Managing Resources on OpenStack

- Managing Resources on OpenStack, on page 1
- Managing Tenants, on page 1
- Managing Networks, on page 9
- Managing Subnets, on page 12
- Managing Flavors, on page 13
- Managing Images, on page 14
- Managing Volumes, on page 15

Managing Resources on OpenStack

Managing Tenants

A tenant identifies a tenant organization or group that is associated with a set of administrators. When you create tenant definitions, the data stored on both regional and local clusters is segmented by tenant. A tenant cannot access the data of another tenant. You can use NETCONF/ REST interface, or the ESC portal to create a tenant definition through ESC.

Note Tenants are not supported on VMware vCenter.

Three types of tenants can be created in ESC:

1. Tenant on the VIM (ESC creates the tenant)—ESC creates and uses the tenant for deployments on default VIM. ESC can delete this tenant.
2. Pre-existing (out-of-band) tenant on the VIM—ESC does not create this tenant, but uses the tenant for deployments on default VIM only. The admin tenant, for example, is a pre-existing tenant, where the ESC itself is deployed. ESC supports deploying resources such as flavors, images and volumes on a pre-existing tenant that is identified by its name or UUID. ESC manages a pre-existing tenant for default VIM only. ESC cannot delete a pre-existing tenant.
3. Tenant within ESC—ESC creates a tenant within ESC, which is independent of any VIM. This tenant acts as the root tenant for deploying VMs on multiple VIMs.

Note that the tenant name must be unique.

Note ESC can create and manage resources such as tenants, networks, subnetworks, images and flavors on the default VIM only. Only deployments are supported on the non-default VIMs (other than the default VIM).

The following attributes manage the tenants in the data model.

- managed_resource attribute
- vim_mapping attribute

The table below further explains the tenant and the attribute mapping in the data model.

Tenant Type	managed_resource	vim_mapping	Description

Tenant on the VIM(created by ESC)	true	true	ESC creates a tenant on the VIM if the managed_resource attribute is set to true. By default, the managed_resource is true. The vim_mapping attribute is true.
			`<tenants>`
			`<tenant>`
			`<name>new-tenant</name>`
			`<managed_resource>true</managed_resource>`
			`</tenant>`
			`</tenants>`

Pre-existing tenant on the VIM	false	true	For a pre-existing tenant, the managed_resource attribute is set to false. The vim_mapping attribute is true.
			`<tenants>`
			`<tenant>`
			`<name>pre-existing</name>`
			`<managed_resource>false</managed_resource>`
			`</tenant>`
			`</tenants>`
			Sample data model using the tenant UUID
			`<tenants>`
			`<tenant>`
			`<name>76eedcae-6067-44a7-b733-fc99a2e50bdf</name>`
			`<managed_resource>false</managed_resource>`
			`</tenant>`
			`</tenants>`

Tenant Type	managed_resource	vim_mapping	Description

Tenant within ESC	-	false	The vim_mapping attribute is set to false to create a tenant within ESC.
			
			`<tenants>`
			`  <tenant>`
			`  <name>esc-tenant-A</name>`
			`  <vim_mapping>false</vim_mapping>`
			`  </tenant>`
			`</tenants>`

-	false	false	Tenant is not created. The request is rejected by ESC.

To deploy VMs on multiple VIMs of the same type (OpenStack VIMs), you must create a tenant with the vim_mapping attribute set to false. This tenant can be created independently or as part of the deployment. This creates a tenant within ESC, which acts as the root tenant for multi VIM deployments. A VIM locator attribute must be specified within the each vm group for multi VIM deployment. For more details, see Deploying VNFs on Single VMware vCenter VIM.

Tenant Quotas

You can set the operational limit, known as quotas for the tenants created in ESC. The quotas can be set during the deployment using the deployment datamodel.

Note Tenant Quotas are not supported on pre-existing tenants and tenants within ESC.

The tenant supports the following quota settings for Compute (Nova) and Network (Neutron):

Compute settings:

- metadata_items
- floating_ips
- cores
- injected_file_path_bytes
- injected_files
- injected_file_content_bytes
- instances
- key_pairs
- ram
- security_groups
- security_group_rules

Compute settings:

- floatingip
- security_group_rule
- security_group
- network
- subnet
- port
- router

The deployment datamodel below shows the quota settings for the tenant.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
 <tenants>
  <tenant>
   <name>tenant-quota-example</name>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property>
                    <name>cores</name>
                    <value>26</value>
                </property>
                <property>
                    <name>metadata_items</name>
                    <value>260</value>
                </property>
                <property>
                    <name>floating_ips</name>
                    <value>26</value>
                </property>
                <property>
                    <name>injected_file_content_bytes</name>
                    <value>26000</value>
                </property>
                <property>
                    <name>injected_file_path_bytes</name>
                    <value>246</value>
                </property>
                <property>
                    <name>injected_files</name>
                    <value>26</value>
                </property>
                <property>
                    <name>instances</name>
                    <value>26</value>
                </property>
                <property>
                    <name>key_pairs</name>
                    <value>26</value>
                </property>
                <property>
                    <name>ram</name>
                    <value>26</value>
                </property>
                <property>
                    <name>security_groups</name>
                    <value>26</value>
```

```
                </property>
                <property>
                    <name>security_group_rules</name>
                    <value>26</value>
                </property>
                <property>
                    <name>floatingip</name>
                    <value>26</value>
                </property>
                <property>
                    <name>security_group_rule</name>
                    <value>26</value>
                </property>
                <property>
                    <name>security_group</name>
                    <value>26</value>
                </property>
                <property>
                    <name>network</name>
                    <value>26</value>
                </property>
                <property>
                    <name>subnet</name>
                    <value>26</value>
                </property>
                <property>
                    <name>port</name>
                    <value>26</value>
                </property>
                <property>
                    <name>router</name>
                    <value>26</value>
                </property>
            </properties>
        </extension>
    </extensions>
  </tenant>
 </tenants>
</esc_datamodel>
```

Note The property name in the deployment datamodel must match the compute and network setting names mentioned above. The tenant creation request is rejected.

Adding Tenants Using Northbound APIs

The following example explains how to create a tenant definition using NETCONF:

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <source>
            <running />
        </source>
        <config>
            <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
                <tenants>
                    <tenant>
                        <name>mytenant</name>
                    </tenant>
                </tenants>
```

```
            </esc_datamodel>
        </config>
    </edit-config>
</rpc>
```

Note For more information about creating and deleting tenant definitions using NETCONF API, see Cisco Elastic Services Controller API Guide. To access the REST API documentation directly from the ESC VM, see REST Northbound API. For more information on adding and deleting networks using the ESC portal, see Managing Resources Using ESC Portal.

Updating Quotas for Tenants

You can update the quotas for the tenants created in ESC. The quota update is only allowed on the tenants for which managed_resource and vim_mapping attributes are set to true. However, updating other configurations, for example, name, vim_mapping, managed_resource, and description are not allowed.

The following deployment data model below shows the process of updating one or multiple properties of quota for the tenants.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
 <tenants>
  <tenant>
   <name>ten-test-1</name>
    <managed_resource>true</managed_resource>
    <vim_mapping>true</vim_mapping>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property>
                    <name>cores</name>
                    <value>15</value>
                </property>
                <property>
                    <name>ram</name>
                    <value>10000</value>
                </property>
            </properties>
        </extension>
    </extensions>
  </tenant>
 </tenants>
</esc_datamodel>
```

The following data model shows how to modify the core properties of the quota of a tenant.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
 <tenants>
  <tenant>
   <name>ten-test-1</name>
    <managed_resource>true</managed_resource>
    <vim_mapping>true</vim_mapping>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property>
                    <name>cores</name>
                    <value>20</value>
                </property>
```

```
                <property>
                    <name>ram</name>
                    <value>10000</value>
                </property>
            </properties>
        </extension>
    </extensions>
  </tenant>
 </tenants>
</esc_datamodel>
```

The following data model shows how to add a non-existing property to the quota of a tenant.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
 <tenants>
  <tenant>
   <name>ten-test-1</name>
    <managed_resource>true</managed_resource>
    <vim_mapping>true</vim_mapping>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property>
                    <name>cores</name>
                    <value>15</value>
                </property>
                <property>
                    <name>ram</name>
                    <value>10000</value>
                </property>
                <property>
                    <name>network</name>
                    <value>10</value>
                </property>
            </properties>
        </extension>
    </extensions>
  </tenant>
 </tenants>
</esc_datamodel>
```

The following example shows how to delete a property from the data model.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
 <tenants>
  <tenant>
   <name>ten-test-1</name>
    <managed_resource>true</managed_resource>
    <vim_mapping>true</vim_mapping>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property nc:operation="delete">
                    <name>cores</name>
                    <value>15</value>
                </property>
                <property>
                    <name>ram</name>
                    <value>10000</value>
                </property>
            </properties>
        </extension>
    </extensions>
```

```
  </tenant>
 </tenants>
</esc_datamodel>
```

Note The property gets deleted from the data model only. The quota values remain the same for that tenant in the OpenStack.

Updating Tenant Quotas with REST API

You can use REST API to create new tenants, or modify quotas of an existing tenant in ESC.

Method Type:

```
PUT
```

URL: */ESCManager/v0/tenants/[tenant_internal_id]*

HTTP Request Headers:

internal_tenant_id : is the tenant ID to be updated

callback : address and port to receive rest callback notifications

Content-Type : application/xml

Example of REST API while creating a tenant with quotas.

```
<tenant xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <name>tenant_internal_id</name>
    <managed_resource>true</managed_resource>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property>
                    <name>port</name>
                    <value>17</value>
                </property>
                <property>
                    <name>ram</name>
                    <value>17021</value>
                </property>
                <property>
                    <name>cores</name>
                    <value>22</value>
                </property>
            </properties>
        </extension>
    </extensions>
</tenant>
```

Example of REST API while creating a tenant with modified or added quotas.

```
<tenant xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <name>tenant_internal_id</name>
    <managed_resource>true</managed_resource>
    <extensions>
        <extension>
            <name>quota</name>
            <properties>
                <property>
                    <name>port</name>
```

```
                    <value>20</value>
                </property>
                <property>
                    <name>ram</name>
                    <value>15000</value>
                </property>
                <property>
                    <name>network</name>
                    <value>5</value>
                </property>
            </properties>
        </extension>
    </extensions>
</tenant>
```

Managing Networks

In ESC, you can configure rich network topologies by creating and configuring networks and subnets, and then instructing either OpenStack or VMware vCenter services to attach virtual machines to ports on these networks.

OpenStack Network

In particular, OpenStack network supports each tenant to have multiple private networks, and allows tenants to choose their own IP addressing scheme, even if those IP addresses overlap with those used by other tenants. This enables very advanced cloud networking use cases, such as building multi-tiered web applications and allowing applications to be migrated to the cloud without changing IP addresses.

ESC supports the following networking functions:

- **Tenant Network**—A tenant network is created for a single network and all its instances. It is isolated from the other tenants.
- **Provider Network**—A provider network is created by the administrator. The attributes are mapped to the physical underlying network or a segment.

  The following attributes define a provider network:

  - network_type
  - physical_network
  - segmentation_id

- **External Network**—An external network typically provides Internet access for your instances. By default, this network only allows Internet access from instances using Network Address Translation (NAT). You can enable Internet access to individual instances using a floating IP address and suitable security group rules. The admin tenant owns this network because it provides external network access for multiple tenants.

ESC also supports ephemeral networks which are short-lived tenant networks purposely created during unified deployment and exists only during the lifetime of that deployment. For more details, see Unified Deployment Request.

Adding Networks Using Northbound APIs

The following example shows how to create a tenant network definition using NETCONF:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <tenants>
   <tenant>
      <name>quicktest4</name>
  <networks>
   <network>
    <name>proto-tenant-network34</name>
    <shared>false</shared>
    <admin_state>true</admin_state>
   </network>
  </networks>
   </tenant>
  </tenants>
</esc_datamodel>
```

The following example shows how to create a subnet for tenant network definition using NETCONF:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
  <tenants>
   <tenant>
      <name>quicktest4</name>
  <networks>
   <network>
    <name>proto-tenant-network27</name>
    <subnet>
     <name>proto-tenant-subnet4</name>
     <ipversion>ipv4</ipversion>
     <dhcp>true</dhcp>
     <address>172.16.0.0</address>
     <netmask>255.255.255.0</netmask>
     <gateway>172.16.0.1</gateway>
    </subnet>
   </network>
  </networks>
   </tenant>
  </tenants>
</esc_datamodel>
```

The following example shows how to create a simple provider network definition using NETCONF:

```
<?xml version="1.0"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
 xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
 xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <networks>
        <network>
          <name>test-net-12</name>
          <shared>true</shared>
          <admin_state>true</admin_state>
          <provider_physical_network>vm_physnet</provider_physical_network>
          <provider_network_type>vlan</provider_network_type>
          <provider_segmentation_id>200</provider_segmentation_id>
        </network>
```

```
    </networks>
</esc_datamodel>
```

The following example shows how to create a subnet for a provider network definition using NETCONF:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
 xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
 xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <networks>
        <network>
            <name>test-net-12</name>
            <subnet>
                <name>test-net-12-subnet</name>
                <ipversion>ipv4</ipversion>
                <dhcp>false</dhcp>
                <address>172.16.0.0</address>
                <gateway>172.16.0.1</gateway>
                <netmask>255.255.255.0</netmask>
            </subnet>
        </network>
    </networks>
</esc_datamodel>
```

The following example shows how to create vxlan-evpn for a provider network type in Cisco VIM:

```
<?xml version="1.0"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
    xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <networks>
        <network>
          <name>ProviderNetworkAttributes-vxlan-evpn</name>
          <shared>true</shared>
          <provider_network_type>vxlan-evpn</provider_network_type>
          <provider_segmentation_id>3010</provider_segmentation_id>
        </network>
    </networks>
</esc_datamodel>
```

The following example shows how to create an external network definition using NETCONF:

```
<network>
 <name>xyz-yesc-net-1</name>
 <shared>false</shared>
 <admin_state>true</admin_state>
 <router_external></router_external>
 <subnet>
  <name>xyz-yesc-subnet-1</name>
  <ipversion>ipv4</ipversion>
  <dhcp>true</dhcp>
  <address>172.16.0.0</address>
  <netmask>255.255.255.0</netmask>
  <gateway>172.16.0.1</gateway>
 </subnet>
</network>
```

Note For more information about creating and deleting network using NETCONF API, see Cisco Elastic Services Controller API Guide. To access the REST API documentation directly from the ESC VM, see REST Northbound API. For more information on adding and deleting networks using the ESC portal, see Managing Resources Using ESC Portal.

Managing Subnets

In ESC, a subnet is assigned to a virtual network. It specifies the IP address, the IP version for a network and so on. You can use NETCONF/ REST interface to create subnet definitions.

Note Subnet is supported on OpenStack only.

Adding Subnet Definitions Using Northbound APIs

The following example shows how to create a subnet definition using NETCONF:

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
 <target>
 <running/>
 </target>
 <config
 <esc_datamodel xmlns="http://www.cisco.com/esc/esc"
xmlns:ns0="http://www.cisco.com/esc/esc"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <networks>
 <network>
 <name>mgmt-net</name>
 <subnet>
 <name>mgmt-net-subnet</name>
 <ipversion>ipv4</ipversion>
 <dhcp>false</dhcp>
 <address>172.16.0.0</address>
 <gateway>172.16.0.1</gateway>
 <netmask>255.255.255.0</netmask>
 </subnet>
 </network>
 </networks>
 </esc_datamodel>
 </config> </edit-config
</rpc>
```

The *no_gateway* attribute allows ESC to create a subnet with the gateway disabled.

In the example below, the *no_gateway* attribute is set to true to create a subnet without gateway.

```
<networks>
        <network>
            <name>mgmt-net</name>
            <subnet>
                <name>mgmt-net-subnet</name>
```

```
                <ipversion>ipv4</ipversion>
                <dhcp>false</dhcp>
                <address>172.16.0.0</address>
                <no_gateway>true</no_gateway><!--  DISABLE GATEWAY -->
                <gateway>172.16.0.1</gateway>
                <netmask>255.255.255.0</netmask>
            </subnet>
        </network>
    </networks>
```

Note For more information about creating subnets using NETCONF API, see Cisco Elastic Services Controller API Guide. To access the REST API documentation directly from the ESC VM, see REST Northbound API. For more information on adding and deleting networks using the ESC portal, see Managing Resources Using ESC Portal.

Managing Flavors

A flavor defines sizes for RAM, disk, and number of cores.

When you deploy VNFs on OpenStack, you either have an option to use out-of-band flavors that are already available on OpenStack or create flavors in ESC. These flavors can be created using NETCONF or REST interface, or the ESC portal, and can be used for multiple deployments. For more information on deployment attributes see, Cisco Elastic Services Controller Deployment Attributes.

Note ESC Release 2.0 and later does not support creating or deleting flavor definitions on VMware vCenter.

Adding Flavors Using Northbound APIs

NETCONF request to create a flavor:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <flavors>
        <flavor>
            <name>test-flavor-indep</name>
              <vcpus>1</vcpus>
              <memory_mb>512</memory_mb>
              <root_disk_mb>0</root_disk_mb>
              <ephemeral_disk_mb>0</ephemeral_disk_mb>
              <swap_disk_mb>0</swap_disk_mb>
        </flavor>
    </flavors>
</esc_datamodel>
```

NETCONF notification upon successful creation of a flavor:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-07-13T13:33:51.805+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
```

```
    <status>SUCCESS</status>
    <status_message>Flavor creation completed successfully.</status_message>
    <flavor>test-flavor-indep</flavor>
    <vm_source>
  </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>CREATE_FLAVOR</type>
    </event>
  </escEvent>
</notification>
```

Note For more information about creating and deleting flavors using NETCONF API, see Cisco Elastic Services Controller API Guide. To access the REST API documentation directly from the ESC VM, see REST Northbound API. For more information on adding and deleting flavors using the ESC portal, see Managing Resources Using ESC Portal.

Managing Images

In ESC, an image is a bootable file system that can be used to launch VM instances.

When you deploy VNFs on OpenStack, you either have an option to use out-of-band images that are already available on OpenStack or create images in ESC. These images can be created using NETCONF or REST interface and can be used for multiple deployments.

An image can be made public or private on OpenStack. By default, the image is public. The visibility attribute is used to mark an image as public or private. A public image can only be created by an admin, whereas a private image does not require admin credentials.

Sample xml is as follows:

```
<images>
      <image>
        <name>mk-test-image</name>
        <src>file:///opt/cisco/esc/esc-confd/esc-cli/dumy.xml</src>
        <disk_format>qcow2</disk_format>
        <container_format>bare</container_format>
        <serial_console>true</serial_console>
        <disk_bus>virtio</disk_bus>
        <visibility>private</visibility>
      </image>
  </images>
```

Both out of band images, and images created by ESC can be public or private.

Adding Images Using Northbound APIs

NETCONF request to create an image:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <images>
        <image>
```

```
            <name>example-cirrosimage-indep</name>

<src>http://172.16.0.0:/share/images/esc_automated_test_images/cirros-0.3.3-x86_64-disk.img</src>

              <disk_format>qcow2</disk_format>
              <container_format>bare</container_format>
              <serial_console>true</serial_console>
              <disk_bus>virtio</disk_bus>
        </image>
    </images>
</esc_datamodel>
```

NETCONF notification upon successful creation of an image:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-07-13T13:46:50.339+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_message>Image creation completed successfully.</status_message>
    <image>example-cirrosimage-indep</image>
    <vm_source>
  </vm_source>
    <vm_target>
  </vm_target>
    <event>
      <type>CREATE_IMAGE</type>
    </event>
  </escEvent>
</notification>
```

Note For more information about adding images using NETCONF API, see Cisco Elastic Services Controller API Guide. To access the REST API documentation directly from the ESC VM, see REST Northbound API. For more information on adding and deleting images using the ESC portal, see Managing Resources Using ESC Portal.

Managing Volumes

A volume is a storage device, similar to a block device in Nova. ESC supports both volumes created by ESC and out-of-band volumes. Further, ESC also supports bootable volumes created by ESC and out of band bootable volumes.

Note The maximum number of volumes that can be attached to a VM through the nova boot command is only two.

Volumes Created by ESC

To create volume as part of the VM group, the <size> and <sizeunits> parameters must be provided in the volumes section of the deployment request. The volume type is the default volume type in Cinder.

The following example shows how to create an ESC volume in the deployment request.

```
<volumes>
```

```
 <volume>
  <name>example</name>
  <volid>1</volid>
  <bus>ide</bus>
  <size>1</size>
  <sizeunit>GiB</sizeunit>
 </volume>
</volumes>
```

Bootable Volumes Created by ESC

A bootable volume is one which is used as a root disk. ESC creates bootable volumes using the image reference name or the UUID in the deployment request. To boot instances from the volume specify the boot_index , otherwise the instance will only be an attached volume.

For example,

```
<volumes>
    <volume>
        <name>cinder-vol1X</name>
        <volid>1</volid>
        <image>cirrosX1.75</image>
        <bus>ide</bus>
        <type>lvm</type>
        <size>1</size>
        <sizeunit>GiB</sizeunit>
        <boot_index>0</boot_index>
    </volume>
</volumes>
```

Out-of-band Volumes

The out-of-band (pre-existing) volume can be specified using the <type> attribute in the deployment request. If the <type> attribute is provided, ESC matches the volume with the type provided.

ESC differentiates an out-of-band volume and volume created by ESC based on the values set in the volumes section of deployment request. The volume (only if the volume is created by ESC) associated to a VM is deleted when a service is undeployed or the VM is scaled down.

Note The support for scale in/out when using out of band volumes is no longer available.

```
<volumes>
 <volume>
  <name>pre-existing</name>
  <volid>1</volid>
  <bus>ide</bus>
  <type>lvm</type>
 </volume>
</volumes>
```

If the <type> attribute is not provided, ESC matches a volume with no type.

ESC matches a volume with the same name. If more than one volume has the same name, ESC will fail the request.

```
<volumes>
 <volume>
```

```
  <name>pre-existing</name>
  <volid>1</volid>
  <bus>ide</bus>
 </volume>
</volumes>
```

Out-of-band Bootable Volumes

Out-of-band bootable volume (for OpenStack only) is a variation of out-of-band volume, where the specified volume is used as a root disk. The VM is booted from that volume, instead of the image. The <boot_index> attribute specifies the out-of-band bootable volumes in the deployment request.

For example,

```
<volumes>
 <volume>
  <name>pre-existing</name>
  <volid>0</volid>
  <bus>ide</bus>
  <type>lvm</type>
  <boot_index>0</boot_index>
 </volume>
</volumes>
```

The out of band bootable volume can be with or without <type> attribute, similar to out of band volumes.

Parameter description

- Name—Specifies the display name of the pre-existing volume.
- Volid—Specifies the order in which volumes are attached. These are consecutive numbers starting from 0 or 1 for every VM group.
- Bus—Specifies the bus type of the volumes to be attached.
- Type—(Optional) If <type> is specified, then ESC matches the volume with the type provided.
- size and sizeunits—Defines a volume created by ESC
- boot_index—(Optional) specifies boot order. Set to 0 to boot from a given volume, similarly to how a VM would be booted from an image. The "bootable" property for that volume in OpenStack must be set to true for this to work.

Tenant-Volume API

The tenant-volume API allows you to create and delete volumes outside a deployment request. The tenant-volume API creates the volume directly under the tenant. You must provide the tenant details to create a volume.

A sample tenant-volume NETCONF API request is as follows:

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>admin</name>
            <volumes>
                <volume>
                    <name>some-volume</name>
                    <type>lvm</type>
                    <size>1</size>
```

```
                <sizeunit>GiB</sizeunit>
            </volume>
        </volumes>
    </tenant>
</tenants>
</esc_datamodel>
```

You can also use the tenant-volume API to create a volume using an existing tenant. For this, the volume name must be unique for that tenant.

Note

- The tenant-volume API is supported by both NETCONF and REST APIs.
- You cannot use the tenant-volume API to create or delete ephemeral or out-of-band volumes.
- The volumes that are managed by ESC only can be deleted.
- You cannot update an existing volume using the tenant-volume API.

Deploying with the Volumes Created by the Tenant-Volume API

ESC treats a volume created by the tenant-volume API as an out-of-band volume. To deploy a volume created by the tenant-volume API, you must provide the <size> and <sizeunit> parameters in the deployment data model. When the <size> and <sizeunit> parameters are not available, ESC looks for the volume created by the tenant-volume API. If this does not exist, then ESC looks for other out-of-band volumes created by other ESCs or other users. If out-of-band volumes are not available, then the deployment request is rejected.

A sample deployment request with a volume created using the tenant-volume API is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications"
xmlns:ns0="http://www.cisco.com/esc/esc" xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>admin</name>
            <deployments>
                <deployment>
                    <name>admin-with-volume</name>
                    <vm_group>
                        <name>cirros</name>
                        <bootup_time>60</bootup_time>
                        <recovery_wait_time>0</recovery_wait_time>
                        <image>Automation-Cirros-Image</image>
                        <flavor>Automation-Cirros-Flavor</flavor>
                        <volumes>
                            <volume>
                                <name>some-volume</name>
                                <volid>1</volid>
                                <bus>ide</bus>
                            </volume>
                        </volumes>
                        <interfaces>
                            <interface>
                                <nicid>0</nicid>
                                <network>mynetwork</network>
                            </interface>
                        </interfaces>
                        <scaling>
```

```
                                <min_active>1</min_active>
                                <max_active>1</max_active>
                                <elastic>true</elastic>
                            </scaling>
                            <kpi_data>
                                <kpi>
                                    <event_name>VM_ALIVE</event_name>
                                    <metric_value>1</metric_value>
                                    <metric_cond>GT</metric_cond>
                                    <metric_type>UINT32</metric_type>
                                    <metric_collector>
                                        <type>ICMPPing</type>
                                        <nicid>0</nicid>
                                        <poll_frequency>3</poll_frequency>
                                        <polling_unit>seconds</polling_unit>
                                        <continuous_alarm>false</continuous_alarm>
                                    </metric_collector>
                                </kpi>
                            </kpi_data>
                            <rules>
                                <admin_rules>
                                    <rule>
                                        <event_name>VM_ALIVE</event_name>
                                        <action>"ALWAYS log"</action>
                                        <action>"TRUE
                                          servicebooted.sh"</action>
                                        <action>"FALSE recover
                                          autohealing"</action>
                                    </rule>
                                </admin_rules>
                            </rules>
                            <config_data />
                        </vm_group>
                    </deployment>
                </deployments>
            </tenant>
        </tenants>
</esc_datamodel>
```

If you provide the <size> and <sizeunit> parameters of a volume, then ESC creates a new volume using these values as part of the deployment. The new volume is treated as an ephemeral volume.

Note For ephemeral volumes, the minimum and maximum scaling value can be more than 1, but for tenants and out-of-band volumes the value can be 1 only.