Deploy the Firewall Threat Defense Virtual Auto Scale Solution on OCI

Deploy Auto Scale Solution

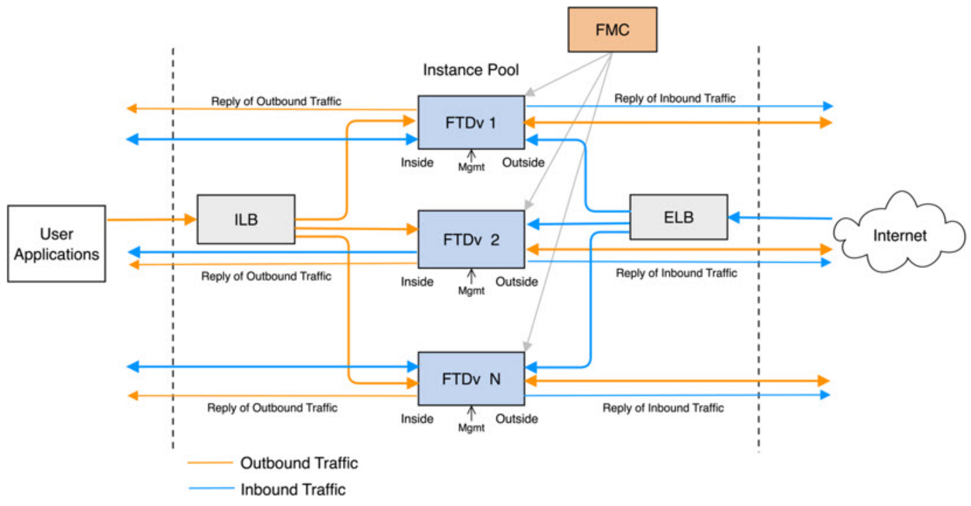
The following sections describe how the components of the Auto Scale solution work for the Firewall Threat Defense Virtual on OCI.

Auto Scale Use Case

The use case for the Firewall Threat Defense Virtual Auto Scale on OCI solution is shown in the following figure. Internet-facing Load Balancer will have a public IP address with ports enabled using Listener and Target Group combination.

Port-based bifurcation is possible for traffic, and it can be achieved via NAT rules. This is explained in the following sections.

Figure 1: Secure Firewall Threat Defense Virtual Auto Scale Use Case Diagram



Scope

This document covers the detailed procedures to deploy the Firewall Threat Defense Virtual Auto Scale for OCI solution.

Important

- Read the entire document before you begin your deployment.
- Make sure the prerequisites are met before you start deployment.
- Make sure you follow the steps and order of execution as described herein.

Prerequisites

Permission and Policies

Following are the OCI permissions and policies that you require to implement the solution:

1. **Users and Group**

Note You must be an OCI User or a Tenancy Administrator to create the Users and Groups.

Create Oracle Cloud Infrastructure user accounts and a group to which the user accounts belong. If the relevant group with user accounts exist, you need not create them. For instructions on creating users and groups, see Creating Groups and Users.

2. **Group Policies**

You need to create the policies and then map them to the group. To create the policies, go to **OCI** > **Identity & Security** > **Policies** > **Create Policy**. Create and add the following policies to the desired group:

- Allow group <*Group_Name*> to use metrics in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to manage alarms in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to manage ons-topics in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to inspect metrics in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to read metrics in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to use tag-namespaces in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to read log-groups in compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to use instance-pools compartment <*Compartment_Name*>
- Allow group <*Group_Name*> to use cloud-shell in tenancy
- Allow group <*Group_Name*> to read objectstorage-namespace in tenancy
- Allow group <*Group_Name*> to manage repos in tenancy

Note You can create policies at tenancy level as well. It is at your discretion how you want to provide all the permissions.

3. **Permission for Oracle Functions**

   To enable a Oracle-Function to access another Oracle Cloud Infrastructure resource, include the function in a dynamic group, and then create a policy to grant the dynamic group access to that resource.

4. **Create Dynamic Group**

   To create dynamic groups, go to **OCI** > **Identity & Security** > **Dynamic Group** > **Create Dynamic Group**

   Specify the following rule while creating the dynamic group:

   ```
   ALL {resource.type = 'fnfunc', resource.compartment.id = '<Your_Compartment_OCID>'}
   ```

   For more details on dynamic groups, see:

   - https://docs.oracle.com/en-us/iaas/Content/Functions/Tasks/functionsaccessingociresources.htm
   - https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingdynamicgroups.htm

5. **Create Policy for Dynamic Group**

   To add policy, go to **OCI** > **Identity & Security** > **Policies** > **Create Policy**. Add the following policy to the group:

   ```
   Allow dynamic-group <Dynamic_Group_Name> to manage all-resources in compartment
   <Compartment_OCID>
   ```

Download files from GitHub

FTDv – OCI Autoscale solution is delivered as a GitHub repository. You can pull or download the files from the repository.

Python3 Environment

A *make.py* file can be found in the cloned repository. This program compresses the oracle functions and template files into a Zip file; copy them to a target folder. In order to do these tasks, the Python 3 environment should be configured.

Note This python script can be used only on Linux environment.

Infrastructure Configuration

The following must be configured:

1. **VCN**

   Create VCN as required for your FTDv application. Create VCN with the Internet Gateway having at least one of the subnet attached with route to internet.

   For information on creating VCN, see https://docs.oracle.com/en-us/iaas/Content/GSG/Tasks/creatingnetwork.htm.

2. **Application Subnets**

   Create subnets as required for your FTDv application. To implement the solution as per this use case, FTDv instance requires 4 subnets for its operation.

   For information on creating subnet, see https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/managingVCNs_topic-Overview_of_VCNs_and_Subnets.htm#.

3. **Outside Subnet**

   Subnet should have route with '0.0.0.0/0' to Internet Gateway. This subnet contains the Outside interface of Cisco FTDv and the Internet-facing Load balancer. Ensure that the NAT Gateway is added for outbound traffic.

   For more information, see the following documents:

   - https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/managingIGs.htm
   - https://docs.oracle.com/en-us/iaas/Content/Network/Tasks/NATgateway.htm#To_create_a_NAT_gateway

4. **Inside Subnet**

   This is similar to the Application Subnets, with or without NAT/Internet gateway.

Note For FTDv health probes, you can reach the metadata server (169.254.169.254) through Port 80.

5. **Management Subnet**

Management subnet should be public so that it supports SSH accessibility to the FTDv.

6. **Function Subnet**

   This subnet is for Oracle Functions deployment.

Note This subnet must have `0.0.0.0/0` route to NAT GW (not Internet GW).

The Public IP of NAT GW of this subnet must be allowed in NSG (Network Security Group) of the Firewall Management Center Virtual and Firewall Threat Defense Virtual.

7. **Security Groups- Network Security Group for FTDv Instance**

   Configure the security group for FTDv instances that allows the Oracle Functions(in same VCN) perform SSH connections to FTDv's management address.

8. **Object Storage Namespace**

   This object storage namespace is used for hosting static website, having configuration.txt file. You must create a pre-authenticated requests for the configuration.txt file. This pre-authenticated URL is used during the template deployment.

Note Ensure that the following configurations that are uploaded are accessible by the FTDv instances through HTTP URL.

When FTDv is booted, it executes the following command`$ copy /noconfirm <configuration.txt file’s pre-authenticated request URL > disk0:Connfiguration.txt`

This command enables FTDv launch to be configured with configuration.txt file.

Secure Firewall Management Center Prerequisites

You can manage the Firewall Threat Defense Virtual devices using the Secure Firewall Management Center, a full-featured, multidevice manager. The Firewall Threat Defense Virtual registers and communicates with the FMC on the Management interface that you allocated to the Firewall Threat Defense Virtual virtual machine.

Create the objects required for Firewall Threat Defense Virtual configuration and management, including a device group to deploy policies and install updates on multiple devices. All the configurations applied on the device group is pushed to the Firewall Threat Defense Virtual instances.

The following sections provide a brief overview of basic steps to prepare the Firewall Management Center. For complete information on the procedure, refer *Secure Firewall Management Center Configuration Guide*. When you prepare the Firewall Management Center, make sure you record the following information:

- Secure Firewall Management Center Public IP Address
- Username and Password (If memory based scaling is enabled, you have to provide 2 user credentials)
- Security Zone Names
- Secure Firewall Management Center Access policy Name

- Secure Firewall Management Center NAT Policy Name
- Device Group Name

Create User in Secure Firewall Management Center

Create a new user in Secure Firewall Management Center with Admin privileges to be used only by Autoscale Manager.

Note You must have an Secure Firewall Management Center user account dedicated to the Firewall Threat Defense Virtual Autoscale solution to prevent conflicts with other FMC sessions.

Procedure

Create new user in Secure Firewall Management Center with Admin privileges. Choose **System** > **Users** and click **Create User**. The username must be Linux-valid:

- Maximum 32 alphanumeric characters, plus hyphen (-) and underscore (_)
- All lowercase
- Should not start with hyphen (-); must have alphabets; should not include a period (.), at sign (@), or slash (/)

Complete user options as required for your environment. See the *Secure Firewall Management Center Configuration Guide* for complete information.

Create Device Group

Device groups enable you to easily assign policies and install updates on multiple devices. A device group should be created and rules should be applied on it. All the configurations applied on the device group are pushed to Firewall Threat Defense Virtual instances.

Procedure

Step 1 Choose **Devices** > **Device Management**.
Step 2 From the **Add** drop-down menu select **Add Group**.
Step 3 Enter the Device group name.
Step 4 Click **Ok** to create the device group.

Create Network and Host Objects

Create the following objects to be used for Firewall Threat Defense Virtual configuration.

Procedure

Step 1 Create Host with its name as *oci-metadata-server* and its IP as *169.254.169.254*.

Step 2 Create a port with its name as health-check-port and its value as 8080 or any other port as required.

Step 3 Create Inside interface, choose **Interface** > **Security Zone**. Select type as **Routed**. Provide a name for the interface, example, *inside-sz*.

Step 4 Create Outside interface, choose **Interface** > **Security Zone**. Select type as **Routed**. Provide a name for the interface, example, *outside-sz*.

Create NAT Policy

Create a NAT policy and create the necessary NAT rules to forward traffic from the outside interface to your application, and attach this policy to the device group you created for Autoscale.

Procedure

Step 1 Choose **Devices** > **NAT**

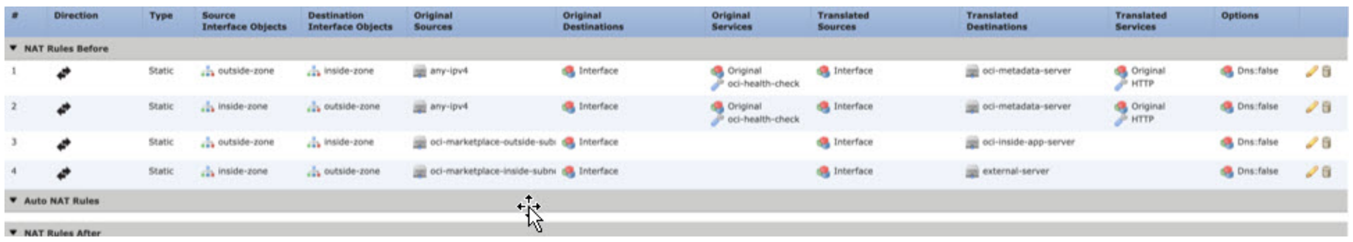
Step 2 From the **New Policy** drop-down list, choose **Threat Defense NAT**.

Step 3 Enter a unique **Name**.

Step 4 Optionally, enter a **Description**.

Step 5 Configure NAT rules. Refer Configure NAT for Threat Defense in the Secure Firewall Management Center Device Configuration Guide for guidelines on how to create NAT rules and apply NAT policies. The following figure shows a basic approach in setting the rules.

Figure 2: NAT Rules



#	Direction	Type	Source Interface Objects	Destination Interface Objects	Original Sources	Original Destinations	Original Services	Translated Sources	Translated Destinations	Translated Services	Options	
▼ NAT Rules Before												
1	⇄	Static	outside-zone	inside-zone	any-ipv4	Interface	Original oci-health-check	Interface	oci-metadata-server	Original HTTP	Dns:false	✎ 🗑
2	⇄	Static	inside-zone	outside-zone	any-ipv4	Interface	Original oci-health-check	Interface	oci-metadata-server	Original HTTP	Dns:false	✎ 🗑
3	⇄	Static	outside-zone	inside-zone	oci-marketplace-outside-sub	Interface		Interface	oci-inside-app-server		Dns:false	✎ 🗑
4	⇄	Static	inside-zone	outside-zone	oci-marketplace-inside-subn	Interface		Interface	external-server		Dns:false	✎ 🗑
▼ Auto NAT Rules												
▼ NAT Rules After												

Step 6 Click **Save**.

Create NAT Rules

A typical NAT rule converts internal addresses to a port on the outside interface IP address. This type of NAT rule is called interface Port Address Translation (PAT). See Configure NAT for Threat Defense in the Secure Firewall Management Center Device Configuration Guide for more information.

Configure the following 2 mandatory rules that are required in your NAT policy:

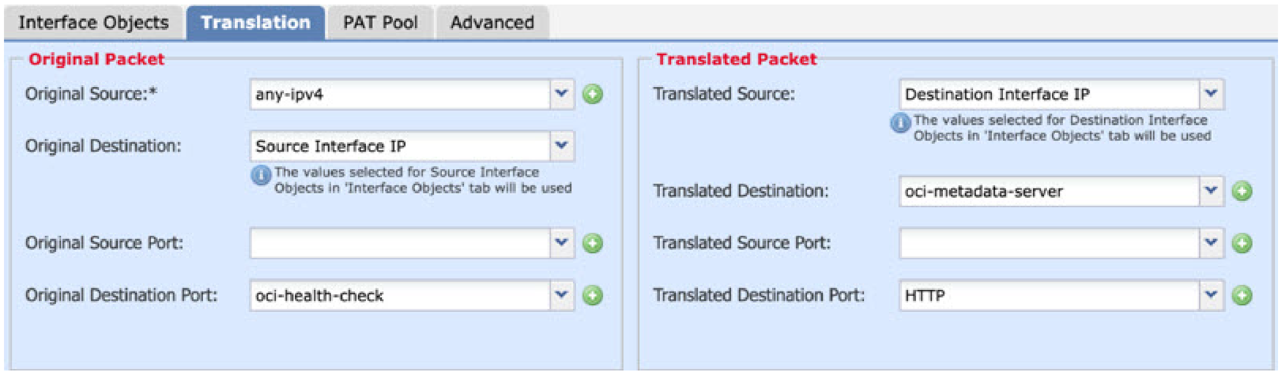
Procedure

Step 1 Configure the following NAT Rule for Inbound health check:

- Source Zone : Outside Zone
- Destination Zone : Inside Zone
- Original-sources : any-ipv4
- Original Destinations: Source Interface IP
- Original source port: Default
- Original-destination-port: health-check-port
- Translated-sources: Destination Interface IP
- Translated-destination: oci-metadata-server
- Translated source port: default
- Translated-destination-port: HTTP

The following figure shows the NAT rule for inbound health check.

Figure 3: Inbound health NAT rule



Original Packet

Original Source:* any-ipv4

Original Destination: Source Interface IP

The values selected for Source Interface Objects in 'Interface Objects' tab will be used

Original Source Port:

Original Destination Port: oci-health-check

Translated Packet

Translated Source: Destination Interface IP

The values selected for Destination Interface Objects in 'Interface Objects' tab will be used

Translated Destination: oci-metadata-server

Translated Source Port:

Translated Destination Port: HTTP

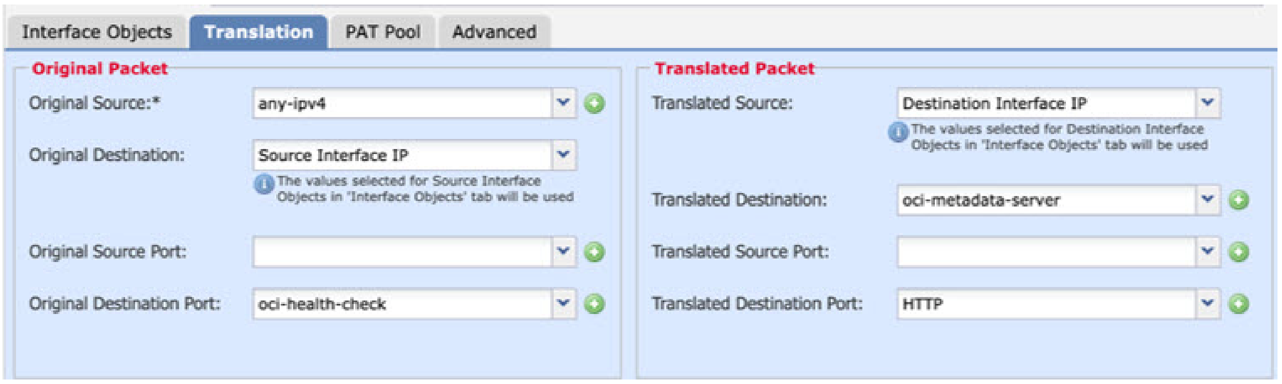
Step 2 Configure the following NAT rule for outbound health check.

- Source Zone : Inside Zone
- Destination Zone : Outside Zone
- Original-sources : any-ipv4
- Original Destinations: Source Interface IP
- Original source port: Default
- Original-destination-port: health-check-port
- Translated-sources: Destination Interface IP

- Translated-destination: oci-metadata-server
- Translated source port: default
- Translated-destination-port: HTTP

The following figure shows the NAT rule for outbound health check.

Figure 4: Outbound health check NAT rule



Original Packet

Original Source:*	any-ipv4
Original Destination:	Source Interface IP
	The values selected for Source Interface Objects in 'Interface Objects' tab will be used
Original Source Port:	
Original Destination Port:	oci-health-check

Translated Packet

Translated Source:	Destination Interface IP
	The values selected for Destination Interface Objects in 'Interface Objects' tab will be used
Translated Destination:	oci-metadata-server
Translated Source Port:	
Translated Destination Port:	HTTP

Similarly, any NAT rules can be added for data traffic, and this configuration pushes them to Firewall Threat Defense Virtual devices.

Create an Access Policy

Configure access control to allow traffic from inside to outside. An Access Policy with all required policies can be created, health port object should be allowed such that traffic on this port is allowed to reach the device. Within an access control policy, access control rules provide a granular method of handling network traffic across multiple managed devices. Proper configuration and sequencing of the rules are essential to build an effective deployment. See the Best Practices for Access Control Rules in the Secure Firewall Management Center Device Configuration Guide.

Assign the device group (created as part of pre-requisites) to the access policy using **Policy Assignments**.

Procedure

Step 1 Choose **Policies** > **Access Control**.

Step 2 Click **New Policy**.

Step 3 Enter a unique Name and, optionally, a Description.

Step 4 Configure security settings and rules for your deployment. For more information, see Access Control in the Secure Firewall Management Center Device Configuration Guide.

Encrypt Password

Note For more information on this procedure, see Create Vaults and Secrets.

Password for FTDv is used to configure all the FTDv instances being used while autoscaling and it is used to create connections for Rest APIs calls for several configuration purpose.

Therefore, you need to save and process the password every now and then. Owing to the frequent changes and vulnerability, editing or saving the password in the plain-text format is not allowed. Password must be in an encrypted format only.

To obtain password in encrypted form:

Procedure

Step 1 Create Vault.

OCI Vault provides services to create and save master encryption keys safely, and methods for encryption and decryption in using them. So Vault should be created (if not having already) in the same compartment as the rest of the autoscale solution.

Go to **OCI** > **Identity & Security** > **Vault** > **Choose or Create New Vault**

.

Step 2 Create Master Encryption Key.

One master encryption key is needed to encrypt the plain text password.

Go to **OCI** > **Identity & Security** > **Vault** > **Choose or Create Key**

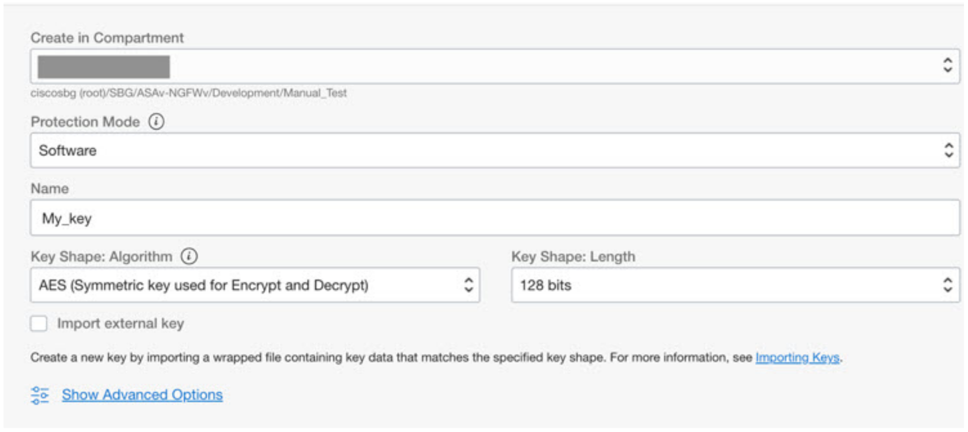
Choose any of the keys from any of the given algorithm with any bit of length.

a. AES – 128, 192, 256

b. RSA – 2048, 3072, 4096

c. ECDSA – 256, 384, 521

Figure 5: Create Key



Create in Compartment
[redacted]
ciscosbg (root)/SBG/ASAv-NGFWv/Development/Manual_Test

Protection Mode: Software

Name: My_key

Key Shape: Algorithm: AES (Symmetric key used for Encrypt and Decrypt)

Key Shape: Length: 128 bits

☐ Import external key

Create a new key by importing a wrapped file containing key data that matches the specified key shape. For more information, see Importing Keys.

Show Advanced Options

Step 3 Create encrypted password.

a. Go to **OCI** > **Open CloudShell (OCI Cloud Terminal)**

b. Execute following command by replacing <*Password*> as your password.

```
echo -n '<Password>' | base64
```

c. From the selected Vault, copy cryptographic endpoint and master encryption key OCID. Replace the following values, and then execute the encrypt command:

- KEY_OCID with Your key's OCID
- Cryptographic_Endpoint_URL with Your vault's cryptographic endpoint URL
- Password with Your password

Encrypt Command

```
oci kms crypto encrypt --key-id Key_OCID --endpoint

Cryptographic_Endpoint_URL --plaintext <base64-value-of-password>
```

d. Copy ciphertext from output of the above command and use it as required.

Preparation of Firewall Threat Defense Virtual configuration File

It is expected that Application is either deployed or its deployment plan is available.

Procedure

Step 1 Collect the following input parameters before deployment:

Parameter	Data Type	Description
tenancy_ocid	String	OCID of the tenancy to which your account belongs. To know how to find your tenancy OCID, see here. The tenancy OCID looks something like this - `ocid1.tenancy.oc1..<unique_ID>`
region	String	The unique identifier of the region in which you want the resources to be created. Example - us-phoenix-1, us-ashburn-1
lb_size	String	A template that determines the total pre-provisioned bandwidth (ingress plus egress) of the external and internal load balancer. Supported values: 100Mbps, 10Mbps, 10Mbps-Micro, 400Mbps, 8000Mbps Example : 100Mbps
availability_domain	String	Example - Tpeb:PHX-AD-1, Tpeb:PHX-AD-2 **Note** To get the availability domain names, see here.
min_and_max_instance_count	comma separated value	The minimum and the maximum number of instances that you would want to retain in the instance pool. Example: 1,5
autoscale_group_prefix	String	The prefix to be used to name all the resources that are created using the template. For example, if the resource prefix is given as 'autoscale', all the resources are named as follows - autoscale_resource1, autoscale_resource2 etc.
mgmt_subnet_ocid	String	OCID of the Management subnet that is to be used.
inside_subnet_ocid	String	OCID of the Inside subnet that is to be used.
function_subnet_ocid	String	OCID of the Function subnet that is to be used.
outside_subnet_ocid	String	OCID of the Outside subnet that is to be used.
mgmt_nsg_ocid	String	OCID of the Management subnet network security group that is to be used.
inside_nsg_ocid	String	OCID of the Inside subnet network security group that is to be used.
outside_nsg_ocid	String	OCID of the Outside subnet network security group that is to be used.

Parameter	Data Type	Description
elb_listener_port	comma separated Values	List of the communication ports for the external load balancer listener. Example: 80
ilb_listener_port	comma separated Values	List of the communication ports for the internal load balancer listener. Example: 80
health_check_port	String	The backend server port of load balancer against which to run the health check. Example: 8080
instance_shape	String	The shape of the instance to be created. The shape determines the number of CPUs, amount of memory, and other resources allocated to the instance. Supported shapes :"VM.Standard2.4" & "VM.Standard2.8"
lb_bs_policy	String	The load balancer policy to be used for the internal and external load balancer's backend set. To know more about how load balancer policies work, see here Supported values: "ROUND_ROBIN", "LEAST_CONNECTIONS", "IP_HASH"
image_name	String	The name of the marketplace image to be used for creating the instance configuration. Default value : " Cisco Firepower NGFW virtual firewall (NGFWv)" **Note** If the user wants to deploy custom image, user has to configure the custom_image_ocid parameter.
scaling_thresholds	Comma separated value	The CPU usage thresholds to be used for scale-in and scale-out. Provide the scale-in and scale-out threshold values as comma separated input. Example : 15,50 where, 15 is the scale-in threshold and 50 is the scale-out threshold.
compartment_id	String	The OCID of the compartment in which to create the resources. Example: **ocid1.compartment.oc1..<unique_ID>**
compartment_name	String	Name of the compartment

Parameter	Data Type	Description
custom_image_ocid	String	OCID of the custom image to be used to create instance configuration if the marketplace image is not to be used. **Note** *custom_image_ocid is optional parameter*
ftdv_password	String	The password for Firewall Threat Defense Virtual in the encrypted form, to SSH into the Firewall Threat Defense Virtual for configuration. Use configuration guide for the instructions on how to encrypt password or see here.
ftdv_license_type	String	Type of Firewall Threat Defense Virtual license either BYOL or PAYG. Currently, BYOL is supported.
cryptographic_endpoint	String	Cryptographic endpoint is a URL, that is used for decrypting password. It can be found in the Vault.
master_encryption_key_id	String	The OCID of key with which the password was encrypted. It can be found in the Vault. **Note** master_encryption_key_id and cryptographic_endpoint both must belong to same vault.
fmc_ip	String	IP address of Secure Firewall Management Center. IP of Firewall Management Center that will be used by customer to manage Firewall Threat Defense Virtual instances. **Note** *Firewall Management Center IP can be private only if it is in the same subnet as Firewall Threat Defense Virtual, otherwise Public IP must be used for all other cases.*
fmc_username	String	Username of the Firewall Management Center account. This username will be used to login into the Firewall Management Center to configure each time the new Firewall Threat Defense Virtual instance comes.
fmc_password	String	Password of Firewall Management Center in encrypted form. For procedure on how to encrypt password, see here.
fmc_device_group_name	String	There must be a device group in Firewall Management Center, all the Firewall Threat Defense Virtual part of this Autoscale solution will be added into that group, so that same policies and configuration can be applied to all of them.

Parameter	Data Type	Description
enable_memory_based_scaling	Bool	Publish Firewall Threat Defense Virtual Memory usage from the Secure Firewall Management Center Virtual. By enabling this flag Scaling can happen based on Memory utilization as well. By default CPU utilization is used.
fmc_metrics_username	String	In case you opt for Memory Utilization by enabling flag enable_memory_based_scaling, an extra Firewall Management Center user account is needed as that will be used continuously to pull memory usage from all the running Firewall Threat Defense Virtual instances.
fmc_metrics_password	String	Password of extra Firewall Management Center account in encrypted form. For procedure on how to encrypt password, see here.
Profile Name		It is the User's profile name in OCI. It can be found under profile section of the user. Example: "oracleidentitycloudservice/ <user>@<mail>.com"
Object Storage Namespace		It is unique identifier created at the time of Tenancy creation. Go to **OCI** > **Administration** > **Tenancy Details**
Authorization Token		This is used as password for docker login which authorizes it to push Oracle-Functions into the OCI container registry. Go to **OCI** > **Identity** > **Users** > **User Details** > **Auth Tokens** > **Generate Token**.

Step 2 Create the *Configuration.json* file with the following content:

```
{
  "licenseCaps": ["BASE", "MALWARE", "THREAT"],
  "performanceTier": "FTDv30",
  "fmcIpforDeviceReg": "DONTRESOLVE",
  "RegistrationId": "cisco",
  "NatId": "cisco",
  "fmcAccessPolicyName": "<autoscale-access-policy-name>",
  "fmcNatPolicyName": "<autoscale-nat-policy-name>",
  "fmcInsideNicName": "inside",
  "fmcOutsideNicName": "outside",
  "fmcInsideNic": "GigabitEthernet0/0",
  "fmcOutsideNic": "GigabitEthernet0/1",
  "fmcOutsideZone": "<outside-zone-name>",
  "fmcInsideZone": "<inside-zone-name>",
  "MetadataServerObjectName": "oci-metadata-server",
  "interfaceConfig": [
    {
      "managementOnly": "false",
      "MTU": "1500",
      "securityZone": {
        "name": "inside-zone"
      },
      "mode": "NONE",
      "ifname": "inside",
      "name": "GigabitEthernet0/0"
      },
      {
        "managementOnly": "false",
        "MTU": "1500",
        "securityZone": {
          "name": "outside-zone"
        },
        "mode": "NONE",
        "ifname": "outside",
        "name": "GigabitEthernet0/1"
      }
    ],
    "trafficRoutes": [
      {
        "interface": "outside",
        "network": "any-ipv4",
        "gateway": "",
        "metric": "2"
      },
      {
        "interface": "inside",
        "network": "oci-metadata-server",
        "gateway": "",
        "metric": "1"
      }
    ]
  }
```

Step 3 Update *Configuration.json* with the configuration settings.

Step 4 Upload Configuration file to Object Storage space.

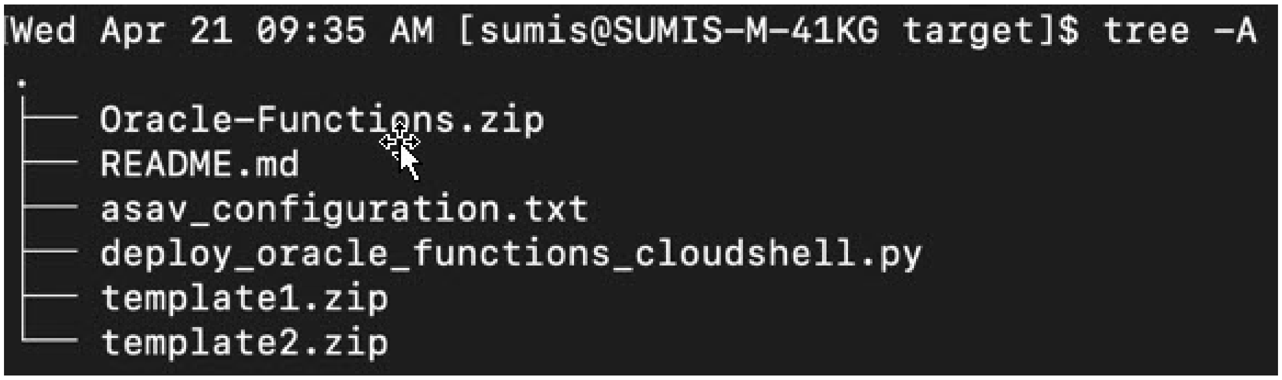
The *configuration.txt* file must be uploaded to the user created object storage space and the pre-authenticated request for the uploaded file should be created.

Note
Ensure that pre-authenticated request URL of configuration.txt is used in the stack deployment.

Note
Expiry period is needed to be defined while creating pre-authenticated URL in OCI, make sure this period is long enough to not expire during solution execution.

Step 5 Create the Zip files.

A *make.py* file can be found in the cloned repository. Execute the `python3 make.py build` command to create the zip files. The target folder has the following files.



```
[Wed Apr 21 09:35 AM [sumis@SUMIS-M-41KG target]$ tree -A
.
├── Oracle-Functions.zip
├── README.md
├── asav_configuration.txt
├── deploy_oracle_functions_cloudshell.py
├── template1.zip
└── template2.zip
```

Deploy the Auto Scale Solution

After completing the pre-requisite steps for deployment, start creating the OCI stack. You can perform a manual deployment or perform deployment using the cloud shell. Deployment scripts and templates for your version are available in the GitHub repository.

Manual Deployment

End-to-end Autoscale solution deployment consist of three steps: Deploy Terraform Template-1 Stack, Deploy Oracle Functions, and then Deploy Terraform Template-2.

Deploy Terraform Template-1 Stack

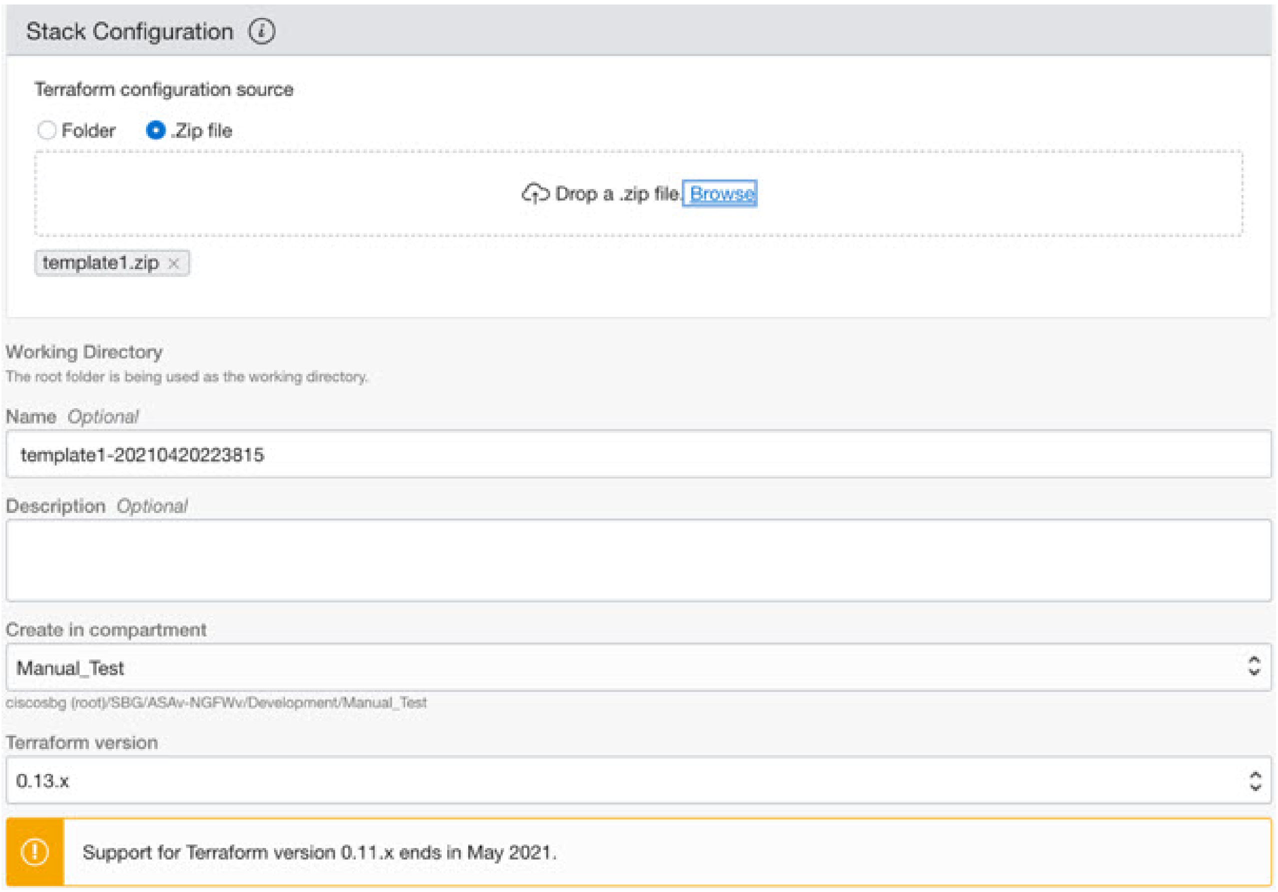
Procedure

Step 1 Log into the OCI portal.

The region is displayed in the upper right corner of your screen. Make sure you are in the intended region.

Step 2 Choose **Developer Service** > **Resource Manager** > **Stack** > **Create Stack**

Choose **My Configuration**, and then select the *Terraform template1.zip* file in the target folder as Terraform Configuration Source as shown in the figure below.



Stack Configuration

Terraform configuration source

Folder [✓ .Zip file]

Drop a .zip file Browse

template1.zip

Working Directory
The root folder is being used as the working directory.

Name *Optional*
template1-20210420223815

Description *Optional*

Create in compartment
Manual_Test
ciscosbg (root)/SBG/ASAv-NGFWv/Development/Manual_Test

Terraform version
0.13.x

Support for Terraform version 0.11.x ends in May 2021.

Step 3 In the **Transform version** drop-down list, select 0.13.x or 0.14.x.

Step 4 In the next step, enter all the details as collected in Collection of Input Parameters.

Note
Enter valid input parameters, otherwise stack deployment may fail in further steps.

Step 5 In the next step, choose **Terraform Actions** > **Apply**.

Post successful deployment, proceed to deploy the Oracle functions.

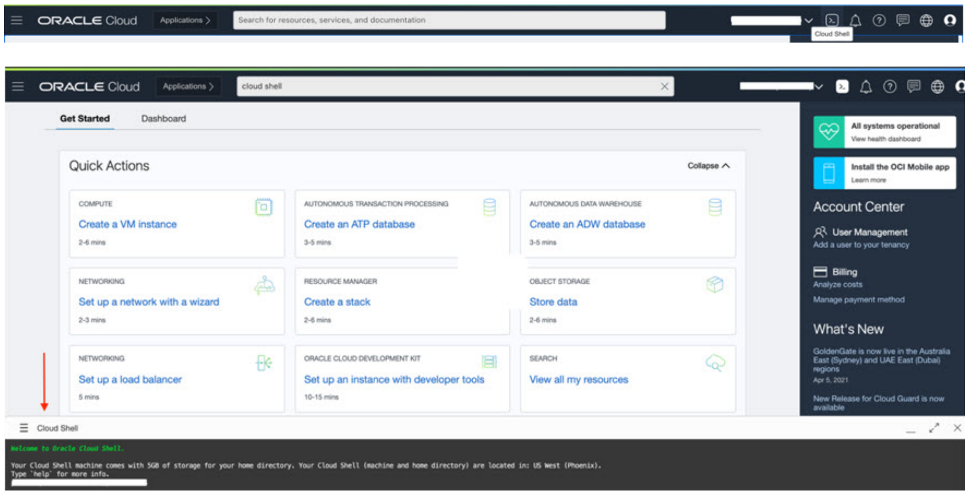
Deploy Oracle Functions

Note *This step must be performed only after successful Terraform Template-1 deployment.*

In OCI, Oracle Functions are uploaded as Docker Images, which are saved into the OCI container registry. Oracle Functions are needed to be pushed into one of the OCI Application (created in Terraform Template-1) at the time of deployment.

Procedure

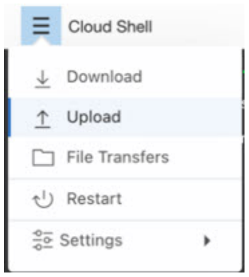
Step 1 Open OCI Cloud Shell.



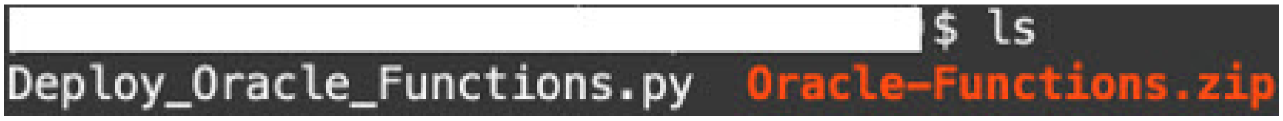
```
Welcome to Oracle Cloud Shell.

Your Cloud Shell machine comes with 5GB of storage for your home directory. Your Cloud Shell (machine and home directory) are located in: US West (Phoenix).
Type `help` for more info.
```

Step 2 Upload *deploy_oracle_functions_cloudshell.py* and *Oracle-Functions.zip*.

From the Cloud Shell's hamburger menu, choose **Upload**.

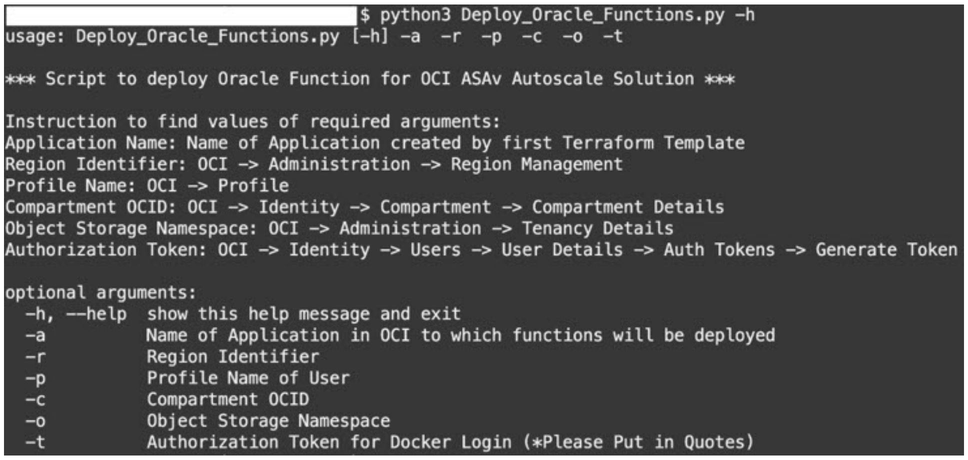


Step 3 Verify files using the **ls** command.



```
[redacted]$ ls
Deploy_Oracle_Functions.py   Oracle-Functions.zip
```

Step 4 Run `python3 Deploy_Oracle_Functions.py -h`. The `deploy_oracle_functions_cloudshell.py` script requires some input parameters whose details can be found using help argument, as shown in figure below.



```
[redacted]$ python3 Deploy_Oracle_Functions.py -h
usage: Deploy_Oracle_Functions.py [-h] -a  -r  -p  -c  -o  -t

*** Script to deploy Oracle Function for OCI ASAv Autoscale Solution ***

Instruction to find values of required arguments:
Application Name: Name of Application created by first Terraform Template
Region Identifier: OCI -> Administration -> Region Management
Profile Name: OCI -> Profile
Compartment OCID: OCI -> Identity -> Compartment -> Compartment Details
Object Storage Namespace: OCI -> Administration -> Tenancy Details
Authorization Token: OCI -> Identity -> Users -> User Details -> Auth Tokens -> Generate Token

optional arguments:
  -h, --help  show this help message and exit
  -a          Name of Application in OCI to which functions will be deployed
  -r          Region Identifier
  -p          Profile Name of User
  -c          Compartment OCID
  -o          Object Storage Namespace
  -t          Authorization Token for Docker Login (*Please Put in Quotes)
```

To run the script pass the following arguments:

Table 1: Arguments and Details

Argument	Particulars
Application Name	It is the name of OCI Application created by Terraform Template-1 deployment. Its value is obtained by combining **“autoscale_group_prefix”** given in Template-1 and suffix **“_application”.**
Region Identifier	Region identifier is the region codeword fixed in the OCI for different regions. Example: 'us-phoenix-1' for Phoenix or “ap-melbourne-1” for Melbourne. To get the list of all region with their region identifiers, go to **OCI** > **Administration** > **Region Management**.
Profile Name	It is simple User’s profile name in OCI. Example: *oracleidentitycloudservice/<user>@<mail>.com* The name can be found under profile section of the user.
Compartment OCID	It is the compartment’s OCID (Oracle Cloud Identifier). Compartment OCID where user have the OCI Application. Go to **OCI** > **Identity** > **Compartment** > **Compartment Details**.
Object Storage Namespace	It is unique identifier created at the time of Tenancy creation. Go to **OCI** > **Administration** > **Tenancy Details**.

Argument	Particulars
Authorization Token	This is used as password for docker login which authorizes it to push Oracle-Functions into the OCI container registry. Specify the token in quotes in the deployment script. Go to **OCI** > **Identity** > **Users** > **User Details** > **Auth Tokens** > **Generate Token**. For some reason, if you are not able to see User Details then click **Developer services** > **Functions**. Go to the application created by Terraform Template-1. Click **Getting Started**, and choose Cloud Shell Setup and among the steps you will find the link to generate auth token as shown below. 5 Generate an Auth Token

Step 5 Run the `python3 Deploy_Oracle_Functions.py` command by passing valid input arguments. It will take some time to deploy all the functions. You can then remove the file and close the Cloud Shell.

Deploy Terraform Template-2

Template 2 deploys the resources related to alarm creation, including alarms, ONS topics for invoking function. The deployment of template 2 is similar to Terraform Template-1 deployment.

Procedure

Step 1 Log into the OCI portal.

The region is displayed in the upper right corner of your screen. Make sure you are in the intended region.

Step 2 Choose **Developer Service** > **Resource Manager** > **Stack** > **Create Stack**.

Select *Terraform template template2.zip* in the target folder as source of Terraform configuration.

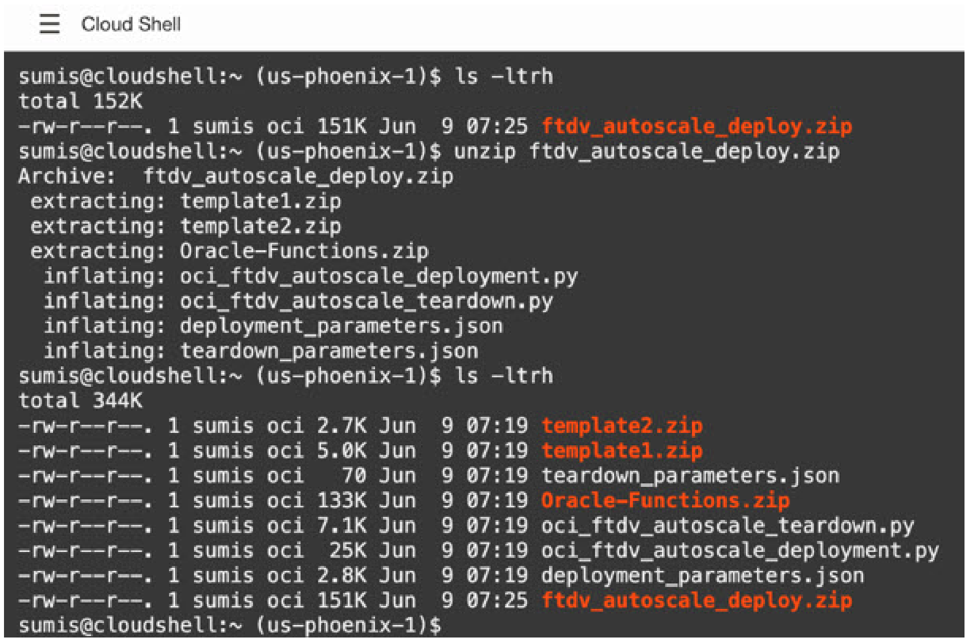
Step 3 In next step, click **Terraform Actions** > **Apply**.

Deployment using cloud shell

To avoid the deployment overhead, you can invoke the easy, end-to-end deployment script to deploy the autoscale solution (terraform template1, template2 and oracle functions).

Procedure

Step 1 Upload the *ftdv_autoscale_deploy.zip* file in the target folder to the cloud shell and extract the files.



```
sumis@cloudshell:~ (us-phoenix-1)$ ls -ltrh
total 152K
-rw-r--r--. 1 sumis oci 151K Jun  9 07:25 ftdv_autoscale_deploy.zip
sumis@cloudshell:~ (us-phoenix-1)$ unzip ftdv_autoscale_deploy.zip
Archive:  ftdv_autoscale_deploy.zip
 extracting: template1.zip
 extracting: template2.zip
 extracting: Oracle-Functions.zip
  inflating: oci_ftdv_autoscale_deployment.py
  inflating: oci_ftdv_autoscale_teardown.py
  inflating: deployment_parameters.json
  inflating: teardown_parameters.json
sumis@cloudshell:~ (us-phoenix-1)$ ls -ltrh
total 344K
-rw-r--r--. 1 sumis oci 2.7K Jun  9 07:19 template2.zip
-rw-r--r--. 1 sumis oci 5.0K Jun  9 07:19 template1.zip
-rw-r--r--. 1 sumis oci   70 Jun  9 07:19 teardown_parameters.json
-rw-r--r--. 1 sumis oci 133K Jun  9 07:19 Oracle-Functions.zip
-rw-r--r--. 1 sumis oci 7.1K Jun  9 07:19 oci_ftdv_autoscale_teardown.py
-rw-r--r--. 1 sumis oci  25K Jun  9 07:19 oci_ftdv_autoscale_deployment.py
-rw-r--r--. 1 sumis oci 2.8K Jun  9 07:19 deployment_parameters.json
-rw-r--r--. 1 sumis oci 151K Jun  9 07:25 ftdv_autoscale_deploy.zip
sumis@cloudshell:~ (us-phoenix-1)$
```

Step 2 Make sure you have updated the input parameters in the *deployment_parameters.json* before executing the `python3 make.py` build command.

Step 3 To start the autoscale solution deployment, run the `python3 oci_ftdv_autoscale_deployment.py` command on the cloud shell.

It will take approximately 10-15 minutes for the solution deployment to complete.

If there is any error during the solution deployment, error log is saved.

Validate Deployment

Validate if all resources are deployed and the Oracle Functions are connected with Alarm & Events. By default, instance pool has minimum and maximum number of instances as zero. You can edit the instance pool in OCI UI with the minimum and maximum number that you want. This will trigger new Firewall Threat Defense Virtual instances.

We recommend that you launch only one instance and check for its workflow, validate its behaviour to ensure that it is working as it is expected. Post this validation, you can deploy the actual requirements of Firewall Threat Defense Virtual.

Note Specify the minimum number of Firewall Threat Defense Virtual instances as **Scale-In protected** to avoid their removal by OCI scaling policies.

Upgrade

Upgrade Autoscale Stack

No support for upgrade in this release. Stacks should be re-deployed.

Upgrade Firewall Threat Defense Virtual VMs

No support for upgrade for Firewall Threat Defense Virtual VMs in this release. The Stack should be re-deployed with the required Firewall Threat Defense Virtual image.

Instance Pool

1. To change minimum and maximum number of instances in the Instance Pool:

   Click **Developer Services** > **Function** > **Application Name(created by Terraform Template 1)** > **Configuration**.

   Change the min_instance_count and max_instance_count respectively.

2. Deletion/Termination of Instance is not equal to Scale-in. If any instance in the Instance Pool is deleted/terminated due to external action and not the scale-in action, instance pool automatically initiates a new instance to recover.

3. Max_instance _count defines threshold limit for Scale-out action, but it can be surpassed by changing the instance count of the Instance Pool through the UI. Ensure that the instance count from UI is less than max_instance_count set in OCI Application. Else, increase the threshold accordingly.

4. Reducing the count of instances in Instance Pool directly from the application does not perform the clean-up actions set programmatically. Due to which backends will not be drained and removed from both the load balancers, if Firewall Threat Defense Virtual has license, it will be lost.

5. Due to some reasons, if Firewall Threat Defense Virtual instance is unhealthy, not responding and unreachable through SSH for some definite period of time, instance is removed from the instance pool forcefully, any license may be lost.

Oracle Functions

- Oracle Functions are actually docker images. These images are saved into root directory of OCI Container registry. These images should not be deleted as it will also delete the function that are used in the Autoscale solution.

- OCI Application created by Terraform template-1, contains crucial environmental variables, which are required by Oracle Functions to work properly. Neither the value nor the format of these environment variables should be changed, unless it is mandated. Any changes made are reflected with new instances only.

Load Balancer Backend Sets

In OCI, Load Balancer attachment to instance pool is only supported using primary interface that is configured as management interface in Firewall Threat Defense Virtual. Hence, inside interface is connected to Internal Load Balancer’s backend set; outside interface is connected to External load balancer’s backend set. These IPs are not automatically added or removed from backend set. The Autoscale solution programmatically handles both of this task. But in case of any external action, maintenance or troubleshooting, there could be situation demanding manual effort to operate on them.

As per requirements, more ports can be opened on Load Balancer using listener and backend sets. Upcoming instances IPs are automatically added to the backend set, however already existing instances IPs should be manually added.

Adding Listener in Load Balancer

To add some port as listener in Load Balancer, go to **OCI** > **Networking** > **Load Balancer** > **Listener** > **Create Listener**.

Register a backend to Backend Set

In order to register an Firewall Threat Defense Virtual instance to Load Balancer, Firewall Threat Defense Virtual instance Outside interface IP should be configured as a backend in the Backend Set of External Load Balancer. Inside interface IP should be configured as backend in Backend set of Internal Load Balancer. Ensure that the port you are using has been added into the listener.

Delete Autoscale Configuration from OCI

Stacks deployed using terraform can be deleted in the same manner, using Resource Manager in OCI. Deletion of stack removes all the resources created by it and all the information associated with these resources are removed permanently.

Note In case of stack deletion, it is recommended to make the Minimum number of instances in Instance pool to 0, wait for instances to be terminated. This will help removal of all instances and won’t leave any residue.

You can perform a manual deletion or use Cloud Shell.

Manual Deletion

The end-to-end autoscale solution deletion consist of three steps: Delete Terraform Template-2 Stack, Delete Oracle Functions, and then Delete Terraform Template-1 Stack.

Delete Terraform Template-2 Stack

To delete the Autoscale configuration, you must begin with Terraform Template-2 stack deletion.

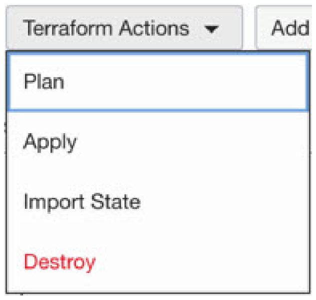
Procedure

Step 1 Log into the OCI portal.

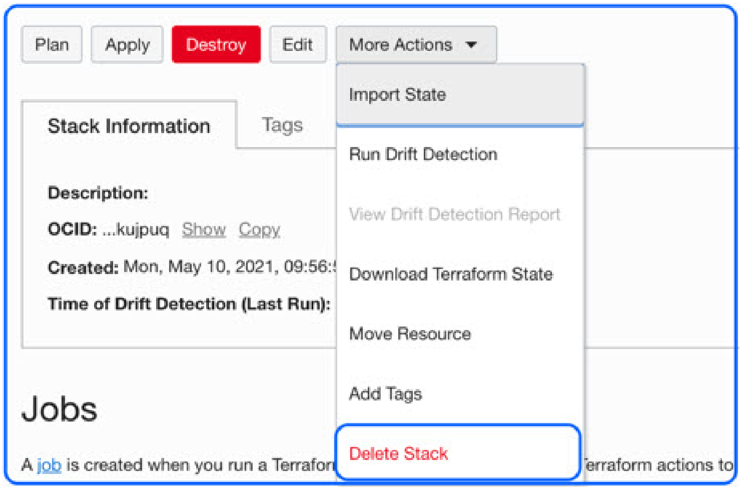
The region is displayed in the upper right corner of your screen. Make sure you are in the intended region.

Step 2 Choose **Developer Services** > **Resource Manager** > **Stack**.

Step 3 Select the stack created by Terraform Template-2, then select **Destroy** in **Terraform Actions** drop-down menu as shown in the figure below:



Destroy Job is created which takes some time to remove resources one after another. You can delete the stack after the destroy job is completed. as shown in the figure below:



Description:
OCID: ...kujpuq Show Copy
Created: Mon, May 10, 2021, 09:56:
Time of Drift Detection (Last Run):

A job is created when you run a Terrafor erraform actions to

Step 4 Proceed to delete the Oracle functions.

Delete Oracle-Functions

The Oracle-Function deployment is not a part of Terraform Template Stack deployment, it is uploaded separately using Cloud Shell. Hence, its deletion is also not supported by Terraform Stack deletion. You must delete all the Oracle-Functions inside the OCI application created by Terraform Template-1.

Procedure

Step 1 Log into the OCI portal.

The region is displayed in the upper right corner of your screen. Make sure you are in the intended region.

Step 2 Choose **Developer Services** > **Functions**. Choose the application name that was created in Template-1 stack.

Step 3 Inside this application visit each function and delete it.

Delete Terraform Template-1 Stack

Note Template-1 Stack deletion will only succeed after deleting all Oracle-Functions.

Same as Terraform Template-2 Deletion.

Procedure

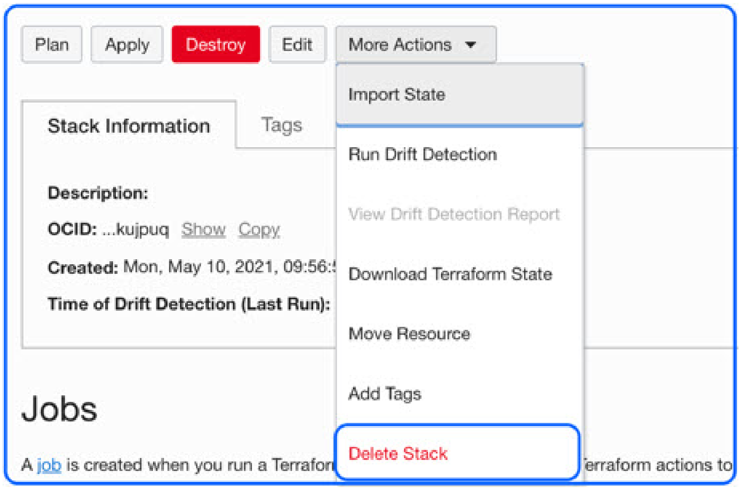
Step 1 Log into the OCI portal.

The region is displayed in the upper right corner of your screen. Make sure you are in the intended region.

Step 2 Choose **Developer Services** > **Resource Manager** > **Stack**.

Step 3 Select the stack created by Terraform Template-2, then click **Destroy** in Terraform **Actions** drop-down menu. Destroy Job will be created which will take some time to remove resources one after another.

Step 4 After the destroy job is completed, you can delete the stack from **More Actions** drop-down menu as shown in the figure below.



Post successful deletion of Terraform Template-1 stack, you must verify whether all the resources are deleted and there is no residue of any kind.

Delete Autoscale Using Cloud Shell

User can use the script to delete the stacks and oracle functions by executing the `python3 oci_ftdv_autoscale_teardown.py` command in the cloud shell. If the stacks are deployed manually, update the stack id of the stack1 and stack2, and update the application id in the *teardown_parameters.json* file.