Deploying Brownfield VMs

This section shows how to deploy a simple Brownfield deployment in ESC. A VM is created, then imported and deployed in ESC as a Brownfield deployment. After a successful Brownfield deployment, ESC monitors and manages the VM like any other VM created in ESC. For more information on Deploying VMs, see Deploying VM, on page 3.

Importing the VNF

To import the active VNF, follow the steps:

1. Through RPC load, send the XML with runtime data to ESC.

   For Example:

   ```
   esc_nc_cli --user <username> --password <password> import-deployment-data CREATE admin
   name-dep /opt/cisco/existing_vms.xml
   ```

2. Deploy the associated `dep.xml` file to ESC, by using the following command:

   ```
   esc_nc_cli --user <username> --password <password> edit-config dep.xml
   ```

3. Fix the deployment issues, if required. To fix the issues, follow the steps:

   a. Re-issue the imported data (as in step 1).

   b. Undeploy the `dep.xml` file.

      For Example:

      ```
      esc_cli --user <username> --password <password> delete-dep aTenantName aDeploymentName
      ```

   c. Re-deploy the `dep.xml` via CLI.

      For Example:

      ```
      esc_cli --user <username> --password <password> edit-config dep.xml
      ```

   d. Repeat steps a to c as needed.

4. Call RPC to declare that the ESC manages the VNF completely.

   For Example:

```
esc_nc_cli --user <username> --password <password> import-deployment-data FINALIZE admin
 name-dep
```

Example of Sample `import.xml`

```
<import>
    <vms>
        <vm_details> <!--First VM details-->
            <flavor/>
            <host/>
            <host_id/>
            <image/>
            <port/>
            <uuid>1bb008e1-d4dd-4107-bc45-7e298a5ac510</uuid>
            <name>parvmoha-bf-vm</name>
            <attached_volume/>
        </vm_details>
        <vm_details>
          ........ <!--add more vms using vm_details tag-->
        </vm_details>
   </vms>
   <deployment_name>parvmoha-dep</deployment_name>
   <project_name>admin</project_name>
   <project_uuid>563fba7044c847a6a370cc10d5ef7d57</project_uuid>
</import>
```

Example of `dep.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>aTenantName</name>
            <managed_resource>false</managed_resource>
            <deployments>
                <deployment>
                    <name>brownfield-deployment</name>
                    <vm_group>
                        <name>g2</name>
                        <vim_vm_name>vm-cirros</vim_vm_name>
                        <image>Automation-Cirros-Image</image>
                        <flavor>Automation-Cirros-Flavor</flavor>
                        <bootup_time>100</bootup_time>
                        <recovery_wait_time>0</recovery_wait_time>
                        <interfaces>
                            <interface>
                                <nicid>0</nicid>
                                <network>esc-net</network>
                              <vim_interface_name>vm-cirros-interface</vim_interface_name>

                            </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_occurrences_true>2</metric_occurrences_true>
                                <metric_occurrences_true>3</metric_occurrences_true>
                                <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>1</max_active>
                        <elastic>true</elastic>
                    </scaling>
                    <recovery_policy>
                        <recovery_type>AUTO</recovery_type>
                        <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
                        <max_retries>1</max_retries>
                    </recovery_policy>
                </vm_group>
            </deployment>
        </deployments>
    </tenant>
</tenants>
</esc_datamodel>
```

Import Limitations

- Only 1 VM per VM group is supported.
- Scaling is not supported.

Deploying VM

It is recommended to verify that no updates are made on any components that are imported during the deployment. For example, VM, subnet, or the port running in OpenStack.

The following example shows how to deploy Brown Field on OpenStack:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
    <tenants>
        <tenant>
            <name>aTenantName</name>
            <managed_resource>false</managed_resource>
            <deployments>
                <deployment>
                    <name>brownfield-deployment</name>
                    <vm_group>
                        <name>g2</name>
                        <vim_vm_name>vm-cirros</vim_vm_name>
```

```
                        <image>Automation-Cirros-Image</image>
                        <flavor>Automation-Cirros-Flavor</flavor>
                        <bootup_time>100</bootup_time>
                        <recovery_wait_time>0</recovery_wait_time>
                        <interfaces>
                            <interface>
                                <nicid>0</nicid>
                                <network>esc-net</network>
                               <vim_interface_name>vm-cirros-interface</vim_interface_name
>
                            </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_occurrences_true>2</metric_occurrences_true>
                                <metric_occurrences_true>3</metric_occurrences_true>
                                <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>1</max_active>
                            <elastic>true</elastic>
                        </scaling>
                        <recovery_policy>
                            <recovery_type>AUTO</recovery_type>
                            <action_on_recovery>REBOOT_THEN_REDEPLOY</action_on_recovery>
                            <max_retries>1</max_retries>
                        </recovery_policy>
                    </vm_group>
                </deployment>
            </deployments>
        </tenant>
    </tenants>
</esc_datamodel>
```

Note When the deployment mode is marked as brownfield **Finalized Import**, it means the VNF is fully managed by ESC . However, when it is marked as fully imported, this deployment cannot be reverted back to import mode.

Creating VM

After deploying, bypass the actual call to create the port and VM in OpenStack on Brownfield mode.

The ESC detects the Brownfield deployment, if only the deployment and tenant name in the `dep.xml` match with the ones loaded in using `import.xml`. Once that is detected, the ESC checks if the VIM is alive, that the image and flavor given in the dep.xml are available.

ESC also verifies if the custom port given in the `dep.xml` is active. When VIM Driver receives the request to create the VM, it bypasses the actual OpenStack create operation and return the VNF's UUID as read from the DB.

Undeploying VM

When a Brownfield deployment is successfully deployed, it is fully managed by ESC only after changing the the Brownfield mode to **Finalized Import**.

A Brownfield deployment can be undeployed from ESC without deleting the VM in the VIM, if the deployment is in brownfield mode.

You can turn the Brownfield mode to **Finalized Import** after a successful deployment. When the Brownfield mode is in **Finalized Import**, it deletes the VM from VIM.

VM Operations in Brownfield Mode

Block all operations on a VM in Brownfield mode. When a Brownfield deployment happens, the VM gets deployed and VM_ALIVE state triggers, if applicable. The deployment then moves to SERVICE_ACTIVE state.

When a Brownfield deployment is deployed and it is in Brownfield mode **ON**, all manually triggered VM actions (START, STOP, RECOVER, REBOOT, ENABLE/DISABLE MONITOR are blocked

The VM deployed as part of Brownfield deployment is initially in Brownfield mode **ON** state. The VM becomes fully managed by ESC only after you manually switch off the Brownfield mode. Hence, any VM operation (manual or auto-triggered) at Brownfield mode is blocked.

If a user tries to manually trigger any service action or VM action to a brownfield deployment that is not yet finalized, receives a validation error. Following is the example of a validation error message:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:esc="http://www.cisco.com/esc/esc"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    /nc:rpc/esc:recoveryVmAction
  </error-path>
    <error-message xml:lang="en">Exception from action callback: Recovery VM action action
 not allowed in brownfield mode.</error-message>
    <error-info>
      <bad-element>recoveryVmAction</bad-element>
    </error-info>
```

```
  </rpc-error>
</rpc-reply>
```

Blocking MONA-triggered Events

A successful deployment of a Brownfield, sends VM_DEPLOYED and VM_ALIVE events. The deployment moves to SERVICE_ACTIVE state. The Brownfield mode is **ON** for this deployment.

If a RECOVERY event is triggered from MONA before manually finalizing the deployment, the deployment moves to SERVICE_ERROR state and ESC sends a notification to Northbound to let it know that the import is failed. Any other VM affecting event, is blocked and ESC then sends VM_ACTION_BLOCKED notification to Northbound to let it know that a VM action is triggered by MONA is blocked

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-10-03T14:06:53.042+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>VM Action [VM_RECOVERY_AUTO] Blocked for VM
[parvmiha-cirros-_Grp1_0_89192a9f-d624-46c7-a264-bb8326ec5d29].</status_message>
    <event>
      <type>VM_ACTION_BLOCKED</type>
    </event>
  </escEvent>
</notification>
```

You can switch Brownfield mode **OFF** to manually recover the VM. Service remains in SERVICE_ACTIVE state in this case because any change in the ESC was blocked. Once the Brownfield mode is **OFF**, svc-action in CONFD rejects manual recovery as the service is in SERVICE_ACTIVE state. Use `recovery-vm-action` command, to recover the VM manually after switching **OFF** the Brownfield mode.

Note VM_ALIVE event notification is not blocked as it is needed for initial transition to SERVICE_ACTIVE state.