers.js /var/www/html/controllers.js
--> Using cache
--> fff3acf9fd62
Step 9/11 : ADD index.html /var/www/html/index.html
--> Using cache
--> e66968acd9cd
Step 10/11 : ARG rl_args
--> Running in e744ae371392
Removing intermediate container e744ae371392
--> 61567fdb6e19
Step 11/11 : RUN SCAN_CMD=$eval "curl https://vscanapidoc.redlock.io/scan.sh 2>/dev/null" && echo "$SCAN_CMD" | sh
--> Running in 7aab545e30e8

{
  "Report": {
    "Packages": [
      {
        "Name": "gnupg2",
        "Version": "2.1.18-8+deb9u3",
        "Vulnerabilities": [
          {
            "Name": "CVE-2018-9234",
            "NamespaceName": "debian:9",
            "Description": "GnuPG 2.2.4 and 2.2.5 does not enforce a configuration in which key certification requires an offline master Certify key, which results in apparently valid certifications that occurred only with access to a signing subkey.",
            "Link": "https://security-tracker.debian.org/tracker/CVE-2018-9234",
            "Severity": "Low",
            "Metadata": {
              "NVD": {
                "CVSSv2": {

```

Note: The output might look different

End of Activity 5

Activity 6 – Kubernetes App Manifest Scanning for Security Misconfigurations

In this activity, you will:

- *Scan your kubernetes application deployment manifest using Prisma Public Cloud Infrastructure-as-Code (IaC) public API for security best practices*
 - *Analyze the result*
 - *Fix all the applicable misconfigurations*
-

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/>

What is Kubernetes?

Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management.

What is a Kubernetes Manifest?

Kubernetes manifest file describes how your containerized application is deployed in kubernetes. There can be one or more objects in a manifest file such as [Deployment](#) (replicated group of [Pods](#)), [Services](#) (proxy), [Volumes](#), and [ConfigMaps](#) (configuration for the application pods/containers). Manifest files can be in JSON or YAML format. YAML format is more common in kubernetes world, so we will use that in this lab, but Prisma Public Cloud Infrastructure-as-Code (IaC) API supports both JSON and YAML format.

What does “scanning” the manifest file mean?

When you scan your kubernetes manifest files using the free Prisma Public Cloud Infrastructure-as-Code (IaC) API, you get back the analysis result that points of any configuration which is vulnerable to exploitation. The scan result will have severity associated with each of the rule violations.

You can include this scan into your CI/CD (Continuous Integration/ Continuous Delivery) pipeline, so all your kubernetes manifests go through an automated sanity check before they are applied to production. CI Build should fail if any of your manifest has a high security security misconfiguration.

This API also allows you to scan [Terraform](#) and [CFT](#) files for security best practices violations. Detailed documentation can be found here: <https://iacscanapidoc.redlock.io/>

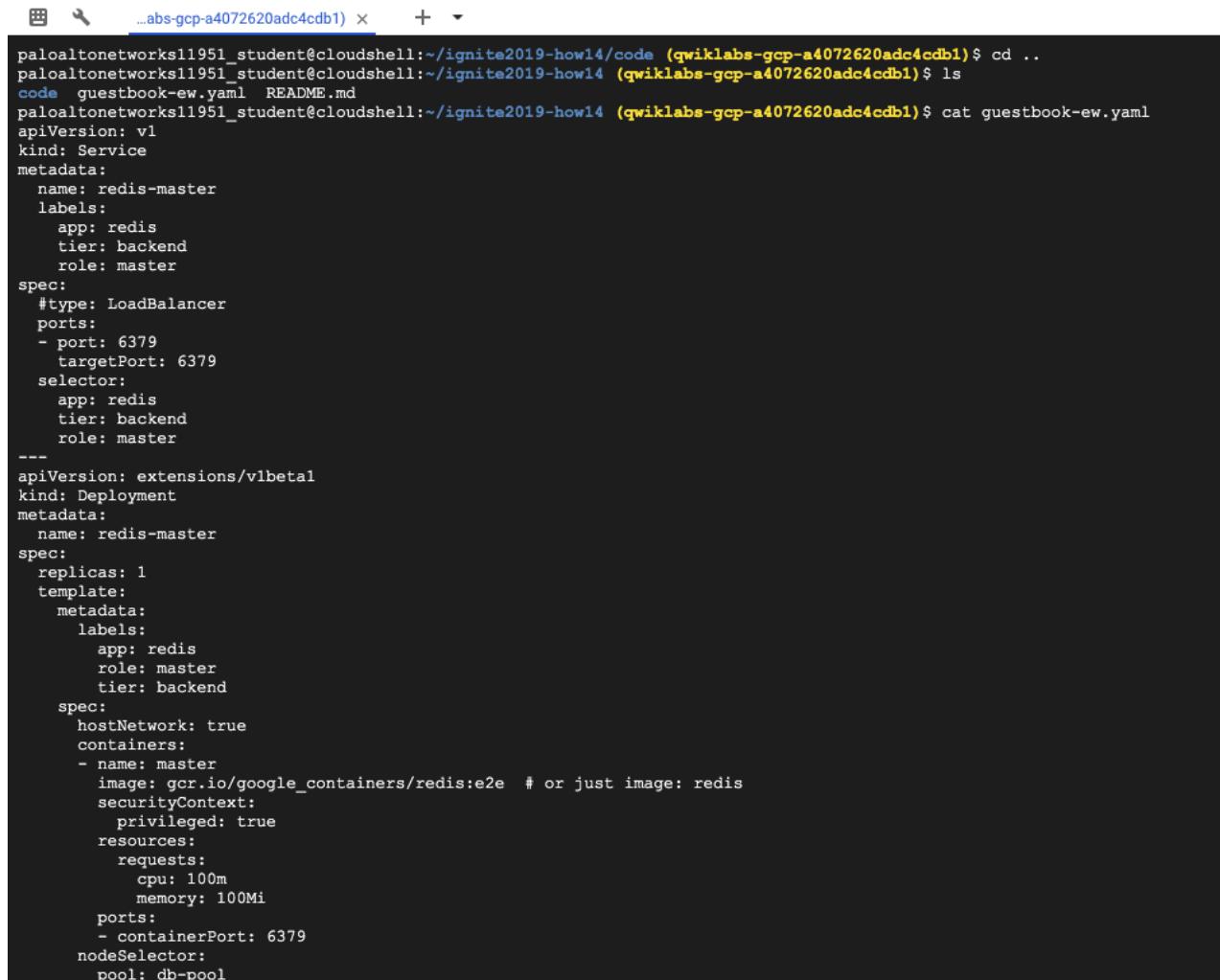
Task 1 – Scan the Application Manifest for Security Best Practices

- Back in the Cloud Shell, explore the manifest files. First, go back to the repo base folder [*ignite2019-how14/*](#) by executing the following command:

```
cd ..
```

- Next view the guestbook application manifest by executing the following command:

```
cat guestbook-ew.yaml
```



```
paloaltonetworks11951_student@cloudshell:~/ignite2019-how14/code (qwiklabs-gcp-a4072620adc4cdb1)$ cd ..
paloaltonetworks11951_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-a4072620adc4cdb1)$ ls
code  guestbook-ew.yaml  README.md
paloaltonetworks11951_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-a4072620adc4cdb1)$ cat guestbook-ew.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-master
  labels:
    app: redis
    tier: backend
    role: master
spec:
  type: LoadBalancer
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
    tier: backend
    role: master
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: redis-master
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: master
        tier: backend
    spec:
      hostNetwork: true
      containers:
      - name: master
        image: gcr.io/google_containers/redis:e2e # or just image: redis
        securityContext:
          privileged: true
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
      nodeSelector:
        pool: db-pool
```

- Scan the guestbook app manifest with Prisma Public Cloud IaC (Infrastructure-as-Code) Scan API using the following command:

```
curl --data-binary @guestbook-ew.yaml -H "Content-Type: application/json" -X POST
https://scanapi.redlock.io/v1/iac | jq .
```

Note: Do not forget the period at the end.

- Analyze the results after the previous curl call:

```
gpatel@cloudshell:~/ignite2019-how14 (cps-containers-dev)$ curl --data-binary @guestbook-ew.yaml -H "Content-Type: application/json" -X POST https://dev.scan.api.redlock.io/v1/iac | jq .
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total Spent   Left Speed
100 3558 100 455 100 3103 1417  9668 --:--:-- --:--:-- 9696
{
  "result": {
    "is_successful": "true",
    "rules_matched": [
      {
        "severity": "high",
        "name": "avoid running privileged containers",
        "rule": "$.spec.template.spec.containers[*].securityContext.privileged == true",
        "id": "92714c07-d12b-4635-aed6-514c5c428c5a"
      },
      {
        "severity": "high",
        "name": "do not share host network with containers",
        "rule": "$.spec.template.spec.hostNetwork == true",
        "id": "99544e17-fc8f-4c77-963e-083ab80c53b0"
      }
    ],
    "severity_stats": {
      "high": 2,
      "low": 0,
      "medium": 0
    }
  }
}
```

As you can see from the scan result, we have 2 potential security misconfigurations in our manifest:

- A container is running in privileged mode which can be dangerous
- Pods in a deployment are sharing network namespace with the host

Both of these are classified as high severity security best practice violations, as you can see from the severity field for both of the violations.

Task 2 – Update the Manifest to Fix the Policy Violations

If these configuration lines are not absolutely necessary then we should remove them. You should work with your developer and security team to discuss other options to avoid these offending configurations which can be potentially exploited. In our case, we will assume we have consulted with our dev and security team and decided to remove both offending violations.

- Open the manifest in your favorite text editor :)

nano guestbook-ew.yaml

- Remove the following lines (line 32)

hostNetwork: true

and (line 36-37)

securityContext:

privileged: true

Use your choice of editor (vi/nano) to modify the *guestbook-ew.yaml* file. To delete a line in the nano editor, you can move your cursor to the line you want to delete and then press *Ctrl + k* to delete that line

→ Save and exit

Press `Ctrl + o` to save and then `Ctrl + x` to exit

→ Rescan the guestbook app manifest again to make sure the policy violations are cleared by executing the following command again:

```
curl --data-binary @guestbook-ew.yaml -H "Content-Type: application/json" -X POST https://scanapi.redlock.io/v1/iac | jq .
```

→ Validate that the policy violations are gone!

```
gpatel@cloudshell:~/ignite2019-how14 (cps-containers-dev)$ curl --data-binary @guestbook-ew.yaml -H "Content-Type: application/json" -X POST https://dev.scan.api.redlock.io/v1/iac | jq .
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload Upload   Total Spent   Left Speed
100 3062  100  35 100 3027     69  6021 --:--:--:--:--:-- 6029
{
  "result": {
    "is_successful": "true"
  }
}
```

You can also scan your Terraform and CFT template files with the same Prisma Public Cloud IaC public API endpoint, and they're all provided for free.

For more details on Kubernetes manifest scanning and IaC API documentation, checkout the documentation page: <https://scanapidoc.redlock.io/>

End of Activity 6

Activity 7 – Launch a two tiered application

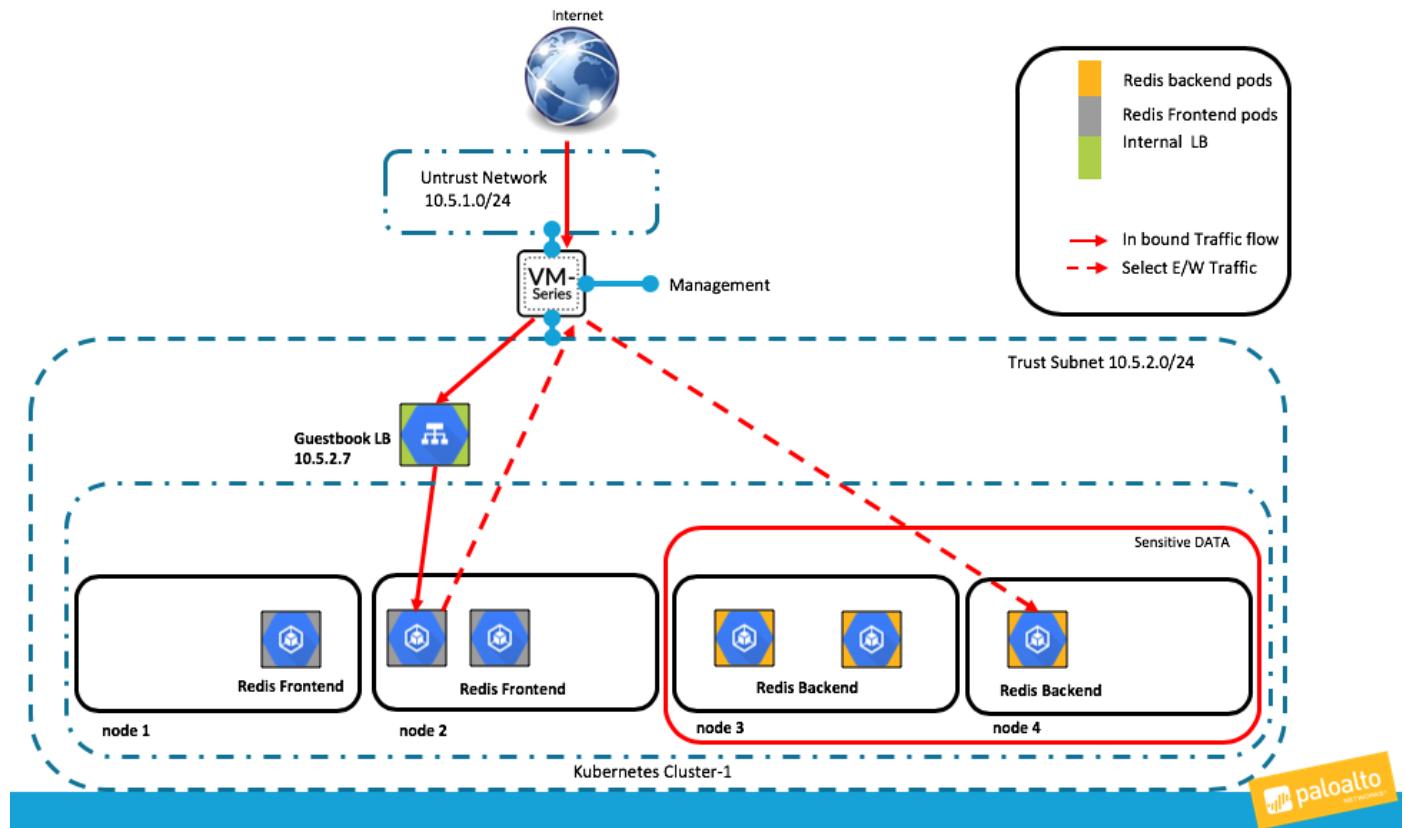
In this activity, you will:

- *Optionally: Explore the application's manifest file that was scanned in Activity 4*
 - *Launch a two tier application within your cluster*
 - *Explore aspects of the application and how a VM-Series Firewall can be used to inspect both North/South and East/West traffic.*
-

Task 1 – Inspect the Guestbook Manifest file

In Activity 4 a Guestbook Manifest file was scanned for CVEs and corrected based on the results of the Prisma Public Cloud IaC public API. 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 customers and provide them with promotions or advertisements.

During the initial deployment a K8s cluster with a VM-Series firewall was deployed. Now the guestbook manifest file that was used in Activity 4 will be deployed to the K8's environment. Once deployed the environment will look like the following diagram:



As you can see this is a two-tiered application with Pods that are dedicated to front-end web services and backend DB services. By selecting which pods are placed in the Node Pool designated for sensitive data, it is possible to analyze select intrapod traffic. This results in sensitive information benefiting from increased protections over cloud native port/protocol access lists.

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/ignite2019-how14/blob/master/guestbook-ew.yaml>

- The manifest can also be viewed by executing the following command in the cloud console:

more guestbook-ew.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 frontend and backend tier. The backend tier will consist of a redis-master and redis-slave for db redundancy. **Node Selectors** are used to make sure the web frontend and DB backend are placed on the desired nodes:

```
paloaltonetworks11960_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-b8aa635eeab06cd5)$ more guestbook-ew.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-master
  labels:
    app: redis
    tier: backend
    role: master
spec:
  type: LoadBalancer
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
    tier: backend
    role: master
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: redis-master
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: master
        tier: backend
    spec:
      containers:
        - name: master
          image: gcr.io/google_containers/redis:e2e # or just image: redis
          resources:
            requests:
              cpu: 100m
              memory: 100Mi
          ports:
          - containerPort: 6379
            nodeSelector:
              pool: db-pool
              ←
apiVersion: v1
kind: Service
metadata:
  name: redis-slave
  labels:
    app: redis
    tier: backend
    role: slave
spec:
  type: LoadBalancer
  ports:
  - port: 6379
  selector:
    app: redis
    tier: backend
    role: slave
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: redis-slave
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: redis
        role: slave
        tier: backend
    spec:
      containers:
        - name: slave
          image: gcr.io/google_samples/gb-redisslave:v1
          resources:
            requests:
              cpu: 100m
              memory: 100Mi
          env:
            - name: GET_HOSTS_FROM
              value: dns
              # If your cluster config does not include a dns service, then to
              # instead access an environment variable to find the master
              # service's host, comment out the 'value: dns' line above, and
              # uncomment the line below:
              # value: env
          ports:
          - containerPort: 6379
            nodeSelector:
              pool: db-pool
              ←
```

Looking at the frontend section of the Manifest file shows the Internal load balancer definition that is being deployed and along with the node selector for the frontend web services:

```
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: frontend  
  annotations:  
    cloud.google.com/load-balancer-type: "Internal"  
  labels:  
    app: guestbook  
    tier: frontend  
spec:  
  # if your cluster supports it, uncomment the following to automatically create  
  # an external load-balanced IP for the frontend service.  
  type: LoadBalancer  
  ports:  
  - port: 80  
    selector:  
      app: guestbook  
      tier: frontend  
  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
  name: frontend  
spec:  
  replicas: 4  
  template:  
    metadata:  
      labels:  
        app: guestbook  
        tier: frontend  
    spec:  
      containers:  
      - name: php-redis  
        image: gcr.io/google-samples/gb-frontend:v4  
        resources:  
          requests:  
            cpu: 100m  
            memory: 100Mi  
        env:  
        - name: GET_HOSTS_FROM  
          value: dns  
          # If your cluster config does not include a dns service, then to  
          # instead access environment variables to find service host  
          # info, comment out the 'value: dns' line above, and uncomment the  
          # line below:  
          # value: env  
        ports:  
        - containerPort: 80  
        nodeSelector:  
          pool: web-pool
```



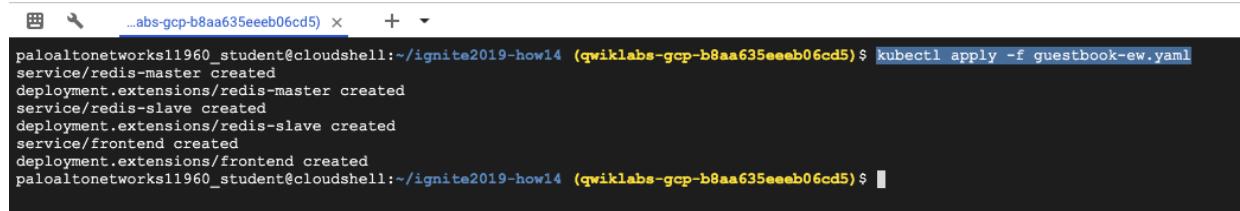
paloalto@networks11900_student:~\$ cloudshell:~/ignite2019-how14 (qwiklabs-gcp-b8aa635eeeb06cd5) \$

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.

Task 2 – Launch the Application

→ Back in your cloud shell type the following command to create the application:

```
kubectl apply -f guestbook-ew.yaml
```

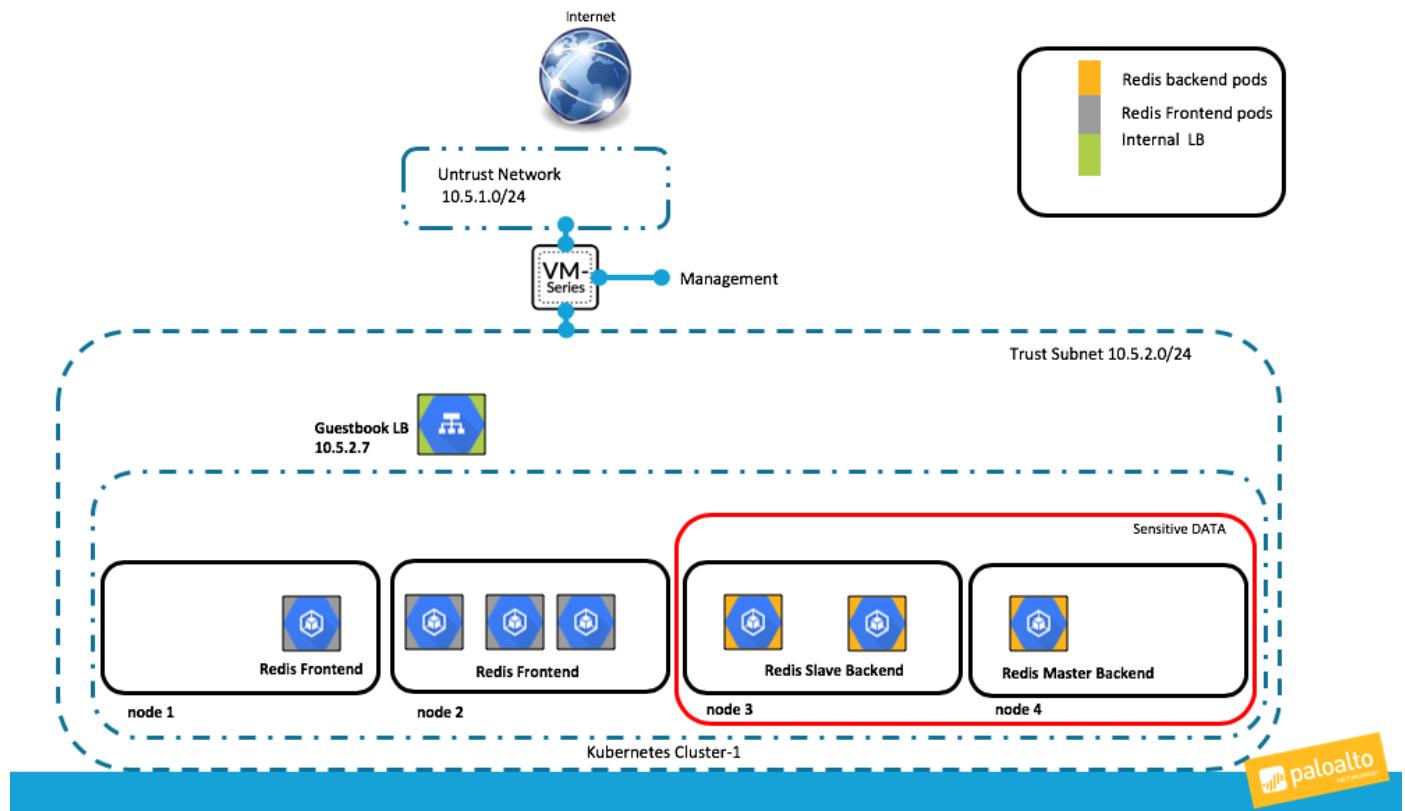


```
paloaltonetworks11960_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-b8aa635eeeeb06cd5)$ kubectl apply -f guestbook-ew.yaml
service/redis-master Created
deployment.extensions/redis-master created
service/redis-slave created
deployment.extensions/redis-slave created
service/frontend created
deployment.extensions/frontend created
paloaltonetworks11960_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-b8aa635eeeeb06cd5)$
```

You should see the services and deployments being created.

Task 3 – Explore what was just deployed

This is now the environment with the newly deployed pods added:



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

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

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE
frontend-549dd7fd98-dvnm	0/1	ContainerCreating	0	20s	<none>	gke-k8s-cluster-1-web-tier-bele76da-82qh	<none>
frontend-549dd7fd98-k5g98	0/1	ContainerCreating	0	20s	<none>	gke-k8s-cluster-1-web-tier-bele76da-82qh	<none>
frontend-549dd7fd98-p9zjd	0/1	ContainerCreating	0	20s	<none>	gke-k8s-cluster-1-web-tier-bele76da-cvk3	<none>
frontend-549dd7fd98-qz9bm	0/1	ContainerCreating	0	20s	<none>	gke-k8s-cluster-1-web-tier-bele76da-cvk3	<none>
redis-master-85d458569f-f8bt2	0/1	ContainerCreating	0	21s	<none>	gke-k8s-cluster-1-default-pool-4e9e74e8-8dp7	<none>
redis-slave-7dcc7fb5dd-4m46h	1/1	Running	0	21s	10.16.0.8	gke-k8s-cluster-1-default-pool-4e9e74e8-8dp7	<none>
paloaltonetworks11985_student	1/1	Running	0	21s	10.16.1.11	gke-k8s-cluster-1-default-pool-4e9e74e8-93hs	<none>

You can see that the Ready and Status of pods as they start up. Verify that all the pods get to created and running. Also notice that the fronted and redis pods are on different node pools.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE
frontend-549dd7fd98-4jwnw	1/1	Running	0	8m	10.16.2.4	gke-k8s-cluster-1-web-tier-la23b339-its3	<none>
frontend-549dd7fd98-9fcvh	1/1	Running	0	8m	10.16.2.3	gke-k8s-cluster-1-web-tier-la23b339-its3	<none>
frontend-549dd7fd98-w4sdc	1/1	Running	0	8m	10.16.3.3	gke-k8s-cluster-1-web-tier-la23b339-kncl	<none>
frontend-549dd7fd98-w9w7k	1/1	Running	0	8m	10.16.3.4	gke-k8s-cluster-1-web-tier-la23b339-kncl	<none>
redis-master-85d458569f-lqwn7	1/1	Running	0	8m	10.16.1.6	gke-k8s-cluster-1-default-pool-e2467308-sxml	<none>
redis-slave-7dcc7fb5dd-9dqdk	1/1	Running	0	8m	10.16.1.7	gke-k8s-cluster-1-default-pool-e2467308-sxml	<none>
redis-slave-7dcc7fb5dd-z6g98	1/1	Running	0	8m	10.16.0.12	gke-k8s-cluster-1-default-pool-e2467308-81nd	<none>
paloaltonetworks11985_student	1/1	Running	0	8m	10.16.0.12	gke-k8s-cluster-1-default-pool-e2467308-81nd	<none>

- 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 initially this may show as pending:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
frontend	LoadBalancer	10.19.248.199	<pending>	80:30261/TCP	18s
kubernetes	ClusterIP	10.19.240.1	<none>	4/TCP	12m
redis-master	ClusterIP	10.19.240.183	<none>	6379/TCP	18s
redis-slave	ClusterIP	10.19.249.193	<none>	6379/TCP	18s

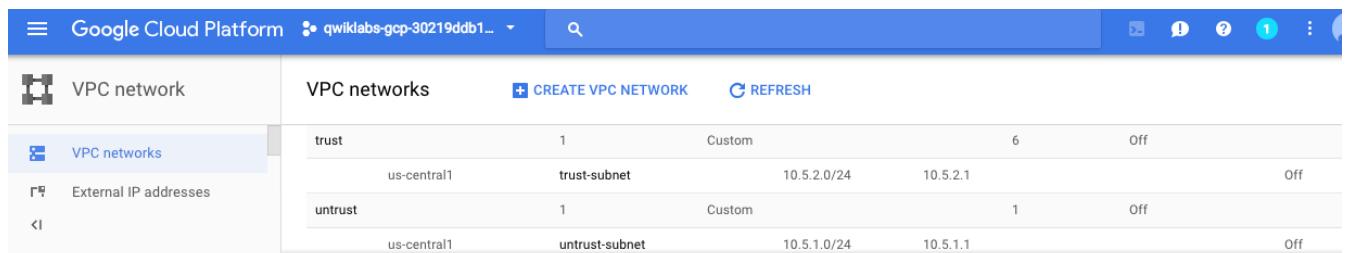
Repeating the “kubectl get svc” command will eventually show a private ip address for the internal load balancer.



```
paloaltonetworks11960_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-b8aa635eeeb06cd5)$ kubectl get svc
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
frontend   LoadBalancer  10.19.252.91  10.5.2.7    30:31274/TCP  12m
kubernetes ClusterIP  10.19.240.1   <none>        443/TCP   1h
redis-master ClusterIP  10.19.253.199  <none>        6379/TCP  12m
redis-slave  ClusterIP  10.19.243.216  <none>        6379/TCP  12m
paloaltonetworks11960_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-b8aa635eeeb06cd5)$
```

The private IP address is from the trust subnet's CIDR block.

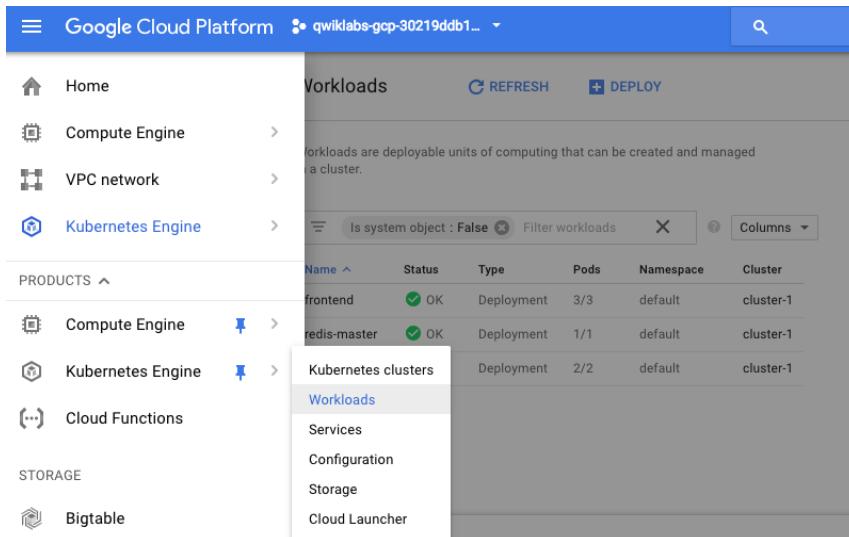
→ navigate to **VPC network > VPC networks** to see the GCP config



VPC network	VPC networks	CREATE VPC NETWORK	REFRESH		
VPC networks	trust	1	Custom	6	Off
	us-central1	trust-subnet	10.5.2.0/24	10.5.2.1	Off
External IP addresses	untrust	1	Custom	1	Off
	us-central1	untrust-subnet	10.5.1.0/24	10.5.1.1	Off

To see the new Kubernetes resources in the GCP console

→ navigate to **Kubernetes Engine > Workloads:**



Name	Status	Type	Pods	Namespace	Cluster
frontend	OK	Deployment	3/3	default	cluster-1
redis-master	OK	Deployment	1/1	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, Services, Configuration, and Storage. The 'Workloads' icon is selected. The main area has a title 'Workloads' with 'REFRESH' and 'DEPLOY' buttons. Below this is a table showing three workloads: 'frontend', 'redis-master', and 'redis-slave'. Each workload is listed with its name, status (OK), type (Deployment), number of pods (3/3, 1/1, 2/2), namespace (default), and cluster (cluster-1). A tooltip above the table explains that workloads are deployable units of computing.

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 page, specifically the 'Load balancing' section. The sidebar includes links for Home, Compute Engine, VPC network, and Kubernetes Engine. Under 'PRODUCTS', there are links for SQL and Spanner. Under 'NETWORKING', there are links for VPC network, Network services, Hybrid Connectivity, and Network Security. The 'Network services' link is expanded, showing 'Load balancing' as a sub-section. A terminal window at the bottom displays the command 'kubectl get svc' with its output, which includes entries for Cloud DNS, Cloud CDN, and Cloud Launcher.

```
l58a20:~$ kubectl get svc
NAME         PORT(S)        AGE
Cloud DNS   80:30895/TCP  1m
Cloud CDN   443/TCP      9m
Cloud Launcher 6379/TCP  1m

```

→ Click on the load balancer to get more details:

The screenshot shows the Google Cloud Platform interface for Network services, specifically the Load balancing section. On the left sidebar, under 'Load balancing', there are four options: Cloud DNS, Cloud CDN, Cloud NAT, and Traffic Director. The main area displays a table for 'Load balancers' with one row visible:

Name	Protocol	Backends
ad37f5a927d7411e9b36442010a8001b	TCP (Internal)	1 regional backend service (1 instance group)

A note at the bottom says: '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:

The screenshot shows the 'Load balancer details' page for the load balancer 'ad37f5a927d7411e9b36442010a8001b'. The 'Frontend' section is highlighted with a red box around the 'Subnetwork' column, which shows 'trust-subnet (10.5.2.0/24)'. Other columns include 'IP:Ports' (10.5.2.7:80) and 'DNS name'.

Protocol	Subnetwork	IP:Ports	DNS name
TCP	trust-subnet (10.5.2.0/24)	10.5.2.7:80	

Below the frontend table, the 'Backend' section shows an instance group 'k8s-ig-e59c5e75a0f61b49' with 4 healthy instances in zone 'us-central1-a'. The 'Advanced configurations' section is collapsed.

End of Activity 7

Activity 8 – Securing Inbound Traffic

In this activity, you will:

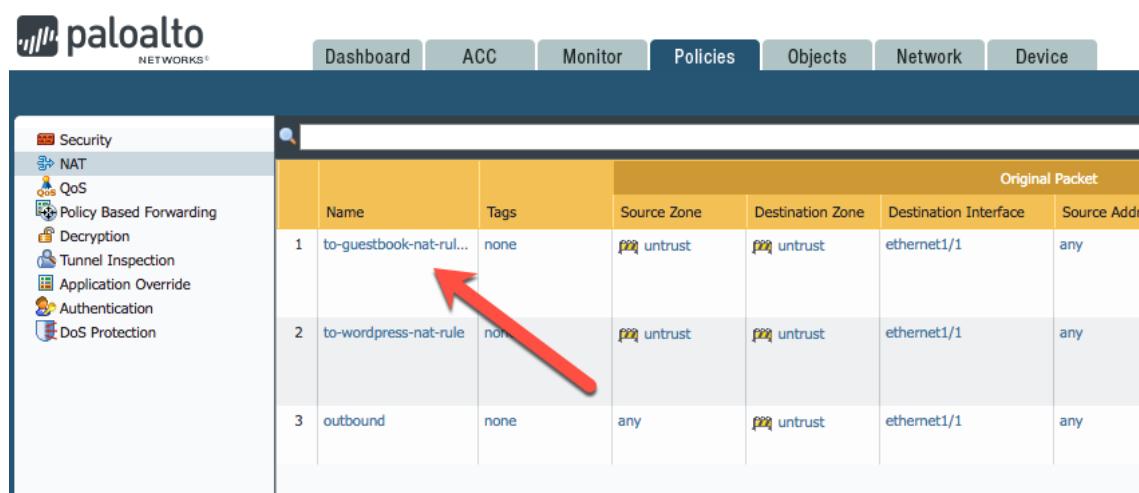
- **Secure traffic that is inbound to your frontend service**
- **Validate that traffic is visible in the Firewall logs**

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 10.5.2.7)

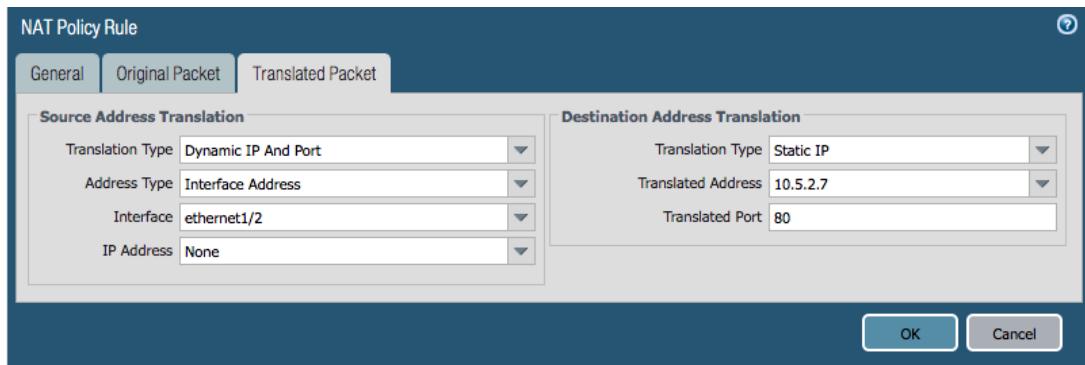
Task 2 – Update the Firewall’s NAT Policy

- Login in to the VM-Series firewall using the management interface’s ip address.
<https://<firewall mgmt ip-addr>>
- Hint:** Check Activity 4, 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:



Name	Tags	Original Packet				Source Address
		Source Zone	Destination Zone	Destination Interface	Source Address	
1 to-guestbook-nat-rul...	none	untrust	untrust	ethernet1/1	any	
2 to-wordpress-nat-rule	none	untrust	untrust	ethernet1/1	any	
3 outbound	none	any	untrust	ethernet1/1	any	

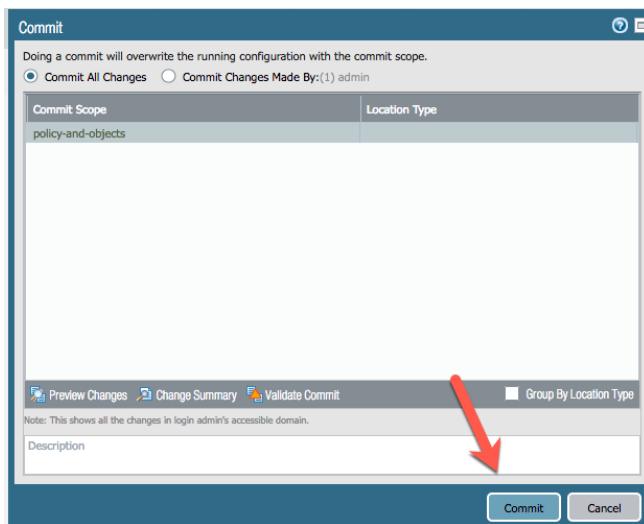
- Click on the **Translated Packet** tab. In the **Translated Address** field, enter the load balancer's ip address from Task 1 and click **OK**. It might not need to be changed.



- 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	2019-06-18 10:15:45
	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, 'VM instances' is selected. The main area shows a table of VM instances. One instance, 'firewall1', is selected. The table has columns: Name, Zone, Recommendation, Internal IP, EDIT, RESET, CLONE, STOP, and DELETE. The 'VM instances' tab is active. The table data is as follows:

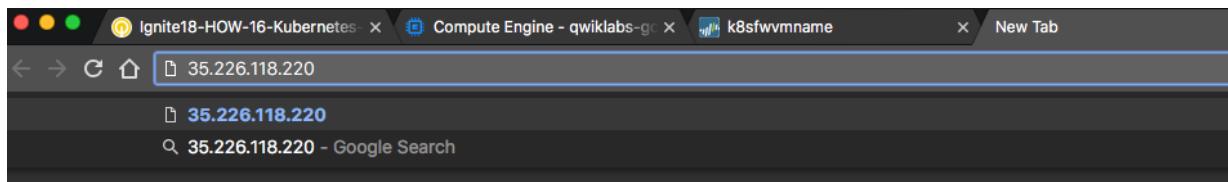
Name	Zone	Recommendation	Internal IP	Actions
firewall1	us-central1-a		10.5.0.4	EDIT RESET CLONE STOP DELETE

Below the table, it says 'VM instance details' with buttons for EDIT, RESET, CLONE, STOP, and DELETE. The 'VM instances' tab is selected. The table data is as follows:

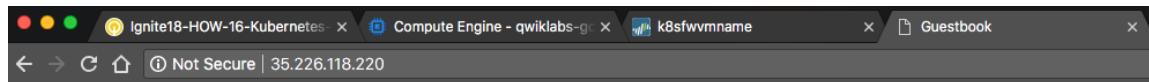
VM instance details					
VM instances					
Creation time May 3, 2018, 9:49:47 AM					
Network interfaces					
Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	IP forwarding
management	management-subnet	10.5.0.4	—	35.225.41.108 (ephemeral)	On
untrust	untrust-subnet	10.5.1.4	—	35.226.118.220 (ephemeral)	
trust	trust-subnet	10.5.2.4	—	None	

Public DNS PTR Record: None

- In your browser navigate to the untrust-ip address of the firewall:
<http://<firewall untrust External IP>>



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



Guestbook

Hello World

Submit

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

Guestbook

Messages

Submit

Hello World

- Open the firewall interface and navigate to the **Monitor** tab. Add the **NAT Dest IP** column.

	Receive Time	Type		
05/24 14:29:17		start	7.54.229	Source User
05/24 14:28:57		start	3.3	
05/24 14:28:57		start	3.3	
05/24 14:28:12		drop	7.54.229	
05/24 14:28:11		drop	7.54.229	
05/24 14:28:06		end	7.54.229	
05/24 14:28:04		end	2.4	
05/24 14:27:56		end	7.54.229	
05/24 14:27:49		end	7.54.229	
05/24 14:27:48		end	7.54.229	
05/24 14:27:39		drop	7.54.229	

- With the **NAT Dest IP** Column visible, it is possible to see the address translation to the internal load balancer IP address. Also notice that there is some E/W traffic between the Frontend and the DB Backend showing up in the logs. More on this in Activity 8.

	Receive Time	Type	From Zone	To Zone	Source	NAT Dest IP	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
🕒	05/23 11:01:54	end	untrust	web	70.42.131.189	10.5.2.7	10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	tcp-fin	4.2k
🕒	05/23 11:01:52	end	web	web	10.16.2.3		10.16.1.9	53	dns	allow	EW	aged-out	458
🕒	05/23 11:01:46	end	web	web	10.16.3.4		10.16.0.5	53	dns	allow	EW	aged-out	458
🕒	05/23 11:01:37	end	web	web	10.16.2.3		10.16.1.11	6379	redis	allow	EW	tcp-fin	708
🕒	05/23 11:01:31	end	web	web	10.16.3.4		10.16.0.8	6379	redis	allow	EW	tcp-fin	708
🕒	05/23 11:01:23	start	web	web	10.16.2.3		10.16.1.11	6379	redis	allow	EW	n/a	307
🕒	05/23 11:01:23	start	web	web	10.16.2.3		10.16.1.9	53	dns	allow	EW	n/a	97
🕒	05/23 11:01:23	start	untrust	web	70.42.131.189	10.5.2.7	10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	n/a	728
🕒	05/23 11:01:17	start	web	web	10.16.3.4		10.16.0.8	6379	redis	allow	EW	n/a	307
🕒	05/23 11:01:17	start	web	web	10.16.3.4		10.16.0.5	53	dns	allow	EW	n/a	97
🕒	05/23 11:01:15	start	untrust	web	168.149.242.101	10.5.2.7	10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	n/a	857

End of Activity 8

Activity 9 – Securing Outbound Traffic

In this activity, you will:

- **Secure outbound traffic from the cluster**
- **Validate traffic is in the Firewall logs**

Task 1 – Add Kube-API server route

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

The screenshot shows the Google Cloud Platform interface under the COMPUTE section. The 'Kubernetes Engine' option is selected, opening a dropdown menu with 'Kubernetes clusters', 'Workloads', 'Services', and 'Configuration'. To the right, a table lists 'Kubernetes clusters' with one entry: 'cluster-1' located in 'us-central1-a' with 2 vCPUs and 7.50 GB of memory. There are 'Connect', 'Edit', and 'Delete' buttons at the bottom of the table.

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

The screenshot shows the 'Kubernetes clusters' details page for 'cluster-1'. The left sidebar shows 'Kubernetes Engine' with 'Kubernetes clusters' selected. The main panel shows cluster details like Master version (1.8.8-gke.0), Endpoint (35.226.171.41), Client certificate (Enabled), and Kubernetes alpha features (Disabled). A red arrow points to the 'Endpoint' field. Other visible fields include 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), and Stackdriver Monitoring (Enabled).

- 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 with 'NETWORKING' options: 'VPC network', 'Network services', and 'Hybrid Connectivity'. Under 'VPC network', there are sub-options: 'VPC networks', 'External IP addresses', 'Firewall rules', 'Routes' (which is highlighted in grey), and 'VPC network peering'. The main content area is titled 'Routes' and shows a table of existing routes. The table has columns for 'Route', 'Destination IP range', and 'Next hop'. There are four rows of default routes listed. At the top right of the table, there are buttons for '+ CREATE ROUTE', 'REFRESH', and 'DELETE'. A red arrow points to the '+ CREATE ROUTE' button.

Create a route with the following parameters:

Name: k8s-mgmt

Network: trust

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

Priority: 10

Next hop: Default internet gateway

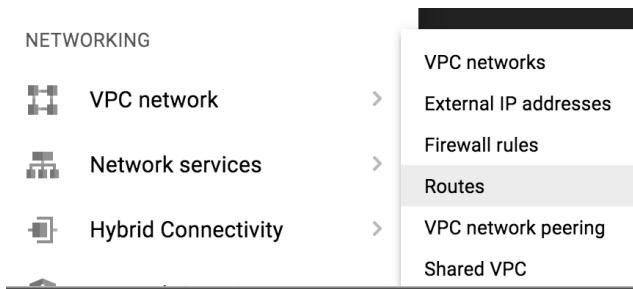
- And then click **Create**.

The screenshot shows the 'Create a route' dialog box. On the left is a sidebar with 'VPC network' navigation items: 'VPC networks', 'External IP addresses', 'Firewall rules', 'Routes' (which is selected and highlighted in blue), 'VPC network peering', 'Shared VPC', and 'Serverless VPC access'. The main form area has the title 'Create a route'. It contains several input fields: 'Name' (set to 'k8s-mgmt'), 'Description (Optional)', 'Network' (set to 'trust'), 'Destination IP range' (set to '35.226.171.41/32'), 'Priority' (set to '1000'), 'Instance tags (Optional)', 'Next hop' (set to 'Default internet gateway'). At the bottom are 'Create' and 'Cancel' buttons, with 'Create' being highlighted by a red arrow. Below the buttons is a note: 'Equivalent REST or command line'.

Task 2 – 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



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

VPC network	Routes	+ CREATE ROUTE	REFRESH	DELETE
	<input type="checkbox"/> default-route-ae75e0c3534835f3	0.0.0.0/0	1000	None
	<input type="checkbox"/> default-route-c28522ea8b1a14f8	10.5.0.0/24	1000	None
	<input type="checkbox"/> default-route-da50f4fa1ac6d95a	10.132.0.0/20	1000	None
	<input type="checkbox"/> default-route-e6bcc1325d731489	10.148.0.0/20	1000	None
	<input type="checkbox"/> default-route-e6d1ad29c60cfdfc	10.146.0.0/20	1000	None
	<input type="checkbox"/> default-route-e76604a7e4330fd1	10.158.0.0/20	1000	None
	<input type="checkbox"/> default-route-f13ee84b623a1bc	0.0.0.0/0	1000	None
	<input type="checkbox"/> default-route-f599dc1e5fb74d0d3	0.0.0.0/0	1000	None
	<input type="checkbox"/> default-route-f83453670643dc95	10.138.0.0/20	1000	None
	<input type="checkbox"/> gke-cluster-1-f2e29940-00b35403-4eee-11e8-93ba-42010a80016f	10.8.0.0/24	1000	None
	<input type="checkbox"/> gke-cluster-1-f2e29940-00cace88-4eee-11e8-93ba-42010a80016f	10.8.1.0/24	1000	None

→ Click on Create Route:

VPC network	Routes	+ CREATE ROUTE	REFRESH	DELETE
	<input type="checkbox"/> default-route-ae75e0c3534835f3	0.0.0.0/0	11	
	<input type="checkbox"/> default-route-c28522ea8b1a14f8	10.5.0.0/24	11	
	<input type="checkbox"/> default-route-da50f4fa1ac6d95a	10.132.0.0/20	11	
	<input type="checkbox"/> default-route-e6bcc1325d731489	10.148.0.0/20	11	

Create a route with the following Parameters:

Name: secure-outbound

Network: trust

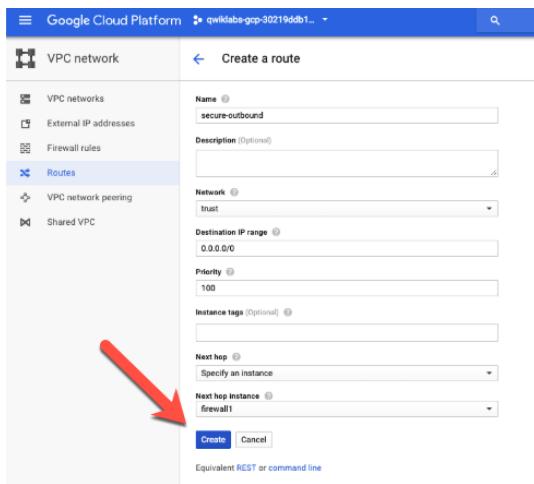
Destination IP Range: 0.0.0.0/0

Priority: 10

Next hop: Specify an instance

Next Hop Instance: firewall1

→ And then click Create



→ Navigate back to the firewall monitor tab and you should now see outbound traffic as well from the cluster node IPs.

Note: You might need to refresh the Monitor page by clicking on the refresh button in the top right corner.

The screenshot shows the Palo Alto Networks Firewall Monitor tab. The left sidebar lists various monitoring categories like Threat, URL Filtering, and Traffic. The main pane displays a table of log entries with the following columns:

Receive Time	Type	From Zone	To Zone	Source	NAT Dest IP	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
05/23 11:20:44	start	web	untrust	10.5.2.5	89.33.8.34	89.33.8.34	0	icmp	allow	Outbound	n/a	160
05/23 11:20:44	start	web	web	10.16.2.4			10.16.0.7	6379	allow	EW	n/a	353
05/23 11:20:44	start	web	web	10.16.2.4			10.16.0.5	53	allow	EW	n/a	98
05/23 11:20:42	start	web	web	10.16.2.4			10.16.0.7	6379	allow	EW	n/a	340
05/23 11:20:42	start	untrust	web	168.149.242.101	10.5.2.7		10.16.0.5	53	allow	EW	n/a	98
05/23 11:20:40	end	web	untrust	10.5.2.2	77.247.110.58	77.247.110.58	0	icmp	allow	to-guestbook-sec-policy-1	n/a	693
05/23 11:20:34	start	web	untrust	10.5.2.2	77.247.110.58	77.247.110.58	0	icmp	allow	Outbound	aged-out	484
05/23 11:18:15	end	web	untrust	10.5.2.5	209.85.200.95	209.85.200.95	443	google-base	allow	Outbound	aged-out	54.0k
05/23 11:18:07	end	web	untrust	10.5.2.6	74.125.201.95	74.125.201.95	443	google-base	allow	Outbound	aged-out	51.5k
05/23 11:18:04	end	web	untrust	10.5.2.2	172.217.212.95	172.217.212.95	443	google-base	allow	Outbound	aged-out	52.0k
05/23 11:11:48	end	untrust	web	168.149.242.101	10.5.2.7		10.5.1.4	80	allow	to-guestbook-sec-policy-1	aged-out	12.9k
05/23 11:11:07	end	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	tcp-rst-from-client	7.0k
05/23 11:11:07	end	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	tcp-rst-from-client	7.0k
05/23 11:11:07	end	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	tcp-rst-from-client	7.0k
05/23 11:11:06	end	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	tcp-rst-from-client	5.9k
05/23 11:11:06	end	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	tcp-rst-from-client	7.0k
05/23 11:11:06	end	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	tcp-rst-from-client	7.0k
05/23 11:10:55	end	web	untrust	10.5.2.6	185.22.154.81	185.22.154.81	0	icmp	allow	Outbound	aged-out	970
05/23 11:10:53	start	web	untrust	10.5.2.3	104.197.174.162	104.197.174.162	443	ssl	allow	Outbound	n/a	444

The source addresses for the outbound traffic are the instance addresses of the Kubernetes cluster nodes.

Name	Zone	Recommendation	In use by	Internal IP
firewall1	us-central1-a			10.50.4 (nic0)
gke-k8s-cluster-1-default-pool-4e9e74e8-8dp7	us-central1-a	k8s-ig-7c8a14505f934663, gke-k8s-cluster-1-default-pool-4e9e74e8-grp		10.5.2.2 (nic0)
gke-k8s-cluster-1-default-pool-4e9e74e8-93hs	us-central1-a	k8s-ig-7c8a14505f934663, gke-k8s-cluster-1-default-pool-4e9e74e8-grp		10.5.2.3 (nic0)
gke-k8s-cluster-1-web-tier-be1e76da-82qh	us-central1-a	gke-k8s-cluster-1-web-tier-be1e76da-grp, k8s-ig-7c8a14505f934663		10.5.2.6 (nic0)
gke-k8s-cluster-1-web-tier-be1e76da-cvk3	us-central1-a	gke-k8s-cluster-1-web-tier-be1e76da-grp, k8s-ig-7c8a14505f934663		10.5.2.5 (nic0)

Task 3 – Test Outbound Pod Traffic

After setting up the outbound routes and looking at the logs in the VM-Series firewall there where most likely traffic showing up that was being originated from the K8s nodes. In this section, you will connect to a Pod and generate outbound traffic to validate visibility.

→ Navigate to the GCP Shell and execute the following command to show the current pods:

```
kubectl get pod -o wide
```

```
paloaltonetworks11985_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-e41669c7b0cce627)$ kubectl get pod -o wide
NAME                               READY   STATUS    RESTARTS   AGE     IP          NODE      NOMINATED NODE
frontend-549dd7fd98-dvnmb         1/1    Running   0          1h      10.16.3.4   gke-k8s-cluster-1-web-tier-be1e76da-82qh   <none>
frontend-549dd7fd98-k5g98          1/1    Running   0          1h      10.16.3.3   gke-k8s-cluster-1-web-tier-be1e76da-82qh   <none>
frontend-549dd7fd98-p9zjd         1/1    Running   0          1h      10.16.2.3   gke-k8s-cluster-1-web-tier-be1e76da-cvk3   <none>
frontend-549dd7fd98-qr9bm         1/1    Running   0          1h      10.16.2.4   gke-k8s-cluster-1-web-tier-be1e76da-cvk3   <none>
redis-master-85d458569f-f8bt2     1/1    Running   0          1h      10.16.0.7   gke-k8s-cluster-1-default-pool-4e9e74e8-8dp7  <none>
redis-slave-7dcc7fb5dd-4s84g      1/1    Running   0          1h      10.16.0.8   gke-k8s-cluster-1-default-pool-4e9e74e8-8dp7  <none>
redis-slave-7dcc7fb5dd-qm46h       1/1    Running   0          1h      10.16.1.11  gke-k8s-cluster-1-default-pool-4e9e74e8-93hs  <none>
paloaltonetworks11985_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-e41669c7b0cce627)$
```

- Copy the first pod name and type in the following command:

```
kubectl exec -it <pod name> /bin/bash
```

In my example the final command would look like this:

```
kubectl exec -it frontend-549dd7fd98-dvnmb /bin/bash
```



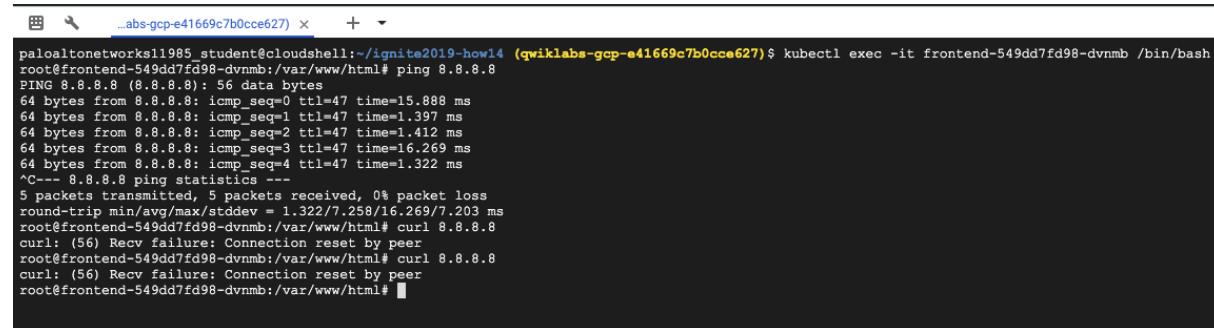
```
paloaltonetworks11985_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-e41669c7b0cce627)$ kubectl exec -it frontend-549dd7fd98-dvnmb /bin/bash
root@frontend-549dd7fd98-dvnmb:/var/www/html#
```

- Next let's attempt to generate outbound internet traffic by pinging a public site:

```
ping 8.8.8.8
```

- Next attempt some web browsing using curl:

```
curl 8.8.8.8
```



```
paloaltonetworks11985_student@cloudshell:~/ignite2019-how14 (qwiklabs-gcp-e41669c7b0cce627)$ kubectl exec -it frontend-549dd7fd98-dvnmb /bin/bash
root@frontend-549dd7fd98-dvnmb:/var/www/html$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=47 time=15.888 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=47 time=1.397 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=47 time=1.412 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=47 time=16.269 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=47 time=1.322 ms
^C--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.322/7.258/16.269/7.203 ms
root@frontend-549dd7fd98-dvnmb:/var/www/html# curl 8.8.8.8
curl: (56) Recv failure: Connection reset by peer
root@frontend-549dd7fd98-dvnmb:/var/www/html# curl 8.8.8.8
curl: (56) Recv failure: Connection reset by peer
root@frontend-549dd7fd98-dvnmb:/var/www/html#
```

You should see that the ping traffic was successful but the web browsing failed.

- Navigate to the firewall and click on the Monitor tab

Receive Time	Type	From Zone	To Zone	Source	NAT Dest IP	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
05/23 12:06:11	deny	web	untrust	10.5.2.6	8.8.8.8	8.8.8.8	80	web-browsing	reset-both	Outbound-deny	policy-deny	351
05/23 12:06:09	end	web	untrust	10.5.2.6	8.8.8.8	8.8.8.8	0	ping	allow	Outbound	aged-out	980
05/23 12:06:09	deny	web	untrust	10.5.2.6	8.8.8.8	8.8.8.8	80	web-browsing	reset-both	Outbound-deny	policy-deny	351
05/23 12:06:06	end	web	untrust	10.5.2.5	77.247.109.201	77.247.109.201	0	icmp	allow	Outbound	aged-out	451
05/23 12:06:01	start	web	untrust	10.5.2.6	8.8.8.8	8.8.8.8	0	ping	allow	Outbound	n/a	490
05/23 12:05:59	start	web	untrust	10.5.2.5	77.247.109.201	77.247.109.201	0	icmp	allow	Outbound	n/a	451
05/23 12:05:36	end	web	untrust	10.5.2.2	92.118.160.37	92.118.160.37	0	icmp	allow	Outbound	aged-out	113
05/23 12:05:31	start	web	untrust	10.5.2.2	92.118.160.37	92.118.160.37	0	icmp	allow	Outbound	n/a	113
05/23 12:01:29	end	web	web	10.5.2.2	10.16.2.3		80	incomplete	allow	EW	aged-out	288
05/23 12:00:14	end	web	untrust	10.5.2.5	206.189.86.188	206.189.86.188	0	icmp	allow	Outbound	aged-out	122

Notice that the web-browsing traffic is being denied but the pings are allowed via a different rule.

- Navigate to the Policies tab and look at the **Outbound-deny** rule Action column.

Name	Tags	Type	Source				Destination		Rule Usage			Application	Service	Action
			Zone	Address	User	HDP Profile	Zone	Address	Hit Count	Last Hit	First Hit			
1 to-guestbook-sec-pol...	none	universal	any untrust	any	any	any	any web	any	5	2019-05-23 11:27:54	2019-05-23 11:01:23	any	service-http	Allow
2 to-wordpress-sec-pol...	none	universal	any untrust	any	any	any	any web	any	2	2019-05-23 12:13:39	2019-05-23 11:48:47	any	TCP8888	Allow
3 EW	none	intrazone	any web	any	any	any	(intrazone)	any	57	2019-05-23 11:43:47	2019-05-23 11:01:23	any	dns	Allow
4 Outbound-deny	none	universal	any web	any	any	any	any untrust	any	52	2019-05-23 12:06:11	2019-05-23 11:43:49	any ssh	any	Deny
5 Outbound	none	universal	any web	any	any	any	any untrust	any	311	2019-05-23 12:11:59	2019-05-23 11:10:53	any web-browsing	any	Allow
6 default-deny-all	none	universal	any	any	any	any	any	any	1	2019-05-23 11:24:34	2019-05-23 11:24:34	any	any	Deny

The **Outbound-Deny** rule is blocking select traffic and providing an extra level of visibility. Now it is clear why the web traffic failed.

End of Activity 9

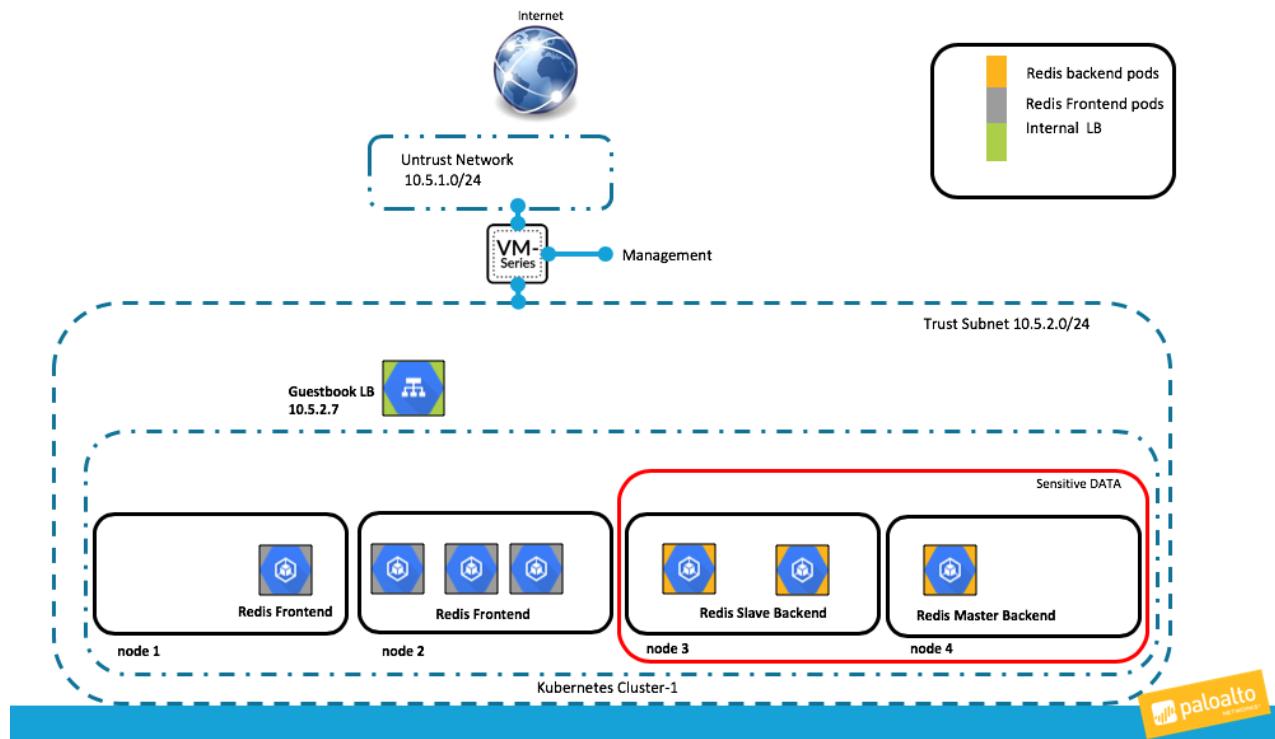
Activity 10 – Investigate Inter Node-Pool traffic

In this activity, you will:

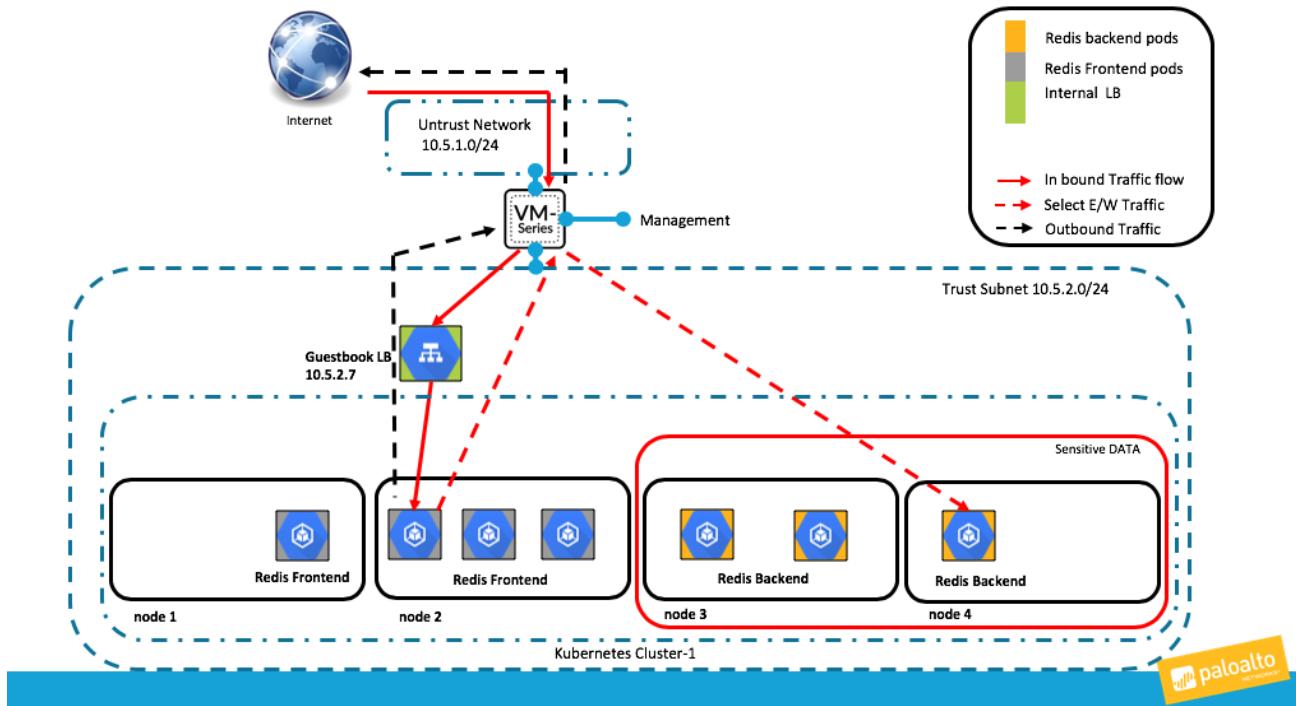
- **Investigate how Inter Node Traffic is being routed through the firewall**
- **Validate that traffic is visible in the Firewall logs**

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. When services are deployed that contain sensitive information it is important to increase the level of visibility. Being able to selectively gain a higher level of security over the cloud native port/protocol approaches increases the overall security posture of applications migrated to the cloud.

In this lab, Node Pools are being used to isolate pods that have sensitive information and traffic destined for these pods are being directed to the firewall for visibility. As shown previously the current environment is setup like this:



The following diagram shows how traffic is moving through the lab environment:



Task 1 – View Node Pool Setup

Each node has been associated with a Node-Pool during the initial setup. You can see the cluster nodes pools opening the GCP and Navigate to **Kubernetes Engine>Clusters**

→ Click k8s-cluster-1

Name	Location	Cluster size	Total cores	Total memory	Master version	Expiration time	Notifications	Labels
k8s-cluster-1	us-central1-a	4	4 vCPUs	15.00 GB	1.11.8-gke.6	Jun 23, 2019		

→ Scroll down and click to expand each node pool

Legacy authorization: Disabled
 Maintenance window: Any time
 Cloud TPU: Disabled
 Application-layer Secrets Encryption: Disabled
 Node auto-provisioning: Disabled
 Vertical Pod Autoscaling: Disabled
 GKE usage metering: Disabled
 Labels: None
 Add-ons
 Permissions

Node Pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit on the top bar. [Learn more](#)

default-pool (2 nodes, version 1.11.8-gke.6)	
web-tier (2 nodes, version 1.11.8-gke.6)	
+ Add node pool	

Look at the **Kubernetes Labels** section. Each node during deployment was given a label that automatically put it into the appropriate pool.

Name	Value
Name	default-pool
Size	2
Node version	1.11.8-gke.6
Node image	Container-Optimized OS with Containerd (cos_containerd) (beta) Change
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Total cores	2 vCPUs
Total memory	7.50 GB
Automatic node upgrades	Disabled
Automatic node repair	Disabled
Autoscaling	Off
Preemptible nodes	Disabled
Boot disk type	Standard persistent disk
Boot disk size in GB (per node)	100
Local SSD disks (per node)	0
Sandbox with gVisor	Disabled
Instance groups	gke-k8s-cluster-1-default-pool-9e39ce4a-grp
Kubernetes labels	
cluster : the-cluster	pool : db-pool
Taints	
No taints set	
GCE instance metadata	
No labels set	
Done Cancel	

Name	Value
Name	web-tier
Size	2
Node version	1.11.8-gke.6
Node image	Container-Optimized OS with Containerd (cos_containerd) (beta) Change
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Total cores	2 vCPUs
Total memory	7.50 GB
Automatic node upgrades	Disabled
Automatic node repair	Disabled
Autoscaling	Off
Preemptible nodes	Disabled
Boot disk type	Standard persistent disk
Boot disk size in GB (per node)	100
Local SSD disks (per node)	0
Sandbox with gVisor	Disabled
Instance groups	gke-k8s-cluster-1-web-tier-729db34-grp
Kubernetes labels	
cluster : the-cluster	pool : web-pool
Taints	
No taints set	
GCE instance metadata	
No labels set	
Done Cancel	

→ Next, click on the **Nodes** tab:

The screenshot shows the Google Cloud Platform interface for the Kubernetes Engine. On the left, there's a sidebar with options like Clusters, Workloads, Services, Applications, Configuration, and Storage. The main area is titled 'Clusters' and shows a single cluster named 'k8s-cluster-1'. Below the cluster name, there are tabs for 'Details', 'Storage', and 'Nodes'. A red arrow points to the 'Nodes' tab, which is currently selected. The 'Details' section contains various configuration details such as Master version (1.11.8-gke.6), Endpoint (35.238.209.14), Client certificate (Enabled), Binary Authorization (Disabled), Kubernetes alpha features (Enabled), Expiration time (Jun 23, 2019, 11:06:48 AM), Total size (4), Master zone (us-central1-a), Node zones (us-central1-a), Network (trust), Subnet (trust-subnet), VPC-native (alias IP) (Disabled), and Pod address range (10.16.0.0/14).

→ Click on one of the nodes:

This screenshot continues from the previous one, showing the 'Nodes' tab selected for the 'k8s-cluster-1'. The main area is titled 'Nodes' and shows a table of nodes. A red arrow points to the 'Status' column header in the table. The table has columns for Name, Status, CPU requested, and CPU alloc. There are four nodes listed: 'gke-k8s-cluster-1-default-pool-9e39ce4a-4dsj' (Status: Ready), 'gke-k8s-cluster-1-default-pool-9e39ce4a-qsvw' (Status: Ready), 'gke-k8s-cluster-1-web-tier-729d6b34-0tgc' (Status: Ready), and 'gke-k8s-cluster-1-web-tier-729d6b34-mwgw' (Status: Ready). The 'Name' column is sorted by name.

Name	Status	CPU requested	CPU alloc
gke-k8s-cluster-1-default-pool-9e39ce4a-4dsj	Ready	671 mCPU	940 mCP
gke-k8s-cluster-1-default-pool-9e39ce4a-qsvw	Ready	772 mCPU	940 mCP
gke-k8s-cluster-1-web-tier-729d6b34-0tgc	Ready	401 mCPU	940 mCP
gke-k8s-cluster-1-web-tier-729d6b34-mwgw	Ready	401 mCPU	940 mCP

- Click on the **Details Tab** and view the **Pod CIDR** Range for this node.

The screenshot shows the 'Node details' page for a node named 'gke-k8s-cluster-1-default-pool-9e39ce4a-4dsj'. The 'Details' tab is selected. In the 'Spec' section, the 'Pod CIDR' field is highlighted with a red box and contains the value '10.16.0.0/24'.

Spec	Pod CIDR	10.16.0.0/24
------	----------	--------------

You can repeat this for the other nodes and see that the nodes in each Node has a defined range for pod deployments.

Task 2 – View GCP Routing Rules

Now lets see how this is used within GCP to redirect traffic to the firewall.

- Within GCP, Navigate to **VPC Network > Routes**

The screenshot shows the 'Routes' page under 'VPC network'. The 'Routes' section is selected. Four specific routes are highlighted with a red box:

- route-dbpool-0: route to 16.0 for web-pool
- route-dbpool-1: route to 16.1 for web-pool
- route-dbpool-2: route to 16.2 for db-pool
- route-webpool-3: route to 16.3 for db-pool

Route	Description	Destination	Next Hop	Tier	Tag
route-dbpool-0	route to 16.0 for web-pool	10.16.0.0/24	10	web-tier	Instance firewall1 (zone us-central1-a)
route-dbpool-1	route to 16.1 for web-pool	10.16.1.0/24	10	web-tier	Instance firewall1 (zone us-central1-a)
route-dbpool-2	route to 16.2 for db-pool	10.16.2.0/24	10	db-tier	Instance firewall1 (zone us-central1-a)
route-webpool-3	route to 16.3 for db-pool	10.16.3.0/24	10	db-tier	Instance firewall1 (zone us-central1-a)

Observe that we have used route tagging during the deployment to also get routes associated with Nodes so traffic that is being sent between the Node-Pools is getting redirected to the VM-Series Firewall.

Task 3 – View the VM-Series Firewall Routing

As a result of the GCP setup, the routing is setup on the VM-Series Firewall is very straight forward. Let's take a look.

- On the VM-Series Firewall, Navigate to **Network > Virtual Routers** and click on **default**.

The screenshot shows the Network > Virtual Routers page. The left sidebar lists various network components like Interfaces, Zones, and GlobalProtect. The main area displays a table for the 'default' virtual router. The table has columns for Name, Interfaces, Configuration, and RIP. The 'Interfaces' column lists 'ethernet1/1' and 'ethernet1/2'. The 'Configuration' column shows 'Static Routes: 2' and 'ECMP status: Disabled'. A red arrow points to the 'default' row in the table.

- Click on **Static Routes**

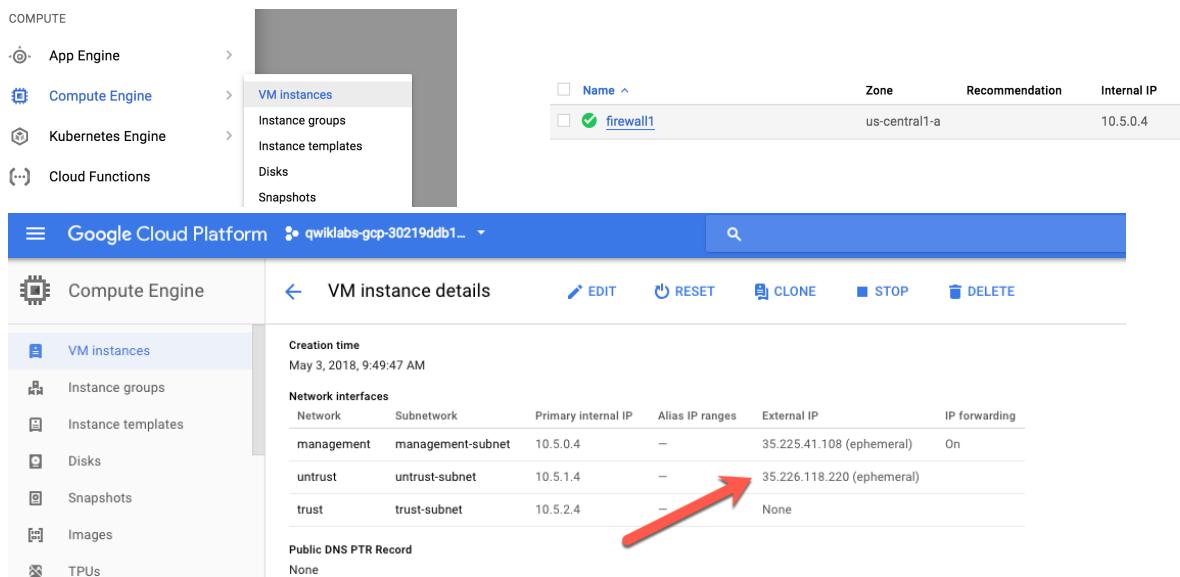
The screenshot shows the 'Virtual Router - default' configuration page under the 'Static Routes' tab. The left sidebar includes options like Router Settings, Static Routes, Redistribution Profile, RIP, OSPF, OSPFv3, BGP, and Multicast. The main area shows the static routes table. The table has columns for Name, Destination, Interface, Type, Value, Admin Distance, Metric, and BFD. Two routes are listed: 'default' (0.0.0.0/0, interface ethernet1/1, type ip-address, value 10.5.1.1) and 'x8' (10.16.0.0/16, interface ethernet1/2, type ip-address, value 10.5.2.1). A red box highlights this table.

Notice that there is a **10.16.0.0/16** route that will cover all the Pod CIDRs on the Nodes. This allows the firewall to send all inter pod traffic back to the GCP fabric without the need for SNAT.

Task 4 – Validate North/South and East/West traffic in the firewall logs

First, repeat the steps done in Activity 6, Task 3.

- Navigate to the **Compute Engine > VM Instances** screen. Then click on the firewall and copy the **External IP address** of the **untrust** interface:

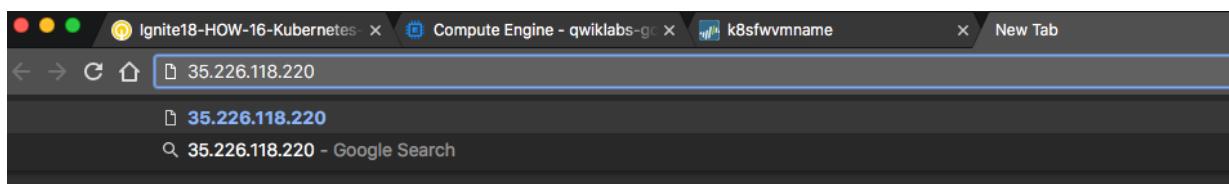


The screenshot shows the Google Cloud Platform Compute Engine interface. In the left sidebar, under the 'COMPUTE' section, 'Compute Engine' is selected. The main area displays 'VM instance details' for a VM named 'firewall1'. The 'Network interfaces' table lists three interfaces: 'management' (Primary internal IP: 10.5.0.4, External IP: 35.225.41.108), 'untrust' (Primary internal IP: 10.5.1.4, External IP: 35.226.118.220), and 'trust' (Primary 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:

<http://<firewall untrust External IP>>



- This time when the Guestbook application comes up, enter something in the text field and click Submit.



If the text is echoed below the Submit button, that is showing that the application successfully wrote the entry to the redis backend.

- Navigate back to the Firewall and click on the Monitor Tab

Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
06/02 19:07:31	end	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	tcp-rst-from-server	5.2k
06/02 19:07:28	end	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	tcp-rst-from-server	6.9k
06/02 19:05:59	start	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	n/a	693
06/02 19:05:56	start	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	n/a	678
06/02 19:00:18	drop	untrust	untrust	209.179.242		10.5.1.4	443	not-applicable	deny	default-deny-all	policy-deny	58
06/02 18:54:34	end	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	tcp-rst-from-server	6.1k
06/02 18:54:29	end	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	tcp-rst-from-server	1.8k
06/02 18:53:13	end	web	web	10.16.3.4		10.16.0.10	53	dns	allow	EW	aged-out	652
06/02 18:52:58	end	web	web	10.16.3.4		10.16.0.13	6379	redis	allow	EW	tcp-fin	708
06/02 18:52:44	start	web	web	10.16.3.4		10.16.0.13	6379	redis	allow	EW	n/a	307
06/02 18:52:39	start	untrust	untrust	169.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-policy-1	n/a	791
06/02 18:52:39	start	web	web	10.16.3.4		10.16.0.10	53	dns	allow	EW	n/a	97
06/02 18:52:38	start	untrust	web	168.149.242.101		10.5.1.4	80	web-browsing	allow	to-guestbook-sec-	n/a	783

Looking at the logs, it is evident that the firewall is seeing the inbound web traffic and the Redis traffic between the Frontend Pods and the Backend Redis pods. Also notice that the East/West traffic is hitting a specific rule that will give the ability to create an application based rule and also enable Threat analysis to protect the backend pods with more security capabilities than the native cloud ACL based constructs.

End of Activity 10

Conclusion

Congratulations! You have now successfully integrated the Palo Alto Networks 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. You have also leveraged the Prisma Public Cloud free public APIs to ensure that the deployment is not introducing known CVEs from the repository via the manifest file.