Installing Cisco Elastic Services Controller on OpenStack

This chapter describes how to install Cisco Elastic Services Controller on OpenStack and includes the following sections:

Installation Scenarios

The following sections briefly describe some of the common deployment scenarios that are addressed by ESC.

Cisco Elastic Services Controller can be installed in different modes as per the requirement. These different modes are configured during installation. The following sections briefly describe some of the common deployment scenarios that are addressed by ESC.

ESC Standalone

In the standalone scenario, a single active VM is deployed for ESC.

ESC with HA

ESC supports High Availability (HA) in the form of Active/Standby and Active/Active models. For Active/Standby model, two ESC instances are deployed in the network to prevent ESC failure and to provide ESC service with minimum interruption. If the active ESC instance fails, the standby instance automatically takes over the ESC services. ESC HA resolves the following single point failures of ESC:

- Network failures
- Power failures
- Dead VM instance
- Scheduled downtime

- Hardware issues
- Internal application failures

Note Installing or upgrading software on the ESC VM is not supported. For further assistance, contact Cisco TAC.

For more information on Installing ESC HA Active/Standby, see Installing Cisco Elastic Services Controller Using the QCOW Image, on page 2 and Installing Cisco Elastic Services Controller on VMware vCenter.

For more information on the Active/Active model, see ESC HA Active/Active Overview Chapter.

Main Components of Cisco Elastic Services Controller Setup

The Cisco Elastic Service Controller (ESC) setup has the following components:

- **Virtual Infrastructure Manager**—Elastic Services Controller (ESC) and its VNFs are deployed in a Virtual Infrastructure Manager (VIM). This might have one or more underlying physical nodes.
- **ESC Virtual Machine**—The ESC VM is a VM that contains all the services and processes used to register and deploy services. This includes the ESC Manager and all other services. ESC provides Netconf API, REST API, and Portal as north bound interfaces to communicate with ESC. ESC VM contains CLI to interact with ESC VM. There are two CLI, one uses the REST API and the other uses Netconf API.

Installing Cisco Elastic Services Controller Using the QCOW Image

You can install Cisco Elastic Services Controller (ESC) on OpenStack by using a QCOW image. ESC will be deployed in the OpenStack as running VM instance to manage VNFs. Therefore, ESC need OpenStack environment parameters to be installed in OpenStack. The installation time varies from 10 to 20 minutes, depending on the host and the storage area network load. This procedure describes how to create ESC Virtual Machine (VM) in OpenStack.

Before you begin

- All system requirements are met as specified in Prerequisites.
- You have the information identified in Preparing for the Installation.
- Copy the ESC image file on the system where you want to install ESC.
- This system must be accessible by OpenStack.

Procedure

Step 1 Log in to the system where you want to install ESC.

Step 2 Check the compatibility of the bootvm.py and the ESC image:

```
./bootvm.py --version
```

For more information on the ESC installer arguments, see **Appendix: A Cisco Elastic Services Controller Installer Arguments**.

Step 3 In a text editor, create a file named PROJECT-openrc.sh file and add the following authentication information. The following example shows the information for a project called admin, where the OpenStack username is also admin, and the identity host is located at controller node.

Note To set the required environment variables for the OpenStack command-line clients, you must create an environment file called an OpenStack rc file, or openrc.sh file. This project-specific environment file contains the credentials that all OpenStack services use. The ESC installation script requires these OpenStack environment parameters to perform authentication and installation on OpenStack. If all the OpenStack credentials are passed through its own arguments, the bootvm.py script doesn't require these parameters.

```
export OS_NO_CACHE=true
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL=http://controller node:35357/v2.0
```

The other OpenStack parameters required for installation are: --os_auth_url, --os_username, --os_password , -os_tenant_name, --bs_os_user_domain_name, --bs_os_project_domain_name, --bs_os_identity_api_version, --bs_os_auth_url, --bs_os_username, --bs_os_password , -bs_os_tenant_name, --bs_os_user_domain_name, --bs_os_project_domain_name, --bs_os_identity_api_version.

For OpenStack V2 API, you need following items to be defined in your global environment variables: --os_password, --os_auth_url, --os_username, --os_tenant_name.

For OpenStack V3 API, set --os_identity_api_version=3 . Other parameters required for OpenStack V3 API are: --os_user_domain_name, --os_project_domain_name, --os_project_name, --os_password, --os_auth_url, --os_username, --os_identity_api_version, --os_ca_cert, --requests_ ca_bundle.

Note The arguments, --os_tenant_name, --os_username, --os_password, --os_auth_url will also by default configure the VIM connector. If you want to skip configuring the VIM connector, pass the parameter (--no_vim_credentials) with the bootvm.py. When no_vim_credentials parameter is provided, the bootvm.py arguments (os_tenant_name, os_username, os_password,os_auth_url) are ignored. For more information on configuring VIM connectors after installation, and managing VIM connectors, see Managing VIM Connectors in the *Cisco Elastic Services Controller User Guide*.

Note --os_ca_cert and --requests_ca_bundle arguments are only required for https connection.

Step 4 On any shell from which you want to run OpenStack commands, source the *PROJECT-openrc.sh* file for the respective project. In this example, you source the *admin-openrc.sh* file for the *admin* project .

```
$ source admin-openrc.sh
```

Step 5 Check the environment variables.

```
$ env | grep OS_
```

Step 6 Register ESC image file in the OpenStack image using the glance command:

```
$ glance image-create \
--name <image_name> \
--is-public=<true or false> or --visibility public or private\
```

```
--disk-format  <disk_format> \
--container-format <container_format> \
--file <file>\
--progress
```

An example configuration is shown below:

```
$  glance image-create \
--name esc-1_0_01_11_2011-01-01 \
--is-public=<true or false> or --visibility public or private \
--disk-format qcow2 \
--container-format bare \
--file esc-1_0_01_11_2011-01-01.qcow2 \
--progress
```

The **glance image-create** command is used to create a new image. The command takes the following arguments:

Note The 'is-public' argument is applicable only for OpenStack Kilo release.

Arguments	Description
name	Name of the image.
is-public	(Optional) Makes the image accessible to the public.
disk-format	Disk format of the image. ESC uses a qcow2 disk format.
container-format	Container format of image. ESC uses a bare container format.
file	Local file that contains disk image to be uploaded during creation.
progress	(Optional) Shows upload progress bar.

To verify whether the image has been registered successfully:

a) Using OpenStack Controller dashboard:

- Log into OpenStack using your credentials.
- Navigate to **Admin** > **Image**.

  Verify if the image appears in the list.

b) Using nova CLI:

```
$ nova image-show <image_name>
```

Step 7 The standard resource requirement of ESC is 4vCPU, 8G RAM, and 40GB disk space. ESC installation script takes the pre-defined "m1.large" flavor which has the definition of 4vCPU, 8G RAM and 80G disk space. To use 40GB disk space, create a flavor with the minimum disk space requirement.

```
$ nova flavor-create ESC_FLAVOR_NAME ESC_FLAVOR_ID 8192 40 4
```

Step 8 To deploy ESC VM, do the following:

a. Ensure that the existing network have connectivity to OpenStack controller. To verify the network connectivity using the nova CLI use:

```
$ nova net-list
```

b. Record the ID of the network that ESC connects to boot the ESC VM by with image and flavor created earlier. The bootvm.py command requires at least one --user_pass argument to create an admin account for linux (ssh/console access) and at least one --user_confd_pass to create an admin account for ConfD (netconf/cli access). The following are the syntax for these mandatory user credential arguments:

```
--user_pass admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE][:OPTIONAL-ROLE]
--user_confd_pass admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE]
```

Generating hashed password is optional. You can select a plain password as and when required.

To generate a hashed password on Ubuntu OS, use the following command:

```
mkpasswd --method=SHA-512 --salt XyZ123 <<< <Password>
```

Starting ESC Release 5.4, invoking the esc_nc_cli command without username and password fails with the following message:

```
admin@esc$ esc_nc_cli --user <username> --password <password> get
esc_datamodel/opdata/status
ERROR Cannot find file /home/admin/.ssh/confd_id_rsa to support public key authentication
 with esc-nc-admin
* Check your arguments or installation.
* Usage without --user --password or --privKeyFile arguments requires ssh keys and
configuration.
* Use the following command to generate new SSH Keys and update to authorized_keys:
sudo escadm confd keygen --user admin
* NOTE: Consult with site security policies for use of public key authentication.
```

You must run the esc_nc_cli command with a username and password:

```
admin@esc$ esc_nc_cli --user admin --password ADMIN-CONFD-PASSWORD get
esc_datamodel/opdata/status
Operational Data
/opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user admin
--password=****** --get -x "esc_datamodel/opdata/status"
<?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>
      </opdata>
    </esc_datamodel>
  </data>
</rpc-reply>
```

The following is an example to install ESC with an authorized public key. In the following example, single quotes are used to avoid conflict with shell reserved characters:

```
--user_pass admin:'$algorithm$salt$hash-of-salt-password':$HOME/.ssh/esc_rsa.pub
--user_confd_pass admin:'$algorithm$salt$hash-of-salt-password':$HOME/.ssh/esc_rsa.pub
```

The public keys are generated as part of a key pair, such as:

```
ssh-keygen -t rsa -b 1024 -C "esc" -N "" -f ~/.ssh/esc_rsa
```

Your public key and identification key are saved in /home/username /.ssh/esc_rsa and esc_rsa.pub files. For more examples on the user credential arguments, see Appendix A: Cisco Elastic Services Controller Installer Arguments.

c. To check the details of ESC VM and get the information including the IP address(es) of the ESC VM, use the following command:

```
$ nova show <esc_vm_name>
```

Additional Installation Options

- **Creating ConfD SSH Keys:** Starting ESC Release 5.4, you can invoke the following command to create SSH keys and configure the public key authentication:

  ```
  sudo escadm confd keygen --user USERNAME
  ```

  Run the above command on each user account on the ESC VM that requires authentication without password. In an Active/Standby deployment, run the command on both the nodes.

  Enabling ConfD SSH keys using bootvm.py: Enabling the following bootvm.py command restores the functionality prior to ESC Release 5.4:

  ```
  --confd_keygen
  ```

  For information on ConfD cli, see Changing the ConfD NetConf CLI Administrator Password Using the Command Line Interface.

- **Deploying ESC in an OpenStack IPv6 environment:**Before deploying ESC instance in IPv6, make sure to source openrc that supports ipv6 addresses. To deploy ESC in IPv6 environment, use the following bootvm arguments:

  ```
  ./bootvm.py <esc_vm_name-ipv6> --poll --user_rest_pass <username>:<password> --image
  <image_name>
  --net <ipv6_network> --ipaddr <ipv6_ip_address> --enable-http-rest --user_pass
  <username>:<password>
  --user_confd_pass <username>:<password> --etc_hosts_file <hosts-file-name> --route
  <default routing configuration>
  ```

- **Deploying ESC in DHCP mode:** If you use the bootvm.py argument without --ipaddr, then the ESC instance will be deployed in a DHCP mode. To deploy ESC in a DHCP network, use the following configuration:

  ```
  ./bootvm.py <esc_vm_name> --image <image_name> --net <IPv6 network> <IPv4 network>
  --flavor <flavor_name>
  --user_pass <username>:<password>
  --user_confd_pass <username>:<password>
  ```

Note By default, ESC only support DHCP in IPv4 networks. If IPv6 is used, you need to log in the ESC VM and run "dhclient -6 ethX" (ethX is the V6 interface name) manually to enable V6 DHCP.

When deploying ESC with multiple network interfaces, with one or more type of DHCP, use `bootvm.py --defroute N` to specify the interface index to be assigned default route and gateway IP.

By default, `N = 0`. So the first network interface on the `bootvm.py` command line is default.

For Example:

```
--defroute 1 will assign <network2> with <default_gateway_ip_address>
./bootvm.py <esc_vm_name> --image <image_id> --net <network1>
<network2> --defroute 1 --gateway_ip <default_gateway_ip_address>
```

- **Using a Bootable Volume in ESC Installation:** You can attach a volume to an ESC instance and launch an instance from inside the volume. For more information, see the sectionUsing a Bootable Volume in ESC Installation.
- **Assigning Floating IP to the ESC:** If you want to associate a floating IP with the ESC instance , do the following:

  1. Check for an available floating IP address and assign it to the ESC VM:

     ```
     $ nova floating-ip-list
     $ nova floating-ip-associate esc_vm_name <ip_address>
     ```

  2. Or create a new floating IP address and assign it to the ESC VM:

     ```
     $ nova floating-ip-create <FLOATING_NETWORK - ID>
     $ nova floating-ip-associate esc_vm_name <ip_address>

     or
     neutron floatingip-create FLOATING_NETWORK
     neutron floatingip-associate floating-ip-ID port-ID
     ```

- **Deploying ESC with static IPs:** To use ESC in a specific network with static IPs, for example, 192.168.0.112 at network1, specify **--ipaddr** and **--gateway_ip** to the bootvm command line , as shown below:

Note Before assigning static IP, make sure the static IP is available and is not being used on other machine.

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network>
--ipaddr <ip_address> --gateway_ip <default_gateway_ip_address> --user_pass
<username>:<password>
--user_confd_pass <username>:<password>
```

- **Deploying ESC with multiple network interfaces:** To use multiple networks for ESC, for example, 192.168.0.112 at network1 and 10.20.0.112 at network2, specify both the IP addresses and the network names of the interfaces in the **--net** and **--ipaddr** arguments in the following command line. In addition,

also choose the default gateway for ESC from the gateways of these networks. Specify the default gateway for ESC through the **--gateway_ip** argument.

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network1>
<network2>  --ipaddr <ip_address1> <ip_address2> --gateway_ip <default_gateway_ip_address>

--user_pass <username>:<password>
--user_confd_pass <username>:<password>
```

Note If **--flavor** is not specified, bootvm.py will use the default flavor "m1.large" in OpenStack.

Ensure that m1.large exist in OpenStack if decide not specified a flavor, and also ensure that the flavor suits ESC deployment requirement.

- **Deploy ESC with log forwarding options:** To forward ESC logs to an rsyslog server, specify the IP address of the rsyslog server while creating an ESC VM. Optionally, you can also specify the port and protocol to use.

  For example, if the IP address of the rsyslog server is 172.16.0.0 the port on the server to forward logs is 514, and the protocol used is UDP, the ESC installation could be

  ```
  ./bootvm.py <esc_vm_name> --image <image_id> --net network1 --rsyslog_server 172.16.0.0

  --rsyslog_server_port 514 --rsyslog_server_protocol udp --user_pass <username>:<password>

  --user_confd_pass <username>:<password>
  ```

- **Disabling the ESC GUI:** To boot up ESC VM with the graphical user interface disabled, modify the --esc_ui_startup argument value, as shown in the command line below:

  ```
  ./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_pass
  <username>:<password>
  --user_confd_pass <username>:<password>
  --esc_portal_startup=False
  ```

- **Enabling REST interface for ESC:** To support the REST interface, specify **--enable-https-rest** argument. You can activate REST interface on both https or http:

  ```
  ./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_pass
  <username>:<password>
  --user_confd_pass <username>:<password> --enable-https-rest

  OR

  ./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_pass
  <username>:<password>
  --user_confd_pass <username>:<password> --enable-http-rest
  ```

-
- **Enabling REST interface for ETSI:** To support the ETSI REST intetrface, specify **--enable-http-etsi** to activate the interface over http, or **--enable-https-etsi** to activate the interface over https.

  ```
  ./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_etsi_pass
  <username>:<password> --enable-https-etsi . . .

  OR

  ./bootvm.py <esc_vm_name> --image <image_id> --net <network> --user_etsi_pass
  ```

```
<username>:<password>
--user_confd_pass <username>:<password> --enable-http-etsi...
```

Note Only the https REST and ETSI interfaces should be enabled in a production environment.

- **Deploying ESC with ETSI OAuth2 Clients:** To add ETSI OAuth2 clients during the installation, use the arguments as shown below. The OAuth2 clients can also be added using the escadm commands after the installation.

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network> --
user_etsi_pass <username>:<password> --etsi_oauth2_pass <clientId>:<clientSecret>
--enable-https-etsi ...
```

- **Deploying ESC with global parameters:** To set the global configurations through the esc_params_file during the installation, use the arguments as shown below. These global configurations can also be changed through REST API after the installation.

Note The default security group is applied to the tenant during tenant creation. By default, the ESC configuration parameter for the security group, *openstack.DEFAULT_SECURITY_GROUP_TO_TENANT* is set to true. The configuration parameter must be set at the time of installation. You can query or update the parameter on ESC VM through the REST API. If the parameter is set to true, you can create and assign default security group during tenant creation. If the parameter is set to false, you cannot create or assign default security group during tenant creation. For details on the parameters that can be configured through esc_params_file, see Appendix A: Cisco Elastic Services Controller Installer Arguments.

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network>  --flavor <flavor_name>
--user_pass <username>:<password>:<public key file> --user_confd_pass
<username>:<password>
--esc_params_file <esc parameter configuration file>
```

- **Deploying ESC with ETSI properties**: To set the ETSI properties through the *etsi_params_file* during installation, use the arguments as shown below. These properties can also be changed in the *etsi-production.properties* file after the installation.

Note For details on the properties that can be configured through the etsi_params_file, see the ETSI Production Properties chapter in the *ETSI NFV MANO User Guide*.

```
./bootvm.py <esc_vm_name> --image <image_id> --net <network> --flavor <flavor_name>
--user_pass <username>:<password>:<public key file> --user_confd_pass
<username>:<password>
--etsi_params_file <etsi properties file>
```

- **Deploying two instances of ESC to build an ESC HA (Active/Standby) pair**: For more information on deploying ESC HA (Active/Standby), see Configuring High Availability in Installing ESC on OpenStack and Installing ESC on VMware chapters.

- **Adding a Dynamic Mapping File**: In Cisco ESC Release 2.1 and earlier, mapping the actions and metrics defined in the datamodel to the valid actions and metrics available in the monitoring agent was enabled using the *dynamic_mappings.xml* file. The file was stored in the ESC VM and was modified using a text editor. ESC 2.2 and later do not have an *esc-dynamic-mapping* directory and *dynamic_mappings.xml* file. If you want to add an existing dynamic_mapping xml file to the ESC VM, do the following:

  1. Backup this file to a location outside of ESC, such as, your home directory.
  2. Create *esc-dynamic-mapping* directory on your ESC VM. Ensure that the read permissions are set.
  3. Install on your ESC VM using the following bootvm argument:

     ```
      --file
     root:root:/opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml:<path-to-local-copy-of-dynamic-mapping.xml>
     ```

  The CRUD operations for mapping the actions and the metrics is available through REST API. To update an existing mapping, delete and add a new mapping through the REST API.

- **Changing the confd password on an ESC VM :** As an administrator, you can configure the confd password through bootvm.py, during the installation time:

  ```
  ./bootvm.py --user_pass <username>:<password> --user_confd_pass
  admin:'PASSWORD-OR-HASH':OPTIONAL-PUBLIC-KEY
  ```

  To reconfigure this password after the installation, execute the following commands:

  ```
  $ /opt/cisco/esc/confd/bin/ssh admin@localhost -p 2024
  $ configure
  $ set aaa authentication users user admin password <your_password>
  $ commit
  $ exit
  ```

Note For the ease of future upgrades, make sure that you keep a copy of all the commands and arguments that are used while installing ESC using the bootvm.py file.

Managing Root Certificates in Cisco Elastic Services Controller

Cisco Elastic Services Controller (ESC) provides a mechanism to enable verification of SSL certificates. Currently, this feature is supported only on OpenStack. Certificate validation is enabled by default during the initial ESC boot up. However, ESC also allows you to configure these SSL certificates. This section describes how to enable/disable certificate validation, add/remove, or list the certificates for Cisco Elastic Service Controller on OpenStack. You can add a root certificate during the ESC bootup or even after ESC bootup is completed.

Enabling/Disabling the Root Certificate Validation

Cisco Elastic Services Controller by default enables certificate validation. You can also enable or disable by modifying the parameter, DISABLE_CERT_VALIDATION, available under the Openstack category in the esc_params.conf file, or through the REST interface, or using the escadm tool.

On ESC active node, use the command, **sudo escadm enable-certificate** or **sudo escadm disable-certificate** to enable and disable the certificate validation, respectively.

Adding a Root Certificate

You can add a root certificate during the ESC bootup or even after ESC bootup is completed. Before adding certificates, ensure the OpenStack environment file, OpenStack RC file has parameters to perform authentication and installation on OpenStack. The --os_auth_url must be specified while passing the parameters. --os_auth_url specifies the secure (https) or unsecured (http) keystone URL used by OpenStack for authentication.

- Add certificate for standalone (only) during the bootup time, i.e, during the ESC VM installation:

```
./bootvm.py test-vm --image <image_name> --net <network> [--cert_file CERT_FILE]
[--confd_aes_key CONFD_AES_KEY]
 /home/cisco/openstack.crt
--user_pass <username>:<password> --user_confd_pass <username>:<password>
```

Note Currently, ESC does not support adding a certificate for HA Active/Standby during the installation as the keepalived service is not running when a certificate is added.

- Add certificate for standalone/HA Active/Standby after booting up the ESC instance. The escadm tool has an **truststore add** option which has the following arguments: The --file argument refers to the CA certificate file. Using this argument you can import any file format supported by the java keytool: X.509 v1, v2, and v3 certificates, and PKCS#7. The --alias argument is unique and refers to the name this specific CA certificate is given.

  1. Copy/Transfer CA Certificate file to ESC active VM .
  2. Add certificate to ESC truststore. To do this, execute the following command:

     ```
     sudo escadm escadm truststore add --alias [ca cert alias] --file [file path]
     ```

     or

     ```
     sudo escadm truststore escadm truststore --alias [ca cert alias] --file [file path]
     ```

  3. Verify the certificate is added.

     ```
     sudo escadm truststore show
     ```

     or

     ```
     sudo escadm truststore show
     ```

Removing a Root Certificate

The escadm tool has a 'truststore delete' option which only takes --alias argument. The --alias argument refers to the name of the CA certificate to be deleted. Use this argument on the standalone/HA Active/Standby ESC VM:

Procedure

Step 1 On (active)ESC use escadm to delete certificate from ESC truststore.

```
sudo escadm truststore delete --alias [ca cert alias]
```

or

```
sudo escadm truststore truststore delete --alias [ca cert alias]
```

Step 2 Verify the certificate is removed.

```
sudo escadm truststore show
```

or

```
sudo escadm truststore show
```

Managing Root Certificates During the Upgrade

- **Image Upgrade**: If you are backing up the ESC DB for upgrade, then no other action is required, the ESC truststore will be restored once the ESC DB is restored. If you are not backing up the ESC DB for upgrade, then each CA certificate needs to be added again to the ESC truststore.
- **RPM Upgrade**: This upgrade method keeps the ESC truststore as is, i.e. all ca certificates in the ESC truststore should remain there after upgrade.

Managing Keystore in Cisco Elastic Services Controller

The keystore is a storage for certificates and keys that an application uses to authenticate itself with different clients.

ESC Keystore holds only one certificate, that is used by all the applications for example, ESCManager, VIMManager, MONA, and so on.

ESCADM provides multiple commands to manage the keystore.

When deploying ESC for the first time, a self signed certificate is created and stored in the keystore by default.

important Notes:

- In an Active/Active mode, the default certificate Common Name (CN) is `db.service.consul`. If you want to set a new certificate, you must set it with the same CN, otherwise, you must add the following configuration to the heat template of all the nodes to disable the certificate validation in MONA when deploying the ESC:

  ```
  mona:
            certificate_validation: false
  ```

- All the escadm keystore commands require root privileges to be executed on the ESC.

ESCADM Keystore Commands

- `escadm  keystore show`

  The `escadm keystore show` command displays information about the certificate currently stored in the keystore. It shows informations like, the creation date, Alias, and certificate fingerprint.

  For example:

  ```
  $ sudo escadm keystore show
  Keystore type: PKCS12
  Keystore provider: SUN
  Your keystore contains 1 entry
  esc, Apr 13, 2020, PrivateKeyEntry,
  Certificate fingerprint (SHA1):
  FF:11:66:3E:93:DD:3A:0B:9A:72:40:16:35:34:D2:22:E1:25:07:80
  ```

- `escadm keystore export [--out <file path>]`

  The export command displays the full content of the certificate. If the `out` option is specified, the content is saved in a file whose path is specified in the previous option.

- `escadm keystore set-- file <file path>`

  The set command replaces the current certificate with a new one residing inside the file whose path is specified in the option `file`.

  Ensure that the file is in PEM format, containing both the certificate and a private key.

  The following example shows how to generate a new self-signed certificate and set it to the keystore:

  1. Use the following command to generate the certificate:

     ```
     openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem -out
     cert.pem
     ```

     The previous commands gives the following two files:

     `key.pem` and `cert.pem`

  2. Use the following commands to combine the two files:

     ```
     cat key.pem > server.pem
     ```

     ```
     cat cert.pem >> server.pem
     ```

  3. To set the new certificate to the keystore, use the following command:

     ```
     $ sudo escadm keystore set --file server.pem
     ```

     ```
     Service "keystore" successfully updated ESC keystore and will take effect once ESC
     services are restarted by running "sudo escadm restart"
     ```

     Note Ensure that you delete any left over files containing private keys and certificate after the setting is done.

     It is mandatory to restart your system for the change to take effect. Use the following command to restart in standalone mode or H/A mode:

     ```
     sudo escadm restart
     ```

If the ESC is running in the Active/Active mode, you must restart all the nodes in the same cluster. However, if the ESC is running in an Active/Active GEO mode, you must stop the GEO service on all the clusters, to make sure a GEO switchover does not take place.

Only in GEO Active/Active mode, to stop GEO service, log in to any node in each cluster and use the following command:

```
$ sudo escadm geo stop --cluster
```

You can login to any node in the cluster where the certificate is originally set. Use the following command:

```
$ sudo escadm stop --cluster
$ sudo escadm start --cluster
```

If you are on a GEO Active/Active mode, once all the node are up and running in passing state, login again to any node to start the GEO service. Use the folowing command to start the service:

```
$ sudo escadm geo start --cluster
```

Using a Bootable Volume in ESC Installation

A volume in OpenStack is a detachable block storage device that can be attached to an ESC instance. You can store and also run ESC instances from a volume.

Note

- Only one ESC instance can be launched from one volume at a time.
- ESC installation with a combination of bootable volume and high availability (Active/Standby and Active/Active) on cinder is not supported.

To launch an ESC instance from a bootable volume, do the following:

Procedure

Step 1 Create a bootable volume in OpenStack based on an ESC image or from a bootable volume. The bootable volume must be at least of 30 GB disk size. For more information, see OpenStack documentation.

Step 2 Deploy ESC VM using the `bootvm.py` command and choose the `--boot_volume` argument instead of the `--image` argument, as shown below:

```
./bootvm.py <esc_vm_name> --boot_volume <volume_name_or_id> --net <network> --user_pass
<username>:<password>
--user_confd_pass <username>:<password> --flavor <flavor_name>
```

Note

- Only one of these arguments, `--image` or `--boot_volume` must be passed to the bootvm.py command. The installation will fail, if both or none of the arguments are used.
- When launching an ESC instance from a bootable volume, volume disk size is considered over the flavor disk size.
- If an ESC instance is deleted, the volume attached to it will not be deleted, as the volume was created out-of-band.