Deploying Virtual Network Functions on OpenStack

- Deploying Virtual Network Functions on OpenStack, on page 1
- Deploying VNFs on Multiple OpenStack VIMs, on page 5

Deploying Virtual Network Functions on OpenStack

This section describes several deployment scenarios for Elastic Services Controller (ESC) and the procedure to deploy VNFs. The following table lists the different deployment scenarios:

Scenarios	Description	Resources	Advantages
Deploying VNFs on a single VIM by creating images and flavors through ESC	The *deployment data model* refers to the images and flavors created and then deploys VNFs.	Images and Flavors are created through ESC using NETCONF/REST APIs.	• The images and flavors can be used in multiple VNF deployments. • You can delete resources (images, flavors, and volumes) created by ESC.
Deploying VNFs on a single VIM using out-of-band images, flavors, volumes, and ports	The *deployment data model* refers to the out-of-band images, flavors, volumes, and ports in OpenStack and then deploys VNFs.	Images, Flavors, Volumes, and Ports are not created through ESC.	• The images, flavors, volumes, ports can be used in multiple VNF deployments. • You cannot delete resources that are not created by through ESC.

Scenarios	Description	Resources	Advantages
Deploying VNFs on multiple VIMs using out-of-band resources	The *deployment data model* refers to out-of-band images, flavors, networks and VIM projects and then deploys VNFs.	Images, Flavors, VIM projects (specified in the locators) and Networks are not created through ESC. They must exist out-of-band in the VIM.	You can specify the VIM (to deploy VMs) that needs to be configured in ESC within a deployment.

To deploy VNFs on multiple OpenStack VIMs, see Deploying VNFs on Multiple OpenStack VIMs.

Deploying VNFs on a Single OpenStack VIM

The VNF deployment is initiated as a service request either originating from the ESC portal or the northbound interfaces. The service request comprises of XML payloads. ESC supports the following deployment scenarios:

- Deploying the VNFs by creating images, and flavors through ESC
- Deploying the VNFs using out-of-band images, flavors, volumes, and ports

Before you deploy the VNFs, you must ensure that the images, flavors, volumes, and ports are available on OpenStack, or you must create these resources. For more details on creating images, flavors, and volumes see Managing Resources Overview.

In a deployment, the out-of-band port must be created by the same tenant as the deployment. For more details on configuring ports, see Interface Configurations.

To deploy VMs on multiple VIMs, see Deploying VNFs on Multiple OpenStack VIMs.

During a deployment, ESC looks for the deployment details in the deployment data model. For more information on the deployment data model, see Cisco Elastic Services Controller Deployment Attributes. If ESC is unable to find the deployment details for a particular service, it uses the existing flavors and images under the *vm_group* to continue the deployment. If ESC is unable to find the image and flavor details, the deployment fails.

Important You can also specify the subnet that is used for a network. The deployment data model introduces a new `subnet` attribute to specify the subnet. See the Cisco Elastic Services Controller Deployment Attributes for more details.

Note When a SERVICE_UPDATE configuration fails, the minimum and maximum number of VMs change causing a scale in or scale out. ESC cannot rollback the minimum or maximum number of VMs in the configuration because of errors caused on OpenStack. The CDB (an ESC DB) would be out of synchronization. In this case, another SERVICE_UPDATE configuration must be performed to do a manual rollback.

For deployments on OpenStack, the UUID or name can be used to refer to the image and flavor. The name has to be unique on the VIM. If there are multiple images with the same name, the deployment cannot identify the right image and the deployment fails.

All deployment and ESC event notifications show tenant UUID. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-01-22T15:14:52.484+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VIM Driver: VM successfully created,
      VM Name:
[SystemAdminxyz_abc_NwDepMod1_0_5e6b7957-20e7-4df9-9113-e5fc8c047e91]</status_message>
    <depname>test_NwDepModVmGrp1</depname>
    <tenant>admin</tenant>
    <tenant_id>62cd11f560b44bf5815eaad41fc94c80</tenant_id>
 </event>
```

Reboot Time Parameter

A reboot time parameter is introduced in the deployment request. This provides more granular control to the reboot wait time of recovery in a deployment. In a deployment, when the VM reboots, the monitor is set with the reboot time. If the reboot time expires before receiving the VM ALIVE event, the next action such as VM_RECOVERY_COMPLETE, or undeploy is performed.

Note The bootup time is used, if the reboot time is not provided.

The data model change is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
   <tenants>
      <tenant>
         <name>tenant</name>
         <deployments>
            <deployment>
               <name>depz</name>
               <vm_group>
                  <name>g1</name>
                  <image>Automation-Cirros-Image</image>
                  <flavor>Automation-Cirros-Flavor</flavor>
                  <reboot_time>30</reboot_time>
                  <recovery_wait_time>10</recovery_wait_time>
                  <interfaces>
                     <interface>
                        <nicid>0</nicid>
                        <port>pre-assigned_IPV4_1</port>
                        <network>my-network</network>
                     </interface>
                  </interfaces>
                  <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>
                           <nicid>0</nicid>
                           <type>ICMPPing</type>
                           <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 />
                  <scaling>
                     <min_active>1</min_active>
                     <max_active>2</max_active>
                     <elastic>true</elastic>
                  </scaling>
                  <recovery_policy>
                     <recovery_type>AUTO</recovery_type>
                     <action_on_recovery>REBOOT_ONLY</action_on_recovery>
                     <max_retries>1</max_retries>
                  </recovery_policy>
               </vm_group>
            </deployment>
         </deployments>
      </tenant>
   </tenants>
</esc_datamodel>
```

Sample notification is as follows:

```
20:43:48,133 11-Oct-2016 WARN  ===== SEND NOTIFICATION STARTS =====
20:43:48,133 11-Oct-2016 WARN  Type: VM_RECOVERY_INIT
20:43:48,134 11-Oct-2016 WARN  Status: SUCCESS
20:43:48,134 11-Oct-2016 WARN  Status Code: 200
20:43:48,134 11-Oct-2016 WARN  Status Msg: Recovery event for
VM [dep-12_CSR1_c_0_37827511-be08-4702-b0bd-1918cb995118] triggered.
20:43:48,134 11-Oct-2016 WARN  Tenant: gilan-test-5
20:43:48,134 11-Oct-2016 WARN  Service ID: NULL
20:43:48,134 11-Oct-2016 WARN  Deployment ID: f6ff8164-fe6d-4589-84fa-f39d676e9231
20:43:48,134 11-Oct-2016 WARN  Deployment name: dep-12
20:43:48,134 11-Oct-2016 WARN  VM group name: CSR1_cirros
20:43:48,134 11-Oct-2016 WARN  VM Source:
20:43:48,134 11-Oct-2016 WARN  VM ID: 90d2066c-9a07-485b-8f72-b51026a62922
20:43:48,134 11-Oct-2016 WARN  Host ID:
69c3fba0a5b5ffff211bd05b9da7e2130d98d005a9bef71ace7d09ff
20:43:48,134 11-Oct-2016 WARN  Host Name: my-server
20:43:48,134 11-Oct-2016 WARN  [DEBUG-ONLY] VM IP: 192.168.0.75;
20:43:48,135 11-Oct-2016 WARN  =====  SEND NOTIFICATION ENDS  =====
20:43:56,149 11-Oct-2016 WARN
20:43:56,149 11-Oct-2016 WARN  ===== SEND NOTIFICATION STARTS =====
20:43:56,149 11-Oct-2016 WARN  Type: VM_RECOVERY_REBOOT
20:43:56,149 11-Oct-2016 WARN  Status: SUCCESS
20:43:56,149 11-Oct-2016 WARN  Status Code: 200
20:43:56,150 11-Oct-2016 WARN  Status Msg: VM
[dep-12_CSR1_c_0_37827511-be08-4702-b0bd-1918cb995118] is rebooted.
20:43:56,150 11-Oct-2016 WARN  Tenant: gilan-test-5
20:43:56,150 11-Oct-2016 WARN  Service ID: NULL
20:43:56,150 11-Oct-2016 WARN  Deployment ID: f6ff8164-fe6d-4589-84fa-f39d676e9231
20:43:56,150 11-Oct-2016 WARN  Deployment name: dep-12
20:43:56,150 11-Oct-2016 WARN  VM group name: CSR1_cirros
20:43:56,150 11-Oct-2016 WARN  VM Source:
20:43:56,151 11-Oct-2016 WARN    VM ID: 90d2066c-9a07-485b-8f72-b51026a62922
```

```
20:43:56,151 11-Oct-2016 WARN    Host ID:
69c3fba0a5b5ffff211bd05b9da7e2130d98d005a9bef71ace7d09ff
20:43:56,151 11-Oct-2016 WARN    Host Name: my-server
20:43:56,152 11-Oct-2016 WARN    [DEBUG-ONLY] VM IP: 192.168.0.75;
20:43:56,152 11-Oct-2016 WARN  =====  SEND NOTIFICATION ENDS  =====
20:44:26,481 11-Oct-2016 WARN
20:44:26,481 11-Oct-2016 WARN  ===== SEND NOTIFICATION STARTS =====
20:44:26,481 11-Oct-2016 WARN  Type: VM_RECOVERY_COMPLETE
20:44:26,481 11-Oct-2016 WARN  Status: FAILURE
20:44:26,481 11-Oct-2016 WARN  Status Code: 500
20:44:26,481 11-Oct-2016 WARN  Status Msg: Recovery: Recovery completed with errors
```

Deploying VNFs on Multiple OpenStack VIMs

You can deploy VNFs on multiple VIMs of the same type using ESC. ESC supports deploying VNFs on multiple OpenStack VIMs. To deploy VMs on a single instance of OpenStack, seeDeploying Virtual Network Functions on OpenStack, on page 1.

To deploy VNFs on multiple VIMs, you must:

- Configure the VIM connector and its credentials
- Create a tenant within ESC

A VIM connector registers the VIM to ESC. To deploy VNFs on multiple VIMs, you must configure the VIM connector and its credentials for each instance of the VIM. You can configure a VIM connector either at the time of installation using the bootvm.py parameters, or using the VIM connector APIs. A default VIM connector is used for a single VIM deployment. For multi VIM deployment, the locator attribute is used to specify the VIM connector.

Typically an ESC, which supports multi VIM deployment has,

- a default VIM on which ESC creates and manages resources,
- and a non-default VIM on which only deployments are supported.

For more details, see Managing VIM Connectors.

A root tenant in the data model hierarchy, which is a tenant within ESC (with the vim_mapping attribute set to false), and an out-of-band VIM tenant placed within the locator attribute must be available for deploying VNFs on multiple VIMs. If the root tenant does not exist, ESC can create a tenant during the multiple VIM deployment itself. You can create more than one ESC tenant. A user can use more than one tenant for multiple VIMs. For more information, see Managing Tenants.

In a multiple VIM deployment, you can specify the target VIM for each VM group. You can deploy each VM group on a different VIM, but the VMs within the VM group are deployed on the same VIM.

You must add a locator attribute to the VM group in the data model to enable multiple VIM deployment. The locator node consists of the following attributes:

Note If the locator attribute is present in the deployment, then the VMs are deployed on the VIM specified in the locator. If the locator attribute is not present in the deployment, then the VMs are deployed on the default VIM. If the default VIM is also not present, then the request is rejected.

- vim_id—the vim id of the target VIM. ESC defines the vim_id and maps it to the vim_connector id. The vim connector must exist before deploying to the VIM specified by the vim_id.
- vim_project—the tenant name created in target VIM. This is an out-of-band tenant or project existing in OpenStack.

Note ESC supports only out-of-band resources (pre-existing resources) such as ports, images, flavors and volumes in a multi VIM deployment. The out of band port must be created by the same tenant as the deployment.

However, multi VIM deployment supports creating only ephemeral volumes using the locator attribute on a non-default VIM. Other resources cannot be created on a non-default VIM.

Recovery of VMs, scale in and scale out of VMs are supported within the same VIM on which the VMs are deployed. The VMs cannot scale or recover on different VIMs.

In the example below, the esc-tenant is a tenant within ESC. There is no mapping to the VIM tenant, and the VMs are not deployed on this esc-tenant. The vim_project, project-test-tenant (within the locator attribute), which is created out-of-band is the tenant on which the VMs are deployed.

```
<tenants>
   <tenant>
      <name>esc-tenant</name>
      <deployments>
         <deployment>
            <name>dep-1</name>
            <vm_group>
               <name>group-1</name>
               <locator>
                  <vim_id>vim-1</vim_id>
                  <vim_project>project-test-tenant</vim_project>
               </locator>
            </vm_group>
         </deployment>
      </deployments>
   </tenant>
</tenants>
```

You can deploy VNFs on a single VIM as well with the locator attribute. That is, the datamodel with the locator attribute can also be used for deploying VMs on a single OpenStack VIM. To deploy without the locator attribute (ESC Release 2.x data model), see Deploying VNFs on a Single OpenStack VIM, on page 2.

The deployment data model is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc" xmlns:ns0="http://www.cisco.com/esc/esc"
 xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:ns2="urn:ietf:params:xml:ns:netconf:notification:1.0"
xmlns:ns3="http://www.cisco.com/esc/esc_notifications">
   <tenants>
      <tenant>
         <name>test-esc-tenant1</name>
         <deployments>
            <deployment>
               <name>dep-1</name>
               <vm_group>
                  <name>g1</name>
                  <locator>
```

```
                        <vim_id>vim1</vim_id>
                        <vim_project>project-test</vim_project>
                     </locator>
                     <bootup_time>150</bootup_time>
                     <recovery_wait_time>30</recovery_wait_time>
                     <flavor>Automation-Cirros-Flavor</flavor>
                     <image>Automation-Cirros-Image</image>
                     <interfaces>
                        <interface>
                           <nicid>0</nicid>
                           <network>my-network</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>
```

A sample multiple VIM deployment data model using out-of-band resources, and creating a root tenant as part of the deployment:

```
<esc_datamodel>
    <tenants>
        <tenant>
          <!-- This root level tenant is an ESC tenant either previously created or created
 here marked by vim_mapping atrribute. -->
           <name>esc-tenant-A</name>
           <vim_mapping>false</vim_mapping>
           <deployments>
               <deployment>
                   <name>dep-1</name>
                   <vm_group>
```

```
                        <name>Grp-1</name>
                        <locator>
                            <vim_id>SiteA</vim_id>
                            <!-- vim_project: OOB project/tenant that should already exist
 in the target VIM -->
                            <vim_project>Project-X</vim_project>
                        </locator>
                        <!-- All other details in vm group remain the same. -->
                        <flavor>Flavor-1</flavor>
                        <image>Image-1</image>
            ...
            ...
          </vm_group>
               </deployment>
            </deployments>
        </tenant>
    </tenants>
</esc_datamodel>
```

All the VIMs specified in a multi VIM deployment must be configured and in CONNECTION_SUCCESSFUL status for the request to be accepted by ESC. If a VIM specified in the deployment is unreachable or in any other status, the request is rejected.

You can apply the affinity and anti-affinity rules for VMs in a multiple VIM deployment. For more information, see Affinity and Anti-Affinity Rules on OpenStack.

Multi VIM deployment supports recovery using the Lifecycle Stages (LCS). For more information on supported LCS, see Recovery Policy (Using the Policy Framework). You can update an existing multi VIM deployment. However, the locator attribute within the VM group cannot be updated. For more information on updating an existing deployment, see Updating an Existing Deployment.