Deploying Virtual Network Functions

You can orchestrate VNFs within a virtual infrastructure domain—either on OpenStack, VMware vCenter or AWS. A VNF deployment is initiated as a service request through northbound interface or the ESC portal. The service request comprises of templates that consist of XML payloads and deployment parameters. This chapter describes the procedures to deploy VNFs (OpenStack or VMware vCenter), and the operations that you can perform during a deployment. For more information on deployment parameters, see Configuring Deployment Parameters.

Important You can assign a static IP address to connect the network to the VNF. The deployment datamodel introduces a new *ip_address* attribute to specify the static IP address. See the Cisco Elastic Services Controller Deployment Attributes for more details.

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.

Scenarios	Description	Resources	Advantages
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.
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>esc-net</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-ucs-28
```

```
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-ucs-28
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, see Deploying 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".

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>esc-net</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.

Deploying Virtual Network Functions on VMware vCenter

This section describes the deployment scenario for Elastic Services Controller (ESC) and the procedure to deploy VNFs on VMware. You can deploy VNFs using out-of-band image definitions. The following table lists the deployment scenarios:

Scenarios	Description	data model templates	Images	Advantages
Deploying VNFs on a single VIM by creating Images through ESC **Important** Images are also referred to as Templates on VMware vCenter.	The process of VNF deployment is as follows: 1. VNF Deployment- The *deployment data model* refers to the images created and then deploys VNFs.	• *deployment data model* • *image data model*	Images are created through ESC using REST APIs.	• The images can be used in multiple VNF deployments. • You can add or delete image definitions through ESC.
Deploying VNFs on a single VIM using out-of-band images	1. VNF Deployment- The *deployment data model* refers to the out-of-band images on VMware vCenter and then deploys VNFs.	• *deployment data model* • Image on VMware vCenter	Images cannot be created or deleted through ESC.	• The images can be used in multiple VNF deployments. • You can view images through ESC portal. • During out-of -band deployment, you can choose images.

Deploying VNFs on Single VMware vCenter 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 resources through ESC
- Deploying the VNFs using out-of-band resources

Before you deploy the VNFs, you must ensure that the resources are available on VMware vCenter, or you must create these resources. See Managing Resources Overview. 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.

Note Deploying VNFs on multiple VIMs is not supported on VMware vCenter.

Note A single ESC instance only supports one vCenter Distributed Switch (vDS):

- A vDS contains one or many ESXi hosts that are clustered.
- If the ESXi hosts are under one compute cluster, the VMware vCenter HA and DRS capabilities must be disabled.
- Clustered Data stores are not supported.
- If the hosts are clustered, only flat data stores under the cluster or under the datacenter are supported.

ESC only supports a default resource pool. You cannot add or create resource pools. When you see the error message "Networking Configuration Operation Is Rolled Back and a Host Is Disconnected from vCenter Server", it is due to a vCenter's limitation. The auto-select for datastore works as follows:

- ESC selects a host first. If deployment is cluster targeted, host will be selected based on the ratio of number of VMs against computing-host's capacity. Otherwise, host is selected as requested for host targeted deployment.
- From the host, datastore is picked based on its free space.

After every redeploy as part of recovery on VMware vCenter, the VM's interface(s) will have different MAC addresses.

Passing OVF Properties to a VM

As a part of deploying a VNF on VMware vCenter, you can pass the name value pair as OVF property to the VM. To pass these configurations while deploying a VNF, you must include additional arguments in the *deployment data model* template.

A sample configuration is as follows:

```
   <esc_datamodel ...>
    ...
    <config_data>
   <configuration>
      <dst>ovfProperty:mgmt-ipv4-addr</dst>
      <data>$NICID_1_IP_ADDRESS/24</data>
   </configuration>
   <configuration>
      <dst>ovfProperty:com.cisco.csr1000v:hostname</dst>
      <data>$HOSTNAME</data>
      <variable>
         <name>HOSTNAME</name>
         <val>csrhost1</val>
         <val>csrhost2</val>
      </variable>
   </configuration>
</config_data>
     ...
   </esc_datamodel>
```

Deploying VNFs on Multiple Virtual Data Centers (Multi-VDCs)

A Virtual Data Center (VDC) combines virtual resources, operational details, rules, and policies to manage specific group requirements. A group can manage multiple VDCs, images, templates, and policies. This group can allocate quotas and assign resource limits for individual groups at the VDC level.

To view the list of VDCs that are available and on the ESC portal, choose **Datacenters**.

Before you Begin

Before you deploy VNFs on multiple VDCs, ensure that the following conditions are met:

- Verify that a standard external network spanning both VDCs is available for the ESC to ping the deployed VMs.
- Verify that at least one management interface on the VMs is connected to the external network.
- Verify that the VDC is present in the vCenter.

Note

- ESC assumes all required resources to be created in VDC are out of band and present in the VDC.
- Currently, ESC can deploy in any VDC present in a vCenter. There is no scoping or restriction of VDCs that ESC can deploy in.

When you deploy a VNF, you must specify the virtual datacenter locator name on which the VNF needs to be provisioned.

A locator element is introduced in deployment request to create and delete resources.

The locator element contains:

- a datacenter name tag—to specify the target VDC for the resource (Deployment, Image, Network and Subnets).
- switch_name—to specify the target VDS to associate the network with.

Using the locator element,

- An image or a template can be created on another VDC by providing the datacenter attribute within the locator. For example,

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <images>
        <image>
            <name>automated-uLinux</name>
            <src>http://VAR_FILE_SERVER_IP/share/images/uLinux/uLinux.ovf</src>
            <locators>
                <datacenter>VAR_VDC2</datacenter>
            </locators>
        </image>
    </images>
</esc_datamodel>
```

- A network can be created and deleted from a VDC.

Note If the network is part of unified deployment, then the datacenter attribute is taken from the deployment attribute in deployment request.

```
<network>
    <locators>
        <datacenter>OTT-03</datacenter>
        <switch_name>dvSwitch</switch_name>
    </locators>
    <name>test-yesc-net-u</name>
    <shared>false</shared>
    <admin_state>true</admin_state>
</network>
```

Cisco Elastic Services Controller Portal allows you to choose the VDC on which the VM is provisioned. When you are creating a service request, you can choose the VDC on which this VM is provisioned. For more information on deploying VNFs on a VDC, see .

The *default_locators* container in ESC operational data shows default locators configured in ESC.

Note The *default_locators* container is not displayed if there are no locators configured.

Sample operational data is as follows:

```
Operational Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin
--privKeyFile=/var/confd/homes/admin/.ssh/confd_id_dsa --privKeyType=dsa --get -x
"esc_datamodel/opdata"
<?xml version="1.0" encoding="UTF-8"?><rpc-reply
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <data>
        <esc_datamodel xmlns="http://www.cisco.com/esc/esc">
            <opdata>
                <status>OPER_UP</status>
                <stats>
                    <hostname>test-ESC-2_2_6_11</hostname>
                    <os_name>Linux</os_name>
                    <os_release>2.6.32-573.22.1.el6.x86_64</os_release>
                    <arch>amd64</arch>
                    <uptime>9481</uptime>
                    <cpu>
                        <cpu_num>4</cpu_num>
                    </cpu>
                </stats>
                <system_config>
                    <active_vim>VMWARE</active_vim>
                    <vmware_config>
                        <vcenter_ip>10.85.103.22</vcenter_ip>
                        <vcenter_port>80</vcenter_port>
                        <vcenter_username>root</vcenter_username>
                    </vmware_config>
                </system_config>
                <default_locators>
                    <datacenter>OTT-ESC-4</datacenter>
                </default_locators>
                <tenants>
```

```
                    <tenant>
                        <name>admin</name>
                        <tenant_id>SystemAdminTenantId</tenant_id>
                    </tenant>
                </tenants>
            </opdata>
        </esc_datamodel>
    </data>
</rpc-reply>
[admin@test-ESC-2_2_6_11 esc-cli]$
```

Deploying Virtual Network Functions on VMware vCloud Director (vCD)

This section describes the deployment scenario for Elastic Services Controller (ESC) and the procedure to deploy VNFs on VMware vCloud Director (vCD). To install ESC on vCD, see the *Cisco Elastic Services Controller Install and Upgrade Guide*.

Resources such as organization, and organization VDC and so on must be created on vCD before deployment. For more information, see Managing Resources on vCloud Director (vCD).

Note ESC supports VMware vCloud Director 8.2.

To deploy the VNF, you must:

1. Add a VIM connector, with the organization and organization user details preconfigured in the VMware vCD. See VIM Connector Configuration for VMware vCloud Director (vCD).

   The vim_vdc leaf under the locator refers to the vDC, the deployment is targeted to.

2. Deploy the VNF with organization VDC, catalog and vApp template parameters preconfigured in the VMware vCD.

See the VMware vCloud Director Documentation to create these resources.

You must set the following key parameters, before deploying the VNFs on vCD:

- VMWARE_VCD_PARAMS—Specify the *VMWARE_VCD_PARAMS* parameter in the extensions section of the datamodel under each deployment section. The *VMWARE_VCD_PARAMS* parameter includes *CATALOG_NAME* and *VAPP_TEMPLATE_NAME*.
- CATALOG_NAME—Specify the name of the preconfigured catalog that contains references to vApp templates and the media images.
- VAPP_TEMPLATE_NAME—Specify the name of the preconfigured vApp template that contains virtual machine image that is loaded with an operating system, application, and data, it ensure that virtual machines are consistently configured across an entire organization.

A sample deployment 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>
         <!-- ESC scope tenant -->
         <name>esc-tenant</name>
         <vim_mapping>false</vim_mapping>
         <deployments>
            <deployment>
               <!-- vApp instance name -->
               <name>vapp-inst1</name>
               <policies>
                  <placement_group>
                     <name>placement-anti-affinity</name>
                     <type>anti_affinity</type>
                     <enforcement>strict</enforcement>
                     <vm_group>g1</vm_group>
                     <vm_group>g2</vm_group>
                  </placement_group>
               </policies>
               <extensions>
                  <extension>
                     <name>VMWARE_VCD_PARAMS</name>
                     <properties>
                        <property>
                           <name>CATALOG_NAME</name>
                           <value>catalog-1</value>
                        </property>
                        <property>
                           <name>VAPP_TEMPLATE_NAME</name>
                           <value>uLinux_vApp_Template</value>
                        </property>
                     </properties>
                  </extension>
               </extensions>
               <vm_group>
                  <name>g1</name>
                  <locator>
                     <!-- vCD vim connector id -->
                     <vim_id>vcd_vim</vim_id>
                     <!-- vCD organization corresponding to the vim connector -->
                     <vim_project>organization</vim_project>
                     <!-- vDC pre-preconfigured in organization -->
                     <vim_vdc>VDC-1</vim_vdc>
                  </locator>
                  <!-- VM name in vAppTemplate -->
                  <image>vm-001</image>
                  <bootup_time>150</bootup_time>
                  <recovery_wait_time>30</recovery_wait_time>
                  <interfaces>
                     <interface>
                        <nicid>0</nicid>
                        <network>MgtNetwork</network>
                        <ip_address>20.0.0.101</ip_address>
                     </interface>
                  </interfaces>
                  <scaling>
                     <min_active>1</min_active>
                     <max_active>1</max_active>
                     <elastic>true</elastic>
                     <static_ip_address_pool>
                        <network>MgtNetwork</network>
                        <ip_address>20.0.0.101</ip_address>
                     </static_ip_address_pool>
```

```
      </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>
         <configuration>
            <dst>ovfProperty:mgmt-ipv4-addr</dst>
            <data>$NICID_0_IP_ADDRESS/24</data>
         </configuration>
      </config_data>
   </vm_group>
   <vm_group>
      <name>g2</name>
      <locator>
         <!-- vCD vim connector id -->
         <vim_id>vcd_vim</vim_id>
         <!-- vCD organization corresponding to the vim connector -->
         <vim_project>organization</vim_project>
         <!-- vDC pre-preconfigured in organization -->
         <vim_vdc>VDC-1</vim_vdc>
      </locator>
      <!-- VM name in vAppTemplate -->
      <image>vm-002</image>
      <bootup_time>150</bootup_time>
      <recovery_wait_time>30</recovery_wait_time>
      <interfaces>
         <interface>
            <nicid>0</nicid>
            <network>MgtNetwork</network>
            <ip_address>20.0.0.102</ip_address>
         </interface>
      </interfaces>
      <scaling>
         <min_active>1</min_active>
         <max_active>1</max_active>
         <elastic>true</elastic>
         <static_ip_address_pool>
            <network>MgtNetwork</network>
            <ip_address>20.0.0.102</ip_address>
         </static_ip_address_pool>
      </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>
                        <configuration>
                           <dst>ovfProperty:mgmt-ipv4-addr</dst>
                           <data>$NICID_0_IP_ADDRESS/24</data>
                        </configuration>
                     </config_data>
                  </vm_group>
               </deployment>
            </deployments>
         </tenant>
      </tenants>
   </esc_datamodel>
```

Deploying Virtual Network Functoins on Amazon Web Services

This section describes the deployment scenario for Elastic Services Controller (ESC) and the procedure to deploy VNFs on Amazon Web Services (AWS). To install ESC on AWS, see the *Cisco Elastic Services Controller Install and Upgrade Guide*.

The following AWS resources must be created on AWS before deployment:

- Amazon Machine Images (AMI)
- Key Pairs
- Elastic IPs
- Security Groups
- Network Elements (such as VPCs, subnets, ACLs, gateways, routes and so on)

See the AWS documentation to create these resources.

For information on VIM connector configuration prior to AWS deployment, see "VIM Connector Configurations for AWS".

Scenarios	Description	Resources	Advantages
Deploying VNFs on a single VIM by creating Amazon Machine Image (AMI) and regions through ESC	The *deployment data model* refers to Amazon Machine Images (AMI), flavors, AWS regions, key pairs, security groups, network interfaces and VIM projects created, and then deploys VNFs.	Amazon Machine Images (AMI), flavors, AWS regions, key pairs, security groups, network interfaces, VIM projects (specified in the locators) and Networks created through ESC.	• You can specify the VIM (to deploy VMs) that needs to be configured in ESC within a deployment. • The images and flavors can be used in multiple VNF deployments. • You can delete resources created by ESC.
Deploying VNFs on multiple VIMs by creating AMIs and regions through ESC	The *deployment data model* refers to Amazon Machine Images (AMI), flavors, AWS regions, key pairs, security groups, network interfaces and VIM projects created and then deploys VNFs.	Images, Flavors, VIM projects (specified in the locators) and Networks created through ESC.	You can specify the VIM (to deploy VMs) that needs to be configured in ESC within a deployment.

For more details, see Deploying VNFs on a Single or Multiple AWS Regions, on page 17.

Deploying VNFs on a Single or Multiple AWS Regions

You can deploy VNFs on a single or multiple AWS regions or VIMs of the same type using ESC.

Note AWS is a Virtual Infrastructure Manager (VIM) for ESC. Further in this document, the terms AWS region and AWS VIM are used interchangeably.

To deploy VNFs on a single or multiple VIMs, you must:

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

A VIM connector registers the VIM to ESC. To deploy VNFs on a single or multiple AWS VIMs, you must configure the VIM connector and its credentials for each region of the VIM. You can configure a VIM connector using the VIM connector APIs. For more information, see VIM Connector Configurations for AWS.

Note A default VIM connector is not supported for AWS deployment.

ESC creates a tenant within ESC with the *vim_mapping* attribute set to false. This tenant is independent of any VIM.

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
   <tenants>
      <tenant>
         <name>aws-sample-tenant</name>
         <vim_mapping>false</vim_mapping>
      </tenant>
   </tenants>
</esc_datamodel>
```

For a single or multiple AWS VIM deployment, you must specify the target region for each VM group.

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

- 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 the target VIM. This is an out-of-band tenant or project existing in OpenStack.
- vim_region—the AWS region in which the VM groups are deployed. This is optional. If the vim region is not specified, then the VMs are deployed in the aws_default_region specified in the VIM connector.

```
<locator>
   <vim_id>AWS_EAST_2</vim_id>
   <vim_region>us-east-1</vim_region>
   <!-- the deployment is going into
North Virginia -->
</locator>
```

If the vim region is not specified,

```
<locator>
   <vim_id>AWS_EAST_2</vim_id>
   <!-- the deployment is going into the default region Ohio (us-east-2)
as defined in the VIM Connector example above -->
</locator>
```

After configuring the VIM connectors and locators, you must pass certain resources as extensions to the deployment. In the example below, the elastic IP, key pair and source destination are passed as extensions to the AWS deployment.

```
<extensions>
   <extension>
      <name>AWS_PARAMS</name>
      <properties>
         <property>
            <name>elastic_ip</name>
            <value>13.56.148.25</value>
         </property>
         <property>
            <name>source_dest_check</name>
            <value>true</value>
         </property>
```

```
         <property>
            <name>key_pair_name</name>
            <value>esc-us-east-1</value>
         </property>
      </properties>
   </extension>
</extensions>
```

A sample AWS deployment is as follows:

```
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
   <tenants>
      <tenant>
         <name>aws-east-1-tenant</name>
         <vim_mapping>false</vim_mapping>
         <deployments>
            <deployment>
               <name>aws-east-1-dep</name>
               <vm_group>
                  <name>aws-vm-east-1</name>
                  <locator>
                     <vim_id>AWS_US_EAST_1</vim_id>
                  </locator>
                  <bootup_time>600</bootup_time>
                  <recovery_wait_time>33</recovery_wait_time>
                  <flavor>t2.micro</flavor>
                  <image>ami-c7bfa6bd</image>
                  <extensions>
                     <extension>
                        <name>AWS_PARAMS</name>
                        <properties>
                           <property>
                              <name>key_pair_name</name>
                              <value>esc-us-east-1</value>
                           </property>
                        </properties>
                     </extension>
                  </extensions>
                  <interfaces>
                     <interface>
                        <nicid>0</nicid>
                        <network>vpc-d7ee1bac</network>
                        <security_groups>
                           <security_group>esc-sg-us-east-1</security_group>
                        </security_groups>
                     </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>
                           <type>ICMPPing</type>
                           <nicid>0</nicid>
                           <poll_frequency>3</poll_frequency>
                           <polling_unit>seconds</polling_unit>
                           <continuous_alarm>false</continuous_alarm>
                           <monitoring_public_ip>true</monitoring_public_ip>
                        </metric_collector>
                     </kpi>
                  </kpi_data>
```

```
                    <rules>
                       <admin_rules>
                          <rule>
                             <event_name>VM_ALIVE</event_name>
                             <action>ALWAYS log</action>
                             <action>FALSE recover autohealing</action>
                             <action>TRUE servicebooted.sh</action>
                          </rule>
                       </admin_rules>
                    </rules>
                    <config_data />
                    <scaling>
                       <min_active>1</min_active>
                       <max_active>1</max_active>
                       <elastic>true</elastic>
                    </scaling>
                 </vm_group>
              </deployment>
           </deployments>
        </tenant>
     </tenants>
</esc_datamodel>
```

Unified Deployment

ESC creates OpenStack resources such as tenants, networks, and subnetworks before deploying a VNF.

During unified deployment, you send a single combined request to create or delete the OpenStack resources, and deploy a VNF. You can create multiple networks and subnetworks, but can create only a single VNF and a single tenant using unified deployment.

A unified deployment request is defined as a new deployment request, and any number of networks and subnetworks located directly inside the deployment definition. Networks and subnets located directly inside the tenant are not considered part of a unified deployment request, and will not be removed during a subsequent undeploy request.

Update the `deployment data model` and the files with the necessary information such as the service and deployment ID, tenant, network and subnetwork ids and so on. You can either use NETCONF or REST APIs. For example, send POST REST and DELETE REST calls.

Note A single NETCONF request can be used to perform multiple actions, such as creating networks and subnetworks; creating images, flavors and deploying VNFs.

See the Elastic Services Controller Deployment Attributes for a list of deployment attributes.

- To create a deployment datamodel with a single deployment request, send POST REST call to:

  `http://[ESC_IP]:8080/v0/deployments/[internal_dep_id]`

- To delete a single deployment request, send DELETE REST call to:

  `http://[ESC_IP]:8080/v0/deployments/[internal_dep_id]`

The VNF will be undeployed, and the network and subnet will be deleted in the specified order.

Note If tenant creation fails as part of a unified deployment request, a manual rollback is needed to clean up ESC. As part of manual rollback, first an undeploy request is required to clean up the deployment, followed by a delete tenant request to clean up the failed tenant creation.

During an undeploy request, any network and subnetwork created as part of the unified deployment request will be deleted along with the VNF. However, the tenant created through unified deployment request will not be deleted.

Undeploying Virtual Network Functions

You can undeploy an already deployed VNF. Use the REST or NETCONF / YANG APIs to undeploy the VNF.

Important You can also undeploy VNFs using the ESC portal. For more information, see ESC Portal Dashboard.

Sample undeploy request is as follows:

```
DELETE /v0/deployments/567 HTTP/1.1
Host: client.host.com
Content-Type: application/xml
Accept: application/xml
Client-Transaction-Id: 123456
Callback:/undeployservicecallback
```

For more details, see Cisco Elastic Services Controller API Guides.

Reboot Parameter

A reboot time parameter is introduced in the deployment request. This provides more flexibility to the operation time of the deployment. In a deployment, when the VM reboots, the monitor is set with the reboot time. If the reboot time expires before the VM alive event, the next action such as vm_recovery_complete, or undeploy is performed.