TX Protect
Deployment

Version 3.20, 2025-01-10

Table of Contents

Introduction to TX Protect Deployment

The ThreatX platform is an agentless deployment that supports both AppSec and DevOps teams without locking either into architectural decisions or sacrificing their autonomy and flexibility. Our agentless architecture ensures that there is no need to disrupt either your applications or your operations.

The ThreatX platform is built for hybrid-cloud and on-premise environments and is application agnostic. If deploying the sensors in your environment, it deploys in minutes via Docker containers and blocks in hours, combining WAF, DDoS, bot, and API protection capabilities into one solution for all your applications and API endpoints.

We regularly update the sensors to provide you with the latest protection against the latest emerging attack patterns, new features, and better insights to the risk profile of your web applications and APIs For the latest information, see TX Protect Documentation.

If the ThreatX SOC hosts your sensors, you might notice the number of sensors fluctuate, or that an individual sensor's uptime has changed. This is because sensors are designed to be added, removed, upgraded, and replaced as needed to ensure optimal site availability and protection.

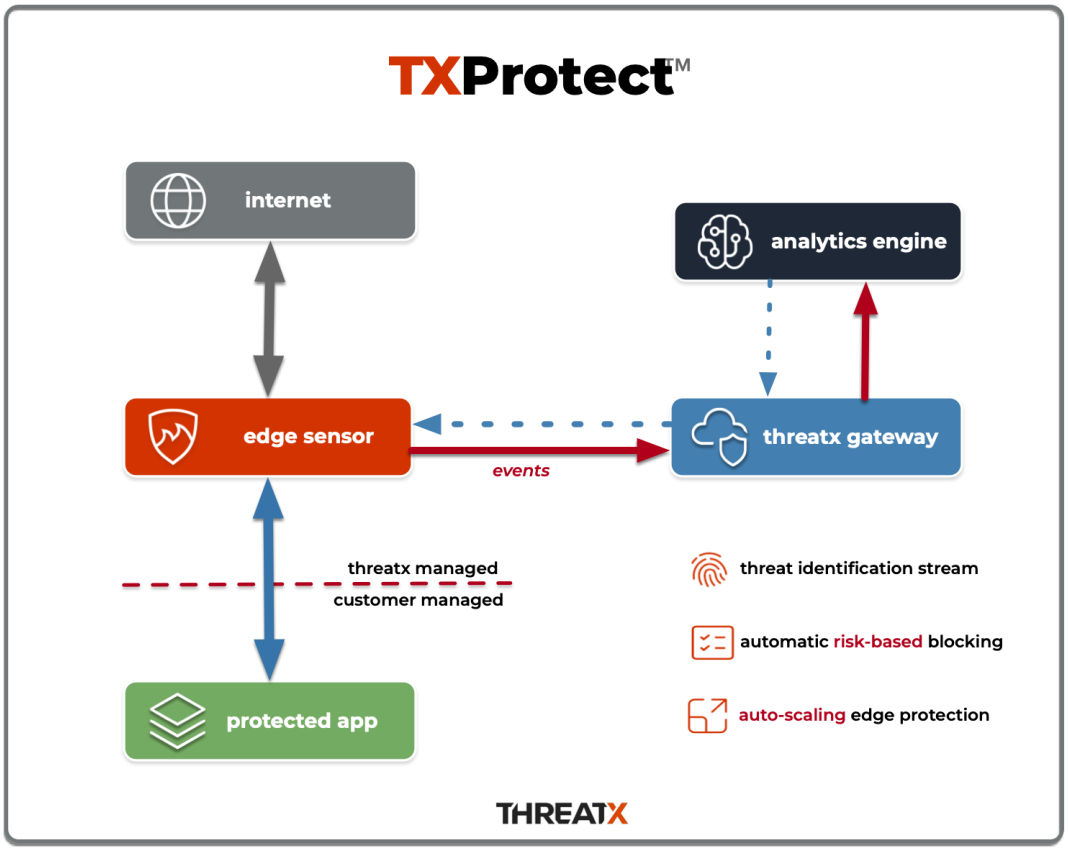
The ThreatX sensors were not designed to monitor site uptime. The ThreatX sensors only see and act on ingress HTTP(s) traffic. Due to the WAFs position in front of your inbound traffic, it is not afforded the same level of insight that a purpose-built monitoring solution would be able to provide.

How Can I Install TX Protect?

Purpose-built for the modern application landscape, ThreatX's web stack agnostic, cloud-native, container-based options deploy in minutes and block in hours, combining WAF, DDoS, bot, and API protection capabilities into one solution for all your applications and API endpoints. TX Protect sensors work with web all stacks

Unlike other sensors such as plugins or source code scanners that need to be installed and upgraded frequently, the TX Protect sensor operates a reverse proxy. This means it decrypts traffic between web clients (such as browsers) on your network with APIs/origin servers before re-encrypting them for you – all without any complicated maintenance.

The TX Protect sensor containers are decoupled from the ThreatX Cloud Analytics platform and can be deployed virtually anywhere, delivering global flexibility and enterprise-grade scalability across complex, geographically dispersed application environments.



The ThreatX platform is flexible, adaptive to customer preference, and compliant with a range of customer network and computing infrastructures. Our agentless architecture lets us deploy our sensors into ThreatX's globally hosted cloud environment, a public cloud infrastructure, and servers hosted by our customers in their data centers. We can honestly say "We've never met an application or API we can't protect!"

Sensor Deployment Options

ThreatX offers four simple deployment methods for the Protect sensor.

ThreatX Cloud (managed)

ThreatX hosts and manages sensor deployment.

Virtual Machine (self-hosted)

ThreatX provides the customer with a machine image compatible with the customer's cloud provider and the customer manages the image deployment, cloud hosting parameters, and cloud-specific support.

Docker (self-hosted)

ThreatX provides the customer with a Docker-based TX Protect sensor container deployed in the customer's data center, and the customer manages the container deployment, container and node parameters, and container-specific support.

Hybrid Deployment

Mix of the ThreatX cloud, public cloud, and Docker deployments deployed when a single deployment model is not feasible. ThreatX will work with the customer to map out the optimal configurations and support models.

On-boarding Checklist

✔ TX Protect Pre-Installation Checklist

Check the box next to any of the requirements that apply to your application…

- ☐ Processes requests with well-formed SQL queries *(E.g., some help desk or bug-tracking software)*
- ☐ Processes requests with well-formed HTML *(E.g., some content management systems)*
- ☐ Requires Two-way SSL/TLS (client authentication)
- ☐ Uses web sockets
- ☐ Requires a specific TLS version or cipher suite restriction *(Default is TLS 1.2 and 1.3)*
- ☐ Supports unique business requirements necessitating custom WAF rules *(E.g, blocking traffic from foreign countries)*
- ☐ Is located behind a firewall or content delivery network (CDN) in which connections from ThreatX service IP addresses would need to be explicitly allowed

If you checked one or more boxes, please contact ThreatX support for assistance with your TX Protect installation.

GCP Terraform Deployment Guide

Summary

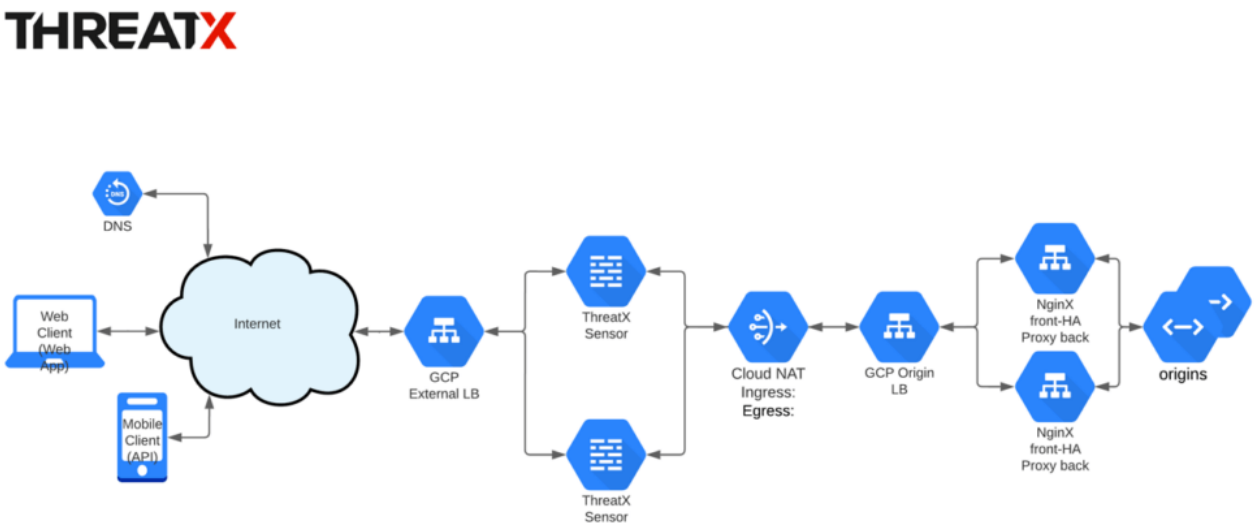
The ThreatX Web Application and API Protection (WAAP) autoscaler sensor is a Terraform module that provides a ThreatX sensor cluster in the Google Cloud Platform (GCP).

The ThreatX Sensor can be deployed behind a GCP Network Load Balancer for high availability. To facilitate HA deployment, ThreatX provides a .tf deployment template. The template may be used ‘as is’ or modified to help deployment into your particular GCP environment.

You must be familiar with Terraform modules to deploy the sensor.

Autoscaler

This template deploys a ThreatX autoscaler behind a network LB, and an egress NAT gateway. ThreatX sensors are deployed in two availability zones within the GCP region as shown in the following configuration example.



The Terraform Module

sensor-deploy.tf

```
module "threatx_sensor" {
  source              = "../"
  customer_name       = "<customer_name>"
  customer_sensor_key = "<customer_sensor_key>"
  deployment_name     = "<deployment_name>" # Unique name for this
deployment (prod, test, etc.)
  waap_version        = "3.20.0"            # WAAP version to deploy.
Default: Latest
  region              = "us-west1"
  jump_host_zone      = "us-west1-a"
  sensor_zones        = ["us-west1-a", "us-west1-b"] # ["zone1","zone2"]
```

```
  deployment_cidr     = "10.128.0.0/28" # CIDR block for subnet
  machine_type        = "e2-medium"     # Default: e2-medium
  target_size         = 2               # Default: 2
  min_replicas        = 2               # Default: 2
  max_replicas        = 10              # Default: 10
  custom_sensor_tags  = ""              # String with comma separation
per tag ("tag1,tag2,tag3")
}
```

Variables

Table 1. Required Module Variables

Parameter	Description
customer_name	ThreatX customer name. Provided by the ThreatX SOC.
customer_sensor_key	ThreatX sensor key. Provided by the ThreatX SOC.
deployment_name	A name for the deployment. It is appended to resource names.
region	Region for the deployment.
sensor_zones	Zones for sensor deployment. At least two should be defined for redundancy.
jump_host_zone	Zone for jump host VM deployment.
deployment_cidr	CIDR block defining subnet created for this deployment. Ensure that the CIDR block is large enough to accommodate max_replicas.

Table 2. Optional Module Variables

Parameter	Description
waap_version	Version of ThreatX WAAP to deploy. Default is latest. Specific versions are not currently supported.
machine_type	Machine type or size for sensors. Default is e2-standard-16.
target_size	Target number of sensor nodes for the autoscaling group. Default is 2.
min_replicas	Minimum number of sensor nodes. Default is 2.
max_replicas	Maximum number of sensor nodes. Default is 10.
custom_sensor_tags	Variable for customer sensor tag customization. Add as comma-separated string, such as "tag1,tag2,tag3".

Outupts

Table 3. Module Outputs

Name	Description
load_balancer_ip	External IP address of the load balancer.
jump_host_ip	External IP address of the jump host.
network_id	Resource ID of the compute network.

Google Cloud Deployment Guide

The ThreatX WAF Sensor can be deployed behind a Google (GCP) Network Load Balancer for high availability. To facilitate HA deployment in GCP, ThreatX provides a .yml deployment template. The template may be used 'as is' or modified to help deployment into your particular GCP environment.

Introduction

ThreatX WAF Sensors can be efficiently deployed in GCP environments. The ThreatX WAF Sensor can be implemented as a single instance or in a multiple-instance High Available (HA) configuration.

The ThreatX WAF Sensor image can be deployed behind a Google (GCP) Network Load Balancer for high availability. Use of a Network Load Balancer allows for architectures in which the client's TLS (formerly SSL) session terminates at the ThreatX WAF. With the client's session information being available to the ThreatX WAF Sensor, this architecture allows TLS client fingerprinting to occur.

A scaled ThreatX image Deployment within GCP creates a "Load Balancer sandwich" consisting of the ingress GCP Network Load Balancer, the ThreatX WAF Sensors deployed within autoscaling target groups, and the egress (to backend) load balancer or gateway depending on your specific origin architecture.

To facilitate HA deployment in GCP, ThreatX provides a .yml deployment template (and associated python files). This template is utilized through the use of Google's command line SDK (gcloud). The template framework implements deployment to a greenfield VPC. The template may be used "as is" or modified to help deployment into your particular GCP environment.

The sensor-deploy.yml file, and other supporting files, can be provided by the ThreatX Security Operations Center (SOC) upon request.

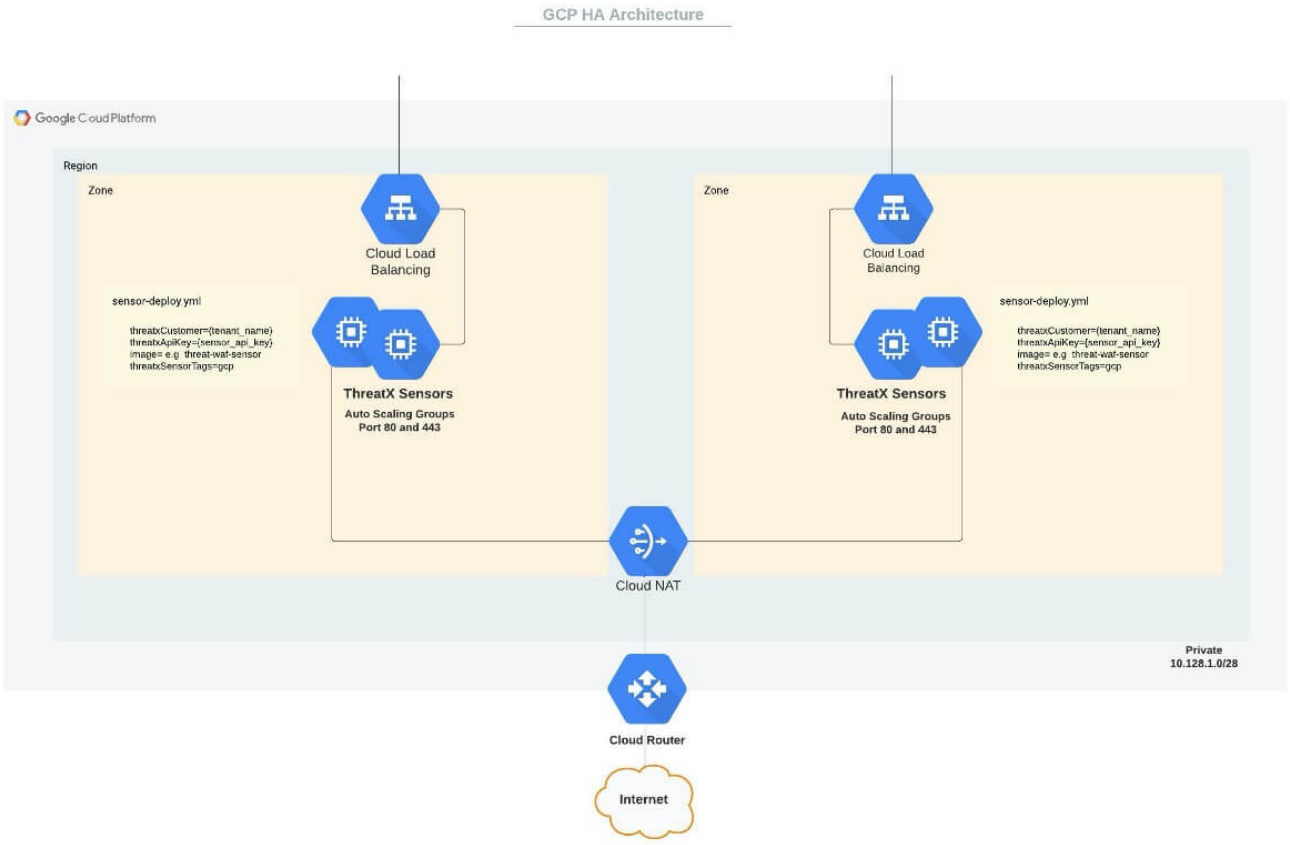


Figure 1. GCP Deployment Model

Because origin server architecture may vary significantly and because the ThreatX deployment template does not deploy these backend systems, they are not reflected in *Figure 1*.

The deployment template parameters are customizable including the GCP network zones and ThreatX target group parameters; any specific network references in *Figure 1* are simply for illustration.

There are several architectural components deployed with default template parameters. These resources are described below:

Network Load Balancer

The two Google Network Load Balancers will provide ingress to defined target groups of ThreatX WAF Sensors. A NLB will be available in each respective zone and utilize (public internet-facing) permanent IPs (not explicitly shown in *Figure 1*) created by the deployment stack.

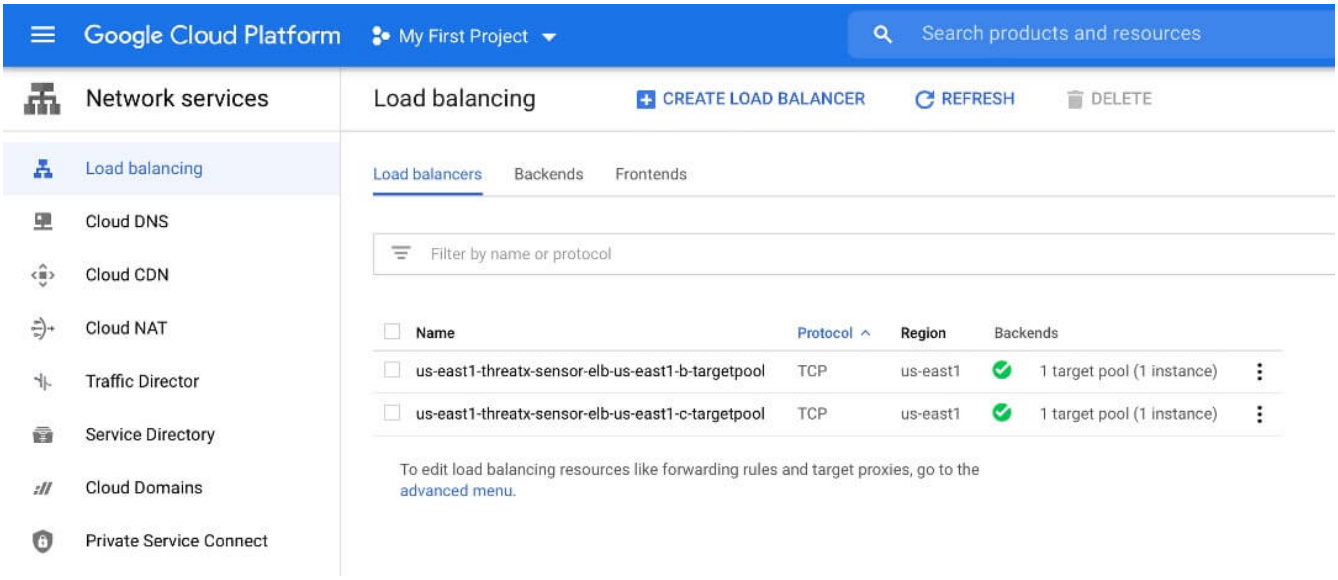


Figure 2. The two NLBs with TCP listeners

The NLB can preserve the client IP and allow the client TLS (aka SSL) to terminate at the ThreatX WAF Sensor instead of at the load balancer. As noted in the introduction, an NLB acting as a TCP load balancer allows the ThreatX WAF Sensors to utilize IP interrogation and TLS fingerprinting techniques fully.

Target Pools

The ThreatX WAF Sensors will be deployed in two target pools corresponding to the NLB zones. These two target pools are distributed across zones for fault tolerance. The target pool mapping to the instance group manager and instances is shown below:

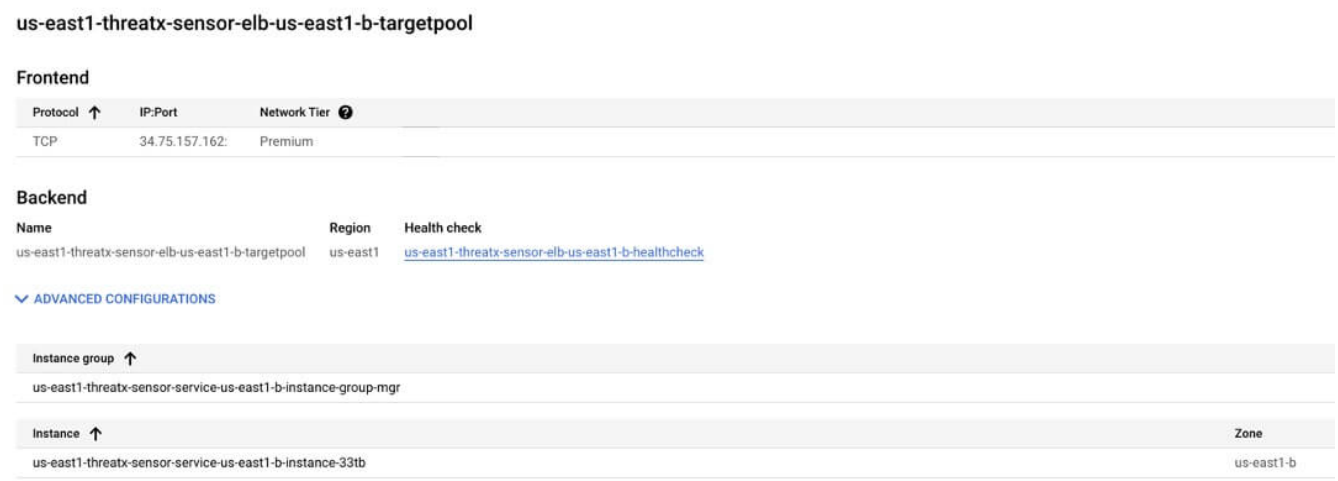


Figure 3. The US-East-1b zone:_

Please note that, as shown in *Figure 3*, a healthcheck is also configured, in each zone, by the deployment script. Instance groups as well as instance templates will be setup in each zone. This is illustrated below.

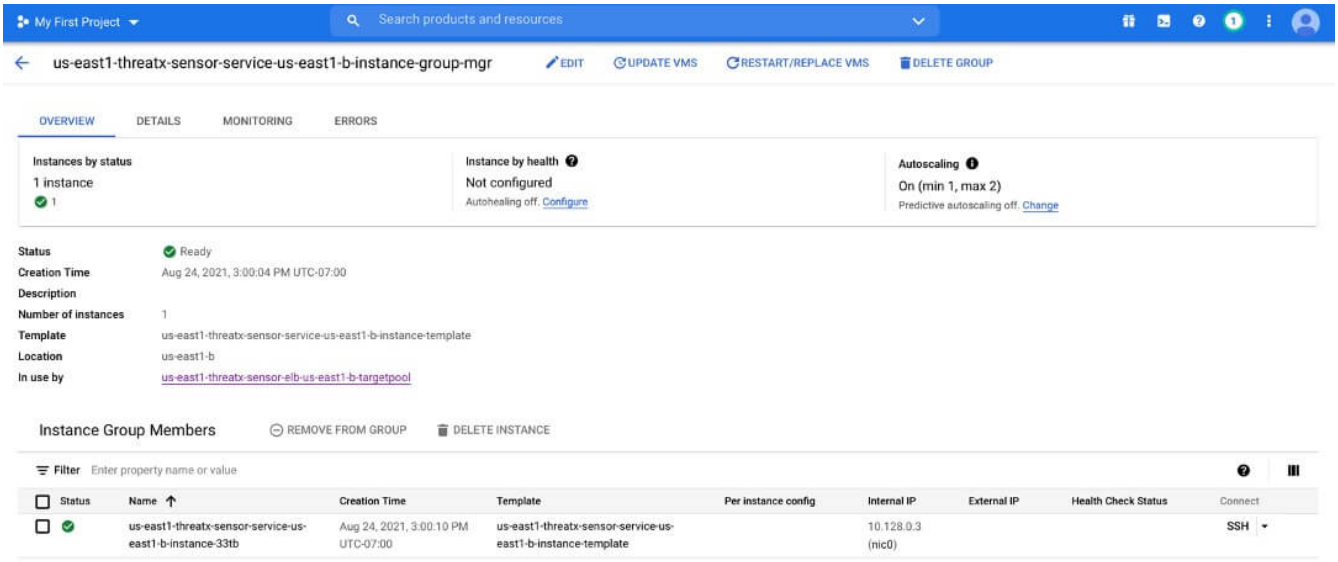


Figure 4. Instance groups as well as instance templates for a single zone._

The defined instance template (seen in *Figure 4*) utilizes a default VM instance type of f1-micro. **It is recommended this be changed (in sensor-deploy.yml) to e2-medium for most production deployments.**

An autoscaler group in each zone is also created. The default pool has a minimum of 1 Sensor deployed (and a maximum number of 2 replicas). The number of replicas may be changed, again, via modification of the sensor-deploy.yml file.

Forwarding Rules

Traffic forwarding rules for each NLB are also provisioned.

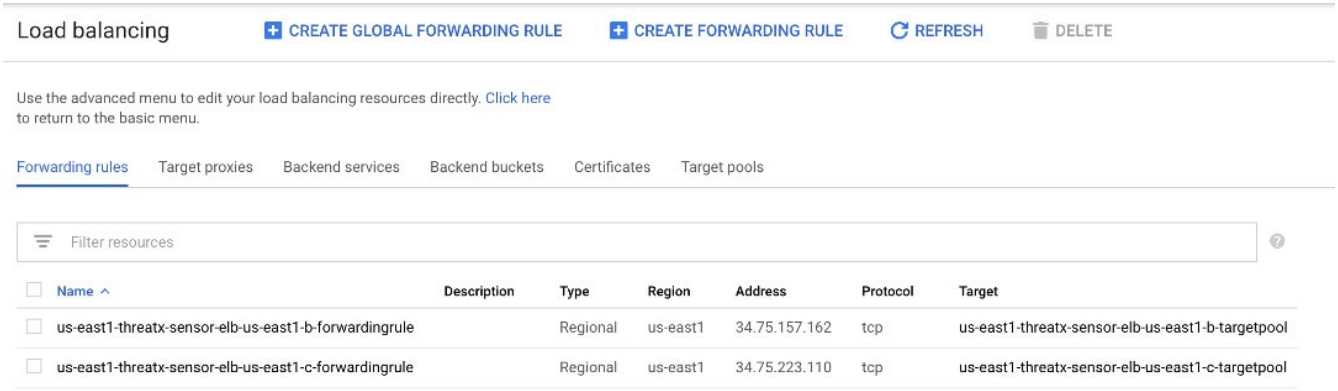


*Figure 5. Traffic forwarding rules_*

ThreatX WAF Sensors

The ThreatX WAF Sensor is available as a GCP image. The automation template will deploy VM compute instances in target pools.

Each ThreatX WAF Sensor must have Internet connectivity to the ThreatX cloud to the pull site, certificate, backend (i.e., origin), routing configuration, and security rules. Once the configuration is obtained, the ThreatX WAF Sensor will inspect and block (or tarpit, or interrogate) traffic, using configuration parameters pulled from the ThreatX cloud. If connectivity to the ThreatX cloud is lost, the ThreatX WAF Sensor will continue to operate using its most recent configuration. If connectivity to the ThreatX cloud is lost, event log messages will be locally cached until connectivity is restored.

The private IP space and NAT Gateway is utilized to enhance Sensor security.

In order to establish connectivity to the ThreatX cloud , both the **tenant name** and a **Sensor API key** must be obtained via the ThreatX management interface.

In *Figure 1*, this mandatory instance configuration information is shown in the “sensor-deploy.yml” callout. In addition to the ThreatX WAF Sensors, a bastion / jump host is deployed, as noted in *Figure 6*.

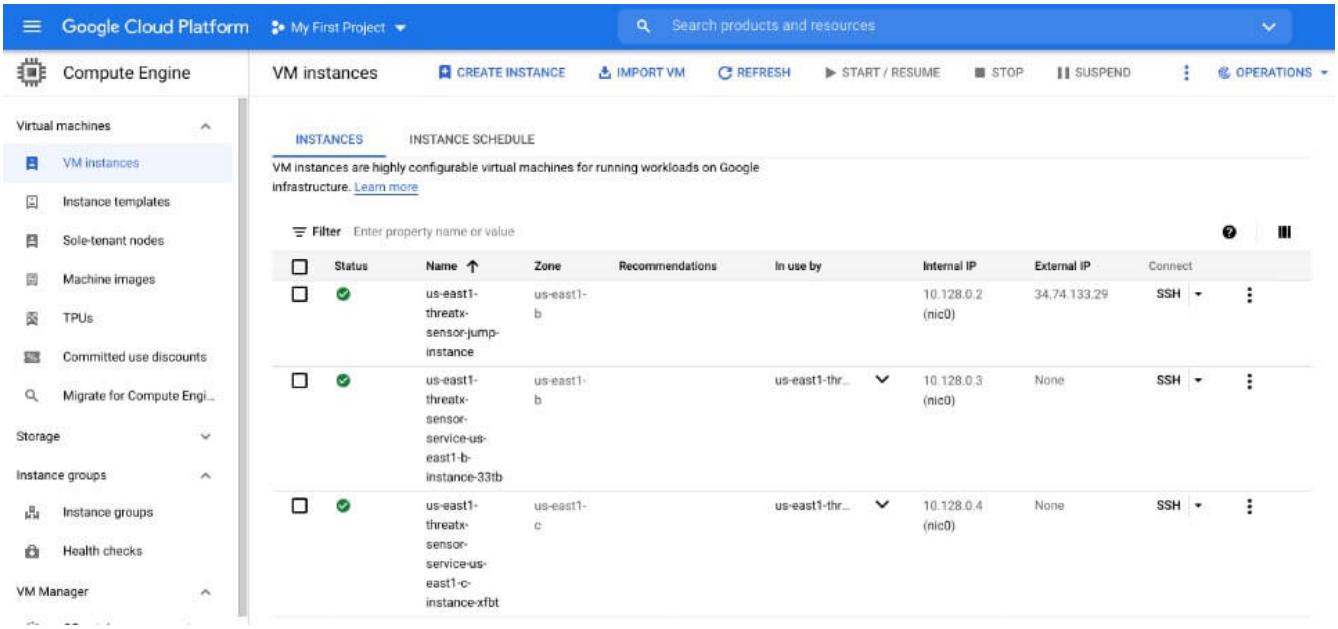


*Figure 6. Deployed bastion / jump host_*

Network Infrastructure

The deployment stack creates multiple different networking components necessary to support the scaled ThreatX deployment. These components are described in more detail below.

Network and Subnetwork

The deployment creates a network and a small subnetwork by default.

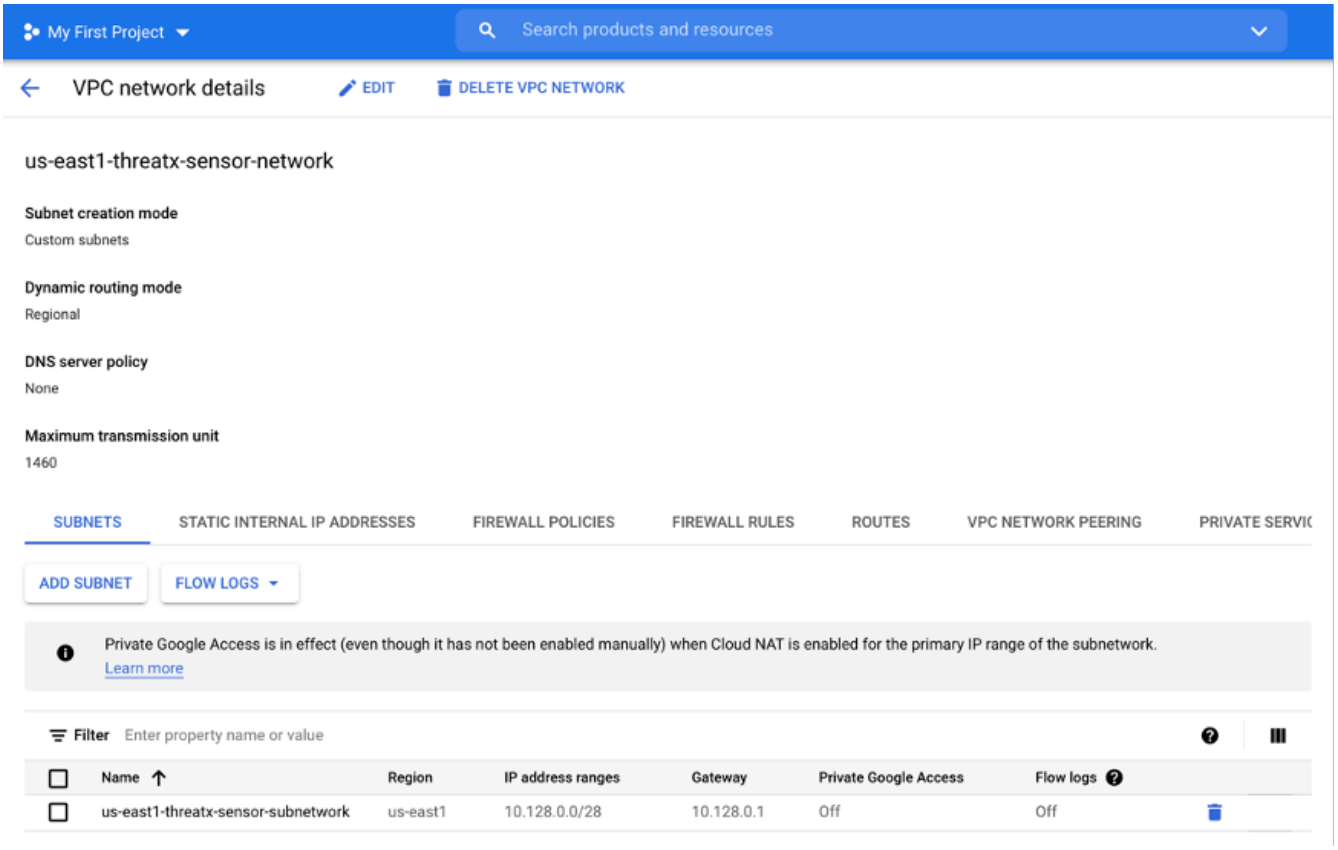


*Figure 7. Default network and subnetwork_*

Gateways and Subnets

As noted above, the ThreatX WAF Sensors must communicate with the ThreatX cloud to obtain configurations. To provide a secure architecture, the private VPC subnets which house the ThreatX WAF Sensors communicate via NAT Gateway to ensure “one-way” communication. The Cloud NAT Gateway is a regional construct.

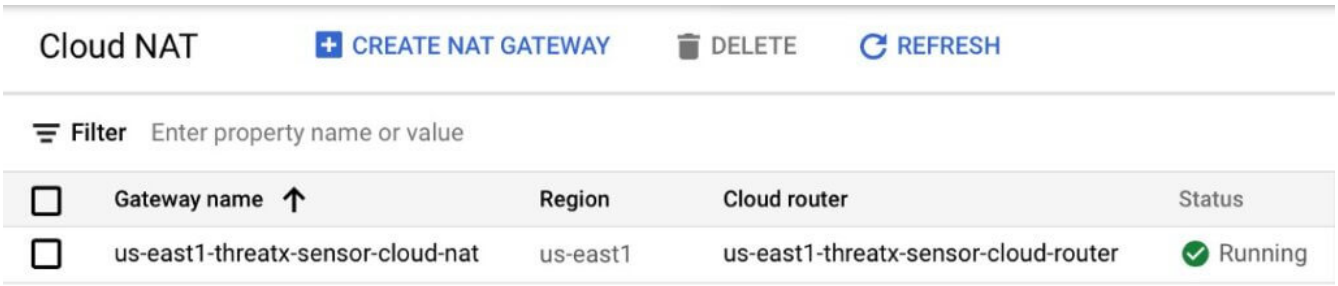


*Figure 8. NAT Gateway_*

Cloud Router

A Cloud Router is also configured and associated to the Cloud NAT Gateway described above.

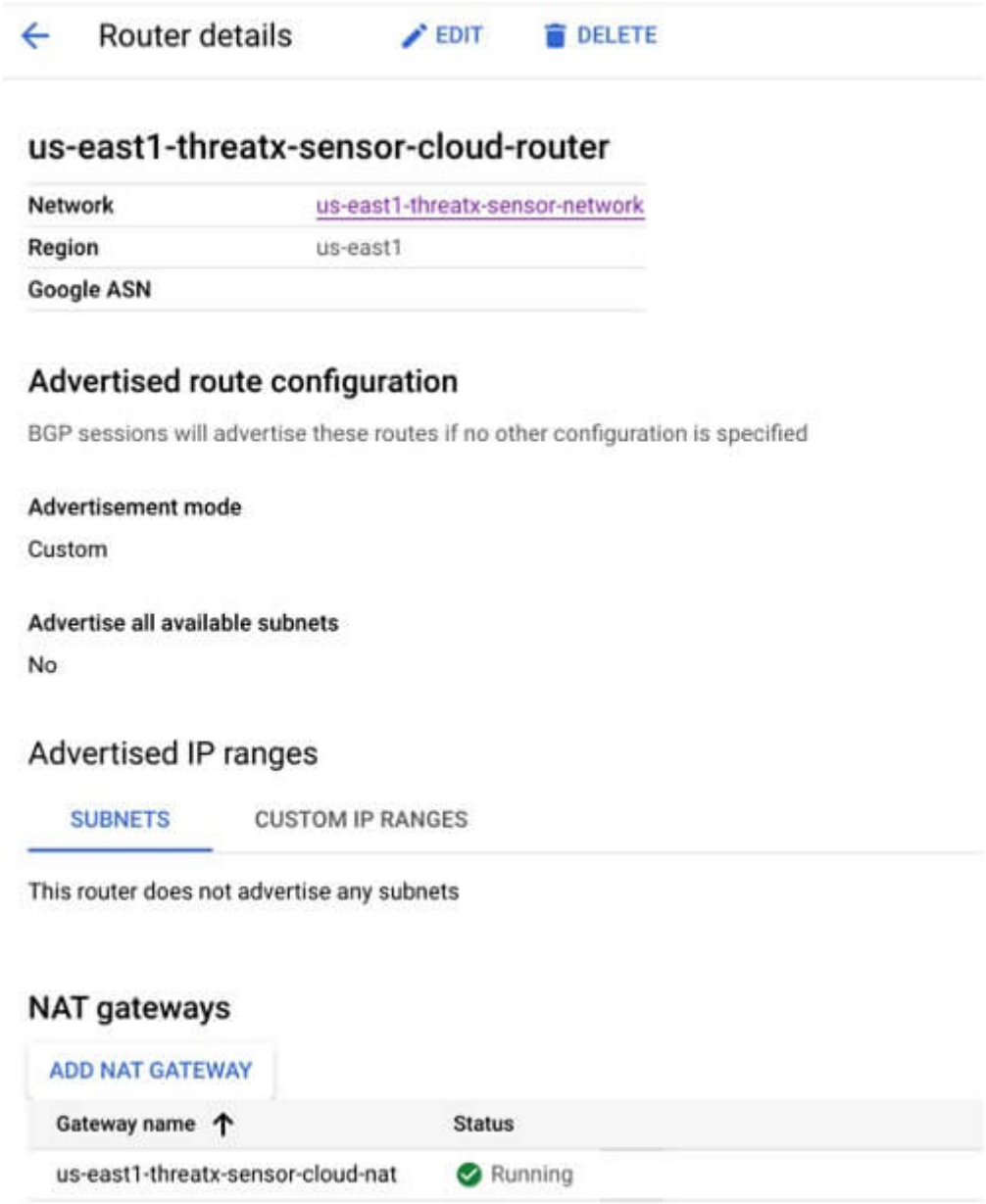


*Figure 9. Cloud Gateway_*

Firewall Rules

Rules will also be created by default to facilitate http(s) and allow for SSH management traffic to the target Sensors. Though SSH traffic rules are provisioned, during normal ThreatX WAF Sensor operation SSH access to the Sensors is not required. This access may be disabled / restricted as desired. The HTTP(S) and internal traffic provisioned firewall rules are shown below.

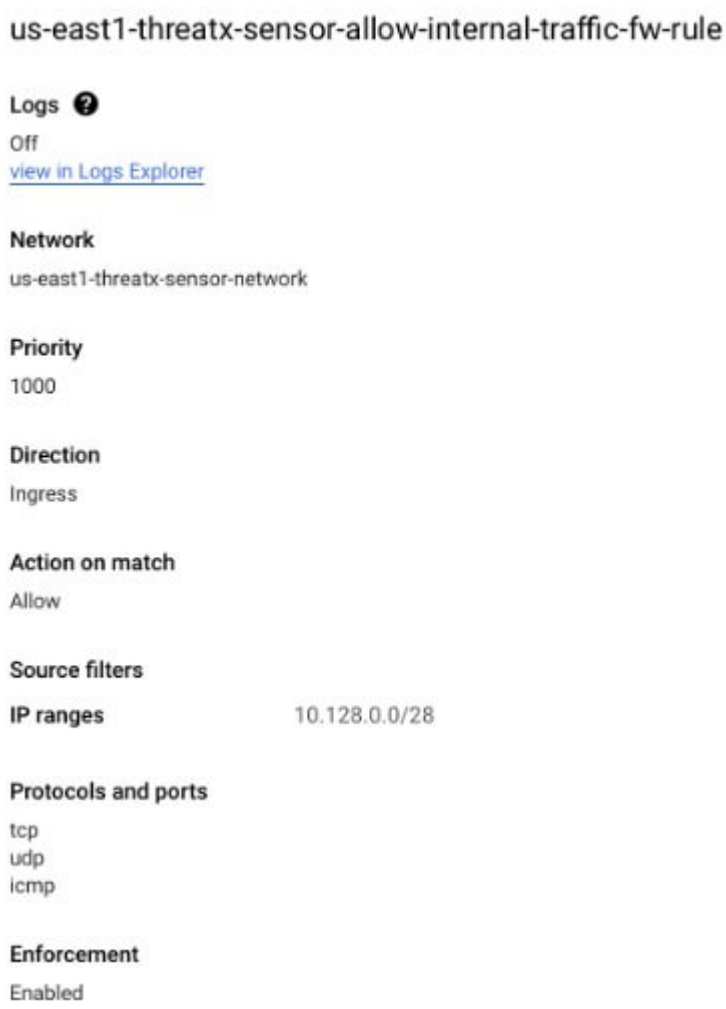


*Figure 10. Figure 10: Firewall Rules_*

Practical Usage

Overview

In utilizing the deployment template, the stack creation process is relatively straightforward. ThreatX, upon request, can supply a package containing .yml file and several .py files as well as a README.md file. The README.md covers many of the deployment process execution steps shown below.

Obtaining the ThreatX Sensor image

As documented, in the README.md, utilize the GCP SDK to create a ThreatX image in your Google project as shown below.

```
gcloud compute --project=<user_project> images create threatx-waf-sensor-latest --source-image=threatx-waf-sensor-latest --source-image-project=cogent-tangent-233801
```

Here `<user_project>` is the Google project ID and will be a string without spaces (e.g. national-portal-513821)

Once successful, the ThreatX image should be present in your project as shown below.

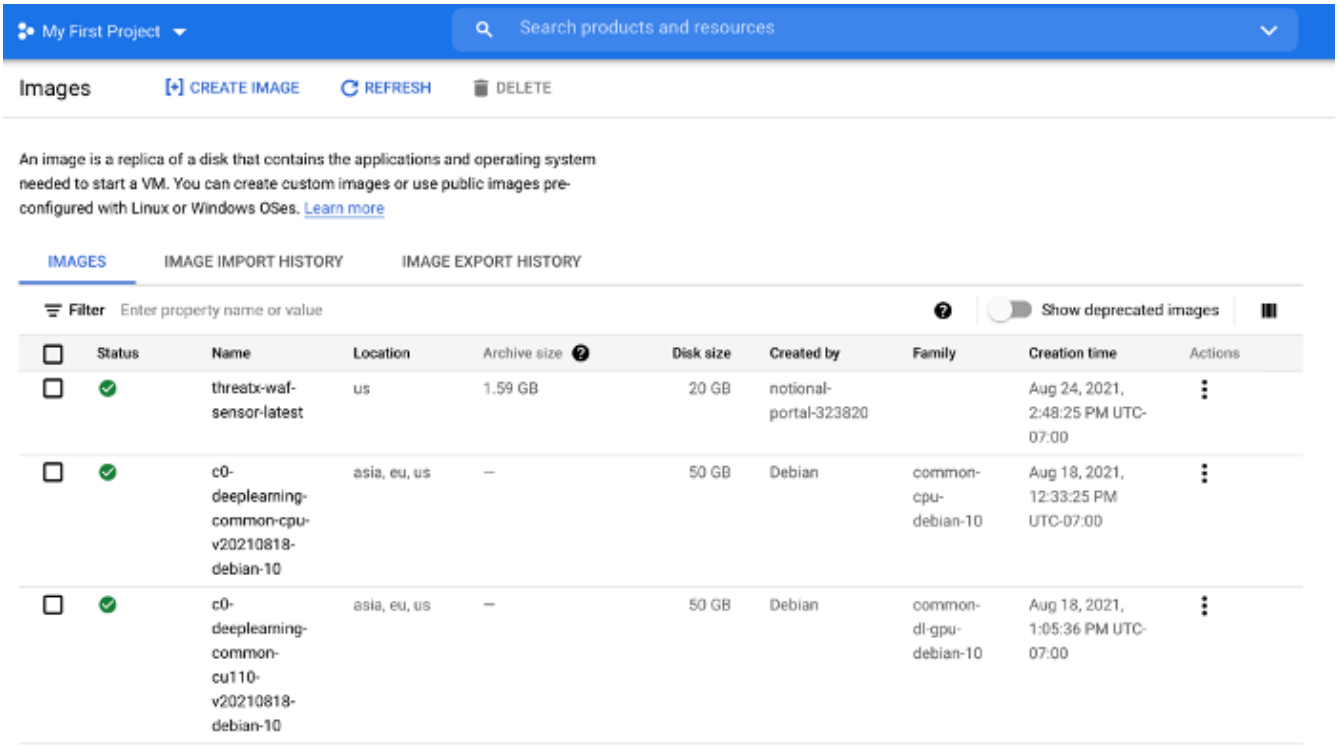


*Figure 11. ThreatX image_*

Modifying the sensor-deploy.yml file

A sample deployment template, sensor-deploy.yml is shown below.

```
imports:
- path: sensor-setup.py

resources:
- name: frontend
  type: sensor-setup.py
  properties:
    threatxCustomer: lab
    threatxApiKey: 2b0d74dadaf897f2514b72cc3b4ac4c8f7cc47cb17cf6c4c0f2cefc25ec2f91c
    threatxSensorTags: gcp-lb,gcp,threatx-gcp
    image: threatx-waf-sensor-latest
    targetSize: 1
    maxReplicas: 2
    utilizationTarget: 0.8
    machineType: f1-micro
    zones:
    - us-east1-b
    - us-east1-c
```

```
imports:
- path: sensor-setup.py

resources:
- name: frontend
  type: sensor-setup.py
  properties:
    threatxCustomer: lab
    threatxApiKey:
    threatxSensorTags: gcp-lb,gcp,threatx-gcp
    image: threatx-waf-sensor-latest
    targetSize: 1
    maxReplicas: 2
    utilizationTarget: 0.8
    machineType: f1-micro
    zones:
    - us-east1-b
    - us-east1-c
```

The parameter inputs are each briefly explained below each parameter. Where possible, the parameters are supplied with default values, which may be adjusted to meet your implementation requirements.

A few parameters may benefit from additional clarification:

threatxCustomer

This is your tenant name and is found in the upper right corner of the interface after logging into Threatx. In the .yml file above the value of the parameter is “lab”.

threatxApiKey

This also may be generated via the ThreatX interface, as shown below. It is a one-time generation of the key, so make sure to capture the value.

threatxSensorTags

Optional standard GCP tags.

image

The local project name of the ThreatX image garnered previously in “Obtaining the ThreatX Sensor image”.

Creating the infrastructure

Once the image has been obtained and the sensor-deploy.yml file has been appropriately modified , the ThreatX infrastructure can be created utilizing the Google SDK with the command in *Figure 14* below.

```
gcloud deployment-manager deployments create threatx-sensor --config=sensor-deploy.yml
```

AWS AMI Deployment Guide

Introduction

The ThreatX WAF Sensor AMI can be used to quickly and easily add application security to applications deployed in AWS VPCs. The AMI can be found by launching an instance and searching for “**ThreatX WAF**” when choosing an AMI.

This AMI will…

- Keeps the ThreatX container image up to when new ec2 instances are launched from the AMI
- Manages the life cycle of containerized WAF instances
- Configured with User-Data

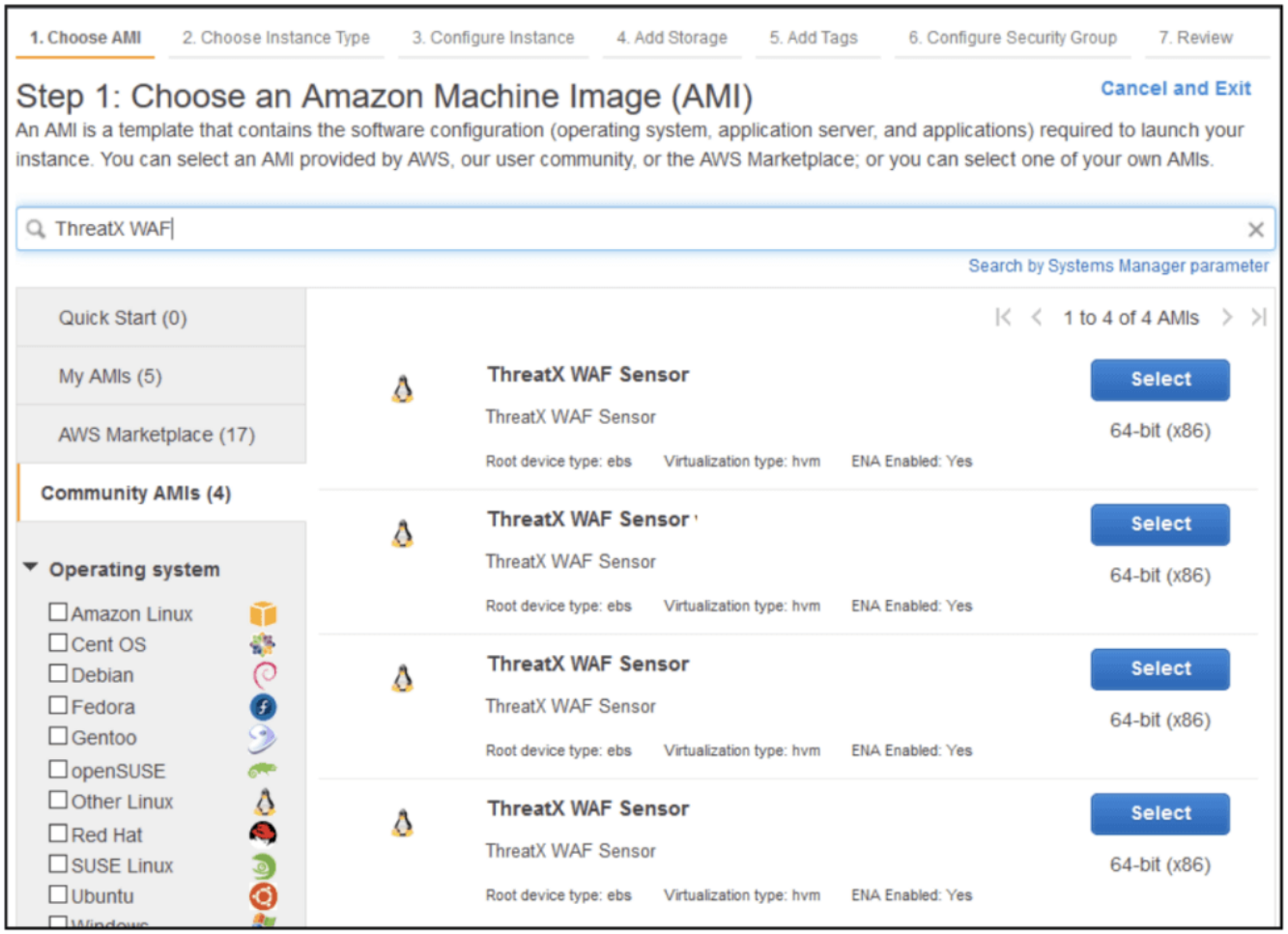


Figure 12. Selecting the ThreatX AMI in the AWS Marketplace

Installation

Minimum Requirements

CPU	2 cores
RAM	1 GB
Disk	20 GB

An instance type of **t3.micro or larger** is recommended.

Configuration

In the simplest deployment, the AMI can be launched with the following User-Data information:

cloud-config

```
#cloud-config
write_files:
    - path: /etc/txconf
      content: |
         CUSTOMER=<customer_name>
         API_KEY=<customer_sensor_key>
         RESOLVER=local
         SENSOR_TAGS=tag1,tag2
```

`SENSOR_TAGS` accepts a comma-separated list of strings

Troubleshooting

Login to the ec2 instance

Login as core user

```
$ ssh -i sshkey.pem core@<instance_url>
```

See the AMI version

```
$ echo $TXWAF_AMI_VERSION
```

Check Logs

Check for problems in the `txwaf` service

```
$ journalctl -u txwaf
```

Check for problems in the docker container

```
$ docker logs txwaf
```

Check for problems in the kernel

```
$ dmesg
```

Enter the `txwaf` container

Get a shell into the ThreatX WAF container

```
$ docker exec -it txwaf bash
```