Cisco Elastic Services Controller 4.4 Install and Upgrade Guide

First Published: 2018-12-21

Last Modified: 2019-01-21

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

PREFACE

Preface ix

Preface ix

Audience ix

Terms and Definitions ix

Obtaining Documentation Request xi

CHAPTER 1

Elastic Services Controller Overview 1

Elastic Services Controller Overview 1

PART I

Installing Cisco Elastic Services Controller on OpenStack 3

CHAPTER 2

Prerequisites 5

Virtual Resource Requirements 5

Software Requirements 5

Preparing for the Installation 6

CHAPTER 3

Installing Cisco Elastic Services Controller on OpenStack 7

Installation Scenarios 7

Main Components of Cisco Elastic Services Controller Setup 8

Installing Cisco Elastic Services Controller Using the QCOW Image 8

Additional Installation Options 11

Managing Root Certificates in Cisco Elastic Services Controller 14

Enabling/Disabling the Root Certificate Validation 15

Adding a Root Certificate 15

Removing a Root Certificate 15

Managing Root Certificates During the Upgrade 16

Using a Bootable Volume in ESC Installation 16

CHAPTER 4 **Configuring High Availability** 17

High Availability Overview 17

How High Availability Works 17

Deploying ESC High Availability 18

Deploying ESC in High Availability Mode on Internal Storage 19

Deploying ESC in High Availability Mode on Replicate External Storage 20

Configuring the Northbound Interface Access 21

Configuring ESC HA with Multiple Interfaces 21

Configuring the ESC HA Virtual IP Address 22

Configuring the ESC L3 HA With BGP 23

Important Notes 26

Troubleshooting High Availability 27

PART II **Installing Cisco Elastic Services Controller on VMware vCenter** 29

CHAPTER 5 **Prerequisites** 31

Virtual Resource and Hypervisor Requirements 31

vCenter Resources 31

Important Notes 32

CHAPTER 6 **Installing Cisco Elastic Services Controller on VMware vCenter** 35

Installing Cisco Elastic Services Controller on VMware vCenter 35

Preparing to Install Cisco Elastic Services Controller 35

Installing the Elastic Services Controller Using the OVA Image 36

Installing Elastic Services Controller Using OVF Tool 38

Powering on Cisco Elastic Services Controller Virtual Machine 40

Next Steps: Cisco Elastic Services Controller Virtual Machine 41

Logging in to Cisco Elastic Services Controller Portal 41

Configuring the Virtual Machine to Automatically Power Up 41

CHAPTER 7 **Configuring High Availability** 43

High Availability Overview 43

How High Availability Works 43
Deploying ESC High Availability with User Data (HA Pair) 44
Deploying ESC High Availability (Standalone Instances) 48
Important Notes for ESC HA 49
Troubleshooting High Availability 49

PART III **Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine (KVM)** 51

CHAPTER 8 **Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine** 53
Installing Cisco Elastic Services Controller in a Kernel-based Virtual Machine 53
Preparing to Install Cisco Elastic Services Controller on a Kernel-based Virtual Machine 53
Installing Elastic Services Controller on a Kernel-Based Virtual Machine 54
Next Steps: Cisco Elastic Services Controller Kernel-based Virtual Machine 55
Logging in to Cisco Elastic Services Controller Portal 55
Verifying ESC installation for a Kernel-based Virtual Machine (KVM) 55
Troubleshooting Tips 56

PART IV **Installing Cisco Elastic Services Controller on Amazon Web Services (AWS)** 57

CHAPTER 9 **Installing Cisco Elastic Services Controller on Amazon Web Services** 59
Prerequisites 59
Installing the Elastic Services Controller Instance in AWS 59

PART V **Installing Cisco Elastic Services Controller on Cisco Cloud Services Platform 2100** 63

CHAPTER 10 **Installing Cisco Elastic Services Controller on Cisco Cloud Services Platform 2100** 65
Prerequisites 65
Installing the Elastic Services Controller Instance in CSP 2100 65
ESC HA Installation 70

PART VI **Post Installation Tasks** 75

CHAPTER 11 **Post Installation Tasks** 77
Changing the ESC Password 77

Changing the ConfD Netconf/CLI Administrator Password Using the Command Line Interface 77
Changing Linux Account Password 78
Changing the ESC Portal Password 78
Configuring Pluggable Authentication Module (PAM) Support for Cisco Elastic Services Controller 79
Authenticating REST Requests 80
REST Authentication 80
Enabling ETSI REST Authentication 80
Changing the REST Interface Password 81
Changing the ETSI REST Interface Password 81
Sending an Authorized REST Request 82
Sending an Authorized ETSI REST Request 82
Configuring Openstack Credentials 83
Reconfiguring ESC Virtual Machine 88
Reconfiguring Rsyslog 88
Reconfiguring NTP 90
Reconfiguring DNS 90
Reconfiguring Hosts 91
Reconfiguring Timezone 91
Verifying ESC Configurations and Other Post-Install Operations 91
Logging in to the ESC Portal 93

PART VII **Upgrading Cisco Elastic Services Controller** 95

CHAPTER 12 **ESC in Maintenance Mode** 97
Setting ESC in a Maintenance Mode 97
Using the escadm Tool 97
Setting ESC in an Operation Mode 98
Backup the Database from the ESC Standalone Instances 98
Backup the Database from the ESC HA Instances 99
Restoring ESC Database 101

CHAPTER 13 **Upgrading Cisco Elastic Services Controller** 103

Upgrading Standalone ESC Instance 104
Deploy the ESC for Upgrade 104
Restoring the ESC Database 105
Important Notes: 105
Upgrading ESC HA Instances 106
Deploying the ESC HA nodes for Upgrade 106
Restoring the ESC Database on New Master and Standby ESC Instances 106
Upgrading VNF Monitoring Rules 107
In-Service Upgrade of the ESC HA Nodes in OpenStack 108
In-Service upgrade in OpenStack using ESC RPM packages 108
In-Service upgrade in OpenStack using ESC qcow2 Image 108
In-Service Upgrade of the ESC HA Nodes in Kernel-Based Virtual Machine (KVM) 111
In-Service Upgrade in KVM using ESC RPM packages 111
In-Service Upgrade in KVM using ESC OVA Image 112
In-Service Upgrade of the ESC HA Nodes in VMware 115
In-Service upgrade in VMware using ESC RPM packages 115
In-Service upgrade in VMware using ESC qcow2 Image 116

PART VIII **Troubleshooting Cisco Elastic Services Controller Installation** 119

CHAPTER 14 **Troubleshooting ESC Issues** 121
ESC System Logs 121
General Installation Errors 126

APPENDIX A **Cisco Elastic Services Controller Installer Arguments** 131

Preface

- Preface, on page ix

Preface

The Cisco Elastic Services Controller Install and Upgrade Guide describes the installation requirements, the installation procedures, and the upgrade procedures for Cisco Elastic Services Controller.

This preface contains the following sections:

Audience

This guide is for network administrators who are installing, provisioning, configuring, and monitoring Virtual Network Functions (VNFs). ESC can be deployed on OpenStack, VMware vCenter, KVM and AWS. The administrator must be familiar with the VIM layer, VMware, and OpenStack resources, and the commands used.

Terms and Definitions

The below table defines the terms used in this guide.

Table 1: Terms and Definitions

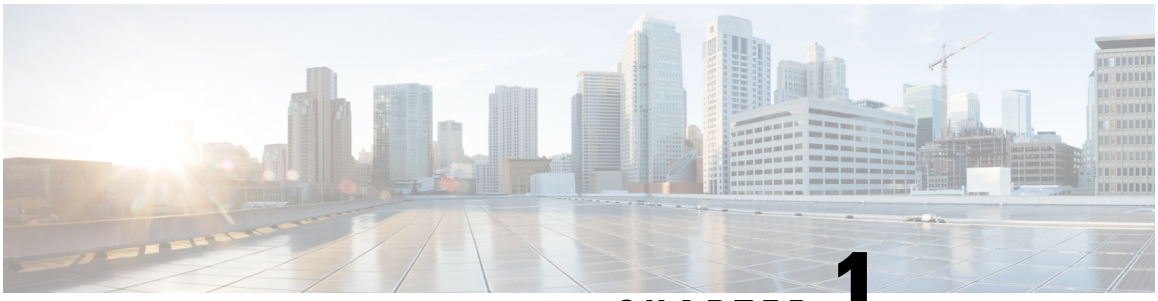
Terms	Definitions
AWS	Amazon Web Services (AWS) is a secure cloud services platform, offering compute, database storage, content delivery and other functionalities.
ESC	Elastic Services Controller (ESC) is a Virtual Network Function Manager (VNFM), performing lifecycle management of Virtual Network Functions.
ETSI	European Telecommunications Standards Institute (ETSI) is an independent standardization organization that has been instrumental in developing standards for information and communications technologies (ICT) within Europe.

Terms	Definitions
ETSI Deployment Flavour	A deployment flavour definition contains information about affinity relationships, scaling, min/max VDU instances, and other policies and constraints to be applied to the VNF instance. The deployment flavour defined in the VNF Descriptor (VNFD) must be selected by passing the *flavour_id* attribute in the InstantiateVNFRequest payload during the instantiate VNF LCM operation.
HA	ESC High Availability (HA) is a solution for preventing single points of ESC failure and achieving minimum ESC downtime.
KPI	Key Performance Indicator (KPI) measures performance management. KPIs specify what, how and when parameters are measured. KPI incorporates information about source, definitions, measures, calculations for specific parameters.
MSX	Cisco Managed Services Accelerator (MSX) is a service creation and delivery platform that enables fast deployment of cloud-based networking services for both Enterprises and Service Providers customers.
NFV	Network Function Virtualization (NFV) is the principle of separating network functions from the hardware they run on by using virtual hardware abstraction.
NFVO	NFV Orchestrator (NFVO) is a functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.
NSO	Cisco Network Services Orchestrator (NSO) is an orchestrator for service activation which supports pure physical networks, hybrid networks (physical and virtual) and NFV use cases.
OpenStack Compute Flavor	Flavors define the compute, memory, and storage capacity of nova computing instances. A flavor is an available hardware configuration for a server. It defines the *size* of a virtual server that can be launched.
Service	A service consists of a single or multiple VNFs.
VDU	The Virtualisation Deployment Unit (VDU) is a construct that can be used in an information model, supporting the description of the deployment and operational behaviour of a subset of a VNF, or the entire VNF if it was not componentized in subsets.
VIM	The Virtualized Infrastructure Manager (VIM) adds a management layer for the data center hardware. Its northbound APIs are consumed by other layers to manage the physical and virtual resources for instantiation, termination, scale in and out procedures, and fault & performance alarms.
VM	A Virtual Machine (VM) is an operating system OS or an application installed on a software, which imitates a dedicated hardware. The end user has the same experience on a virtual machine as they would have on dedicated hardware.
VNF	A Virtual Network Function (VNF) consists of a single or a group of VMs with different software and processes that can be deployed on a Network Function Virtualization (NFV) Infrastructure.
VNFM	Virtual Network Function Manager (VNFM) manages the life cycle of a VNF.

Obtaining Documentation Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*, at: http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.



CHAPTER 1

Elastic Services Controller Overview

• Elastic Services Controller Overview, on page 1

Elastic Services Controller Overview

Cisco Elastic Services Controller (ESC) is a Virtual Network Functions Manager (VNFM), which performs life cycle management of Virtual Network Functions (VNFs). ESC provides agent-less and multi vendor VNF management by provisioning virtual services, and monitoring their health and load. ESC provides the flexibility to define monitoring rules, and associate actions to be triggered based on the outcome of these rules. As a VNFM, in addition to the typical life cycle management operations, ESC also supports automatic VM recovery when a VM fails and performs automatic scaling in and out functions. ESC fully integrates with Cisco and other third party applications.

- As part of the Cisco Orchestration Suite, ESC is packaged with Cisco Network Services Orchestrator (NSO), and available within Cisco Solution, Managed Services Accelerator Solution (MSX).
- As a standalone product, ESC is available as a Virtual Network Function Manager for several Cisco VNFs such as CSR1K, ASAv, WSA and many others.

ESC is deployed in a virtual machine within OpenStack, VMware vCenter, KVM or AWS and manages its VNFs in a Virtual Infrastructure Manager (VIM).

ESC fully integrates with Cisco and other third party applications. As a standalone product, the Elastic Services Controller can be deployed as a VNF Manager. ESC integrates with Network Services Orchestrator (NSO) to provide VNF management along with orchestration. Elastic Services Controller as a VNF Manager targets the virtually managed services and all service provider NFV deployments such as virtual video, WiFi, authentication and others.

ESC can manage both basic and complex VNFs . Basic VNFs include a single VM such as a vFW, vRouter and others.

Complex VNFs include multiple VMs that are orchestrated as a single entity with dependencies between them.

IPv6 Support

Elastic Services Controller provides IPv6 support on OpenStack for:

- VNF Management
- HA— ESC manages VNFs on IPv4 and IPv6 (OpenStack and KVM only).

Elastic Services Controller provides IPv6 support for northbound interface (for example, NFVO to VNFM), and southbound interface (for example, VNFM to VNF). In order to support both northbound and southbound IPv6 concurrently, the following pre-requisites must be met:

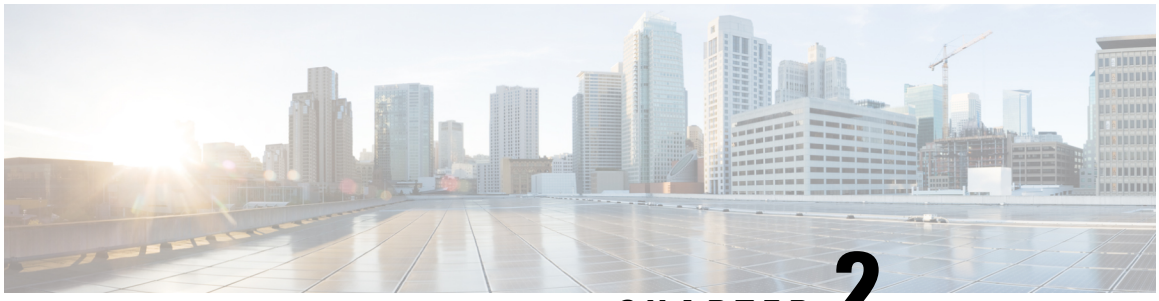
- OpenStack cloud computing is set up and configured for ipv6, including the endpoints (that are ipv6 based).
- The OpenStack cloud computing must contain a Controller, endpoints, and a few Compute hosts, with an ipv6 management and os_api based networks.
- The ESC default security group rules support the IPv6 traffic.

Note When you are deploying a VM, you can attach an out-of-band port of an IPv6 subnet to a VM. However, if you are deleting this VM, you cannot attach the same IPv6 address to another VM due to a known OpenStack issue.

PART I

Installing Cisco Elastic Services Controller on OpenStack

- Prerequisites, on page 5
- Installing Cisco Elastic Services Controller on OpenStack, on page 7
- Configuring High Availability, on page 17



CHAPTER 2

Prerequisites

The following sections detail the prerequisites for installing Cisco Elastic Services Controller:

- Virtual Resource Requirements, on page 5
- Software Requirements, on page 5
- Preparing for the Installation, on page 6

Virtual Resource Requirements

The following table lists the virtual resource requirements for Cisco Elastic Services Controller:

Requirements (for 1000 VNFs/VM management)	Description
Virtual CPUs	4 VCPUs
Memory	8 GB RAM
Disk Space	30 GB

Software Requirements

The following table lists the software requirements on OpenStack:

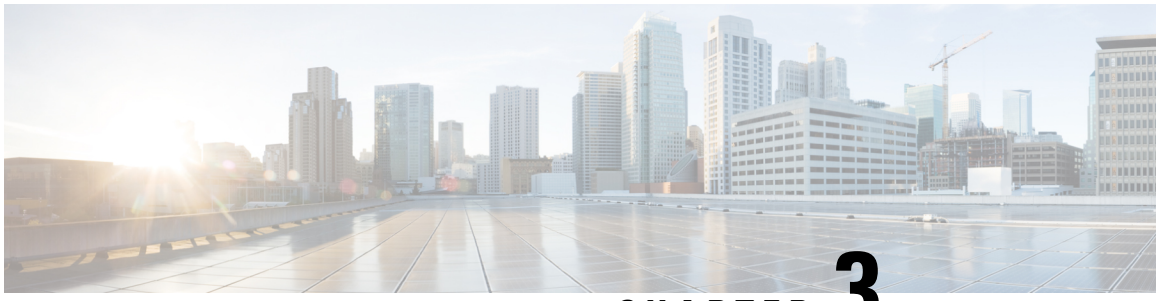
Requirements	Description
Supported Browsers	Any of the following: • Mozilla Firefox 40.x or later • Google Chrome 44.x or later

Requirements	Description
OpenStack Version	Any of the following: • Newton • Ocata • Queens • Keystone V2 and V3

Preparing for the Installation

Before you perform the installation, ensure that you are prepared by reviewing this checklist:

Requirements	Your Information/ Notes
For Pre-installation Configuration	
QCOW image location	
QCOW image	
VM per Instance	
IP address	
Subnet mask	
Hostname	
Domain name	
Gateway IP address	
Admin password	
ESC Release Package	
ESC.qcow2	An image file for booting up the ESC instance
bootvm.py	The installation script compatible with python 2.7.6 and Python 3.4



CHAPTER 3

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, on page 7
- Main Components of Cisco Elastic Services Controller Setup, on page 8
- Installing Cisco Elastic Services Controller Using the QCOW Image, on page 8
- Managing Root Certificates in Cisco Elastic Services Controller, on page 14
- Using a Bootable Volume in ESC Installation, on page 16

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 an Active-Passive model. Two ESC instances are deployed in the network to prevent ESC failure and to provide ESC service with minimum interruption. If the primary 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

For more information on deploying ESC HA, see 'Configuring High-Availability' in Installing ESC on OpenStack and Installing ESC on VMware chapters.

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, on page 5.
- You have the information identified in Preparing for the Installation, on page 6.
- Copy the ESC image file on the system where you want to install ESC.
- This system must be accessible by OpenStack.

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 30GB disk space. ESC installation script takes the pre-defined "m1.large" flavor which has the definition of 4vCPU, 8G RAM and 80G disc space. To use 30GB disk space, create a flavor with the minimum disk space requirement.

```
$ nova flavor-create ESC_FLAVOR_NAME ESC_FLAVOR_ID 8192 30 4
```

Step 8 To deploy ESC VM, do the following:

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

   ```
   $ nova net-list
   ```

2. 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>
```

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 key are generated as part of a key pair with, 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.

3. 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

- **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.

- **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.

- **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-rest . . .

OR

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

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

- **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 two instances of ESC to build an ESC HA pair**: For more information on deploying ESC HA, 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/confd_cli  -u admin
  $ 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 master 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 during the installation as the keepalived service is not running when a certificate is added.

- Add certificate for standalone/HA after booting up the ESC instance. The escadm tool has an **add-certificate** 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 master VM .
  2. Add certificate to ESC truststore. To do this, execute the following command:

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

  3. Verify the certificate is added.

     ```
     sudo escadm list-certificate
     ```

Removing a Root Certificate

The escadm tool has a 'delete-certificate' 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 ESC VM:

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

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

Step 2 Verify the certificate is removed.

```
sudo escadm list-certificate
```

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.

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 on cinder is not supported.

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

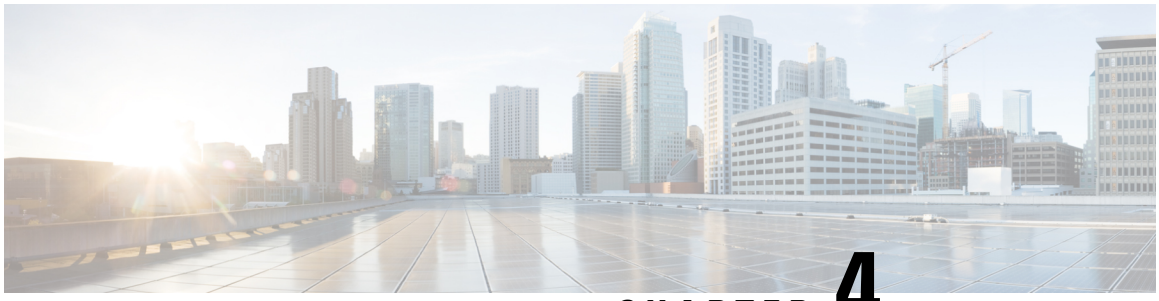
Step 1 Create a bootable volume in OpenStack based on an ESC image or from a bootable volume. The bootable volume must be at least of 30 GB disk size. For more information, see OpenStack documentation.

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.



CHAPTER 4

Configuring High Availability

This chapter contains the following sections:

- High Availability Overview, on page 17
- How High Availability Works, on page 17
- Deploying ESC High Availability, on page 18
- Configuring the Northbound Interface Access, on page 21
- Important Notes, on page 26
- Troubleshooting High Availability, on page 27

High Availability Overview

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

- Network failures
- Power failures
- Dead VM instance
- Scheduled downtime
- Hardware issues
- Internal application failures

How High Availability Works

ESC HA network can be either set up as a single installation of a ESC HA pair or deployed as two standalone ESC nodes that are converted into HA pair after re-configuring these nodes post deployment. A HA deployment consists of two ESC instances: a primary and a standby. Under normal circumstances, the primary ESC instance provides the service. The corresponding standby instance is passive. The standby instance is in constant communication with the primary instance and monitors the primary instances' status. If the primary ESC instance fails, the standby instance automatically takes over the ESC services to provide ESC service with minimum interruption.

The standby also has a complete copy of the database of the primary, but it does not actively manage the network until the primary instance fails. The KeepAliveD service monitors both primary and standby instances activity status. When the primary instance fails, the standby takes over automatically. The standby instance takes over primary instance to manage the services while primary instance restoration is taking place.

When the failed instance is restored, if required you can manually initiate a switch-over and resume network management via the primary instance.

Both primary and standby ESC instances are connected to the northbound orchestration system through an IPv4 or IPv6 network. For the northbound system, a unique virtual IP address is assigned to access the current primary ESC High Availability instance. The deployed VNFs are connected to both ESC primary and standby instances through another IPv6 network.

ESC HA nodes are managed by KeepAliveD and DRBD (Replication tool to keep the ESC database synchronized) sync network services. While the KeepAliveD service monitors both primary and standby instances status, the DRBD service monitors primary instance DB and sync the changes to the standby instance DB. These two services can be co-located on same VIP network or in two separate networks. VM handshake between ESC instances occurs through the KeepAliveD over the IPv4 or IPv6 network.

Deploying ESC High Availability

To deploy Cisco Elastic Services Controller (ESC) High Availability (HA), ESC standalone instances can be installed on two separate nodes - Primary and Standby. For more information see, How High Availability Works, on page 17. You can connect the Primary and Standby instances to either a Cinder volume or Replication based volume (DRBD).

The following deployment mechanisms can be used to deploy ESC HA:

- Internal Storage—When ESC HA is configured with Internal storage, the Primary and the Standby instances have individual databases which are always synchronized. In this solution, ESC HA is designed with database replication and DRBD is used as the tool for disk-level replication. The database in the Primary instance simultaneously propagates the data to the database in the Standby instance thus requiring no external storage. In the event of a Primary instance failing, the Standby instance get assigned the role of the Primary instance along with its own synchronized database.

  ESC HA is deployed using Internal storage, the ESC instances reply on the virtual IP address (that is kad_vip argument), and the interface of vrrp instance (that is kad_vif argument) to select the Primary ESC instance. To establish a reliable heartbeat network, it is recommended that the Primary and Standby ESC instances are on different physical hosts. The reliability of the physical links between the ESC instances (such as, network interface bonding) can also be taken into consideration.

- Replicate External-Storages — In this type of architecture, ESC HA is configured with DRBD and both Primary and Standby instance store their data in two external storages (OpenStack Cinder volumes). Each ESC node is attached by a Cinder volume and ESC data files are stored in the cinder volume. The data in two ESC node are synchronized through the database replication mechanism provided by DRBD.

The table lists the differences between the HA options :

	Internal Storage Based ESC HA	**Replicate External Storage Based ESC HA**
Data sharing method	Data replication between HA nodes	Data replication between two external storages (cinder volume)

Installation Method	Post-installation Configuration Bootvm Installation	Bootvm Installation
VIM Support	OpenStack, VMware, KVM	OpenStack only
Dependency	VIM independent	Rely on OpenStack cinder
Advantages	• No dependency on specific VIM components. • Flexible to build of HA clusters from commodity hardware, without the requirement for shared-storage.	• Use database replication mechanism for data synchronization • Two cinder volumes are used as external storage and are attached to ESC node.
Limitations	The data consistency may be affected in a double fault condition (occurs when both ESC nodes have problems).	The data consistency may be affected in a double fault condition (occurs when both ESC nodes have problems).

Deploying ESC in High Availability Mode on Internal Storage

When you boot ESC instances on Primary and Standby instances, you need to specify the following *bootvm.py* command arguments to deploy ESC HA on an internal storage:

- kad_vip

Note When ESC HA is deployed, the *kad_vip* argument allows end users to access the Primary ESC instance.

- kad_vif
- ha_node_list

These arguments enable the *bootvm.py* command to automatically set up the internal storage on the OpenStack. For more information on using the *bootvm.py* command arguments, see Appendix A: Cisco Elastic Services Controller Installer Arguments.

To deploy ESC HA instances, use the bootvm script on both the nodes with the following arguments:

```
ON HA NODE 1:

$ ./bootvm.py <ESC_HA_Node1>\
--user_pass <username>:<password>\
--user_confd_pass <username>:<password>\
--gateway_ip <default gateway IP address>\
--net <network name1>\
--ipaddr <static ip address>
--image <image_name>\
--avail_zone nova:<openstack zone>\
--ha_node_list=<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>\
--db_volume_id <cinder volume id>\
--kad_vip <virtual IP address>\
```

```
--kad_vif <VRRP_Interface_Instance>\
--ha_mode drbd


ON HA NODE 2:

$ ./bootvm.py <ESC_HA_Node2>\
--user_pass <username>:<password>\
--user_confd_pass <username>:<password>\
--gateway_ip <default gateway IP address>\
--net <network name1>\
--ipaddr <static ip addresses>\
--image <image_name>\
--avail_zone nova:<openstack zone>\
--ha_node_list=<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>\
--db_volume_id <cinder volume id>\
--kad_vip <virtual IP address>\
--kad_vif <VRRP_Interface_Instance>\
--ha_mode drbd
```

OR

You can also use **escadm** tool to re-configure ESC HA parameters on each of the standalone ESC VMs. Three parameters "--ha_node_list , --kad_vip, --kad_vif" are all required to configure ESC HA. For example:

```
$ sudo bash
$ escadm ha set --ha_node_list='<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>' --kad_vip <virtual IP
 address> --kad_vif <VRRP_Interface_Instance>
$ sudo escadm restart
```

Deploying ESC in High Availability Mode on Replicate External Storage

Replicate external storage ESC HA requires two cinder volumes for database storage.

Before you begin

- Networks and IP addresses that both ESC instances will connect to
- Keepalived interface and virtual IP for HA switchover

Step 1 Create two cinder volumes in OpenStack. The configured cinder volume size should be 3GB.

```
$ cinder create --display-name  cindervolume_name_a[SIZE]
$ cinder create --display-name  cindervolume_name_b[SIZE]
```

Step 2 Check the status of the created cinder volume and find the uuids for deployment.

```
$ cinder list
```

Step 3 Deploy ESC HA instances. Use the bootvm script on both the nodes with the following arguments:

```
ON HA NODE 1:

$ ./bootvm.py <ESC_HA_Node1>\
--user_pass <username>:<password>\
--user_confd_pass <username>:<password>\
--gateway_ip <default gateway IP address>\
```

```
--net <network name1>\
--ipaddr <static ip address>\
--image <image_name>\
--avail_zone nova:<openstack zone>\
--kad_vip <virtual IP address>\
--kad_vif <VRRP_Interface_Instance>\
--ha_node_list=<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>\
--db_volume_id <cinder volume id>\
--ha_mode drbd_on_cinder

ON HA NODE 2:

$ ./bootvm.py <ESC_HA_Node2>\
--user_pass <username>:<password>\
--user_confd_pass <username>:<password>\
--gateway_ip <default gateway IP address>\
--net <network name1>\
--ipaddr <static ip address>\
--image <image_name>\
--avail_zone nova:<openstack zone>\
--kad_vip <virtual IP address>\
--kad_vif <VRRP_Interface_Instance>\
--ha_node_list=<ESC_HA_NODE1_IP> <ESC_HA_NODE2_IP>\
--db_volume_id <cinder volume id>\
--ha_mode drbd_on_cinder
```

Step 4 After both VMs are rebooted; the keepalived state on one of ESC VM should be one of ESC VM should be in MASTER state and the other one should be in BACKUP state. You can check ESC HA state by using following command: *$ sudo escadm status --v*.

Configuring the Northbound Interface Access

When you configure ESC HA, you can also specify a virtual Anycast IP address to the HA pair. The northbound interface as well as the service portal uses virtual Anycast IP address to access the ESC Primary HA instance. When deploying ESC HA, use the following arguments with the *./bootvm.py* script.

- --ha_node_list
- --kad_vip
- --kad_vif

For more details on these arguments, see section **Appendix A: Cisco Elastic Services Controller Installer Arguments**.

The following section explains how to configure ESC HA with multiple interfaces and to configure the virtual Anycast IP address.

Configuring ESC HA with Multiple Interfaces

You can configure ESC HA with DRDB synchronization and VRRP heartbeat broadcasting on a network interface for data synchronization and VNF monitoring. You can use an additional network interface to allocate Virtual IP for the northbound access. To configure the multiple interfaces on ESC HA nodes, use --ha_node_list, --kad_vip, --kad_vif arguments to specify these multiple network interfaces configuration. For details on these arguments, see section **Appendix A: Cisco Elastic Services Controller Installer Arguments**.

Example configuration steps are shown below:

```
./bootvm.py <esc_ha1> \
--user_pass <username>:<password>
--user_confd_pass <username>:<password>
--image <image_id> \
--net <net-name> \
--gateway_ip <default_gateway_ip_address> \
--ipaddr <ip_address1> <ip_address2> \
--ha_node_list < IP addresses HA nodes1> < IP addresses for HA nodes2> \
--kad_vip <keepaliveD VIP of the HA nodes and the interface for keepaliveD VIP> \  (for
example: --kad_vip 192.0.2.254:eth2)
--kad_vri <virtual router id of vrrp instance>
--kad_vif <virtual IP of the HA nodes or the interface of the keepalived VRRP> \ (for
example: --kad_vif eth1 )
--ha_mode <HA installation mode> \
--route <routing configuration> \  (for example:192.0.2.254/24:192.168.0.1:eth1 )
--avail_zone nova:<openstack zone> \
```

Similarly, a three network interface can be configured for ESC HA nodes. An example three interfaces configuration is shown below with the following assumptions :

- Network 1 is an IPv6 network used for northbound connection. ESC VIP is allocated in this network and the Orchestrator send requests to ESC through ESC VIP.
- Network 2 is an IPv4 network used for ESC sync traffic (DRDB synchronization) and VRRP heartbeat. This network is also used for OpenStack connection and VNF monitoring.
- Network 3 is another IPv4 network used for management. The SA, rsyslog, etc. can use this network to manage ESC.

./bootvm.py esc-ha-0 --image ESC-2_2_x_yyy --net network-v6 network --gateway_ip 192.168.0.1 --ipaddr 2001:cc0:2020::fa 192.168.0.239 192.168.5.239 --ha_node_list 192.168.0.239 192.168.0.243 --kad_vip [2001:cc0:2020::fc/48]:eth0 --kad_vif eth1 --ha_mode drbd --route 172.16.0.0:eth1 --avail_zone nova: zone name

./bootvm.py esc-ha-1 --image ESC-2_2_x_yyy --net network-v6 network lab-net-0 --gateway_ip 192.168.0.1 --ipaddr 2001:cc0:2020::fa 192.168.0.239 192.168.5.239 --ha_node_list 192.168.0.239 192.168.0.243 --kad_vip [2001:cc0:2020::fc/48]:eth0 --kad_vif eth1 --ha_mode drbd --route 172.16.0.0:eth1 --avail_zone nova: zone name

Configuring the ESC HA Virtual IP Address

In this option, the value of kad_vip argument should be a virtual IP, which allows the service portal and the northbound to access the Primary ESC and send requests to ESC HA service through virtual IP (VIP).

If northbound and both ESC HA nodes are located in the same network, you can connect directly through the virtual IP (VIP). If northbound doesn't sit on the same network as ESC HA, assign a floating IP to ESC HA VIP using the procedure below:

1. Create a port with the VIP address (kad_vip) in the same network as ESC's kad_vip connects.

   ```
   neutron port-create network --name network_vip --fixed-ip
   subnet_id=network-subnet,ip_address=192.168.0.87
   ```

2. Deploy ESC HA . See **Configuring High-Availability** section in Installing ESC on OpenStack.

Note Make sure the *kad_vip* using the same IP address as the port created above.

3. Associate a floating IP with the port created above. The first uuid is the floating ip id and the second one is the port id.

   ```
   neutron floatingip-associate <floating IP> <port ID>
   ```

   Access ESC HA through the floating IP and it will connect to the ESC Primary node.

4. For the portal access, make sure the keepalive network is accessible by your browser and the virtual IP is the IP address to access the portal of the Primary node.

   For example, if the VIP is 192.0.2.254, access ESC HA portal with https://192.0.2.254:9001/.

Configuring the ESC L3 HA With BGP

To configure BGP for ESC HA, there are two options:

1. Directly booting ESC HA L3 with BGP
2. Using post configuration from existing ESC HA pair

To configure BGP for ESC HA, the following network parameters are required:

- BGP remote IP
- IP of the interface for BGP anycast routing
- BGP local AS number for routing configuration
- BGP remote AS number for routing configuration
- BGP routing configuration
- --bgp_local_ip
- --bgp_local_router_id

Note You must configure BGP router with neighbors, and restart it. Verify that the router is able to ping the AnyCast IP.

On the BGP router, set two neighbors. The below BGP configuration is designed for Bird router. The configuration is router specific. For each types of router, the procedure is different:

The below configurations are given according to the bootvm command :

```
protocol bgp E3 from EXABGP {
  neighbor 198.18.42.222 as 65012;
}

protocol bgp E4 from EXABGP {
  neighbor 198.18.61.222 as 65011;
}
```

Booting an ESC VM with BGP options

```
   ######################################################################
   #   ESC on bgp-001.novalocal is in MASTER state.
   ######################################################################

[admin@bgp-001 ~]$ health.sh
============== ESC HA (MASTER) with DRBD =================
vimmanager (pgid 4007) is running
monitor (pgid 4135) is running
mona (pgid 4167) is running
drbd (pgid 0) is master
snmp (pgid 5375) is running
etsi is disabled at startup
pgsql (pgid 4586) is running
keepalived (pgid 3068) is running
portal (pgid 5315) is running
confd (pgid 4417) is running
filesystem (pgid 0) is running
escmanager (pgid 4615) is running
=======================================
ESC HEALTH PASSED
[admin@bgp-001 ~]$


   ######################################################################
   #   ESC on bgp-002.novalocal is in BACKUP state.
   ######################################################################

[admin@bgp-002 ~]$ health.sh
============== ESC HA (BACKUP) with DRBD =================
vimmanager is stopped
monitor is stopped
mona is stopped
drbd (pgid 0) is backup
snmp is stopped
etsi is disabled at startup
pgsql is stopped
keepalived (pgid 3069) is running
portal is stopped
confd is stopped
filesystem is stopped
escmanager is stopped
=======================================
ESC HEALTH PASSED
[admin@bgp-002 ~]$
```

Use below values for BGP post configuration:

```
./bootvm.sh <NETWORK_VM_name> \
--image <ESC_image> \
--ipaddr <static_IP_address1> <IP_address2> <IP_address_3>\
--gateway_ip <gateway IP address of NETWORK> \
--net <net_id1> <net_id2> <net_id3> \
--network_params_file <network_params_file> \
--host_mapping_file <host_mapping_file> \
--avail_zone <openStack_zone> \
--bgp_remote_ip <BGP_remote_IP_address> \
--bgp_local_as <BGP_local_AS_#> \
--bgp_remote_as <BGP_remote_AS_#>\
--bgp_local_router_id <local_BGP_reouter_id> \
--bgp_anycast_ip <BGP_anycast_IP> \
--bgp_md5 <BGP_MD5>
```

Where,

```
--ip_addr: ----> the local IP address of the ESC VM
--net: ----> the network id(s) in OpenStack that ESC will connect to.
--bgp_anycast_ip:  ----> the IP address that NCS will communicate with
--bgp_remote_ip:  ----> this IP address of the external router that ESC will peer with
--bgp_local_as:  ----> local AS for the ESC "router"
--bgp_remote_as:   ----> AS number for the external router ESC will peer with
--bgp_local_router_id: ----> id for the esc "router"
--bgp_md5:    ----> optional - md5 to be used to pair with external router
```

Configuring BGP HA Post Configuration

1. For each HA instance, create the network interface file:

```
# cat /etc/sysconfig/network-scripts/ifcfg-lo:2
IPV6INIT='no'
IPADDR='10.0.124.124'                    <------------------ bgp anycast IP
BROADCAST='10.0.124.255'
NETWORK='10.0.124.0'
NETMASK='255.255.255.0'
DEVICE='lo:2'
ONBOOT='yes'
NAME='loopback'
```

2. For each HA instance:

```
Bring lo:2 up
# ifup lo:2
```

To configure BGP for ESC HA, use the escadm tool in ESC Virtual Machine, as shown below:

```
$ sudo bash
# escadm bgp set --local_ip LOCAL_IP --anycast_ip ANYCAST_IP --remote_ip REMOTE_IP --local_as
 LOCAL_AS --remote_as REMOTE_AS
--local_router_id LOCAL_ROUTER_ID
# escadm reload
# reboot
```

Example:

```
[root@bgp-001 admin]# escadm bgp set --local_ip 198.18.42.124 --anycast_ip 10.0.124.124
--remote_ip 192.168.0.2 --local_as 65124 --remote_as 65000 --local_router_id 198.18.42.124

[root@bgp-002 admin]# escadm bgp set --local_ip 198.18.42.125 --anycast_ip 10.0.124.124
--remote_ip 192.168.0.2 --local_as 65114 --remote_as 65000 --local_router_id 198.18.42.125
```

Configuring a BGP Router

To configure a BGP router, log in to the BGP router to configure BGP Anycast routing. The required parameters are:

`<Router_AS_#>`same as--*bgp_remote_as*above

`<Esc_ip_address>`must be the ESC VM's IP address configured for BGP advertisement.

`<ESC_AS_#>`same as--*bgp_local_as*shown above

```
configure

router bgp <Router_AS_#>

neighbor <ESC_IP_address>

remote-as <ESC_AS_#>
  address-family ipv6 unicast
   route-policy anycast-in in
   route-policy anycast-out out
```

```
route-policy anycast-in
  pass
end-policy

route-policy anycast-out
  drop
end-policy

commit
```

Important Notes

- **ESC HA**
  - An HA failover takes about 2 to 5 minutes. The ESC service will not be available during the switchover time.
  - When the switchover is triggered during transactions, all incomplete transactions will be dropped. The requests should be re-sent by northbound if it does not receive any response from ESC.
- **External Storage**
  - If the Primary ESC instance is suspended by OpenStack command, the switch over will be triggered but the cinder volume won't be attached to the new Primary ESC instance. This is not a valid use case for ESC HA.
- **Internal Storage**
  - Two ESC instances have to be deployed to establish the HA solution. The ESC HA will start to work when both ESC instances are successfully deployed and are able to connect to each other. If you just deploy one ESC instance with HA parameters, the ESC instance keeps Switching-to-Master state and will not be able to provide any service until it reaches its peer.
  - Split-brain scenario can still happen in this ESC HA solution, although the chance is very low.
- **ETSI-specific Notes**

  ESC supports ETSI MANO northbound API defined by the European Telecommunications Standards Institute (ETSI) for NFV Management and Orchestration. The ETSI MANO API is another programmatic interface based on the REST architecture. For more information, see ETSI MANO Compliant Lifecycle Operations in the *Cisco Elastic Services Controller User Guide*. Consider the following notes while enabling ETSI service on ESC which is in HA mode:

  - The server.address value in the *etsi-vnfm.properties* file must be set to a Virtual IP (VIP) address. This IP address can be used to communicate back to the ETSI services using API callbacks. If the virtual IP address is not specified, the ETSI service startup may fail.
  - The ETSI VNFM service and the escadm script generate and maintain the security.user.name and security.user.password property values. You should not change it manually.The security.user.password is encoded.

Troubleshooting High Availability

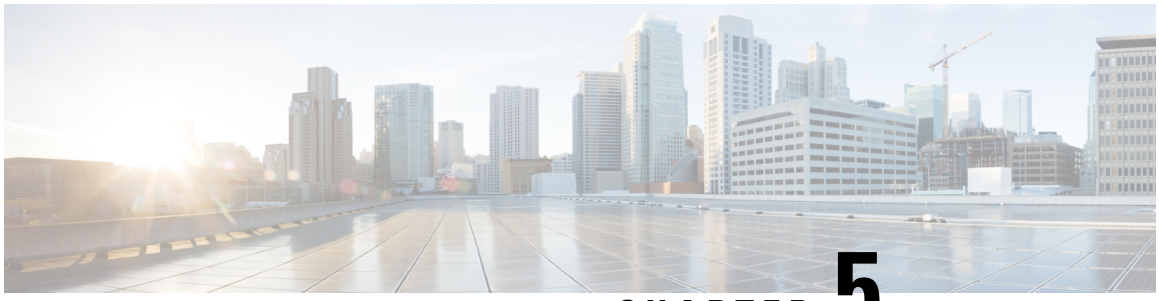
- Check for network failures. If a network problem occurs, you must check the following details:
  - The IP address assigned is correct, and is based on the OpenStack configuration.
  - The gateway for each network interface must be pingable.
- Check the logs for troubleshooting:
  - The ESC Admin logs at */var/log/esc/escadm.log*
  - The ESC manager log at */var/log/esc/escmanager.log*
  - The ESC HA log at */var/log//esc/esc_haagent.log*
  - The KeepAliveD log at */var/log/messages* by grep keepalived
- Check for DRBD (Replication based ESC HA) for Internal Storage solution:
  - Check the DRBD configuration file at

    `/etc/drbd.d/esc.res`
  - Access the DRBD log
  - `/var/log/messages|grep drbd`

PART II

Installing Cisco Elastic Services Controller on VMware vCenter

- Prerequisites, on page 31
- Installing Cisco Elastic Services Controller on VMware vCenter, on page 35
- Configuring High Availability, on page 43



CHAPTER 5

Prerequisites

The following sections detail the prerequisites for installing Cisco Elastic Services Controller:

- Virtual Resource and Hypervisor Requirements, on page 31
- vCenter Resources, on page 31
- Important Notes, on page 32

Virtual Resource and Hypervisor Requirements

The following table lists the prerequisites to install Cisco Elastic Services Controller on VMware vCenter or vSphere:

See the VMware Compatibility Guide to confirm that VMware supports your hardware platform.

Requirement	Description
System Requirements	
Virtual CPUs	4 VCPUs
Memory	8 GB RAM
Disk Space	30 GB
Hypervisor Requirements	
VMWare vCenter	ESC supports VMware and vCenter versions 5.5, 6.0, and 6.5.

vCenter Resources

Resources to be created/installed on vCenter:

- **Datacenters**: At least one datacenter. For more details, see the **Important Notes** below.
- **Hosts**: Host configuration based on your targeted performance objectives. Each Host under the single vDS must have at least two physical Network Interface Card (NIC) connected, (one for vCenter

Management Interface by default, and the other used to assign to VDS's uplink portgroup). This setup is required for data access across hosts.

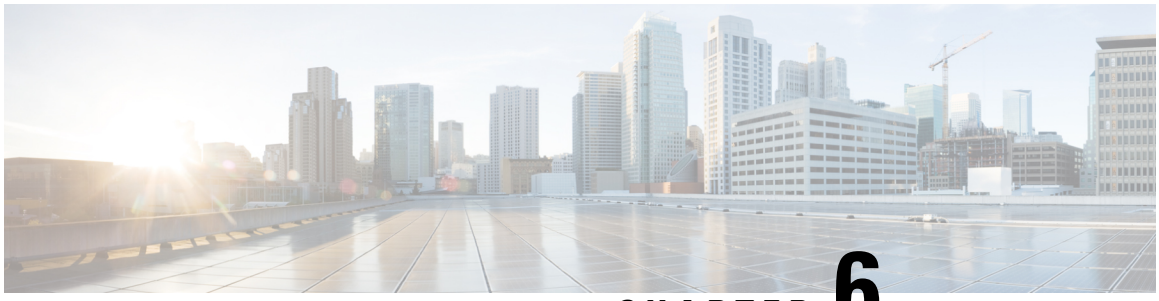
- **Compute Clusters**: Clusters can be created to group several hosts together.
- **Datastores**: Shared datastore is required if user wants to leverage DRS.
- **Distributed Switches**: At least one distributed switch that will contains all the VNF supporting networks.

Important Notes

Keep in mind the following important notes while installing ESC on a VMware:

- A single ESC instance will only support:
  - Multiple Datacenter supported deployment, network, image, subnet creation
  - One vSphere Distributed Switch (VDS)
- DPM, HA, and vMotion must be off.
- If DRS is enabled, it has to be in the "Manual Mode".
- Fault Tolerance is not supported.
- Datastore Cluster is not supported, only flat datastore(s) structure under the cluster or under the datacenter are supported.
- ESC only supports a default resource pool. Adding and creating resource pools are not supported.
- Image (Template) created through ESC are stored under **/esc-ovas** folder .
- Day-0, smart license, and other supported files are packed into a ISO file, and uploaded to the same folder where the VM rest, then mount it as a CD-ROM to the VM.
- ESC/VIM does not respond for the name and file content passed in for generating ISO file. They have to be provided according to each template's requirements. e.g. for ASAv, the day-0 config has to be named as "day0-config", and smart license token has to be named as "idtoken".
- 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. See the Troubleshooting guide , page 91 to increase the timeout for rollback.
- The following VM features and operations are not supported in all versions of the Cisco CSR 1000V. If still these operations are used or performed, there may be risk of encountering dropped packets, dropped connections, and other error statistics.
  1. DRS
  2. Suspend
  3. Snapshot
  4. Resume

- Although deployments can be processed without shared storage, ESC does not guarantee optimized computing resource. Shared storage(s) should associate with as many as possible hosts, which will give more opportunity to DRS to balance resources.
- Every time a redeploy happens as part of recovery on VMware, VM's interface(s) will have different MAC addresses.
- All the VM group defined in a datamodel must accompany with a "zone-host" placement policy, meaning the deployment has to be either host-targeted or cluster-targeted.
- Recovery may fail, if a VM has PCI/PCIe passthrough device(s) attached, when it's recovered to a computing-host (picked based on ESC placement algorithm) which does not have any PCI/PCIe passthrough enabled device available .
- For PCI/PCIe passthrough working, DRS has to be off .
- If you experience a PowerOn error on a VM that has PCI/PCIe passthrough device(s) attached to it, update the VM or the image (template) the VM is cloned from, using the solution described here.



CHAPTER 6

Installing Cisco Elastic Services Controller on VMware vCenter

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

- Installing Cisco Elastic Services Controller on VMware vCenter, on page 35
- Next Steps: Cisco Elastic Services Controller Virtual Machine, on page 41

Installing Cisco Elastic Services Controller on VMware vCenter

Cisco Elastic Services Controller can be installed in a VMware ESXi hypervisor and can be accessed or managed using vSphere client of VMware. You can install Cisco Elastic Services Controller in a VMware environment using an Open Virtual Appliance (OVA) package.

The VMware vSphere client can be connected directly to your ESXi installation, or it can be connected to a vCenter server which in turn is connected to your vSphere installation. Connecting through vCenter provides a number of capabilities that connecting directly to ESXi does not. If a vCenter server is available and associated with the ESXi installation, it should be used.

Preparing to Install Cisco Elastic Services Controller

In order to install Cisco Elastic Services Controller and configure its network connection, you have to answer several questions. Some of these questions concern the networking environment in which the virtual machine is being installed, and some of them concern values which are unique to the particular virtual machine being installed.

Before you perform the installation, ensure that you are prepared by reviewing this checklist:

Requirements	Your Information/ Notes
OVA image location	
OVA image	
VSphere Web Client	
Hostname	

Requirements	Your Information/ Notes
IP address	
Subnet mask	
Network	
vCenter IP	
vCenter Port	
vCenter credentials	
Datacenter Name	
Datastore Host	
Computer Cluster Name	

Installing the Elastic Services Controller Using the OVA Image

To install Cisco Elastic Services Controller, you must first download the correct installation file.

Using vSphere, connect directly to the ESXi installation or the vCenter server, and select the ESXi installation where the OVA is to be deployed.

This procedure describes how to deploy the Elastic Services Controller OVA image on VMware.

Before you begin

- Set your keyboard to United States English.
- Confirm that the Elastic Services Controller OVA image is available from the VMware vSphere Client.
- Make sure that all system requirements are met as specified in the Chapter 6: Prerequisites.
- Gather the information identified in Preparing to Install Cisco Elastic Services Controller.

Step 1 Using the VMware vSphere Client, log in to the vCenter server.

Step 2 Choose **vCenterHome** > **Hosts and Clusters**. Right click the host where you want to deploy ESC, and then choose **Deploy OVF Template**.

Step 3 In the wizard, provide the information as described in the following table:

Screen	Action
Select source	Select the Elastic Services Controller OVA.
Review details	Review OVF Template details.
Select name and folder	Enter a name and select a folder for the VM.

Screen	Action
Select configuration	Select any one of the following deployment configurations: • Large, 1 Network • Large, 2 Networks • Large, 3 Networks
Select a resource	Select the host or cluster to run the ESC template.
Select storage	Select a location to store the files for the VM and a provisioning type. The storage can be local or shared remote, such as NFS or SAN. You can choose either **Thin provisioned format** or **Thick provisioned format** to store the VM virtual disks.
Select networks	Based on the Network configuration for the deployment selected in the **Select configuration** option, you can allocate the pre-configured networks in vCenter to ESC network interfaces.
Customize template	
Bootstrap Properties	
Username	Administrator username for remote login.
Password	Administrator password.
Host name	VM Hostname.
Network IP	VM IP address.
Network Gateway	Gateway IP address.
Enable Https Rest	Enable external REST interface over HTTPS on port 8443.
Enable Portal startup	Enable Portal startup at port 9001 (for https).
VIM Settings of vCenter Server	
vCenter IP	IP address of vCenter server for VNF deployment .
vCenter Port	Port of the vCenter server.
vCenter Username	Username to access vCenter server.
vCenter Password	Password to access vCenter server.
Datacenter Name	Name of the Datacenter in target vCenter for VNF deployment (Default VDC after Multi-VDC supported)
Datastore Name	The destination datastore of all the image (template) create through ESC .

Screen	Action
Datastore Host	The destination computing-host of all the image (template) create through ESC.
Ready to Complete	Review the deployment settings. **Caution** Any discrepancies can cause VM booting issues. Carefully review the IP address, subnet mask, and gateway information for accuracy.
Public Key	Administrator authorized public key for remote login.
ConfD Username	Administrator username for netconf and ConfD CLI.
ConfD Password	Administrator password for netconf and ConfD CLI.
ConfD Public Key	Administrator authorized public key for netconf and ConfD CLI.

Step 4 Check the **Power on after deployment** check box to power on the VM after deployment.

Step 5 Click **Finish**.
A progress indicator shows the task progress until Elastic Services Controller is deployed.

Step 6 After Elastic Services Controller is successfully deployed, click **Close**.

Step 7 Power on the Elastic Services Controller VM.

Installing Elastic Services Controller Using OVF Tool

In addition to installing the Elastic Services Controller using the OVA image, you can use the VMware OVF Tool, a command-line client, to install Elastic Services Controller on VMware vCenter or vSphere.

To install Elastic Services Controller (ESC) from the command line, do the following:

Step 1 Use the probe mode to learn the properties of the OVA package. The probe mode allows you to investigate the contents of a source.

To invoke the probe mode, use the **ovftool** command with only a source and no target.

```
>ovftool <source locator>
```

The following example shows the result of probing the ESC OVA.

```
NETWORK_OVA=(Path to the OVA Package)

NETWORK_HOSTNAME="$(User Name)"
NETWORK_GATEWAY="192.0.2.1"
NETWORK_NET1_IP="192.0.2.0.xx/24" #
NETWORK_NET2_IP="192.51.100.xx/24"
ADMIN_USERNAME="(admin name)"
ADMIN_PASSWORD="(password)"
HTTPS_REST="True"

VMWARE_VCENTER_PORT='80'
```

```
VMWARE_VCENTER_IP='192.0.2.0.xx'
VMWARE_DATASTORE_HOST='192.0.2.0.xx'
VMWARE_DATACENTER_NAME='DC-NETWORK-1'
VMWARE_DATASTORE_NAME='cluster-datastore1'
VMWARE_COMPUTE_CLUSTER_NAME='DC-CLUSTER-1'
VMWARE_VCENTER_USERNAME='root'
VMWARE_VCENTER_PASSWORD='password'
VMWARE_VCENTER_FOLDER="$USER"

# All valid deployment options:
#          4CPU-8GB  (default)
#          4CPU-8GB-2Net
#          4CPU-8GB-3Net
DEPLOYMENT_OPTION="4CPU-8GB-2Net"
```

Step 2 Before you deploy the ESC OVA, configure the properties of the OVA packages. Ensure the following OVA package properties are updated for the ESC OVA: NETWORK_OVA, NETWORK_HOSTNAME, VMWARE_VCENTER_FOLDER, NETWORK_NET1_IP, NETWORK_NET2_IP, and VMWARE_VCENTER_FOLDER.

The OVA descriptors contain configuration properties for the OVA package. You can set only one property at a time, but you can have multiple instances of the option per command. For multiple property mappings, repeat the option, separating them with a blank, for example *--prop:p1=v1 --prop:p2=v2 --prop:p3=v3*.

```
>.ovftool/ovftool\
--powerOn \
--acceptAllEulas \
--noSSLVerify \
--datastore=$VMWARE_DATASTORE_NAME \
--diskMode=thin \
--name=$NETWORK_HOSTNAME \
--deploymentOption=$DEPLOYMENT_OPTION \
--vmFolder=$VMWARE_VCENTER_FOLDER \
--prop:admin_username=$ADMIN_USERNAME --prop:admin_password=$ADMIN_PASSWORD \
--prop:admin_username=admin \
--prop:admin_password='Strong4Security!' \
--prop:confd_admin_username=admin \
--prop:confd_admin_password='Strong4Security!' \
--prop:network_hostname=$NETWORK_HOSTNAME \
--prop:vmware_vcenter_port=$VMWARE_VCENTER_PORT \
--prop:vmware_vcenter_ip=$VMWARE_VCENTER_IP \
--prop:vmware_datastore_host=$VMWARE_DATASTORE_HOST \
--prop:vmware_datacenter_name=$VMWARE_DATACENTER_NAME \
--prop:vmware_vcenter_username=$VMWARE_VCENTER_USERNAME \
--prop:vmware_datastore_name=$VMWARE_DATASTORE_NAME \
--prop:vmware_compute_cluster_name=$VMWARE_COMPUTE_CLUSTER_NAME \
--prop:vmware_vcenter_password=$VMWARE_VCENTER_PASSWORD \
--prop:net1_ip=$NETWORK_NET1_IP \
--prop:net2_ip=$NETWORK_NET2_IP \
--prop:gateway=$NETWORK_GATEWAY \
--prop:https_rest=$HTTPS_REST \
--net:"Network1=VM Network" --net:"Network2=MgtNetwork" --net:"Network3=VNFNetwork" \
    $NETWORK_OVA
vi://$VMWARE_VCENTER_USERNAME:$VMWARE_VCENTER_PASSWORD@$VMWARE_VCENTER_IP/$VMWARE_DATACENTER_NAME/
host/$VMWARE_COMPUTE_CLUSTER_NAME
```

Following are some advanced examples of passing user credentials through properties.

Advanced usage with password hash:

```
--prop:admin_username=admin \
--prop:admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmzxMXLDugfzTbTXmMQDw146VzpxQvMumeaa125.agYHZUqQ8L.sdM2v0'
```

```
 \
--prop:confd_admin_username=admin \
--prop:confd_admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmzxMXLDugfzTbTXmMQDw146VzpxQvMumeaa125.agYHZUqQ8L.sdM2v0'
 \
```

Advanced usage with password hash and authorized public key:

```
--prop:admin_username=admin \
--prop:admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmzxMXLDugfzTbTXmMQDw146VzpxQvMumeaa125.agYHZUqQ8L.sdM2v0'
 \
--prop:admin_public_key='ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAu+nkTtu2pShVbTYL+mmKxtmzM5dNXFy8IeX/lH5fXsODH1EAySlzHGFXq36RT5vIG/
+c2uV8fRsWaY7xXDrdGICxfkPuEj2UQH2MQx2yFjMFcaSAT56hsqE= admin@net' \
--prop:confd_admin_username=admin \
--prop:confd_admin_password='$6$wnOi$UDOQmkKm2tQtr2jDVNhoo4wS42ffYYmzxMXLDugfzTbTXmMQDw146VzpxQvMumeaa125.agYHZUqQ8L.sdM2v0'
 \
--prop:confd_admin_public_key='ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAu+nkTtu2pShVbTYL+mmKxtmzM5dNXFy8IeX+xXu6TT2sTsxkKtVy8uOAeBp1qKzkp+
c2uV8fRsWaY7xXDrdGICxfkPuEj2UQH2MQx2yFjMFcaSAT56hsqE= admin@net' \
```

Note You need to replace the variables (IP addresses, root password, VM names, and so on) in the examples above with values from your own system.

Step 3 To deploy the OVA package with the VMware OVF Tool, use the following command syntax:

```
>ovftool <source locator> <target locator>
```

where <*source locator*> is the path to the OVA package and <*target locator*> is the path target for the virtual machine, OVA package or VI. A VI location refers to any location on a VMware product, such as vSphere, VMware Server or ESXi. For more information on the VMware OVF Tool, see the VMware OVF Tool user documentation.

The ESC VM is deployed on VMware and powered on automatically.

Powering on Cisco Elastic Services Controller Virtual Machine

To power on the Cisco Elastic Services Controller virtual machine (VM):

Note You must set the memory and CPUs based on the requirements prior to clicking the power on. Once you start the VM you cannot change the memory or CPU settings until you shut down.

Step 1 After deploying the VM, select the virtual machine name in vSphere, right-click on it and select **Open Console**.

Step 2 Click the **Power on** button ( ). During the initial boot of the newly deployed machine, you will be prompted to enter a root (system) password, which is not the Cisco Elastic Services Controller portal password.

Note This is the root password for the underlying Linux operating system on which the Cisco Elastic Services Controller portal is installed. You will be asked to enter this password twice. You will need root access to the underlying Linux operating system at various times in the future, so make sure that you remember this password.

The boot process can take a while.

The End User License Agreement window appears on the first boot. Read the license agreement in its entirety, and only if you understand and accept the license terms, enter y (Yes).

Next Steps: Cisco Elastic Services Controller Virtual Machine

Logging in to Cisco Elastic Services Controller Portal

To log in to the ESC Portal, see the Logging in to the ESC Portal, on page 93

Configuring the Virtual Machine to Automatically Power Up

You can configure the ESXi hypervisor to automatically power up the ESC VM when power is restored to the ESXi hypervisor layer.

Note You must manually power up the VM.

Step 1 In the vSphere client, select the ESXi machine to which you are connected. It is not a specific VM that you have to select but the ESXi hypervisor on which they reside.

Step 2 Select the **Configuration** tab.

Step 3 Click the **Virtual Machine Startup/Shutdown** link under the **Software** area. You should see the VM in the list shown in window.

Step 4 Click the **Properties...** link present at the top right corner of the page. If you do not see that, resize the window until you do.

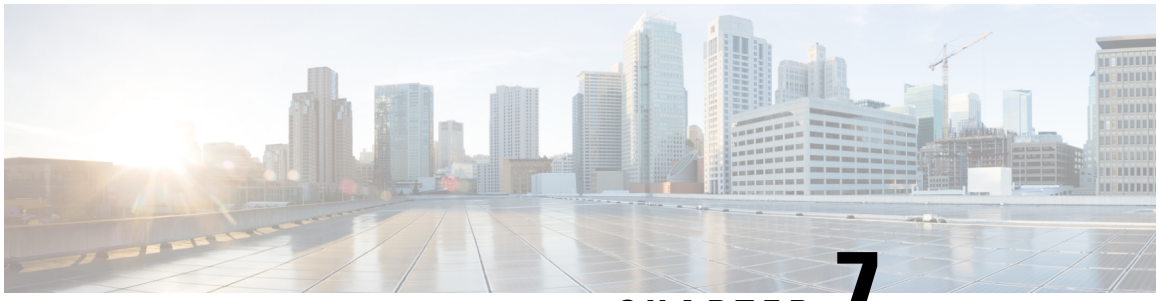
The Virtual Machine Startup and Shutdown page is displayed.

Step 5 Check the **Allow virtual machines to start and stop automatically with the system** check box.

Step 6 Select the virtual machine running ESC and use the **Move Up** button on the right to move it up into the group labeled **Automatic Startup**

Step 7 Click **OK**

This ensures that whenever power is restored to the ESXi hypervisor, the ESC VM powers up automatically.



CHAPTER 7

Configuring High Availability

This chapter contains the following sections:

- High Availability Overview, on page 43
- How High Availability Works, on page 43
- Deploying ESC High Availability with User Data (HA Pair), on page 44
- Deploying ESC High Availability (Standalone Instances), on page 48
- Important Notes for ESC HA, on page 49
- Troubleshooting High Availability, on page 49

High Availability Overview

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

- Network failures
- Power failures
- Dead VM instance
- Scheduled downtime
- Hardware issues
- Internal application failures

How High Availability Works

A High Availability deployment consists of two ESC instances: a primary and a standby. Under normal circumstances, the primary ESC instance provides the services. The corresponding standby instance is passive. The standby instance is in constant communication with the primary instance and monitors the primary instances' status. If the primary ESC instance fails, the standby instance automatically takes over the ESC services to provide ESC service with minimum interruption.

The standby also has a complete copy of the database of the primary, but it does not actively manage the network until the primary instance fails. When the primary instance fails, the standby takes over automatically. Standby instance takes over primary instance to manage the services while primary instance restoration taken place.

When the failed instance is restored, failback operations can be initiated to resume network management via the original primary instance.

ESC instances are managed by using KeepAliveD service. The VM handshake between ESC instances occurs through the KeepAliveD over the IPv4 network.

Deploying ESC High Availability with User Data (HA Pair)

Before you begin:

- Cisco Elastic Services Controller (ESC) High Availability (HA) requires a network to keep alive and replicate database between primary and standby nodes. Both ESC VMs must have at least one network interface connecting to the same network and must be able to communicate to each other through the network.
- Ensure the two ESC VMs are located in different hosts and datastores so that single point failures can be prevented.

You can deploy ESC HA on VMware vCenter or vSphere in either of two ways:

- Deploying ESC HA with user data as a High Availability pair (Supported from ESC 4.2)
- Deploying ESC HA as two standalone instances and then using post configuration to set them as a High Availability pair. For more information,see the section on "Deploying ESC High Availability (Standalone Instances)".

To deploy ESC HA on VMware vCenter or vSphere with user data as a High Availability pair, define the user data file for each HA instance and then point the user data for each instance via ovftool. The encoding of user data is done via a set of commands in the ovftool script, and the result of this is set as a variable to the –prop:user-data= property in the ovftool.

Note The admin user/password and confd user/password properties are mandatory OVF properties. These properties cannot be defined in the user-data files.

- Define the two VMs for ESC HA.

  User Data 1

```
#cloud-config
ssh_pwauth: True
write_files:
 - path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
   content: |
     network:
       version: 1
       config:
       - type: nameserver
         address:
         - 161.44.124.122
       - type: physical
```

```
        name: eth0
        subnets:
        - type: static
          address: 172.16.0.0
          netmask: 255.255.255.0
          routes:
          - gateway: 172.16.0.0
            network: 0.0.0.0
            netmask: 0.0.0.0
 - path: /opt/cisco/esc/esc-config/esc-config.yaml
   content: |
      resources:
        confd:
          option: start-phase0
        drbd:
          nodes:
          - 172.16.0.0
          - 172.16.1.0
          run_forever: true
        esc_service:
          depend_on: filesystem
          type: group
        escmanager:
          depend_on:
          - pgsql
          - mona
          - vimmanager
        etsi:
          depend_on: pgsql
          startup: false
        filesystem:
          depend_on: drbd:master
        keepalived:
          vip: 172.16.2.0
        portal:
          depend_on: escmanager
          startup: false
        snmp:
          startup: false
runcmd:
 - [ cloud-init-per, once, escadm_ovf_merge, sh, -c, "/usr/bin/escadm ovf merge"]
 - [ cloud-init-per, once, escservicestart, sh, -c, "chkconfig esc_service on && service
 esc_service start"]
```

User data 2

```
#cloud-config
ssh_pwauth: True
write_files:
 - path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
   content: |
     network:
       version: 1
       config:
       - type: nameserver
         address:
         - 161.44.124.122
       - type: physical
         name: eth0
         subnets:
         - type: static
           address: 172.16.1.0
           netmask: 255.255.255.0
           routes:
           - gateway: 172.16.0.0
```

```
            network: 0.0.0.0
            netmask: 0.0.0.0
 - path: /opt/cisco/esc/esc-config/esc-config.yaml
   content: |
      resources:
        confd:
          option: start-phase0
        drbd:
          nodes:
          - 172.16.0.0
          - 172.16.1.0
          run_forever: true
        esc_service:
          depend_on: filesystem
          type: group
        escmanager:
          depend_on:
          - pgsql
          - mona
          - vimmanager
        etsi:
          depend_on: pgsql
          startup: false
        filesystem:
          depend_on: drbd:master
        keepalived:
          vip: 172.16.2.0
        portal:
          depend_on: escmanager
          startup: false
        snmp:
          startup: false
runcmd:
 - [ cloud-init-per, once, escadm_ovf_merge, sh, -c, "/usr/bin/escadm ovf merge"]
 - [ cloud-init-per, once, escservicestart, sh, -c, "chkconfig esc_service on && service
 esc_service start"]
```

- OVFtool should be called twice - once for each VM instance. Each instance needs to provide a "--prop:user-data" property to point to its hashed user-data.
- Here is an example to boot a pair of HA instances that use 172.16.0.0 and 172.16.1.0 (floating) IPs to its instances, and 172.16.2.0 as a KAD_VIP.

```
user_data_1=`cat ./user-data-1`
user_data_2=`cat ./user-data-2`
dec_user_data_1=`echo "$user_data_1" | base64 | tr -d '[:space:]'`
dec_user_data_2=`echo "$user_data_2" | base64 | tr -d '[:space:]'`
# vcenter-16 is the developer lab for vmware5
ESC_OVA=/scratch/BUILD-${ESC_IMAGE}/BUILD-${ESC_IMAGE}/ESC-${ESC_IMAGE}.ova
# All valid deployment options:
#          2CPU-4GB
#          4CPU-8GB  (default)
#          4CPU-8GB-2Net
#          4CPU-8GB-3Net
DEPLOYMENT_OPTION="4CPU-8GB-2Net"
deploy_vmware_vm1() {
/usr/bin/ovftool \
--powerOn \
--acceptAllEulas \
--noSSLVerify \
--datastore=$VM_WARE_DATASTORE_NAME \
--diskMode=thin \
--name=$INSTANCE_NAME"-0" \
--deploymentOption=$DEPLOYMENT_OPTION \
```

```
--vmFolder=$FOLDER \
--prop:admin_username=$ESC_VM_USERNAME --prop:admin_password=$ESC_VM_PASSWORD \
--prop:esc_hostname=$INSTANCE_NAME"-0" \
--prop:rest_username=$REST_USERNAME \
--prop:rest_password=$REST_PASSWORD \
--prop:portal_username=$PORTAL_USERNAME \
--prop:portal_password=$PORTAL_PASSWORD \
--prop:confd_admin_username=$CONFD_USERNAME \
--prop:confd_admin_password=$CONFD_PASSWORD \
--prop:vmware_vcenter_port=$VMWARE_VCENTER_PORT \
--prop:vmware_vcenter_ip=$VM_WARE_VCENTER_IP \
--prop:vmware_datastore_host=$VM_WARE_DATASTORE_HOST \
--prop:vmware_datacenter_name=$VM_WARE_DATACENTER_NAME \
--prop:vmware_vcenter_username=$VM_WARE_VCENTER_USERNAME \
--prop:vmware_datastore_name=$VM_WARE_DATASTORE_NAME \
--prop:vmware_vcenter_password=$VM_WARE_VCENTER_PASSWORD \
--prop:net1_ip=$NET1_IP1 \
--prop:net2_ip=$NET2_IP1 \
--prop:gateway=$ESC_GATEWAY \
--prop:https_rest=$HTTPS_REST \
--prop:user-data=$dec_user_data_1 \
--net:"Network1=VM Network" --net:"Network2=MgtNetwork" --net:"Network3=VNFNetwork" \
   $ESC_OVA vi://$VM_WARE_VCENTER_USERNAME:$VM_WARE_VCENTER_PASSWORD@$VM_WARE_VCENTER_IP/
$VM_WARE_DATACENTER_NAME/host/$VM_WARE_DATASTORE_CLUSTER
}
deploy_vmware_vm2() {
/usr/bin/ovftool \
--powerOn \
--acceptAllEulas \
--noSSLVerify \
--datastore=$VM_WARE_DATASTORE_NAME \
--diskMode=thin \
--name=$INSTANCE_NAME"-1" \
--deploymentOption=$DEPLOYMENT_OPTION \
--vmFolder=$FOLDER \
--prop:admin_username=$ESC_VM_USERNAME --prop:admin_password=$ESC_VM_PASSWORD \
--prop:esc_hostname=$INSTANCE_NAME"-1" \
--prop:rest_username=$REST_USERNAME \
--prop:rest_password=$REST_PASSWORD \
--prop:portal_username=$PORTAL_USERNAME \
--prop:portal_password=$PORTAL_PASSWORD \
--prop:confd_admin_username=$CONFD_USERNAME \
--prop:confd_admin_password=$CONFD_PASSWORD \
--prop:vmware_vcenter_port=$VMWARE_VCENTER_PORT \
--prop:vmware_vcenter_ip=$VM_WARE_VCENTER_IP \
--prop:vmware_datastore_host=$VM_WARE_DATASTORE_HOST \
--prop:vmware_datacenter_name=$VM_WARE_DATACENTER_NAME \
--prop:vmware_vcenter_username=$VM_WARE_VCENTER_USERNAME \
--prop:vmware_datastore_name=$VM_WARE_DATASTORE_NAME \
--prop:vmware_vcenter_password=$VM_WARE_VCENTER_PASSWORD \
--prop:net1_ip=$NET1_IP2 \
--prop:net2_ip=$NET2_IP2 \
--prop:gateway=$ESC_GATEWAY \
--prop:https_rest=$HTTPS_REST \
--prop:user-data=$dec_user_data_2 \
--net:"Network1=VM Network" --net:"Network2=MgtNetwork" --net:"Network3=VNFNetwork" \
   $ESC_OVA vi://$VM_WARE_VCENTER_USERNAME:$VM_WARE_VCENTER_PASSWORD@$VM_WARE_VCENTER_IP/
$VM_WARE_DATACENTER_NAME/host/$VM_WARE_DATASTORE_CLUSTER
}
deploy_vmware_vm1
deploy_vmware_vm2
```

- Once the VMs are deployed successfully, you can check the status of ESC HA. You will find that one VM instance is booted as MASTER while the other VM instance is a BACKUP.

Deploying ESC High Availability (Standalone Instances)

To deploy ESC HA on VMware vCenter or vSphere, two separate standalone nodes need to be installed first. After the standalone ESC instances are installed, reconfigure these nodes to turn them into Primary and Standby using the following:

- kad_vip
- kad_vif
- ha_node_list

Note

- On each ESC VM, we need to run *escadm* tool to configure ESC HA parameters and then reload and restart the escadm service.
- When you are deploying ESC HA, the kad_vip argument allows end users to access the Primary ESC instance.

Step 1 Log in to the ESC Standalone instances.

Step 2 As an admin user, run the *escadm* tool on both the Primary and Standby instances and provide the corresponding arguments.

- **kad_vip**— Specifies the IP address for Keepalived VIP (virtual IP) plus the interface of Keepalived VIP [ESC-HA]
- **kad_vif**— Specifies the interface for Keepalived virtual IP and keepalived VRRP [ESC-HA]. You can also use this argument to only specify the interface for keepalived VRRP, if the VIP interface is already specified using the *kad_vip* argument.
- **ha_node_list**— Specifies list of IP addresses for HA nodes in the Primary/Standby cluster for DRDB synchronization. This argument is utilized for replication-based HA solution only. For ESC instances with multiple network interfaces, the IP addresses should be within the network that *--kad_vif* argument specifies .

```
$ sudo escadm ha set --kad_vip= <ESC_HA_VIP> --kad_vif= <ESC_KEEPALIVE_IF> --ha_node_list=
<ESC_NODE_1_IP> <ESC_NODE_2_IP>
$ sudo escadm reload
$ sudo escadm restart
```

Step 3 After the restart, one ESC VM should be in Primary state and the other one should be in Standby state.

Step 4 Add the VIP to the allowed address pairs for both VMs so that the VIP is reachable from outside.

Step 5 Verify the status of each ESC instance.

```
# sudo escadm status
```

The following table lists few other command to check the status:

Status	CLI Commands
ESC HA Role	`cat /opt/cisco/esc/keepalived_state`

ESC Health	`sudo escadm health`
ESC Service Status	If you want to see more details (such as status of the VIM manager, SNMP, portal, ESC manager, keepalived status and so on), add '–v': `sudo escadm status --v` To check the detailed status, check the /var/log/esc/escadm.log

Important Notes for ESC HA

- The HA failover takes about 2 to 5 minutes based on the number of managed VNFs to be operational. ESC service will not be available during the switchover time.
- When the switchover is triggered during transactions, all incomplete transactions will be dropped. The requests should be re-sent by Northbound interface if it does not receive any response from ESC.

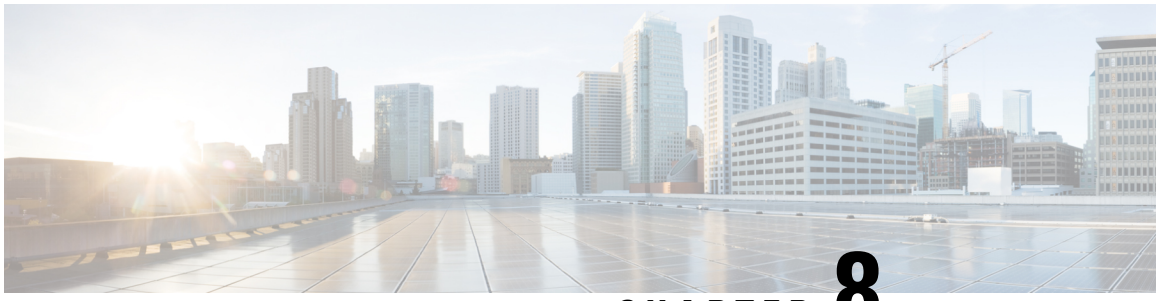
Troubleshooting High Availability

- Check for network failures. If a network problem occurs, you must check the following details:
  - The IP address assigned is correct, and is based on the OpenStack configuration.
  - The gateway for each network interface must be pinged.
- Check the logs for troubleshooting:
  - The ESC manager log at */var/log/esc/escmanager.log*
  - The KeepAliveD log at */var/log/messages* by grep keepalived
  - The ESC service status log at */var/log/esc/escadm.log*

PART III

Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine (KVM)

- Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine, on page 53



CHAPTER 8

Installing Cisco Elastic Services Controller on a Kernel-based Virtual Machine

This chapter describes how to install Cisco Elastic Services Controller on a Kernel-based Virtual Machine and includes the following sections:

- Installing Cisco Elastic Services Controller in a Kernel-based Virtual Machine, on page 53
- Next Steps: Cisco Elastic Services Controller Kernel-based Virtual Machine, on page 55

Installing Cisco Elastic Services Controller in a Kernel-based Virtual Machine

Cisco Elastic Services Controller can be installed in a Kernel-based Virtual Machine. You can install Cisco Elastic services controller in a Kernel-based Virtual Machine using libvirt.

Preparing to Install Cisco Elastic Services Controller on a Kernel-based Virtual Machine

If you plan to run Cisco Elastic Services Controller on a kernel-based virtual machine, make sure the following are setup:

	Notes
Python 2.7 or 3.x	Installed by default on Linux
python-setuptools	Installed by default on Linux

	Notes

pip	**On RHEL**:
	`# easy_install pip`
	Since the installation using pip compiles source files, the gcc and python development packages are also required on RHEL. To install these packages on RHEL:
	`# yum install gcc python-devel`
	On Ubuntu: Installed by default. Since the installation using pip compiles source files, the gcc and python development packages are also required on Ubuntu. To install these packages on Ubuntu:
	`# apt-get install python-dev`

OpenStack clients	`# pip install python-keystoneclient`
	`# pip install python-cinderclient`
	`# pip install python-novaclient`
	`# pip install python-neutronclient`

genisoimage	**On RHEL**:
	`# yum install  genisoimage`
	On Ubuntu:
	`# apt-get install genisoimage`

libvirt and virtinst	**On RHEL 6.x**:
	`# yum install libvirt-python python-virtinst`
	On RHEL 7.x:
	`# yum install libvirt-python virt-install`
	On Ubuntu:
	`# apt-get install libvirt-dev`
	`# pip install libvirt-python`

Note libvirt will create the default network automatically.

Installing Elastic Services Controller on a Kernel-Based Virtual Machine

To install standalone Elastic Services Controller (ESC) on a kernel-based virtual machine, do the following:

Step 1 Load the variables from the openerc file that contains OpenStack credentials:

```
cat ./openrc.sh
export OS_TENANT_NAME='<OS tenant username>'
export OS_USERNAME='<OS username>'
export OS_PASSWORD='<OS password>'
```

```
export OS_AUTH_URL='http://<Openstack Host>:5000/v2.0/'

source ./openrc.sh
```

Step 2 Copy the ESC qcow2 image and the bootvm.py into the kernel-based VM.

Step 3 Boot ESC on a kernel-based VM on the default network that was created when libvirt was installed, use one of the following command:

```
./bootvm.py --user_pass <username>:<password> --user_confd_pass <username>:<password> --libvirt
--image <image_name> esc-vm --net <default network>
```

Step 4 Boot ESC on a kernel-based VM on the default network with static IP, using the following command:

```
./bootvm.py --user_pass <username>:<password> --user_confd_pass <username>:<password> --libvirt
--image <image_name> esc-vm --net <network> --ipaddr <ip_address>
```

Step 5 Get a list of used IP addresses in your network. Use IP addresses that are not in the list for both HA bootvm.py command and for kad_vip. Determine the first 3 octets of your network (i.e. 192.168.122) and pass it in the below command :

```
arp -an | grep 192.168.122
```

Step 6 To install ESC on a kernel-based VM in high availability, use the following command twice for both the HA nodes:

Note For the second bootvm.py command, use the other HA instance name.

```
./bootvm.py --user_pass <username>:<password> --user_confd_pass <username>:<password> --libvirt
--image <image_name> --ha_mode drbd --gateway_ip <default_gateway_ip_address> --ipaddr <ip_address>
 --ha_node_list <ha peer ip addresses separated by comma> --kad_vip <vip address> esc-ha-1 --net
<network>
```

Next Steps: Cisco Elastic Services Controller Kernel-based Virtual Machine

Logging in to Cisco Elastic Services Controller Portal

To log in to the ESC Portal, see the Logging in to the ESC Portal, on page 93

Verifying ESC installation for a Kernel-based Virtual Machine (KVM)

After deploying ESC on a Kernel-based virtual machine, use the following procedure to verify the deployment.

Step 1 Check that the ESC VMs have booted using the following command:

```
$ virsh list
```

Step 2 Get the IP address of the ESC VM, using the following command:

```
$ arp -an | grep <ip_address>
```

Step 3 Connect to ESC using SSH and verify the processes are running:

```
$ ssh USERNAME@ESC_IP
```

Troubleshooting Tips

When SSH access is not available, due to network conditions or ESC startup failures, you can connect to ESC through console(if enabled in ESC VM image) or VNC access. To access ESC VM through VNC, do the following:

1. Identify the vnc port.

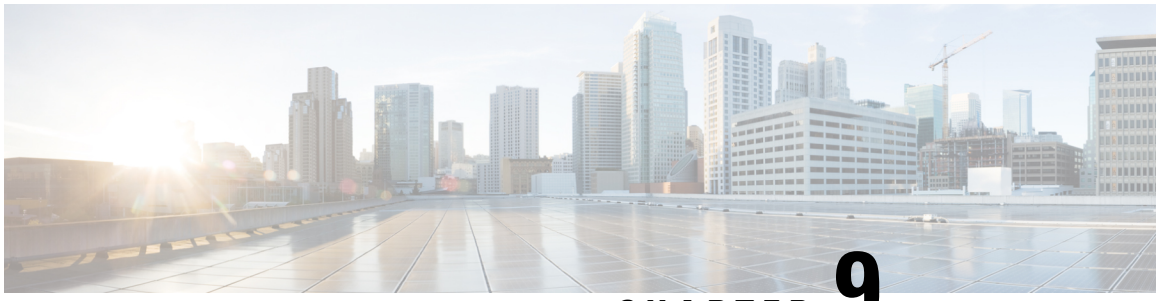
   ```
   virsh dumpxml 10 | fgrep vnc
   ```

2. Create a ssh tunnel to the local vnc port to allow connection from your remote VNC client.

PART IV

Installing Cisco Elastic Services Controller on Amazon Web Services (AWS)

- Installing Cisco Elastic Services Controller on Amazon Web Services, on page 59



CHAPTER 9

Installing Cisco Elastic Services Controller on Amazon Web Services

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

- Prerequisites, on page 59
- Installing the Elastic Services Controller Instance in AWS, on page 59

Prerequisites

Following are the prerequisites that you must complete before you start installing the ESC instance in AWS.

Note If the ESC AMI images are shared with your AWS account, you can ignore these prerequisites and directly use the AMI image for ESC installation.

Step 1 Configure AWS CLI . You can use pip to install AWS CLI. For more details, refer to the AWS documentation.

Step 2 Configure the credentials for AWS CLI based on your account information.

Step 3 Create a Amazon S3 Bucket. Use this for bucket for uploading ESC image.

Note You must have a role named vmimport that allows importing VM and you must attach an IAM policy to the role. For more information, refer to the documentation on the creation of S3 bucket in AWS.

Step 4 Extract the vmdk file from ESC ova file.

```
$ tar xvf ESC-<latest image file>.ova ESC-<latest image file>-disk1.vmdk
```

Installing the Elastic Services Controller Instance in AWS

Once you have completed the tasks specified in the prerequisites section, you can use the procedure below to deploy and launch ESC instance in AWS.

Step 1 **Upload and register ESC image.**

a) Upload the vmdk image to the S3 bucket.

```
aws s3 cp <esc-vmdk-file> s3://<S3 bucket name>/
```

b) Register the image.

```
aws ec2 import-image --description "<esc-vmdk-file>" --disk-containers file://containers.json
```

Step 2 **Create user data**.

a) Create a user for ESC VM. Without a user, you would not be able to access the VM. It is recommended to configure 'admin' user with sudo access and ssh key.

b) Create the esc-config.yaml in user-data using write_files command.

Each instance can have up to 15 interfaces, depending on the type of instances.

Note If you want to use two interfaces, ensure that you create the two network interfaces before hand. These interfaces on different subnets must belong to the same availability zone. Add the interface details in the 'Configure Instance Details' tab when launching the instance from AWS console.

c) Enable esc_service and start it.

Following is an example of a complete user data:

```
#cloud-config
# It is recommended to disable password authentication for ssh when ESC runs in public cloud such as
 AWS.
ssh_pwauth: False
users:
 - name: admin
   # Put admin in 'esc-user' group, otherwise some scripts of ESC might fail when running as admin.
   groups: esc-user
   gecos: User created by cloud-init
   # This is an example of the hashed password for 'admin'.
   passwd:
$6$rounds=656000$pswsUsR7Iz9NIfA4$7E1sEGV8rhDieNDhc824lYwL3cQ8Rsgp9Nds.OZBe9rG/DE56YWk0kDZoB.DsjATrj9pcBnAe.rSOpWll2r0N/

   # The public key for admin user. Replace it with your public key to login.
   ssh-authorized-keys:
     - ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCqGLe4EVVI/rQy4e4jZUEnc5PvYItc39x5fz9rRggZzpwYzKXSj+UnWQMgvkIai+
Mv5vTPiEYTSNZx9PmUKayZaLr/2GiLPmnPNEgyzvjD5v77vV3Ag7eHfLXKLYbu7ausYqFKEFbNgSTGC1PWhoz2geY4zNO9hS3eMnNvxNSLpbo3ftzamQoqtWSx2aRc81M/
piy6NcBzJ3JeH4rOk9bQ+QxRAYm3bOlq/qRfuoxmrsgd68xAIXeDWyGumEThXN9MDEcQMIWO54fiPQgkqKbZWztH2EBnE9/B6rZCRBUUvdoQhQt2L/
hbCZN1k+oqQ53rlG/BjT09CGfYbgoHq2v
   # false allows you to sudo with the password.
   lock-passwd: false
   homedir: /home/admin
   # sudo settings
   sudo: ALL=(ALL) ALL
write_files:
 - path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
   content: |
     network:
       version: 1
       config:
       # You must define the name server when you use the static IP address.
       - type: nameserver
         address:
         - 172.31.0.2
```

```
      # Define physical network interface
      - type: physical
        name: eth0
        subnets:
        # Define the static IP address
        - type: static
          address: 172.31.5.66
          netmask: 255.255.240.0
          # Define the routes
          routes:
          - gateway: 172.31.0.1
            # 0.0.0.0 means the default gateway
            network: 0.0.0.0
            netmask: 0.0.0.0
 # ESC service config file
 - path: /opt/cisco/esc/esc-config/esc-cfg.yaml
   content: |
     confd:
       # AAA users for ConfD
       init_aaa_users:
       # Public key for ConfD user 'admin'
      - key: c3NoLXJzYSBBQUFBQjNOemFDMXljMkVBQUFBREFRQUJBQUFCQVFDeFkwMzByaEMzSXlWekF2bStISVlmMmpkdm
RUZndTTEpCRjVPTjZoUEgvK2FBTkkzb0NCSmJndjhPdjRtVXUvYmlCYmsyS240QW52Ni9ROE1YWGducnZST241MlJuODN2ejRCWTAw
T1h2SzZrT2YrUnZkSDFtNjhscVlrWU9uZVErNEtOak5tQXRwV0huT0xCZE1mZ2pzTmF1SlF1QVJUMEtDS2VBS3k4aUVqSUZpZDhWZ3
NiSlA0aDNpTzdjcTkza0E1ZGFQb0xiNWRKRVp3ZW15WS9ENGp6ZnJUeDVKWFFuMy80SDdaQVZPaWcyNzBGUnlGVkZhNFl1VXNYcDk1d3
QveHdpc0RUREVCYTYydjkxQzdXamtaNy9rYkRlRW9VSU9OZExqdEdvbU84c2JRUUJoZHBVTTZlNXJkeUl2VzQ3YTZYOFA5N2lBR3JrQ09
qMWVHNkYgeGlhb3hpbnlAWElBT1hJTlktTS1SRVhXCg==
         # Note: 'admin' is the only user supported and you cannot change the name here.
         name: admin
         # Hashed password for admin user.
         passwd:
$6$rounds=656000$d4hZHtniblo4/b0m$fD3./1H3jcPlWAENviFlu70i5wKnH9DIasDwTkL.p70UFZlfalzD9O7utL1NdKXwudnNhxIOrvYagkBfq6AWh.

     # No specific settings for esc service. Leave it empty.
     esc_service: {}
runcmd:
 - [ cloud-init-per, once, escservicestart, sh, -c, "chkconfig esc_service on && service esc_service
 start"]
```

Following is an example to define two interfaces in user data:

```
 - path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
   content: |
     network:
       version: 1
       config:

       - type: physical
         name: eth0

         subnets:

         - type: static
           address: 172.31.5.66
           netmask: 255.255.240.0
           # Define the routes
           routes:
           - gateway: 172.31.0.1
             # 0.0.0.0 means the default gateway
             network: 0.0.0.0
             netmask: 0.0.0.0

       - type: physical
         name: eth1
```

```
subnets:

- type: static
  address: 172.31.51.220
  netmask: 255.255.240.0
```

Step 3 **Launch ESC VM in AWS**

Launch ESC VM using one of the following method:

- **From Portal**:
  1. Go to EC2 Management Console, IMAGES/AMIs. Select the image you imported and click **Launch**.
  2. Choose an instance type. Choose t2.xlarge as the instance type.
  3. Configure the Instance Details. Add details such as User Data, Storage, Tag name, and so on. While using two interfaces, create these network interfaces and them here.
  4. Configure a security group. Enable ssh only.
  5. Click **Launch**.
- **From Command Line**: Choose the image, subnet, security group and use the following command to instantiate ESC VM.

```
aws ec2 run-instances --subnet-id <subnet id> --image-id <image id> --security-group-ids <security
 group id> --count 1
--instance-type <instance> --key-name <key name>  --user-data <user data file location>
--associate-public-ip-address
```

Note ESC does not support HA installation on AWS.

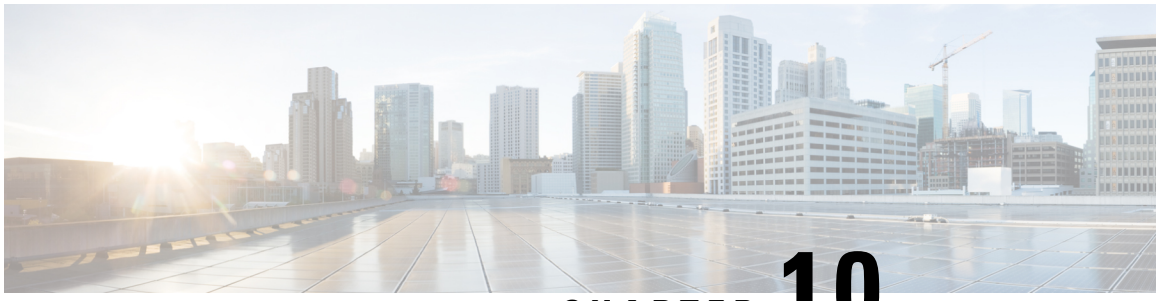
What to do next

After you launch the ESC VM, check the status of the ESC service using the `$ sudo escadm status` command.

PART V

Installing Cisco Elastic Services Controller on Cisco Cloud Services Platform 2100

- Installing Cisco Elastic Services Controller on Cisco Cloud Services Platform 2100, on page 65



CHAPTER 10

Installing Cisco Elastic Services Controller on Cisco Cloud Services Platform 2100

This chapter describes how to install Cisco Elastic Services Controller on CSP 2100 and includes the following sections:

- Prerequisites, on page 65
- Installing the Elastic Services Controller Instance in CSP 2100, on page 65

Prerequisites

Following are the prerequisites that you require before you start installing the ESC instance in CSP 2100.

- Virtual CPUs 2 (minimum)
- Memory 8 GB
- Disk size 80

Installing the Elastic Services Controller Instance in CSP 2100

Once you have completed the tasks specified in the prerequisites section, you can use the following procedure to deploy and launch ESC instance in CSP 2100. Following are the three deployment alternatives available for CSP 2100.

- ESC with Single and Dual Interfaces
- ESC HA Installation

ESC with Single and Dual Interface

To install ESC in CSP, you must create the user-data in the following format as the day0 configuration file:

A sample for single interface describing the day zero file as config drive and user data is as follows:

```
#cloud-config
users:
 - name: admin          # The user's login name
   gecos: admin         # The user name's real name
   groups: esc-user     # add admin to group esc-user
```

```
   passwd:
$6$rounds=4096$Yo11pRsFO$iT5SGMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1

                        # The hash -- not the password itself -- of the password you want
                        #           to use for this user. You can generate a safe hash via:

                        #               mkpasswd --method=SHA-512 --rounds=4096
   lock-passwd: false   # Defaults to true. Lock the password to disable password login
                        # Set to false if you want to password login
   homedir: /home/admin # Optional. Set to the local path you want to use. Defaults to
/home/<username>
   sudo: ALL=(ALL) ALL  # Defaults to none. Set to the sudo string you want to use

ssh_pwauth: True        # Defaults to False. Set to True if you want to enable password
authentication for sshd.

write_files:
# System Configuration
- path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
  content: |
    network:            # Network configuration
      version: 1        # Network config version 1
      config:
      - type: physical  # Represents a “physical” network device, typically Ethernet-based.

        name: eth0      # The network device name
        subnets:        # Multiple subnet entries can be defined under this portion.{}
       - type: static  # Specify the subnet type. set to 'static' to configure this interface
 with a static IPv4.
          address: 10.0.0.40   # ESC's Static IP Address
          netmask: 255.255.255.128  # Netmask
          dns_nameservers: ['10.0.0.1']
          routes:
          - gateway: 10.0.0.1 # Default Gateway
            network: 0.0.0.0
            netmask: 0.0.0.0
# ESC Configuration
- path: /opt/cisco/esc/esc-config/esc-cfg.yaml
  content: |
    confd:
      init_aaa_users:
      - name: admin
        passwd:
$6$rounds=4096$Yo11pRsFO$iT5SGMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1

      escmanager:
        open_ports:
        - '8080'
        - '8443'
        url:
        - http://0.0.0.0:8080/ESCManager
        - https://0.0.0.0:8443/ESCManager
    esc_service: {}
# Networking
- path: /opt/cisco/esc/esc-config/esc-config.xml
  content: |
    <?xml version="1.0" ?>
    <esc>
        <esc-cloud>
            <http_rest enabled="true"/>
            <auth enabled="false"/>
            <https_rest enabled="false"/>
        </esc-cloud>
        <esc-service>
```

```
            <service-init/>
        </esc-service>
    </esc>
# Params
- path: /opt/cisco/esc/esc-config/esc_params.conf
  content: |
    default.enable_cascade_deletion=true
bootcmd:
- [ cloud-init-per, once, disable_ipv6_eth0, sh, -c, "echo net.ipv6.conf.eth0.disable_ipv6
 = 1 >> /etc/sysctl.conf"]
# Update the ESC-HOST value with your own hostname
- [ cloud-init-per, once, update_host_name, sh, -c, "echo ESC-HOST >> /etc/hostname &&
hostnamectl set-hostname ESC-HOST"]
- [ cloud-init-per, once, update_hosts, sh, -c, "echo 127.0.0.1 ESC-HOST >> /etc/hosts"]
runcmd:
- [ cloud-init-per, once, confd_keygen_root, sh, -c, "/usr/bin/escadm confd keygen --user
root"]
- [ cloud-init-per, once, confd_keygen_admin, sh, -c, "/usr/bin/escadm confd keygen --user
 admin"]
- [ cloud-init-per, once, esc_service_start, sh, -c, "chkconfig esc_service on && service
esc_service start"] # You must include this line
```

A sample for dual interfaces describing the day zero file as config drive and user data is as follows:

You can configure an ethernet-based physical network device with a static IPv4 in ESC .

```
#cloud-config
users:
 - name: admin          # The user's login name
   gecos: admin         # The user name's real name
   groups: esc-user     # add admin to group esc-user
   passwd:
$6$rounds=4096$Yo11pRsFO$iT5SGMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1

                        # The hash -- not the password itself -- of the password you want
                        #           to use for this user. You can generate a safe hash via:

                        #               mkpasswd --method=SHA-512 --rounds=4096
   lock-passwd: false   # Defaults to true. Lock the password to disable password login
                        # Set to false if you want to password login
   homedir: /home/admin # Optional. Set to the local path you want to use. Defaults to
/home/<username>
   sudo: ALL=(ALL) ALL  # Defaults to none. Set to the sudo string you want to use

ssh_pwauth: True        # Defaults to False. Set to True if you want to enable password
authentication for sshd.

write_files:
# System Configuration
- path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
  content: |
    network:            # Network configuration
      version: 1        # Network config version 1
      config:
     - type: physical  # Represents a “physical” network device, typically Ethernet-based.

        name: eth0      # The network device name
        subnets:        # Multiple subnet entries can be defined under this portion.{}
       - type: static  # Specify the subnet type. set to 'static' to configure this interface
 with a static IPv4.
          address: 10.0.0.40   # ESC's Static IP Address
          netmask: 255.255.255.128  # Netmask
          dns_nameservers: ['10.0.0.1']
          routes:
          - gateway: 10.0.0.1 # Default Gateway
```

```
            network: 0.0.0.0
            netmask: 0.0.0.0
      - type: physical  # Represents a “physical” network device, typically Ethernet-based.

        name: eth1      # The network device name
        subnets:        # Multiple subnet entries can be defined under this portion.{}
       - type: static  # Specify the subnet type. set to 'static' to configure this interface
 with a static IPv4.
          address: 192.168.0.40   # ESC's Static IP Address
          netmask: 255.255.255.128  # Netmask
          dns_nameservers: []
          routes:
          - gateway: 192.168.0.1 # Default Gateway
            network: 192.168.0.0
            netmask: 255.255.255.128
# ESC Configuration
- path: /opt/cisco/esc/esc-config/esc-cfg.yaml
  content: |
    confd:
      init_aaa_users:
      - name: admin
        passwd:
$6$rounds=4096$Yo11pRsFO$iT5SQMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1

      escmanager:
        open_ports:
        - '8080'
        - '8443'
        url:
        - http://0.0.0.0:8080/ESCManager
        - https://0.0.0.0:8443/ESCManager
    esc_service: {}
# Networking
- path: /opt/cisco/esc/esc-config/esc-config.xml
  content: |
    <?xml version="1.0" ?>
    <esc>
        <esc-cloud>
            <network gateway="10.0.0.1" hostname="VAR_LOCAL_HOSTNAME" ipv6_autoconf="no"
networking="yes" networking_ipv6="yes" nozeroconf="yes">
                <interface bootproto="none" device="eth1" ipaddr="192.168.0.40"
ipv4_failure_fatal="yes" ipv6init="no" name="System_eth1" netmask="255.255.255.128"
nm_controlled="no" onboot="yes" type="Ethernet"/>
                <resolv/>
            </network>
            <http_rest enabled="true"/>
            <auth enabled="false"/>
            <https_rest enabled="false"/>
        </esc-cloud>
        <esc-service>
            <service-init/>
        </esc-service>
    </esc>
# Params
- path: /opt/cisco/esc/esc-config/esc_params.conf
  content: |
    default.enable_cascade_deletion=true
bootcmd:
- [ cloud-init-per, once, disable_ipv6_eth0, sh, -c, "echo net.ipv6.conf.eth0.disable_ipv6
 = 1 >> /etc/sysctl.conf"]
# Update the ESC-HOST value with your own hostname
- [ cloud-init-per, once, update_host_name, sh, -c, "echo ESC-HOST >> /etc/hostname &&
hostnamectl set-hostname ESC-HOST"]
- [ cloud-init-per, once, update_hosts, sh, -c, "echo 127.0.0.1 ESC-HOST >> /etc/hosts"]
```

```
runcmd:
- [ cloud-init-per, once, confd_keygen_root, sh, -c, "/usr/bin/escadm confd keygen --user
root"]
- [ cloud-init-per, once, confd_keygen_admin, sh, -c, "/usr/bin/escadm confd keygen --user
 admin"]
- [ cloud-init-per, once, esc_service_start, sh, -c, "chkconfig esc_service on && service
esc_service start"] # You must include this line
```

Creating ESC passwords to use in Day0 Files

When using the Cloud-Init day0 file to deploy an ESC instance, the passwords must be passed in as a hash, and not a plain text.

To create a hashed password, use the mkpasswd tool. The following example shows how to use the mkpasswd tool to create a hashed password.

```
~$ mkpasswd --method=SHA-512 --rounds=4096
Password:
$6$rounds=4096$Yo11pRsFO$iT5SGMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1
```

Step 1 **Upload user-data file to CSP**

To deploy ESC, the user-data file must be first uploaded to the CSP node.

Note The path to upload images and day0 files is: /osp/repository

```
scp user-data-esc admin@<CSP_IP_ADDRESS>:/osp/repository
```

Step 2 **Deploying ESC VM**

You must edit configuration to be sent to the CSP node hosting the ESC VM.

Following is the deployment datamodel for single interface. For dual interface, you have two interfaces.`<name>ESC-SA-2-IF</name>`

```
<?xml version="1.0"?>
<services xmlns="http://www.cisco.com/ns/test/service">
  <service>
    <name>ESC-SA-1-IF</name>
    <memory>2048</memory>
    <numcpu>1</numcpu>
    <disk_size>8.0</disk_size>
    <disk-resize>true</disk-resize>
    <iso_name>ESC-4_4_0_xxx</iso_name>
    <power>on</power>
    <ip>10.0.0.40</ip>
    <!-- add the ip for display in the CSP web/console interfaces -->
    <vnc_password>CSP_ADMIN_PASSWORD</vnc_password>
    <!-- to secure the VNC console session -->
    <vnics>
      <vnic>
        <nic>0</nic>
        <vlan>12</vlan>
        <tagged>false</tagged>
        <type>access</type>
        <passthrough_mode>none</passthrough_mode>
        <model>virtio</model>
        <network_name>enp11s0f0</network_name>
      </vnic>
    </vnics>
    <disk_type>ide</disk_type>
    <day0_filename>user-data-esc</day0_filename>
    <day0-dest-filename>user-data</day0-dest-filename>
```

```
    <day0-volume-id>cidata</day0-volume-id>
  </service>
</services>
```

Step 3 **Sending Configuration**

Use a netconf-console (shipped with ConfD) to deploy ESC on a CSP node.

```
$ netconf-console --port=2022 --host=<CSP_IP_ADDRESS> --user=CSP_ADMIN_USERNAME
--password=CSP_ADMIN_PASSWORD
```

Step 4 **Configuring the VIM Connector**

After ESC has booted, configure the VIM Connectors.

When installing ESC in CSP, no VIM connectors are added by default. To manage VNFs, you must create the VIM connector.

Step 5 **Adding the VIM Connectors**

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*.

ESC HA Installation

To install ESC in CSP, you must create the user-data in the following format as the day0 configuration file. For HA, you must define one file for each VM.

For creating ESC passwords to use in Day0 Files, see the **Creating ESC passwords to use in Day0 Files** section.

A sample for ESC HA installation on ode 1 describing the day zero file as config drive and user data is as follows:

```
user-data sample - HA Node 1
#cloud-config
users:
 - name: admin
   gecos: admin
   groups: esc-user
   passwd:
$6$rounds=4096$Yo11pRsFO$iT5SGMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1

   lock-passwd: false
   homedir: /home/admin
   sudo: ALL=(ALL) ALL

ssh_pwauth: True

write_files:
# System Configuration
- path: /etc/cloud/cloud.cfg.d/sys-cfg.yaml
  content: |
    network:            # Network configuration
      version: 1        # Network config version 1
      config:
      - type: physical  # Represents a “physical” network device, typically Ethernet-based.

        name: eth0      # The network device name
        subnets:        # Multiple subnet entries can be defined under this portion.{}
       - type: static  # Specify the subnet type. set to 'static' to configure this interface
```

```
 with a static IPv4.
          address: 10.0.0.40   # ESC's Static IP Address
          netmask: 255.255.255.128  # Netmask
          dns_nameservers: ['10.0.0.1']
          routes:
          - gateway: 10.0.0.1 # Default Gateway
            network: 0.0.0.0
            netmask: 0.0.0.0
      - type: physical  # Represents a “physical” network device, typically Ethernet-based.

        name: eth1      # The network device name
        subnets:        # Multiple subnet entries can be defined under this portion.{}
       - type: static  # Specify the subnet type. set to 'static' to configure this interface
 with a static IPv4.
          address: 192.168.0.40   # ESC's Static IP Address
          netmask: 255.255.255.128  # Netmask
          routes:
          - gateway: 192.168.0.1 # Default Gateway
            network: 192.168.0.0
            netmask: 255.255.255.128
# ESC Configuration
- path: /opt/cisco/esc/esc-config/esc-cfg.yaml
  content: |
    ha:
      vri: 9 # this number is ranged from 1 ~ 254. You can randomly choose one but make
sure it won't conflict with other VRRP ID in the same subnet
      mode: drbd             # Only 'drbd' is supported now.
      vip: 10.0.0.42  # The VIP for ESC HA
      vif: eth0
      nodes:
      - ipaddr: 10.0.0.40  # ESC HA's own IP Address
      - ipaddr: 10.0.0.41 # The peer ID Address of ESC HA
    confd:
      init_aaa_users:
      - name: admin
        passwd:
$6$rounds=4096$Yo11pRsFO$iT5SGMJ6z8WEnmj8TKMdInblgWeb/UChmrsQs3aspx8j.yUuuhxKk2XScOkerWWxpqD5F0sLfC5kzT5t2xGkL1

    esc_service: {}
# Params
- path: /opt/cisco/esc/esc-config/esc_params.conf
  content: |
    default.enable_cascade_deletion=true
bootcmd:
- [ cloud-init-per, once, disable_ipv6_eth0, sh, -c, "echo net.ipv6.conf.eth0.disable_ipv6
 = 1 >> /etc/sysctl.conf"]
# Update the ESC-HA-1 value with your own hostname
- [ cloud-init-per, once, update_host_name, sh, -c, "echo ESC-HA-1 >> /etc/hostname &&
hostnamectl set-hostname ESC-HA-1"]
- [ cloud-init-per, once, update_hosts, sh, -c, "echo 127.0.0.1 ESC-HA-1 >> /etc/hosts"]
runcmd:
- [ cloud-init-per, once, confd_keygen_root, sh, -c, "/usr/bin/escadm confd keygen --user
root"]
- [ cloud-init-per, once, confd_keygen_admin, sh, -c, "/usr/bin/escadm confd keygen --user
 admin"]
- [ cloud-init-per, once, esc_service_start, sh, -c, "chkconfig esc_service on && service
esc_service start"] # You must include this line
```

Step 1 **Uploading user-data file to CSP**

To deploy ESC, the user-data file must be first uploaded to the CSP node.

Note The path to upload images and day0 files is: /osp/repository

```
scp user-data-esc-ha-1 CSP_ADMIN_USERNAME@<CSP_IP_ADDRESS>:/osp/repository

scp user-data-esc-ha-2 CSP_ADMIN_USERNAME@<CSP_IP_ADDRESS>:/osp/repository
```

Step 2 **Deploying ESC VM**

You must edit configuration to be sent to the CSP node hosting the ESC VM.

Following is the deployment datamodel for ESC HA on node 1 :

```
deployESC-HA-1.xml
<?xml version="1.0"?>
<services xmlns="http://www.cisco.com/ns/test/service">
  <service>
    <name>ESC-HA-1</name>
    <memory>2048</memory>
    <numcpu>1</numcpu>
    <disk_size>8.0</disk_size>
    <disk-resize>true</disk-resize>
    <iso_name>ESC-4_4_0_xxx</iso_name>
    <power>on</power>
    <ip>10.0.0.40</ip>
    <!-- add the ip for display in the CSP web/console interfaces -->
    <vnc_password>VNC_PASSWORD</vnc_password>
    <!-- to secure the VNC console session -->
    <vnics>
      <vnic>
        <nic>0</nic>
        <vlan>1</vlan>
        <tagged>false</tagged>
        <type>access</type>
        <passthrough_mode>none</passthrough_mode>
        <model>virtio</model>
        <network_name>enp11s0f0</network_name>
      </vnic>
      <vnic>
        <nic>1</nic>
        <vlan>1</vlan>
        <tagged>false</tagged>
        <type>access</type>
        <passthrough_mode>none</passthrough_mode>
        <model>virtio</model>
        <network_name>enp11s0f1</network_name>
      </vnic>
    </vnics>
    <disk_type>ide</disk_type>
    <day0_filename>user-data-esc</day0_filename>
    <day0-dest-filename>user-data</day0-dest-filename> <!-- 'user-data' must always be used for the
 day0-dest-filename. -->
    <day0-volume-id>cidata</day0-volume-id>
  </service>
</services>
```

Following is the deployment datamodel for ESC in HA on node 2 :

```
deployESC-HA-2.xml
<?xml version="1.0"?>
<services xmlns="http://www.cisco.com/ns/test/service">
  <service>
    <name>ESC-HA-2</name>
    <memory>2048</memory>
    <numcpu>1</numcpu>
    <disk_size>8.0</disk_size>
    <disk-resize>true</disk-resize>
    <iso_name>ESC-4_4_0_xxx</iso_name>
```

```
    <power>on</power>
    <ip>10.0.0.41</ip>
    <!-- add the ip for display in the CSP web/console interfaces -->
    <vnc_password>VNC_PASSWORD</vnc_password>
    <!-- to secure the VNC console session -->
    <vnics>
      <vnic>
        <nic>0</nic>
        <vlan>1</vlan>
        <tagged>false</tagged>
        <type>access</type>
        <passthrough_mode>none</passthrough_mode>
        <model>virtio</model>
        <network_name>enp11s0f0</network_name>
      </vnic>
      <vnic>
        <nic>1</nic>
        <vlan>1</vlan>
        <tagged>false</tagged>
        <type>access</type>
        <passthrough_mode>none</passthrough_mode>
        <model>virtio</model>
        <network_name>enp11s0f1</network_name>
      </vnic>
    </vnics>
    <disk_type>ide</disk_type>
    <day0_filename>user-data-esc</day0_filename>
    <day0-dest-filename>user-data</day0-dest-filename> <!-- 'user-data' must always be used for the
 day0-dest-filename. -->
    <day0-volume-id>cidata</day0-volume-id>
  </service>
</services>
```

Step 3 Sending Configuration

Use a netconf-console (shipped with ConfD) to deploy ESC on a CSP node.

```
$ netconf-console --port=2022 --host=<CSP_IP_ADDRESS> --user=<CSP_ADMIN_USERNAME>
--password=<CSP_ADMIN_PASSWORD> --edit-config=deployESC-HA-1.xml

$ netconf-console --port=2022 --host=<CSP_IP_ADDRESS> --user=<CSP_ADMIN_USERNAME>
--password=<CSP_ADMIN_PASSWORD> --edit-config=deployESC-HA-2.xml
```

Step 4 Configuring the VIM Connector

After ESC has booted, configure the VIM Connectors.

When installing ESC in CSP, no VIM connectors are added by default. To manage VNFs, you must create the VIM connector.

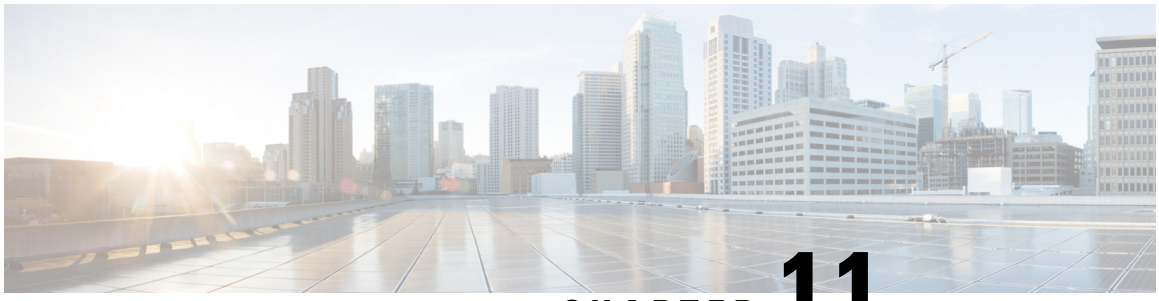
Step 5 Adding the VIM Connectors

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*.

PART VI

Post Installation Tasks

- Post Installation Tasks, on page 77



CHAPTER 11

Post Installation Tasks

This chapter contains the following sections:

- Changing the ESC Password , on page 77
- Configuring Pluggable Authentication Module (PAM) Support for Cisco Elastic Services Controller, on page 79
- Authenticating REST Requests, on page 80
- Configuring Openstack Credentials , on page 83
- Reconfiguring ESC Virtual Machine, on page 88
- Verifying ESC Configurations and Other Post-Install Operations , on page 91
- Logging in to the ESC Portal, on page 93

Changing the ESC Password

You will be forced to change the default password on the first time login. Portal will not let you bypass this step and will keep returning you to this page until you change the default password. After the first time password change, you can change your password using the procedures described in this section. Also, if the user has multiple browsers or tabs or the SAME user is logged on by 2 or more computers and one of the user changes the password then everyone will be logged off and asked to re-enter the new password. The user session has an expiry of 1 hour so if the user is inactive on the portal for an hour then portal will expire the session and the user will have to re-login. If you forgot your password, you can also update or randomly generate the password.

This section discusses how to change the passwords.

Changing the ConfD Netconf/CLI Administrator Password Using the Command Line Interface

After you install ESC, to change the Confd admin password, do the following:

Step 1 Log in to the ESC VM.

```
$ ssh USERNAME@ESC_IP
```

Step 2 Switch to the admin user.

```
[admin@esc-ha-0 esc]$ sudo bash
[sudo] password for admin:
```

Step 3 Load the ConfD CLI:

```
$ /opt/cisco/esc/confd/bin/confd_cli  -u admin
```

Step 4 Set the new admin password:

```
$ configure
$ set aaa authentication users user admin password <new password>
```

Step 5 Save the changes.

```
$ commit
```

Changing Linux Account Password

Step 1 Log in to ESC VM.

```
 $ ssh USERNAME@ESC_IP
```

Step 2 To update or generate a random password, use the following command:

```
/usr/bin/pwqcheck
/usr/bin/pwqgen
```

Changing the ESC Portal Password

The user can update or reset the default admin password.

Step 1 Log in to ESC VM.

Step 2 Switch to the root user.

Step 3 To update the default admin password or randomly generate a password, use one of the following method:

- Using escadm utility:

  To update the default admin password (admin/********):

  ```
  [root@anyname-v44-52 admin]# escadm portal set --username admin --password ********
  Successfully updated password for username admin
  ```

  To generate a random password:

  ```
  [root@anyname-v44-52 admin]# escadm portal set --username admin
  Would you like to use the generated password: "Accent5omit&Wide"?[y|n]y
  Successfully updated password for username admin
  ```

  The *--must_change* variable will ask the user to change their password at the next login.

  The *--must_change* variable is not applicable for REST users.

  ```
  [root@anyname-v44-52 admin]# escadm portal set --username admin --must_change
  ```

```
Would you like to use the generated password: "Rainy4Dozen&Behave"?[y|n]y
Successfully reset password for username admin. User must change the password at the next login.
```

- To reset to a specific password:

```
[root@anyname-v44-52 admin]# escadm portal set --username admin --password P@55w0rd! --must_change
Successfully reset password for username admin. User must change the password at the next login.
```

- Using the bootvm command line:

```
--user_portal_pass admin:<new password>
```

- Using the ESC Portal:

1. Log in to ESC portal using your username and password.
2. Choose **Accounts Setting** on the Navigation menu.
3. Enter the old password in the Old password field, then enter a new password in the New Password and Confirm Password fields.
4. Click **Update Password**.

Configuring Pluggable Authentication Module (PAM) Support for Cisco Elastic Services Controller

You can configure ESC services to use Pluggable Authentication Modules (PAM) for user authentication in ESC. With Cisco Elastic Services supporting PAM, you can also enable LDAP authentication in ESC. If PAM is not configured, ESC will continue to use the default authentication method for each ESC service. The following table lists the commands to enable PAM authentication for each ESC service.

Table 2: Configuring PAM for ESC Services

ESC Service/Component	Command
ESCManager (REST interface)	`sudo escadm escmanager set --auth PAM:[:<pam_service_name>]`
ESC Monitor (Health API)	`sudo escadm monitor set --auth PAM:[:<pam_service_name>]`
Confd	`sudo escadm confd set --auth PAM:[:<pam_service_name>]`
Portal	`sudo escadm portal set --auth PAM[:<pam_service_name>]`
ETSI	`sudo escadm etsi set --pam_service <pam_service_name>`

Note

- The SSHD service that runs inside the ESC VM already uses PAM authentication by default.
- If any component sets PAM authentication without specifying the PAM service, ESC defaults to the PAM service 'system-auth'.

Authenticating REST Requests

ESC REST and ETSI REST APIs use http basic access authentication where the ESC client will have to provide a username and password when making REST requests. The user name and password will be encoded with Base64 in transit, but not encrypted or hashed. HTTPS will be used in conjunction with Basic Authentication to provide the encryption.

This section discusses ESC REST and ETSI REST authentications, how to change the default password of the interfaces, and how to send an authorized requests from the ESC client.

REST Authentication

By default, the REST authentication is enabled. To disable REST authentication, you can pass the argument **--disable-rest-auth** to bootvm. Cisco does not recommend you to use this in a production environment.

ESC also supports https communication over port 8443. ESC will generate a self-signed certificate that the client will need to trust to get the https communication going. By default, the REST https is disabled and restricted to localhost. To enable it pass the argument **--enable-https-rest** to ESC bootvm.py, ESC installation script.

Note Make sure to pass either **--enable-https-rest** or **--enable-http-rest** or both the arguments to the bootvm.py script to enable http and https interfaces to the REST API. You must pass **--user_rest_pass** while using either **--enable-https-rest** or **--enable-http-rest** when REST authentication is not disabled. To enable https or http after ESC VM is booted, use the escadm command specified below.

```
sudo escadm escmanager set --url
http://127.0.0.1:8080/ESCManager,https://0.0.0.0:8443/ESCManager
```

You must change the configuration of the peer instance if ESC is in HA mode.

Enabling ETSI REST Authentication

If the ETSI REST http or https interfaces are enabled, then all requests to an ETSI API must contain authentication data. You can use the **--enable-http-etsi** or **--enable-https-etsi** arguments respectively to enable http and https interfaces to the ESC bootvm.py installation script.

You can enable both interfaces simultaneously, but only the https interface should be enabled in a production environment.

Note To enable http or https after the ESC VM has booted, use the escadm command specified below:

```
sudo escadm etsi enable_http_rest
```

OR

```
sudo escadm etsi enable_https_rest
```

Then restart the ETSI service.

Changing the REST Interface Password

The REST interface has only one default username/password (admin/<default_password>). The password can be updated after the bootup using escadm tool from the ESC VM CLI. You can also update the password through the REST API.

Step 1 Log in to ESC VM.

Step 2 To replace the existing password with a new one, use one of the below options:

- Using the escadm tool from the ESC VM CLI, you can generate a random password:

```
[root@test-v44-52 admin]# escadm rest set --help
usage: escadm rest set [-h] [-v] --username USERNAME [--password PASSWORD]

optional arguments:
  -h, --help           show this help message and exit
  -v, --v, --verbose   show verbose output
  --username USERNAME
  --password PASSWORD  new password or use randomly generated password if no
                       password provided
```

- Using the REST API:

```
http://[ESCVM_IP]:8080/ESCManager/v0/authentication/setpassword?userName=admin&password=yourPassword
```

or

```
https://[ESCVM_IP]:443/ESCManager/v0/authentication/setpassword?userName=admin&password=yourPassword
```

Changing the ETSI REST Interface Password

The ETSI REST interface has only one default username/password (admin/<default_password>). The password can be updated after the bootup using escadm tool from the ESC VM CLI.

Step 1 Log in to ESC VM.

Step 2 To set the default ETSI REST username and password, use the following command:

```
sudo escadm etsi rest_user <username>:<password>
```

or

```
[admin@xyz-esc-4-4-0-59-keep ~]$ escadm etsi set --help
```

```
usage: escadm etsi set [-h] [-v] [--startup {0,1,true,false,manual,auto}]
[--rest_user REST_USER] [--pam_service PAM_SERVICE]

optional arguments:
-h, --help show this help message and exit
-v, --v, --verbose show verbose output
--startup {0,1,true,false,manual,auto}
set to false|0|manual to disable etsi at startup.
--rest_user REST_USER
Set the user for rest. Format username:password
--pam_service PAM_SERVICE
Specify a PAM service to use for authentication. This
will override the rest user. To revert to the using
the rest user for authentication, supply an empty
string.
```

Sending an Authorized REST Request

To send an authorized request, the ESC client should send the request with the following header:

```
Authorization: Basic YWRtaW46Y2lzY28xMjM=
```

where *YWRtaW46Y2lzY28xMjM=* is the Base64 encoded string of the default username/password.

Most libraries and web clients have an interface for providing the username/password and the application will encode the username/password and add the HTTP Basic Auth header.

Example using the default credentials:

For HTTP:

http://[ESCVM_IP]:8080/ESCManager/v0/tenants/

For HTTPS:

https://[ESCVM_IP]:8443/ESCManager/v0/tenants/

Sending an Authorized ETSI REST Request

To send an authorized request, the ESC client should send the request with the following header:

```
Authorization: Basic YWRtaW46Y2lzY28xMjM=
```

where *YWRtaW46Y2lzY28xMjM=* is the Base64 encoded string of the default username/password.

Most libraries and web clients have an interface for providing the username/password and the application will encode the username/password and add the HTTP Basic Auth header.

Example using the default credentials:

For HTTP:

http://[ESCVM_IP]:8250/vnflcm/v1/vnf_lcm_op_occs

For HTTPS:

http://[ESCVM_IP]:8251/vnflcm/v1/vnf_lcm_op_occs

Configuring Openstack Credentials

If ESC was deployed without passing VIM credentials, you can set VIM credentials through ESC VIM and through VIM User APIs (REST or Netconf API).

Note ESC will accept the northbound configuration request only if the following conditions are met:

- ESC has VIM or a VIM user configured through APIs(REST/Netconf).
- ESC has VIM or a VIM user configured, and ESC is able to reach the VIM.
- ESC has VIM or a VIM user configured, and ESC is able to authenticate the user.

Configuring using Netconf API

- **Passing VIM credential using Netconf :**

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <!--represents a vim-->
    <vim_connector>
      <!--unique id for each vim-->
      <id>my-server-30</id>
      <!--vim type [OPENSTACK|VMWARE_VSPHERE|LIBVIRT|AWS|CSP]-->
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>http://<os_ip:port>/v3</value>
        </property>
        <!-- The project name for openstack authentication and authorization -->
        <property>
          <name>os_project_name</name>
          <value>vimProject</value>
        </property>
        <!-- The project domain name is needed for openstack v3 identity api -->
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
      </properties>
      <users>
        <user>
          <id>admin</id>
          <credentials>
            <properties>
              <property>
                <name>os_password</name>
                <value>********</value>
              </property>
              <!-- The user domain name is needed for openstack v3 identity api -->
              <property>
                <name>os_user_domain_name</name>
                <value>default</value>
```

```
            </property>
          </properties>
        </credentials>
      </user>
    </users>
  </vim_connector>
 </vim_connectors>
</esc_system_config>
```

Note

- In 3.0, multiple VIM connectors are supported but within one ESC instance, all the VIM connectors have to in one VIM type (e.g. OpenStack, VMware).
- One VIM is chosen as the default VIM which supports all pre 3.0 config requests and datamodels.
- Deployments can be done on the VIM that is not the default VIM. The deployment to a non default VIM has to have all out-of-band resources (except ephemeral volumes). No other configurations like image, flavor, network, and so on can be done on the VIM that is not the default VIM.
- The default VIM connector will be auto provisioned and does not need to be configured in the following scenarios:
  - If VIM credentials have been passed during ESC boot up.
  - If upgrading from 2.3.x to 3.0.
- The change in the datamodel for Openstack create VIM connector would be handled during upgrade by migration. The 'os_tenant_name' and 'os_project_domain_name' properties would be moved to the VIM Connector properties and 'os_ternant_name' will be renamed to 'os_project_name'.
- For the default VIM Connector, once it is properly authenticated, its properties cannot be updated.
- VIM user can be deleted, recreated, or its properties can be updated at anytime.

- **Updating VIM Connector using Netconf:**

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector nc:operation="replace">
      <id>example_vim</id>
      <type>OPENSTACK</type>
      <properties>
        <property>
          <name>os_auth_url</name>
          <value>{auth_url}</value>
        </property>
        <property>
          <name>os_project_name</name>
          <value>vimProject</value>
        </property>
        <!-- The project domain name is only needed for openstack v3 identity api -->
```

```
        <property>
          <name>os_project_domain_name</name>
          <value>default</value>
        </property>
        <property>
          <name>os_identity_api_version</name>
          <value>3</value>
        </property>
      </properties>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Updating VIM user using Netconf:

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">
  <vim_connectors>
    <vim_connector>
      <id>example_vim</id>
        <users>
          <user nc:operation="replace">
            <id>my_user</id>
            <credentials>
             <properties>
              <property>
                <name>os_password</name>
                <value>********</value>
              </property>
              <!-- The user domain name is only needed for openstack v3 identity api
-->
              <property>
                <name>os_user_domain_name</name>
                <value>default</value>
              </property>
             </properties>
           </credentials>
          </user>
        </users>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Deleting VIM connector using Netconf:

```
<esc_system_config xmlns="http://www.cisco.com/esc/esc">  <vim_connectors>
    <vim_connector nc:operation="delete">
      <id>example_vim</id>
    </vim_connector>
  </vim_connectors>
</esc_system_config>
```

- Deleting VIM Connector using command:

```
$/esc_nc_cli delete-vim-connector <vim connector id
```

- Deleting VIM user using command:

```
$/esc_nc_cli delete-vim-user <vim connector id> <vim user id>
```

Configuring using REST API

- Adding VIM using REST:

```
POST /ESCManager/v0/vims/
HEADER: content-type, callback

<?xml version="1.0"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>example_vim</id>
  <type>OPENSTACK</type>
  <properties>
    <property>
      <name>os_auth_url</name>
      <value>{auth_url}</value>
    </property>
    <property>
      <name>os_project_name</name>
      <value>vimProject</value>
    </property>
    <!-- The project domain name is only needed for openstack v3 identity api -->
    <property>
      <name>os_project_domain_name</name>
      <value>default</value>
    </property>
    <property>
      <name>os_identity_api_version</name>
      <value>3</value>
    </property>
  </properties>
</vim_connector>
```

- Adding VIM user using REST:

```
POST /ESCManager/v0/vims/{vim_id}/vim_users
HEADER: content-type, callback

<?xml version="1.0"?>
<user xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>my_user</id>
  <credentials>
    <properties>
      <property>
        <name>os_password</name>
        <value>********</value>
      </property>
      <!-- The user domain name is only needed for openstack v3 identity api -->
      <property>
        <name>os_user_domain_name</name>
        <value>default</value>
      </property>
    </properties>
  </credentials>
</user>
```

- Update VIM using REST:

```
PUT /ESCManager/v0/vims/{vim_id}
HEADER: content-type, callback

<?xml version="1.0"?>
<vim_connector xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <!--unique id for each vim-->
  <id>example_vim</id>
```

```
  <type>OPENSTACK</type>
  <properties>
    <property>
      <name>os_auth_url</name>
      <value>{auth_url}</value>
    </property>
    <property>
      <name>os_project_name</name>
      <value>vimProject</value>
    </property>
    <!-- The project domain name is only needed for openstack v3 identity api -->
    <property>
      <name>os_project_domain_name</name>
      <value>default</value>
    </property>
    <property>
      <name>os_identity_api_version</name>
      <value>3</value>
    </property>
 </properties>
</vim_connector>
```

- Update VIM user using REST:

```
PUT /ESCManager/v0/vims/{vim_id}/vim_users/{vim_user_id}
HEADER: content-type, callback

<?xml version="1.0"?>
<user xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id>my_user</id>
  <credentials>
    <properties>
      <property>
        <name>os_password</name>
        <value>********</value>
      </property>
      <!-- The user domain name is only needed for openstack v3 identity api -->
      <property>
        <name>os_user_domain_name</name>
        <value>default</value>
      </property>
    </properties>
  </credentials>
</user>
```

- Delete VIM using REST:

```
DELETE /ESCManager/v0/vims/{vim_id}
```

- Delete VIM user using REST:

```
DELETE /ESCManager/v0/vims/{vim_id}/vim_users/{user_id}
```

- Notification example after each VIM or VIM user configuration is done :

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-10-06T16:24:05.856+00:00</eventTime>
  <escEvent xmlns="http://www.cisco.com/esc/esc">
    <status>SUCCESS</status>
    <status_code>200</status_code>
    <status_message>Created vim connector successfully</status_message>
    <vim_connector_id>my-server-30</vim_connector_id>
    <event>
```

```
      <type>CREATE_VIM_CONNECTOR</type>
    </event>
  </escEvent>
</notification>
```

Important Notes:

- In ESC 3.0, you can add multiple VIM Connector for Openstack VIM. Each VIM Connector can have only one VIM User.
- VIM username and password can be updated at anytime. VIM endpoint will not be able to update after a resource is created through ESC.
- After VIM is connected and VIM user is authenticated, VIM can no longer be deleted or updated, only VIM user can be deleted or updated.
- The name of a VIM property or VIM user credentials property is not case sensitive, e.g. OS_AUTH_URL and os_auth_url is the same to ESC.

Reconfiguring ESC Virtual Machine

This section covers the following:

- Reconfiguring Rsyslog
- Reconfiguring NTP
- Reconfiguring DNS
- Reconfiguring Hosts
- Reconfiguring Timezone

Reconfiguring Rsyslog

Rsyslog parameters are optional. If there is a need for customization after booting an ESC VM, you can edit the files in ESC VM (/etc/rsyslog.d/).

Step 1 **Editing the Rsyslog file**:

- If you haven't specified the log forwarding configuration at the bootup time, you may create a file under /etc/rsyslog.d/ like /etc/rsyslog.d/log-forwarding.conf.
- If you have specified the log forwarding through installation, you may just need to edit the file. The file could be /etc/rsyslog.d/20-cloud-config.conf. In this file, to forward logs to multiple rsyslog servers, edit the following line:

```
*.* @[server_ip]:port
```

Note

- Use '@@' before specifying server ip address (if TCP is the protocol used to forward logs to the rsyslog server).
- Use '@' before specifying server ip address (if UDP is the protocol used to forward logs to the rsyslog server).
- server_ip can either be ipv4/ipv6 address of the rsyslog server.
- '[ ]' around the server_ip is required to separate it from ' :port# ', if an ipv6 server address is specified.

For further information on Rsyslog configuration, see the Red Hat documentation.

Step 2 **Configuring the ESC log file:**Configure which ESC log files you want to forward to the rsyslog server:

a) Navigate to /etc/rsyslog.d/ Create or modify a configuration file, such as **log-esc.conf**. Make a copy of sample log-esc.conf .

b) Specify the following block for every file you want to forward to rsyslog server.

```
$InputFileName /var/log/esc/escmanager.log
$InputFileTag esc-manager:
$InputFileStateFile stat-esc-manager
$InputFileSeverity info
$InputRunFileMonitor
```

For example:

```
$InputFileName /var/log/esc/file1.log
$InputFileTag file1:
$InputFileStateFile stat-file1
$InputFileSeverity info
$InputRunFileMonitor

$InputFileName /var/log/esc/file2.log
$InputFileTag file2:
$InputFileStateFile stat-file2
$InputFileSeverity info
$InputRunFileMonitor
```

Step 3 Restart the rsyslog service

```
# service rsyslog restart
```

Step 4 Configure the server side to receive forwarded logs.

a) On a designated server, go to /etc/rsyslog.conf, and uncomment the lines shown below, depending on if you want to listen to logs from clients based on TCP or UDP:

```
#$ModLoad imudp
#$UDPServerRun 514
```

b) Exit the file. Run this command as the last step.

```
sudo service rsyslog restart
```

Now, the server is listening for logs on port 514, using TCP/UDP.

Reconfiguring NTP

Step 1 Open the NTP configuration file /etc/ntp.conf in a text editor such as vi, or create a new one if it does not already exist:

```
# vi /etc/ntp.conf
```

Step 2 Add or edit the list of public NTP servers. If you don't specify the NTP server through the installation, the file should contain the following default lines, but feel free to change or expand these according to your needs:

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
server <your_ntp_server_ip> iburst
```

The iburst directive at the end of each line speeds up the initial synchronization.

Step 3 Once you have the list of servers complete, in the same file, set the proper permissions, giving the unrestricted access to localhost only. Make sure those lines are there in your configure file.

```
restrict default kod nomodify notrap nopeer noquery
restrict -6 default kod nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict -6 ::1
```

Step 4 Save all changes, exit the editor, and restart the NTP daemon:

```
# service ntpd restart
```

Step 5 Make sure that ntpd is started at boot time:

```
# chkconfig ntpd on
```

Reconfiguring DNS

Step 1 The /etc/resolv.conf file contains the configuration for the DNS client (resolver). It typically looks something like this:

```
search domain.com
nameserver 8.8.4.4
```

Step 2 You may modify the IP address of the "nameserver" item or add new nameserver records.

```
search domain.com
nameserver <your_first_dns_ip>
nameserver <your_second_dns_ip>
```

Step 3 Restart Network Service.

```
service network restart
```

Reconfiguring Hosts

The /etc/hosts file allows you to add, edit, or remove hosts . This file contains IP addresses and their corresponding hostnames. If your network contains computers whose IP addresses are not listed in DNS, it is recommended that you add them to the /etc/hosts file.

Step 1 Add the IP addresses that are not listed in DNS to the /etc/hosts file.

Step 2 Restart your network for the changes to take effect.

```
service network restart
```

Reconfiguring Timezone

For ESC VM, in /etc the file "localtime" is a link to or copy of a file containing information about your time zone. Access your zone information files from /usr/share/zoneinfo. To change the time zone, find your country, your city or a city in the same time zone from zone information files in /usr/share/zoneinfo and link it to the localtime in the /etc file.

```
$ ln \-sf /usr/share/zoneinfo/America/Los_Angeles /etc/localtime
```

Verifying ESC Configurations and Other Post-Install Operations

This section covers various post-install checks and operations using the escadm tool.

Verifying Existing ESC Configurations

You can use `escadm dump` command for displaying current ESC configurations in yaml format. The output will show the various services in ESC.

```
$ sudo escadm dump

resources:
  confd:
    init_aaa_users:
    - name: admin
      passwd:
    option: start-phase0
  esc_service:
    group:
    - confd
    - mona
    - vimmanager
    - pgsql
    - escmanager
    - portal
    - monitor
    - snmp
    type: group
  escmanager: {}
  mona: {}
  monitor: {}
  pgsql: {}
  portal: {}
```

```
  snmp:
    run_forever: true
  vimmanager: {}
```

Verifying VIM configurations

You can use `escadm vim show` command to verify the vim settings are correctly populated:

```
$ sudo escadm vim show

[
    {
        "status": "CONNECTION_SUCCESSFUL",
        "status_message": "Successfully connected to VIM",
        "type": "OPENSTACK",
        "id": "default_openstack_vim",
        "properties": {
            "property": [
                {
                    "name": "os_auth_url",
                    "value": "http://10.85.103.143:35357/v3"
                }
            ]
        }
    }
]
```

Troubleshooting ESC Services Startup Issues

Problem: Issues encountered while verifying ESC services status at the installation time using `sudo escadm status`.

Causes: Some services take time to start or have trouble starting.

Solution:

1. Identify the issues using one of the following method:
    - Check the log /var/log/esc/escadm.log

      ```
      $ cat /var/log/esc/escadm.log
      2017-06-01 20:35:02,925: escadm.py(2565): INFO: promote drbd to master...
      2017-06-01 20:35:02,934: escadm.py(2605): INFO: Waiting for at least one drbd to be
       UptoDate...
      2017-06-01 20:35:02,942: escadm.py(2616): INFO: Waiting for peer drbd node to be
      demoted...
      2017-06-01 20:35:14,008: escadm.py(2423): INFO: mount: /dev/drbd1
      /opt/cisco/esc/esc_database
      2017-06-01 20:35:14,017: escadm.py(1755): INFO: Starting filesystem service: [OK]
      2017-06-01 20:35:15,039: escadm.py(1755): INFO: Starting vimmanager service: [OK]
      2017-06-01 20:35:16,116: escadm.py(1755): INFO: Starting monitor service: [OK]
      2017-06-01 20:35:17,163: escadm.py(1755): INFO: Starting mona service: [OK]
      2017-06-01 20:35:18,440: escadm.py(1755): INFO: Starting snmp service: [OK]
      2017-06-01 20:35:21,397: escadm.py(1770): INFO: Starting confd service:[FAILED]
      2017-06-01 20:35:28,304: escadm.py(1755): INFO: Starting pgsql service: [OK]
      2017-06-01 20:35:29,331: escadm.py(1755): INFO: Starting escmanager service: [OK]
      2017-06-01 20:35:30,354: escadm.py(1755): INFO: Starting portal service: [OK]
      2017-06-01 20:35:31,523: escadm.py(1755): INFO: Starting esc_service service: [OK]
      ```

    - Add '-v' to escadm status to show the verbose output of the ESC services.

```
$ sudo escadm status --v
0 ESC status=0 ESC HA Master Healthy
pgsql (pgid 61397) is running
vimmanager (pgid 61138) is running
monitor (pgid 61162) is running
mona (pgid 61190) is running
drbd is master
snmp (pgid 61541) is running
filesystem (pgid 0) is running
<<service>> is dead
keepalived (pgid 60838) is running
portal (pgid 61524) is running
confd (pgid 61263) is running
escmanager (pgid 61491) is running
```

2. Confirm the status of the identified services that has issues and manually start these services.

```
$ sudo escadm <<service>> status// If the status is stopped or dead, manually start the
 services using the next command.

$ sudo escadm <<service>> start --v
```

Logging in to the ESC Portal

Note

- The ESC portal is enabled by default. You must ensure that the ESC portal is not disabled during installation. For more information on enabling or disabling the ESC portal, see Installing Cisco Elastic Services Controller Using the QCOW Image, on page 8.
- When you log in to the ESC portal for the first time you are prompted to change the default password.

To log in to the ESC portal, do the following:

Before you begin

- Register an instance of ESC. For more information on registering the ESC instance see, Installing Cisco Elastic Services Controller Using the QCOW Image, on page 8
- Ensure that you have the username and password.

Step 1 Using your web browser, enter the IP address of ESC and port 443.

Example:

For example, if the IP address of ESC is 192.0.2.254, enter:

https://192.0.2.254: 443 [login via https]

A Security Alert message is displayed.

Step 2 Click **Yes** to accept the security certificate. The Login page is displayed.

Step 3 Enter the username and password and click **Login** .

If you are logging in for the first time, the login page reappears, prompting you to change your password.

Step 4 Enter the old password in the Old Password field, then enter a new password in the New Password and Confirm Password fields.

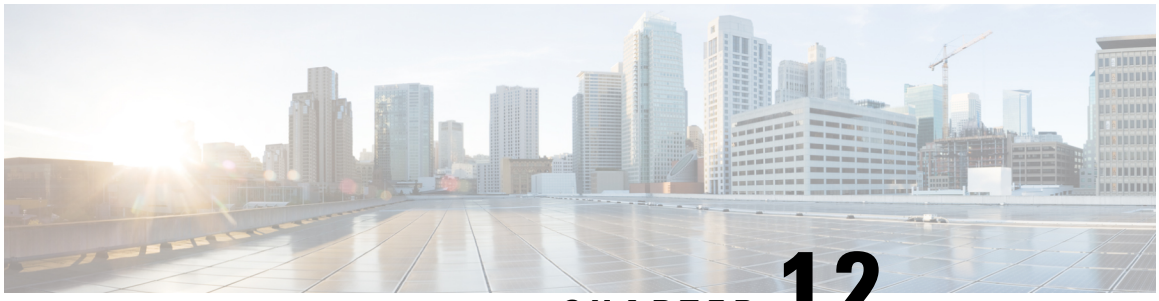
Step 5 Click **Update Password** or press **Enter**.

Note
- If the UI becomes unresponsive, restart the UI by executing the **sudo escadm portal restart** from the ESC shell prompt.
- ESC Portal only supports one user.
- Currently, a pre-installed self-signed certificate supports HTTPS. The user must confirm the self-signed certificate before proceeding with the ESC Portal.
- In HTTPS communication mode, if the URL protocol type returned by OpenStack is not HTTPS, the access to the VNF Console may be disabled. For security reasons, while running in HTTPS more non-secure communication will be rejected.

PART VII

Upgrading Cisco Elastic Services Controller

- ESC in Maintenance Mode, on page 97
- Upgrading Cisco Elastic Services Controller, on page 103



CHAPTER 12

ESC in Maintenance Mode

This chapter contains the following chapters:

- Setting ESC in a Maintenance Mode, on page 97
- Backup the Database from the ESC Standalone Instances, on page 98
- Backup the Database from the ESC HA Instances, on page 99
- Restoring ESC Database, on page 101

Setting ESC in a Maintenance Mode

ESC must be put to maintenance mode to backup and restore ESC database. To do so, use the escadm tool as specified in the below section.

Before you begin

Note From ESC release 4.4, ESC continues to be in maintenance mode after HA switch over or DB restoration, if ESC was in maintenance mode before HA switch over or DB restoration.

During maintenance mode,

- Northbound requests are blocked by ESC and ESC responds with maintenance mode notification.
- Only REST requests receive response that ESC is unavailable temporarily. ConfD requests get the maintenance mode rejection message, or an OK message for all idempotent request such as create tenant request when the tenant already exists.
- Monitoring actions are paused.
- All ongoing requests and transactions continue to progress.

Using the escadm Tool

ESC can be put to maintenance mode using the escadm tool.

Step 1 Put ESC to maintenance mode from the VM shell:

```
sudo escadm op_mode set --mode=maintenance

Set mode to MAINTENANCE
Operation Mode = MAINTENANCE
```

Step 2 To query operation mode at any time,

```
sudo escadm op_mode show
```

Example:

```
Operation Mode = OPERATION
```

Step 3 Set maintenance mode when there is no in-flight transaction. Using the ipt_check flag with the escadm tool, you can choose to set ESC in the maintenance mode only if there are no ongoing transactions in ESC. Set the flag to true, if you do not want ESC to set in the maintenance mode, if there are ongoing transactions in ESC.

```
sudo escadm op_mode set --mode=maintenace --ipt_check=true
```

With the **ipt_check** option set to true, escadm tool checks if there is any on going operation, if so, the escadm tool will not set ESC to maintenance mode.

Setting ESC in an Operation Mode

Put ESC in operation mode using the escadm tool:

```
sudo escadm op_mode set --mode=operation
```

Response is as follows:

```
Set mode to OPERATION
Operation Mode = OPERATION
```

Verify ESC's operation mode at any time using the following command:

```
sudo escadm op_mode show
```

Backup the Database from the ESC Standalone Instances

- The following assumptions should be taken into consideration :
  - A third machine is required to store the database and log backups.
  - ESC does not support database schema downgrade. Restoring database to the older ESC version could cause unexpected problems.
- Before you start the backup process, ensure you have an external storage space (could be in the OpenStack controller or any system accessible by ESC). The backup/restore could be expressed in a generic format which will be used by the escadm tool: scp://<username>:<password>@<backup_ip>:<filename> . In this format, the credentials, IP address and file storage path of the third machine are required. You may also use localhost IP as the backup IP to backup database in a location of ESC VM and then copy the files to the external storage

To backup the ESC database from a standalone ESC or a HA (master node):

Step 1 Log in to ESC VM and set it to maintenance mode, run:

```
$ sudo escadm op_mode set --mode=maintenance
```

Step 2 To make sure ESC is in maintenance mode, run:

```
$ sudo escadm op_mode show
```

Step 3 Backup database. Execute the commands below as a root user:

```
# sudo escadm backup --file /tmp/db_file_name.tar.bz2
scp://<username>:<password>@<backup_vm_ip>:<filename>
```

Step 4 To put ESC back to operation mode, run:

```
$ sudo escadm op_mode set --mode=operation
$ sudo escadm op_mode show
```

Step 5 Collect all the logs from the old ESC VM and back it up. Execute the below command as a root user.

```
# sudo escadm log collect
```

A timestamped log file will be generated in: /var/tmp/esc_log <timestamp>.tar.bz2

Note If a dynamic mapping file is used by ESC service, the dynamic mapping file should be backed up at the same time with ESC logs. The default path of the dynamic mapping file is */opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml.*

Step 6 After a successful database back-up, shut down the old ESC VM using Horizon/Kilo or Nova commands. For ESC VM instances based in VMware vSphere, shutdown the primary instance through VMware client dashboard. An example of shutting down a VM in OpenStack is shown below :

```
 $ nova stop OLD_ESC_ID
```

Step 7 Detach the old port from the old VM and rename the old ESC node. Examples of detaching and renaming the VM in OpenStack is shown below:

```
nova interface-detach ESC_NAME port-id-of-ESC_NAME
nova rename ESC_NAME ESC_NAME.old
```

In VMWare, assign a different IP address to the old VM and then rename the old VM.

Backup the Database from the ESC HA Instances

- The following assumptions should be taken into consideration :
  - A third machine is required to store the database and log backups.
  - ESC does not support database schema downgrade. Restoring database to the older ESC version could cause unexpected problems.
- Before you start the backup process, ensure you have an external storage space available (could be in the OpenStack controller or any system accessable by ESC). The backup/restore could be expressed in a generic format which will be used by the escadm tool: scp://<username>:<password>@<backup_ip>:<filename> . In this format, the credentials, IP address

and file storage path of the third machine are required. You may also use localhost IP as the backup IP to backup database in a location of ESC VM and then copy the files to the external storage.

To backup the ESC database from a standalone ESC or a HA (master node):

Step 1 **Perform the following steps on the Standby ESC node.**

a) Connect to the standby ESC instance using SSH:

```
$ ssh <username>@<backup_vm_ip>
```

b) Verify that the ESC instance is standby and note the name of the standby ESC HA instance :

```
$ sudo escadm status --v
```

If the output value shows "BACKUP", the node is the standby ESC node.

c) Change access to an admin user.

```
sudo bash
```

d) Collect all the logs from the standby ESC VM and back it up.

```
$ sudo escadm log collect
```

A timestamped log file will be generated in: /var/tmp/esc_log <timestamp>.tar.bz2

e) Shutdown the standby ESC instance through OpenStack Kilo/Horizon using Nova command or VMware client. An example of shutting down the VM on OpenStack is shown below:

```
$ nova stop OLD_ESC_STANDBY_ID
```

Step 2 **Perform the following steps on the Master ESC node.**

a) Connect to the primary ESC instance using SSH:

```
$ ssh <username>@<master_vm_ip>
```

b) Change access to an admin user.

```
$ sudo bash
```

c) Verify that the ESC instance is Master and note the name of the Master ESC HA instance

```
$ sudo escadm status --v
```

If the output value shows "MASTER", the node is the Primary ESC node.

d) Back up the database files from the master node of ESC HA:

```
$ sudo escadm backup --file /tmp/db_file_name.tar.bz2
scp://<username>:<password>@<backup_vm_ip>:<filename>
```

e) Collect the logs from the master ESC VM and back it up.

```
$ sudo escadm log collect
```

A timestamped log file will be generated in: /var/tmp/esc_log <timestamp>.tar.bz2

Note If a dynamic mapping file is used by ESC service, the dynamic mapping file should be backed up at the same time with ESC logs. The default path of the dynamic mapping file is */opt/cisco/esc/esc-dynamic-mapping/dynamic_mappings.xml* .

Step 3 Shutdown the primary ESC instance through OpenStack Kilo/Horizon using Nova command. For ESC VM instances based in VMware vSphere, shutdown the primary instance through VMware client dashboard. An example of shutting down the VM on OpenStack is shown below:

```
$ nova stop OLD_ESC_MASTER
```

Use the **nova list** command or the **nova show OLD_ESC_STANDBY** command, to verify if the ESC HA instances have been successfully shut down.

Step 4 (Only for OpenStack) Detach the port from the old ESC VM and rename the old VM.

If upgraded ESC VM needs to operate with same IP addresses and same instance names as old instances, detach the ports from each instance, shutdown the old ESC VMs and then rename the old ESC instances.

If you intend to use old VMware primary instance, assign a different IP address and rename the VM name. If not, you can delete the old VM and use the same IP address for the new upgraded VMware primary instance. After deleting the old VM, you can continue with the old instance name and the IP address.

OpenStack commands for detaching the ports and renaming the old VMs are shown below:

```
nova interface-list ESC_NAME
nova interface-detach ESC_NAME port-id-of-ESC_NAME
nova rename ESC_NAME ESC_NAME.old
```

Restoring ESC Database

Before you begin

To restore the database,

- In standalone ESC instance, stop ESC services. Run `# sudo escadm stop`.
- In HA type instances, stop escadm on the Backup first, and later on the Master ESC HA instance. Run `# sudo escadm stop`.
- All the services must be stopped. To check the status, run `# sudo escadm status --v`.

Step 1 Restore the database. Execute the following command from the ESC VM:

```
$ scp <username>@<server_ip>:/path/db.tar.bz2     /tmp/
$ sudo escadm restore -file /tmp/db.tar.bz2
```

Step 2 Enter the ESC password in the URL, or manually enter it after executing the above command.

Step 3 Restart the ESC service to complete the database restore by running the following command:

```
$ sudo escadm restart
```

Note ESC maintenance mode blocks the northbound request and VNF monitoring. However, if there are some ongoing transactions because of northbound requests before ESC entering maintenance mode, those transactions may have following restriction with backup and restore:

- ESC reports an error for the deployment, network creation, and subnet creation requests, if these transactions are interrupted by backup and restore. Northbound handles these error messages but it may cause network or subnet leakage in some cases (For example, ESC is interrupted before getting the UUID from OpenStack).
- ESC reports an error for service chain upgrade, and requires service chain undeployment and deployment (rather than downgrade and upgrade) to re-create the service.

CHAPTER 13

Upgrading Cisco Elastic Services Controller

Cisco Elastic Service Controller supports two type of upgrades:

- **Backup and Restore Upgrade**: This upgrade process involves stopping the ESC keepalive daemon (for ESC HA), backing up the database, stopping and renaming (or deleting) the ESC instances, re-installing the ESC instances, and restore database. For information on the supported ESC versions for ESC 4.4 upgrade, see the table below.
- **In-service upgrade**: ESC supports in-service upgrade for high-availability nodes with a minimum downtime.

You can upgrade the ESC instance as a standalone instance or as a high availability pair. The upgrade procedure is different for standalone and high availability pair.

This chapter lists separate procedures on how to upgrade ESC standalone and ESC High Availability instance. You must review these instructions before you decide to upgrade the ESC instance. See the Installation Scenarios, on page 7 for more information on the installation scenarios.

- ESC only support direct upgrade from previous two minor releases. For example, ESC 2.3 will support direct upgrade from ESC 2.1 and ESC 2.2. For any release older than the supported versions for direct upgrade, you need to perform the staged upgrade.
- Upgrading ESC using RPM Package (referred to as RPM Upgrade in this chapter) applies only to the ESC upgrade between ESC patch releases with the same release number. For Example, the upgrade from ESC 3.1.0.116 to ESC 3.1.0.150. If you want to upgrade ESC between minor releases (for example, upgrade from ESC 2.3.1 to ESC 2.3.2) or major releases (for example, upgrade from ESC 3.0 to ESC 4.0), you can upgrade through Backup and Restore upgrade process using qcow2 image.
- For ESC upgrade, you should be familiar with ESC installation process.
    - For OpenStack, refer to the OpenStack installation procedures, see Chapter 4: Installing Cisco Elastic Services Controller in OpenStack.
    - For VMware, refer to the VMware Installation installation procedures, see Chapter 7: Installing Cisco Elastic Services Controller in VMware vCenter.

- For ESC HA, please refer to the ESC HA installation procedures, see Chapter 5 : Configuring High Availability for OpenStack and Chapter 8: Configuring High Availability for VMware.

Table 3: Supported ESC Versions for Upgrading to ESC 4.4

Virtual Infrastructure Manager	Supported Versions for Backup and Restore Upgrade	Supported Versions for In-Service Upgrade
OpenStack	4.3, 4.2, 4.1	4.3, 4.2, 4.1
VMware	4.3, 4.2, 4.1	4.3, 4.2, 4.1

IMPORTANT NOTES

- ESC portal now displays the notification data that was present in the database, even after the upgrade. This feature is supported only from ESC 2.1. If you are upgrading from 1.1 to 2.1 or later, you will not be able to see the notifications from the 1.1 release on the ESC portal as this data was not present in the database.
- After upgrading to the new ESC version, ESC service will manage the life cycle of all VNFs deployed in the previous release. To apply any new features (with new data models) to the existing VNFs, you must undeploy and redeploy these VNFs.

- Upgrading Standalone ESC Instance, on page 104
- Upgrading ESC HA Instances, on page 106
- In-Service Upgrade of the ESC HA Nodes in OpenStack, on page 108
- In-Service Upgrade of the ESC HA Nodes in Kernel-Based Virtual Machine (KVM), on page 111
- In-Service Upgrade of the ESC HA Nodes in VMware, on page 115

Upgrading Standalone ESC Instance

To upgrade standalone ESC instance, perform the following tasks:

1. Back up the ESC database. For more information, see Backup the Database from the ESC Standalone Instances.

Note To backup any custom scripts used by the deployments, and for restoring them before restoring the database, you must re-install the scripts on the backup as well.

2. Redeploy the ESC instance. For more information, see the below section, **Deploy the ESC for Upgrade**.
3. Restore the ESC database on the new ESC instance. For more information, see the below section, **Restoring the ESC Database**.

Deploy the ESC for Upgrade

After backing up and shutting down of the old ESC VM, a new/upgraded (based on new ESC package) ESC VM should be installed. All parameters for ESC installation should be the same as the old ESC VM deployment.

- For OpenStack, you need to register the new ESC qcow2 image using the Glance command with a new image name and then use new bootvm.py script and new image name to install ESC VM.

Note In OpenStack, if an old ESC VM was assigned with floating IP, the new ESC VM should be associated with the same floating IP after the installation.

- For VMWare, you need to use the new ESC OVA file to install ESC VM. All other configurations and property values should be the same as the old VM.

Restoring the ESC Database

Restore the ESC database on the new ESC instance , using the following procedure:

Step 1 Connect to the new ESC instance using SSH.

```
$ ssh USERNAME@NEW_ESC_IP
```

Step 2 Switch to the root user.

```
$ sudo bash
```

Step 3 Stop the ESC service.

```
$ sudo escadm stop
```

Step 4 Check ESC service status to make sure all the services are stopped.

```
$ sudo escadm status
```

Step 5 Restore the database files.

```
$ scp://<username>:<password>@<backup_ip>:<filename>
$ sudo escadm restore --file /path/where/file/scp-ed/to/db.tar.bz2
```

Step 6 Restart the ESC service:

```
$ sudo escadm restart
```

After ESC service is started, the standalone ESC upgrade is complete. You can check the health of the new ESC service by running `$ sudo escadm status` in the new ESC VM.

Step 7 In Openstack, after restoring the database successfully, delete the old ESC instance:

```
$ nova delete OLD_ESC_ID
```

Important Notes:

After upgrading to the new ESC version, ESC service will keep doing life cycle management of all VNFs deployed by the old version. However, to apply any new features (with new data models) to the VNFs deployed by the ESC with old version is not guaranteed. If you want to apply any new feature of the new ESC version to existing VNFs, you have to undeploy and redeploy those VNFs.

Upgrading ESC HA Instances

To upgrade ESC HA nodes, perform the following tasks:

1. Back up the database from an old ESC HA primary instance. For more information, see Backup the Database from the ESC HA Instances.

Note To backup any custom scripts used by the deployments, and for restoring them before restoring the database, you must re-install the scripts on the backup as well.

2. Deploy new ESC HA nodes based on new ESC version. For more information, see the below section, **Deploy the ESC HA nodes for Upgrade**.
3. Restore the Database on Primary ESC instance (Standby ESC instance will sync with the Primary ESC instance). For more information, see the below section, **Restoring the ESC Database on New Master and Standby Instances**.

Deploying the ESC HA nodes for Upgrade

After backing up and shutting down the two old ESC VMs, based on new ESC package install the new ESC VMs.

- For OpenStack, you need to register the new ESC qcow2 image using the Glance command with a new image name and then to use new bootvm.py script and new image name to install ESC VM. All other bootvm.py arguments should be the same as used to setup an old VMs.
- For VMWare, there are two steps to bring up HA pair in VMware: 1) setup two standalone instances 2) reconfigure each instance with HA info. All other configurations and property values should be the same as the old VMs.
- If VIP is used for Northbound access, keep VIP the same for the new deployment as used to reconfigure the old HA pair.

Restoring the ESC Database on New Master and Standby ESC Instances

Shut down the Standby ESC instance.

Step 1 Connect to the standby ESC instance using SSH.

```
$ ssh USERNAME@ESC_STANDBY_IP
```

Step 2 Verify that the ESC instance is standby and note the name of the standby ESC HA instance :

```
$ sudo escadm status
```

If the output value shows "BACKUP", the node is the standby ESC node.

Note If a dynamic mapping file (dynamic_mapping.xml) is used by ESC service, the dynamic mapping file should be restored into the backup ESC VM. Before power off the standby ESC node, you need to copy the backup dynamic mapping file (dynamic_mapping.xml) to the path */opt/cisco/esc/esc-dynamic-mapping/* .

Step 3 Shutdown the standby ESC instance through OpenStack Kilo/Horizon using Nova command. For ESC VM instances based in VMware vSphere, shutdown the primary instance through VMware client dashboard. An example of shutting down the standby ESC instance in OpenStack is shown below:

```
$ nova stop NEW_ESC_STANDBY_ID
```

Restore the database on the new Master ESC instance.

Step 4 Connect to the primary ESC instance using SSH.

```
$ ssh USERNAME@ESC_MASTER_IP
```

Step 5 Switch to the root user.

```
$ sudo bash
```

Step 6 Verify that the ESC instance is primary.

```
$ sudo escadm status
```

If the output value shows 'MASTER' , the node is the master ESC node.

Step 7 Stop the ESC services on the master node and verify the status to ensure the services are stopped.

```
$ sudo escadm stop
$ sudo escadm status
```

Step 8 Restore the database files.

```
$ sudo escadm restore --file /tmp/db.tar.bz2
$ scp://<username>:<password>@<backup_ip>:<filename>
```

Note If a dynamic mapping file (dynamic_mapping.xml) is used by ESC service, the dynamic mapping file should be restored into the ESC VM. Before starting the ESC node, you need to copy the backup dynamic mapping file (dynamic_mapping.xml) to the path */opt/cisco/esc-dynamic-mapping/*.

Step 9 Reboot the VM to restart the full ESC service:

```
$ sudo escadm restart
```

Step 10 Use the `$ sudo escadm status` to check the status of the ESC service.

Step 11 Start the standby ESC node.

Power on the standby ESC node through OpenStack Nova/Horizon or VMware client. After starting the standby node, ESC HA upgrade process should be complete.

Step 12 Delete the old HA instance through OpenStack Nova/Horizon or VMware client. An example of deleting the VM on OpenStack is shown below:

```
$ nova delete OLD_ESC_MASTER_RENAMED OLD_ESC_STANDBY_RENAMED
```

Upgrading VNF Monitoring Rules

In ESC 2.1 and earlier, mapping the actions and metrics defined in the datamodel to the valid actions and metrics available in the monitoring agent is enabled using the *dynamic_mappings.xml* file. The file is stored in the ESC VM and can be modified using a text editor. ESC 2.2 and later do not have an *esc-dynamic-mapping*

directory and *dynamic_mappings.xml* file. The CRUD operations for mapping the actions and the metrics is available through REST API.

To upgrade the VNF monitoring rules, you must back up the *dynamic_mappings.xml* file and then restore the file in the upgraded ESC VM. For more information, see the backup and restore procedures. For upgrade of HA instance, see Upgrading ESC HA Instances. For upgrade of the standalone instance, see Upgrading Standalone ESC Instance.

In-Service Upgrade of the ESC HA Nodes in OpenStack

In-Service upgrade in OpenStack using ESC RPM packages

Procedure

	Command or Action	Purpose
Step 1	Backup ESC database and log files.	
Step 2	Log into the ESC HA secondary VM and stop the escadm service.	`$ sudo escadm stop`

Step 3	Ensure the ESC VM is in STOP state. ESC may take some time to switch to the STOP state. If ESC status turns into STOP state, please note that it won't be the part of HA cluster and you will lose HA function temporarily.	`$ sudo escadm status`
		
		`Expected output:`
		`ESC status=0 ESC HA is stopped`

Step 4	Copy the RPM file for upgrade to the ESC VM and execute the rpm command for upgrade.	`$ sudo rpm -Uvh /home/admin/cisco-esc-3.1.0-145.x86_64.rpm`
Step 5	Start the escadm service.	`$ sudo escadm start`
Step 6	Log into the ESC HA Primary VM and repeat step 3 to step 6 in Primary VM. Please note that after stop escadm service in Primary ESC VM, a failover will be triggered and the upgraded secondary VM will take over the Primary role.	
Step 7	Check the ESC version on each instance to verify the version is upgraded correctly and make sure ESC service is running properly in new Primary VM.	`# esc_version` `# health.sh (in Primary VM)`

In-Service upgrade in OpenStack using ESC qcow2 Image

Step 1 Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
# sudo escadm log collect
```

Note A timestamped file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

Step 2 Redeploy secondary ESC instance. Register new ESC image on the secondary instance, and wait for the data to be synchronized.

a) Delete the secondary instance through Horizon/Kilo using OpenStack Nova client. In OpenStack controller, running following command through nova client.

```
nova delete <secondary_vm_name>
```

b) Register new ESC image into OpenStack Glance for redeployment usage.

```
glance image-create --name <image_name> --disk-format qcow2 --container-format bare --file
<esc_qcow2_file>
```

c) Redeploy the secondary ESC VM instance based on newer image version. Re-install new the secondary instance by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the former ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to use the same values. Once the new ESC instance with upgraded version is up, it will be in secondary state.

d) Log into the new instance and run the following command to check the synchronization state of the new ESC node.

```
# drbd-overview
```

Wait until the output of drbd-overview show both nodes are "UpToDate" like the output below. It means the new ESC instance has completed the data synchronization from the primary instance.

```
esc/0 Connected Secondary/Primary UpToDate/UpToDate
```

Step 3 Stop keepalived service on Secondary instance, Power off primary instance, and then start Secondary keepalived service.

a) Log into the primary instance, set ESC primary node into maintenance mode.

```
$ sudo escadm op_mode set --mode=maintenance
```

Make sure there is no in-flight transaction ongoing before moving to the next step. To verify there are no in-flight transactions, use the following command:

```
For ESC 2.3:
$ sudo escadm ip_trans
```

For versions older than ESC 2.3, check escmanager log at (/var/log/esc/escmanager.log) and make sure there are no new transaction in escmanager log.

b) Log in to the upgraded secondary instance and shut down the ESC service.

```
$ sudo escadm stop
```

c) Power off the primary instance through OpenStack Nova client/Horizon and make sure it is off. In OpenStack Controller, run:

```
$ nova stop <primary_vm_name>
$ nova list | grep <primary_vm_name>
```

d) Log into the previously upgraded secondary instance which is in stopped state and restart the ESC service. The secondary ESC instance will take the primary role (switchover will be triggered) and start providing services with new version.

```
$ sudo escadm restart
```

Step 4 Check the ESC version on the new primary instance to verify the version is upgraded correctly.

```
$ sudo escadm status (check ha status)

Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version)
version : 3.x.x
release : xxx
```

Step 5 Re-deploy the old primary instance with the new ESC image.

Delete the old primary instance and redeploy it by using the new ESC package (bootvm.py and new registered image).

a) Log in to the new deployed instance and check ha status. The new instance should be in secondary state:

```
$ sudo escadm status --v
```

b) Run the following command to check the synchronization state of the new ESC secondary node:

```
# drbd-overview
```

Wait until the output of drbd-overview shown as UpToDate.

c) For the new ESC secondary node, make sure the health check is passed and the ESC version are upgraded correctly.

```
$ sudo escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy
$ esc_version (check esc version)version : 2.x.x
release : xxx
$ health.sh
Expected output:
ESC HEALTH PASSED
```

Step 6 Go back in to the first upgraded primary instance and check the health and keepalived state.

```
$ drbd-overview
Expected output:
1:esc/0  Connected Primary/Secondary UpToDate/UpToDate /opt/cisco/esc/esc_database ext4 2.9G 52M 2.7G
 2%

$ sudo escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version) Expected output:
version : 2.x.x
release : xxx

$ health.sh (check esc health)
Expected output:
ESC HEALTH PASSED
```

Note Quick rollback: In case of an upgrade failure, shutdown the upgraded instance and start the old primary instance to have a quick rollback.

Rollback Procedure for In-service Upgrade

1. Copy the database and log backup files to a location out of ESC VMs.
2. Delete any remaining ESC instance and redeploy ESC HA VMs using qcow2 image with old version.
3. Restore the database. Follow the procedures in the section, Upgrading ESC HA Instance with Backup and Restore for HA database restore.
4. After database restore, you should have ESC service back with the old version.

In-Service Upgrade of the ESC HA Nodes in Kernel-Based Virtual Machine (KVM)

In-Service Upgrade in KVM using ESC RPM packages

Use this procedure to upgrade ESC high-availability nodes with a minimum service interruption on a Kernel-based virtual machine.

Step 1 Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
$ sudo escadm log collect
```

Note A timestamped log file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

Step 2 Log into the ESC HA secondary VM and stop the ESC service.

```
$ sudo escadm stop
```

Step 3 Make sure the secondary ESC VM is in STOP state.

```
$ sudo escadm status --v
```

If ESC status=0 esc ha is stopped.

Step 4 In secondary VM, execute the rpm command for upgrade:

```
$ sudo rpm -Uvh /home/admin/cisco-esc-<latest rpm filename>.rpm
```

Step 5 Log into the primary instance, set ESC primary node into maintenance mode.

```
$ sudo escadm op_mode set --mode=maintenance
```

Make sure there are no in-flight transactions and no new transactions during the upgrade. From ESC 2.3, you may use following commands to check in-flight transactions.

```
$ sudo escadm ip_trans
```

For any build older than ESC 2.3, you may need to check escmanager log for transactions at (/var/log/esc/escmanager.log).

Step 6 Power off ESC primary node and make sure it is completely shut down. In KVM ESC controller, execute the following commands:

```
$ virsh destroy <primary_vm_name>

$ virsh list --all
```

Step 7 Log in the upgraded ESC instance (previous secondary one), start the ESC service. The upgraded VM will take over primary role and provide ESC service.

```
$ sudo escadm restart
$ start esc_monitor
```

Step 8 Check the ESC version on the new primary instance to verify the upgraded version is correct. Once it is in the Primary state, make sure ESC service is running properly in the new Primary VM.

```
$ sudo escadm status
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version

$ health.sh
Expected output:
ESC HEALTH PASSED
```

Step 9 Power on the old primary instance. In KVM ESC controller, execute the following commands:

```
$ virsh start <primary_vm_name>
```

Step 10 Log into the VM which is still with old ESC version and repeat step 2, 3, 4, and 7 in the VM.

In-Service Upgrade in KVM using ESC OVA Image

Step 1 Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
$ sudo escadm log collect
```

Note A timestamped log file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

Step 2 Redeploy secondary ESC instance. Register new ESC image on the secondary instance.

a) Delete the secondary instance through lib vert Virsh commands. On KVM host, run the following command:

```
$ Virsh destroy the <secondary_vm_name>
$ Virsh undefine --remove-all-storage <secondary_vm_name>
```

b) Copy the new ESC image into Kvm Host for redeployment usage:

```
sshpass -p "host Password' scp  /scratch/BUILD-2_x_x_x/BUILD-2_x_x_x/ESC-2_x_x_x.qcow2 root@HOSTIP:
```

c) Redeploy the secondary ESC VM instance based on newer image version. Re-install new the secondary instance by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the former ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to use the same values. Once the new ESC instance with upgraded version is up, it will be in secondary state.

d) Log into the new instance and run the following command to check the synchronization state of the new ESC node.

```
$ drbd-overview
```

wait until the output of drbd-overview show both nodes are "UpToDate" like the output below. It means the new ESC instance has completed the data synchronization from the primary instance.

```
esc/0 Connected Secondary/Primary UpToDate/UpToDate
```

Step 3 Stop keepalived service on Secondary instance, Power off primary instance, and then start Secondary keepalived service.

a) Log into the primary instance, set ESC primary node into maintenance mode.

```
$ sudo escadm op_mode set --mode=maintenance
```

Make sure there is no in-flight transaction ongoing before moving to the next step. To verify there are no in-flight transactions, use the following command:

```
For ESC 2.3:
$ sudo escadm ip_trans
```

For versions older than ESC 2.3, check escmanager log at (/var/log/esc/escmanager.log) and make sure there are no new transaction in escmanager log.

b) Log in to the upgraded secondary instance and shut down the keepalived service.

```
$ sudo escadm stop
```

c) Power off the primary instance and make sure it has been completely turned off. In KVM ESC Controller, run:

```
$ virsh destroy <primary_vm_name>
$ virsh list --all
```

d) Log into the previously upgraded secondary instance which is in stopped state and start the ESC service. The secondary ESC instance will take the primary role (switchover will be triggered) and start providing services with new version.

```
$ sudo escadm restart
```

Step 4 Check the ESC version on the new primary instance to verify the version is upgraded correctly.

```
$ sudo escadm status (check ha status)

Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version)
version : 4.1.x
release : xxx

$ health.sh (check esc health)

Expected output:
ESC HEALTH PASSED
```

Step 5 Re-deploy the old primary instance with the new ESC image.

Delete the old primary instance and redeploy it by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the old ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to be the same values.

a) Log in to the new deployed instance and check ha status. The new instance should be in secondary state:

```
$ sudo escadm status
```

b) Run the following command to check the synchronization state of the new ESC secondary node:

```
$ drbd-overview
```

Wait until the output of drbd-overview shown as UpToDate.

c) For the new ESC secondary node, make sure the health check is passed and the ESC version are upgraded correctly.

```
$ sudo escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy
$ esc_version (check esc version)version : 4.1.x
release : xxx
$ health.sh
Expected output:
ESC HEALTH PASSED
```

Step 6 Go back in to the first upgraded primary instance and check the health and keepalived state.

```
$ drbd-overview
Expected output:
1:esc/0  Connected Primary/Secondary UpToDate/UpToDate /opt/cisco/esc/esc_database ext4 2.9G 52M 2.7G
 2%

$ sudo escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version) Expected output:
version : 2.x.x
release : xxx

$ health.sh (check esc health)
Expected output:
ESC HEALTH PASSED
```

Note Quick rollback: In case of an upgrade failure, shutdown the upgraded instance and start the old primary instance to have a quick rollback.

Rollback Procedure for In-service Upgrade

1. Copy the database and log backup files to a location out of ESC VMs.
2. Delete any remaining ESC instance and redeploy ESC HA VMs using qcow2 image with old version.
3. Restore the database. Follow the procedures in the section, Upgrading ESC HA Instance with Backup and Restore for HA database restore.
4. After database restore, you should have ESC service back with the old version.

In-Service Upgrade of the ESC HA Nodes in VMware

In-Service upgrade in VMware using ESC RPM packages

Use this procedure to upgrade the ESC high-availability nodes one node at a time with a minimum service interruption. This process leverages the ESC HA replication and failover capability to smoothly move ESC service to the new upgraded node without the manual database restore.

Step 1 Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
# sudo escadm log collect
```

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

Step 2 Log into the ESC HA secondary VM and stop the keepalived service.

```
$ sudo escadm stop
```

Step 3 Make sure the secondary ESC VM is in STOP state.

```
$ sudo escadm status --v
```

If ESC status=0 esc ha is stopped.

Step 4 In secondary VM, execute the rpm command for upgrade:

```
$ sudo rpm -Uvh /home/admin/cisco-esc-2.2.9-50.rpm
```

Step 5 Log into the primary instance, set ESC primary node into maintenance mode.

```
$ sudo escadm op_mode set --mode=maintenance
```

Make sure there are no in-flight transactions and no new transactions during the upgrade. From ESC 2.3, you may use following commands to check in-flight transactions.

```
$ sudo escadm ip_trans
```

For build older than ESC 2.3, you may need to check escmanager log and make sure no new transactions are recorded in this log file. The log file can be located at (/var/log/esc/escmanager.log).

Step 6 Power off ESC primary node. In VMware vSphare Client, select **Home > Inventory > VMs and Templates**, right click the primary instance name from the left panel, and select **Power > Power Off**.

Step 7 Log in to the upgraded ESC instance (previous secondary one), and start the keepalived service. The upgraded VM will take over primary role and provide ESC service.

```
$ sudo escamd restart
```

Step 8 Check the ESC version on the new primary instance to verify the upgraded version is correct. Once it is in the Primary state, make sure ESC service is running properly in the new Primary VM.

```
$ sudo escadm status
```

```
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version

$ health.sh
Expected output:
ESC HEALTH PASSED
```

Step 9 Power on the old primary instance. In VMware vSphare Client, select **Home > Inventory > VMs and Templates**, right click the primary instance name from the left panel, then select **Power > Power On**.

Step 10 Log into the VM which is still with old ESC version and repeat step 2, 3, 4, and 7 in the VM.

In-Service upgrade in VMware using ESC qcow2 Image

Step 1 Backup ESC database and log files.

a) Perform ESC database backup from primary node. For more information on backing up the database, see Backup the Database from the ESC HA Instances.

b) Collect and backup all logs from both primary and secondary VMs. To backup the log, use the following command:

```
# sudo escadm log collect
```

Note A timestamped log file will be generated in: /var/tmp/esc_log-<timestamp>.tar.bz2

c) Copy the database backup file and logs files (generated in /tmp/esc_log-.tar.bz2)* out of ESC VMs.

Step 2 Redeploy secondary ESC instance. Register new ESC image on the secondary instance, and wait for the data to be synchronized.

a) Delete the secondary instance. To delete the secondary ESC instance, you need to first "Power Off" the instance through vSphere Client and then use the **Delete from Disk** option. In VMware vSphare Client, select **Home > Inventory > VMs and Templates**, right click the instance name from the left panel, then select **Power > Power Off**. Now to delete the secondary instance, select **Home > Inventory > VMs and Templates**, right click the instance name from the left panel, then select **Delete from Disk**.

b) Redeploy the secondary ESC VM instance based on newer image version. Re-install new the secondary instance by using the new ESC package (bootvm.py and new registered image). Once the new ESC instance with upgraded version is up, it will be in secondary state.

c) Log into the new instance and run the following command to check the synchronization state of the new ESC node.

```
$ drbd-overview
```

Wait until the output of drbd-overview show both nodes are "UpToDate" like the output below. It means the new ESC instance has completed the data synchronization from the primary instance.

```
esc/0 Connected Secondary/Primary UpToDate/UpToDate
```

Step 3 Stop keepalived service on Secondary instance, Power off primary instance, and then start Secondary keepalived service.

a) Log into the primary instance, set ESC primary node into maintenance mode.

```
$ sudo escadm op_mode set --mode=maintenance
```

Make sure there is no in-flight transaction ongoing before moving to the next step. To verify there are no in-flight transactions, use the following command:

```
For ESC 2.3:
$ sudo escadm ip_trans
```

For versions older than ESC 2.3, check escmanager log at (/var/log/esc/escmanager.log) and make sure there are no new transaction in escmanager log.

b) Log in to the upgraded secondary instance and shut down the keepalived service.

```
$ sudo escadm stop
```

c) Power off the primary instance and make sure the primary instance has been powered off. In VMware vSphare Client, select **Home > Inventory > VMs and Templates**, right click the instance name from the left panel, then select **Power > Power Off**.

d) Log into the previously upgraded secondary instance which is in stopped state and start the keepalived service. The secondary ESC instance will take the primary role (switchover will be triggered) and start providing services with new version.

```
$ sudo escadm start
```

Step 4 Check the ESC version on the new primary instance to verify the version is upgraded correctly.

```
$ sudo escadm status --v(check ha status)

Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version)
version : 3.x.x
release : xxx

$ health.sh (check esc health)

Expected output:
ESC HEALTH PASSED
```

Step 5 Re-deploy the old primary instance with the new ESC image.

Delete the old primary instance and redeploy it by using the new ESC package (bootvm.py and new registered image). All other installation parameters should be the same as the old ESC VM deployment. For example, hostname, ip address, gateway_ip, ha_node_list, kad_vip, kad_vif have to be the same values. To delete, in the VMware vSphare Client, access, **Home > Inventory > VMs and Templates**, right click the instance name from the left panel, then select **Delete from Disk**.

a) Log in to the new deployed instance and check ha status. The new instance should be in secondary state:

```
$ sudo escadm status
```

b) Run the following command to check the synchronization state of the new ESC secondary node:

```
$ drbd-overview
```

Wait until the output of drbd-overview shown as UpToDate.

c) For the new ESC secondary node, make sure the health check is passed and the ESC version are upgraded correctly.

```
$ sudo escadm status (check ha status)
Expected output:
```

```
0 ESC status=0 ESC Master Healthy
$ esc_version (check esc version)version : 3.x.x
release : xxx
$ health.sh
Expected output:
ESC HEALTH PASSED
```

Step 6 Go back in to the first upgraded primary instance and check the health and keepalived state.

```
$ drbd-overview
Expected output:
1:esc/0  Connected Primary/Secondary UpToDate/UpToDate /opt/cisco/esc/esc_database ext4 2.9G 52M 2.7G
 2%

$ sudo escadm status (check ha status)
Expected output:
0 ESC status=0 ESC Master Healthy

$ esc_version (check esc version) Expected output:
version : 3.x.x
release : xxx

$ health.sh (check esc health)
Expected output:
ESC HEALTH PASSED
```

Note Quick rollback: In case of an upgrade failure, shutdown the upgraded instance and start the old primary instance to have a quick rollback.

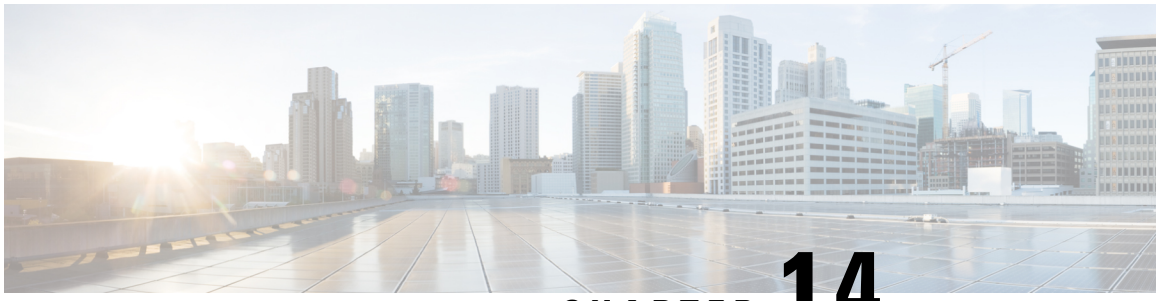
Rollback Procedure for In-service Upgrade

1. Copy the database and log backup files to a location out of ESC VMs.
2. Delete any remaining ESC instance and redeploy ESC HA VMs using qcow2 image with old version.
3. Restore the database. Follow the procedures in the section, Upgrading ESC HA Instance with Backup and Restore for HA database restore.
4. After database restore, you should have ESC service back with the old version.

PART VIII

Troubleshooting Cisco Elastic Services Controller Installation

- Troubleshooting ESC Issues, on page 121



CHAPTER 14

Troubleshooting ESC Issues

This chapter contains the following sections:

- ESC System Logs, on page 121
- General Installation Errors, on page 126

ESC System Logs

Log messages are created for ESC events throughout the VNF lifecycle. These can be external messages, messages from ESC to other external systems, error messages, warnings, events, failures and so on. The log file can be found at `/var/log/esc/escmanager_tagged.log`.

The log message format is as follows:

```
date=<time-date>] [loglevel=<loglevel>] [tid=<transactionid>] [cl=<classifications>]
[tags=<tags>] [msg=<message>
```

Sample log is as follows:

```
date=15:43:58,46022-Nov-2016]
[loglevel=ERROR ] [tid=0793b5c9-8255-47f3-81e6-fbb59f6571f7] [cl=OS ]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:CSCvd94541,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:test,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0] [msg=sleepingfor5seconds
to allow vm to become ACTIVE instance id:
162344f7-78f9-4e45-9f23-34cf87377fa7
name:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0
```

When a request is received, a RequestDetails object is created which autogenerates a unique transaction id. This value is carried forward across all threads. Classifications and tags are optional. These are prefixes added to the log messages to enhance readability, and help in debugging. With classifications and tags, the log messages can be easily parsed and filtered by the log analysis tools.

The following classifications are supported:

NBI	"com.cisco.esc.rest""com.cisco.esc.filter"(North Bound Interface - Clientinterface)
SBI	"com.cisco.esc.rest"- source is a callback handler or"EventsResource"(South Bound Interface - i.e. between ESC and the VIM)

SM	"com.cisco.esc.statemachines". stands for StateMachine. This classification indicates logs in the StateMachine category.
MONITORING	"com.cisco.esc.monitoring""com.cisco.esc.paadaptor"(MONA related logs)
DYNAMIC_MAPPING	"com.cisco.esc.dynamicmapping""com.cisco.esc.db.dynamicmapping"(MONA related logs)
CONFD	"com.cisco.esc.confd"
CONFD_NOTIFICATION	"com.cisco.esc.confd.notif""com.cisco.esc.confd.ConfdNBIAdapter"
OS	"com.cisco.esc.vim.openstack"
LIBVIRT	"com.cisco.esc.vim.vagrant
VIM	"com.cisco.esc.vim"
REST_EVENT	"ESCManager_Event""com.cisco.esc.util.RestUtils". indicates REST notifications in logs.
WD	"com.cisco.esc.watchdog"
DM	"com.cisco.esc.datamodel""com.cisco.esc.jaxb.parameters"(Datamodel and resource objects)
DB	"com.cisco.esc.db"(Database related logs)
GW	"com.cisco.esc.gateway"
LC	"com.cisco.esc.ESCManager"(Start up related logs)
SEC	"com.cisco.esc.jaas"
MOCONFIG	"com.cisco.esc.moconfig"(MOCONFIG object related logs --this is internal for ESC developers)
POLICY	"com.cisco.esc.policy"(Service/VM Policy related logs)
TP	"com.cisco.esc.threadpool"
ESC	"com.cisco.esc" Any other packages not mentioned above

The following tags are supported:

- **Workflow [wf:]**—Generated using action and resource from RequestDetails object. Example "wf:create_network"
- **Event type [eventType:]**—Event that triggered the current action. Example: "eventType:VM_DEPLOY_EVENT"
- **Resource based**—These values are generated based on the type of parameter used by the event. The hierarchy, that is, the tenant, the vm group and so on is added to the log.

Tenant	[tenant:<tenant name>]

Network	[tenant:<tenant id>, network:<network name>] **Note** The tenant appears only if applicable.
Subnet	[tenant:<tenant name or id>, network:<network name or id>, subnet:<subnet name>] **Note** The tenant appears only if applicable.
User	[tenant:<tenant name>, user:<user name or id>] **Note** The tenant appears only if applicable.
Image	[image:<image name>]
Flavor	[flavor:<flavor name>]
Deployment	[tenant:<tenant name or id>, depName:<deployment name>]
DeploymentDetails	[tenant:<tenant name or id>, depName:<deployment name>, vmGroup:<vm group name>, vmName:<vm name>]
Switch	[tenant:<tenant name or id>, switch:<switch name>]
Volume	[volume:<volume name>]
Service	[svcName:<Service Registration name>]

Further, ESC logs can also be forwarded to an rsyslog server for further analysis and log management.

Filtering Logs Using Confd APIs

You can query and retrieve logs (for example, deployment logs, or error logs) in ESC using log filters introduced in the confd APIs. New filters for Tenant, Deployment Name, and VM Name are introduced. This enables you to query the ESC logs further for most recent error logs using the log filters in Confd APIs. You can also retrieve ESC logs related to the communication between ESC and the OS (by setting the classification tag to "OS").

The log format to retrieve confd API logs:

```
date=<time-date>] [loglevel=<loglevel>] [tid=<transactionid>] [cl=<classifications>]
[tags=<tags>] [msg=<message>
```

The sample log is as follows:

```
date=15:43:58,46022-Nov-2016] [loglevel=ERROR ] [tid=0793b5c9-8255-47f3-81e6-fbb59f6571f7]
 [cl=OS ]
[tags=wf:create_vm,eventType:VM_DEPLOY_EVENT,tenant:test,depName:test-dep,vmGrpName:test-VNF,
vmName:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0]
[msg=sleepingfor5seconds to allow vm to become ACTIVE instance id:
162344f7-78f9-4e45-9f23-34cf87377fa7 name:test-dep_test_0_dc3f406c-05ca-43b3-af21-0841e3b029a0
```

The parameters for log level, classification and tags are dependent on each other to retrieve the logs. You can successfully retrieve the logs with the following combination.

- log_level=ERROR, classifications=OS, tags=(depName:test-dep)
- log_level=ERROR, classifications=OS, tags=(tenant: test)

The log filter returns a value when all the following conditions are met:

- Log level
- Classifications (if provided)
- Tags (if provided)

Note If there are more than one classification listed, it has to match at least one of the classifications. The same applies to the tags as well.

For example, the following log filter criteria does not return the log sample mentioned earlier:

```
log_level=ERROR, classifications=VIM, tags=(depName:test-dep)
```

It does not return any value though the log level and tags match, the classification VIM does not match.

The data model is as follows:

```
rpc filterLog {
    description "Query and filter escmanager logs using given parameters";
    tailf:actionpoint escrpc;
    input {
      leaf log_level {
        mandatory false;
        description "One of DEBUG / INFO / WARNING / ERROR / TRACE / FATAL. Results will
include all logs at and
                     above the level specified";
        type types:log_level_types;
        default ERROR;
      }
      leaf log_count {
        mandatory false;
        description "Number of logs to return";
        type uint32;
        default 10;
      }
      container classifications {
        leaf-list classification {
         description "Classification values to be used for the log filtering.  For example:
 'OS', 'SM'.
                      Logs containing any of the provided classification values will be
returned.";
          type types:log_classification_types;
        }
      }
      container tags {
        list tag {
          key "name";
          leaf name {
            mandatory true;
           description "Tag name to be used for the log filtering.  For example: 'tenant',
 'depName'.
                        Logs containing any of the provided tag name plus the tag values
will be returned.";
            type types:log_tag_types;
          }
          leaf value {
            mandatory true;
            description "Tag value pairs to be used for the log filtering.  For example:
'adminTenant', 'CSRDeployment'";
            type string;
          }
```

```
      }
    }
  }
  output {
    container filterLogResults {
      leaf log_level {
        description "Log level used to filter for the logs.";
        type types:log_level_types;
      }
      list logs {
        container classifications {
          leaf-list classification {
            description "Classifications used to filter for the logs.";
            type types:log_classification_types;
          }
        }
        container tags {
          list tag {
            key "name";
            leaf name {
              mandatory true;
              description "Tag name used to filter for the logs.";
              type types:log_tag_types;
            }
            leaf value {
              mandatory true;
              description "Tag value used to filter for the logs.";
              type string;
            }
          }
        }
        leaf log_date_time {
          description "Timestamp of the log.";
          type string;
        }
        leaf log_message {
          description "The log message.";
          type string;
        }
      }
    }
  }
}
```

You can query for the confd API logs through the netconf console or esc_nc_cli

- Through the netconf-console, run the following query:

  ```
  /opt/cisco/esc/confd/bin/netconf-console --port=830 --host=127.0.0.1 --user=admin
  --privKeyFile=/home/admin/.ssh/confd_id_dsa --privKeyType=dsa --rpc=log.xml
  ```

- Using the esc_nc_cli, run the following query:

  ```
  ./esc_nc_cli filter-log log.xml
  ```

  The sample log.xml is as follows:

  ```
  <filterLog xmlns="http://www.cisco.com/esc/esc">
    <log_level>INFO</log_level>
    <log_count>1</log_count>
    <classifications>
      <classification>OS</classification>
      <classification>SM</classification>
    </classifications>
    <tags>
  ```

```
    <tag>
      <name>depName</name>
      <value>CSR_ap1</value>
    </tag>
    <tag>
      <name>tenant</name>
      <value>admin</value>
    </tag>
  </tags>
</filterLog>
```

The response is as follows:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <filterLogResults xmlns="http://www.cisco.com/esc/esc">
    <log_level>INFO</log_level>
    <logs>
      <classifications>
        <classification>OS</classification>
        <classification>SM</classification>
      </classifications>
      <tags>
        <tag>
          <name>depName</name>
          <value>CSR_ap1</value>
        </tag>
        <tag>
          <name>tenant</name>
          <value>admin</value>
        </tag>
      </tags>
      <log_date_time>13:06:07,575 31-Oct-2016</log_date_time>
      <log_message> No pending work flow to start.</log_message>
    </logs>
  </filterLogResults>
</rpc-reply>
```

Note The logging API responses are in XML format. If the log messages contain any XML characters, then the characters will be escaped so not to break the XML conformance.

General Installation Errors

This section cover some of the common installation problems and how to troubleshoot them.

Problem/Error	Possible Reason	User Action

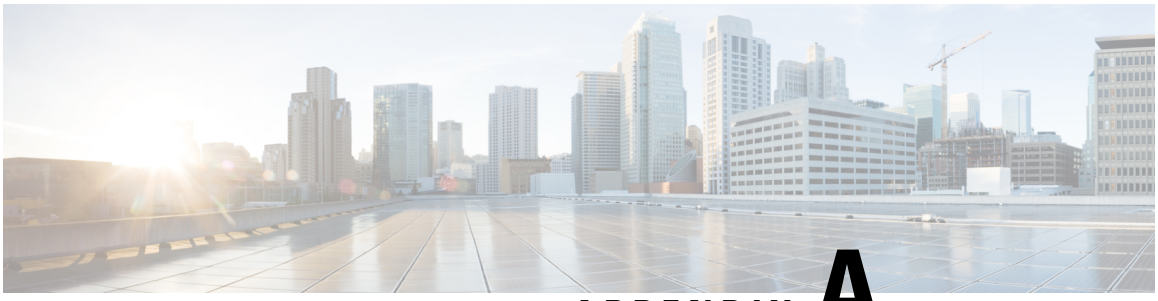
Issues encountered while verifying ESC services status at the installation time using `sudo escadm status`	Some services take time to start or have trouble starting.	**1.** Identify the issues using one of the following method:
		• Check the log /var/log/esc/escadm.log
		`$ cat /var/log/esc/escadm.log`
		• Add '-v' to escadm status to show the verbose output of the ESC services and see the services are up and running.
		2. Confirm the status of the identified services that has issues and manually start these services.
		`$ sudo escadm <<service>> status//`
		`If the status is stopped or dead, manually start the services`
		`using the next command.`
		`$ sudo escadm <<service>> start --v`

Authentication related error during the ESC installation	OpenStack credential-related arguments are missing.	Source your OpenStack RC file and verify the OpenStack clients work properly
ESC HA-related Issues		
Network Problem		Check for the following: • The static IP addresses for both ESC nodes used are based on the OpenStack configuration. • The gateway for each network interface is accessible.

Problem/Error	Possible Reason	User Action
ESC master node stays in the "switching-to-master" state	This could be because of the following issues: • ESC HA node can not reach its peer during the initial installation. • ESC service (tomcat) can't start properly due to database problems (etc. database migration, database file corruption). • Confd can't start due to the CDB file corruption. • Postgresql can't start or init due to the file system issues. • The connection between ESC nodes is too slow.	Check for the: • Connectivity between ESC master node and standby node. For initial installation, ESC master service won't be up if it can't reach the standby node. You need to make sure you have both ESC nodes successfully deployed and they can reach each other. • ESC logs at /var/log/esc/esc_haagent.log (ESC 2.X) or /var/log/esc/escadm.log (ESC 3.X) to identify the services that has issues. • Log at /var/log/esc/escmanager.log for esc_service and postgresql issues.
MTU issues for ESC HA		For the ESC VMs, reduce the MTU for network interfaces from 1500 to 1450. Follow the below steps to reduce the MTU value: 1. Identify the interface you would like to change. Access the interface from the location, /etc/sysconfig/network-scripts/ifcfg-ethX, where X represent the interface number you want to change. 2. Use a text editor like VIM to add or edit the MTU items for the interface, for instance, set MTU = 1450 3. Restart the interface. `network service restart`

Problem/Error	Possible Reason	User Action

Other Issues		There are several useful logs to check for ESC HA troubleshooting.
		• ESC manager log is located at /var/log/esc/escmanager.log
		• ESC HA log about esc service startup/stop is located at /var/log/esc/esc_haagent.log for ESC 2.X and /var/log/esc/escadm.log for ESC 3.X
		• For Keepalived configuration:
		  • Check the configuration file at /etc/keepalived/keepalived.conf
		  • Keepalived log is located at /var/log/messages by grep keepalived or vrrp
		• For DRBD configuration:
		  • DRBD configuration can be verified by checking the file at /etc/drbd.d/esc.res
		  • The DRBD log is located at /var/log/messages by grep drbd



APPENDIX A

Cisco Elastic Services Controller Installer Arguments

You need to specify the following *bootvm.py* script arguments to boot ESC instances.

Arguments	Description
esc_hostname	Specifies the host name of the ESC VM instance.
--image	Specifies the image id used in the OpenStack glance to boot up the ESC instance.
--boot_volume	Specify the volume name or id of the external bootable volume from where you want to launch ESC instance.
--net	Specifies the Network IDs or names in OpenStack that ESC connects to.
--ipaddr	(Optional) Specifies the IP addresses that ESC will be assigned in the network. **Note** The IP address must correspond to the net_id in the --net argument.
--gateway_ip	(Optional) Specifies the default gateway IP address of ESC.
--os_auth_url	(Optional) Specifies the OpenStack keystone url used by os_auth_url for authentication.
--os_username	(Optional) Specifies the OpenStack keystone username used by os_username for authentication.
--os_password	(Optional) Specifies the OpenStack keystone password used by os_password for authentication.
--os_tenant_name	(Optional) Specifies the OpenStack tenant name used by os_tenant_name for ESC deployment.
--bs_os_auth_url	(Optional) Specifies the OpenStack keystone url used by bs_os_auth_url for authentication.

Cisco Elastic Services Controller Installer Arguments

Arguments	Description
--bs_os_username	(Optional) Specifies the OpenStack keystone username used by bs_os_username for authentication.
--bs_os_password	(Optional) Specifies the OpenStack keystone password used by bs_os_password for authentication.
--bs_os_tenant_name	(Optional) Specifies the OpenStack tenant name used by bs_os_tenant_name for ESC deployment.
--flavor	(Optional) Specifies the OpenStack flavor id to boot the ESC VM.
--security_rules_file	(Optional) Specifies the file to define security rules (IP, Port security) for ESC VM.
--etc_hosts_file	(Optional) Specifies the file for adding more entries to the ESC vm's hosts file (/etc/hosts).
--avail_zone	(Optional) Specifies the OpenStack zone used for ESC deployment.
--esc_params_file	(Optional) Specifies the default parameter file for ESC deployment.
--db_volume_id	(Optional) Specifies the cinder volume id to mount for database storage in ESC HA [ESC-HA].
-- ha_node_list	(Optional) Specifies list of IP addresses for HA nodes in the Primary/Standby cluster. For ESC nodes with multiple network interfaces, these IPs should be the addresses in the network used for data synchronization. **Note** This argument is utilized for replication-based HA solution only.
--kad_vip	(Optional) Specifies the IP address for Keepalived VIP (virtual IP) plus the interface of Keepalived VIP [ESC-HA]. An example format for specifying the interface of VIP is --kad_vip 192.0.2.1:eth2 or --kad_vip [2001:cc0:2020::fc]:eth2
--kad_vif	(Optional) Specifies the interface for Keepalived virtual IP and keepalived VRRP [ESC-HA]. You can also use this argument to only specify the interface for Keepalived VRRP, if the VIP interface is already specified using the *kad_vip* argument.
--kad_vri	Specified the virtual router id of vrrp instance. Accepted values for kad_vri are 0 to 254. ESC VMs in the same HA should use the same kad_vri number. If kad_vip is not used for L3 HA, the kad_vir has to be used, otherwise, you can skip kad_vri argument.
--route	Specifies the routing configuration for ESC VM.
--ntp_server	(Optional) Specifies the NTP server address.

Arguments	Description
--rsyslog_server	(Optional) Specifies the IP address of rsyslog server that ESC sends the log to
--rsyslog_server_port	(Optional) Specifies the port of rsyslog server that ESC sends the log to.
--rsyslog_server_protocol	(Optional) Specifies the protocol to be used by the ESC to forward logs to the server.
--secure	(Optional) Enables secure configuration. You can specify the following values: • **A**—Root is completely locked out. You cannot login as a root even from the console. • **B**—SELinux runs in the enforcing mode. • **C**—IPv4/IPv6 tables are started. • **D**—SSH password authentication is disabled. You need the private key to ssh into ESC vm. • **E**—host keys for confd will be re-created.
--host_mapping_file	(Optional) Specifies the host mapping file for VNF deployment.
--version	(Optional) Prints the version of bootvm.py and exits.
--rng_virtio	Enables installing and deploying the ESC VM on Libvirt/KVM with the RNG Virtio device. The default values are: device=/dev/random rate_period=1000 rate_bytes=1024

Arguments	Description

--user_pass	This along with --user_confd_pass are mandatory arguments from 3.0 onwards.
	
	This argument adds a user to access the ESC VM. Use this argument to specify a user without administrative privileges, i.e, a non-admin/non-root user. Use the following format: user_name:password. The bootvm.py command requires at least one --user_pass argument to create an admin account for linux (ssh/console access) . The following is the syntax for the mandatory user credential argument:
	
	`--user_pass admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE][:OPTIONAL-ROLE]`
	
	This user can only do the following:
	• Login to ESC through SSH.
	• Access and drive the Netconf CLI, such as, esc_nc_cli, netconf-console, and so on.
	• Read ESC -related logs from /var/logs/esc
	• Access REST interface through localhost
	
	This user cannot:
	• Access the ESC DB and reconfigure ESC system.
	• Access the system-level logs
	• Configure the system level components, such as: Rsyslog, Keepalived, DRDB, and so on.
	• Access the encryption keys and values from REST interface or ESC logs.
	
	Following is an example of --user_pass for admin account and stronger clear text passwords. Use single quotes to avoid conflict with shell reserved characters:
	
	-user_pass admin:'Strong4Security!'.
	
	Another example to install ESC using a password hash for both admin accounts. Use single quotes to avoid conflict with shell reserved characters:
	
	--user_pass admin:'$algorithm$salt$hash-of-salt-password'.
	
	ESC 2.1 and later, accepts the public key for this attribute. For example, the following will generate 'admin321' as the password for user 'admin' and use */tmp/abc.pub* as the key file to inject the public key for it:
	
	`--user_pass admin:admin321:/tmp/abc.pub`

Arguments	Description
--user_confd_pass	Used to change confd users. The bootvm.py command requires at least one --user_confd_pass to create an admin account for ConfD (netconf/cli access). The following is the syntax for the mandatory user credential argument: `--user_confd_pass admin:'PASSWORD-OR-HASH'[:OPTIONAL-PUBLIC-KEY-FILE]` Following is an example of --user_confd_ pass for admin account and stronger clear text passwords. Use single quotes to avoid conflict with shell reserved characters: --user_confd_ pass:'Strong4Security!'. Another example, to install ESC using a password hash for both admin accounts. Use single quotes to avoid conflict with shell reserved characters: --user_confd_ pass:'$algorithm$salt$hash-of-salt-password'. ESC 2.1 and later, accepts the public key for this attribute. For example, the following will generate 'admin321' as the password for user 'admin' and use */tmp/abc.pub* as the key file to inject the public key for it: --user_confd_ pass:admin321:/tmp/abc.pub
--esc_portal_startup	(Optional) Starts the ESC portal.
--log	(Optional) Specifies the log file. By default, logs to stdout.
--esc_monitor_check_ips	(Optional) Specifies the IP addresses that must be monitored by esc_monitor (for HA failover).
--enable-https-rest	(Optional) Enables a secure REST Interface for the created ESC VM.
--enable-http-rest	(Optional) Enables an unsecured REST Interface for the created ESC VM.
--disable-rest-auth	(Optional) Disables REST API authentication. **Note** REST authentication should not be disabled in a production environment.
--enable-snmp-agent	(Optional) Enables automatic start-up of the SNMP service. The default value is False.
--ha_mode	Specifies the ESC HA mode for HA installation. Specify one of the following available options for HA: **no_ha**: No HA, **cinder**: Shared Cinder Volume, **drbd**: Built-in DRBD, **drbd_on_cinder**: DRBD over Cinder Volume
--enable-https-etsi	(Optional) Enables a secure ETSI REST Interface for the created ESC VM.

Arguments	Description
--enable-http-etsi	(Optional) Enables an unsecured ETSI REST Interface for the created ESC VM. Enabling this interface is not recommended in a production environment.
--encrypt_key	Specifies the key for encryption.
--proxy	Uses the proxy on a given port.
--noproxy	Lists the hosts which do not use proxy.
--kad_unicast_src_ip	Specifies the source IP address of unicast. Should be the IP address of interface that ESC VM uses for unicast (L3) VRRP communication. Example: --kad_unicast_src_ip 10.0.0.1
--kad_unicast_peer	Specified the peer IP addresses of unicast. Should be the ip address of interface that ESC peer VM uses for unicast (L3) VRRP communication. Example: --kad_unicast_peer 10.0.0.1
--placement_hint	Use this argument to specify the placement of ESC HA virtual machines using the server group, samehost, differenthost filters. Example: • --placement_hint different_host=2b299428-e7a7-4528-8566-9a4970183c6a [ID should be the VM uuid] • --placement_hint same_host=2b299428-e7a7-4528-8566-9a4970183c6a [ID should be the VM uuid] • --pacement_hint group=4c7758ab-e9cb-4cf0-8f02-344ec666365b [ID should be the server group uuid]
--format {json}	Use this argument to capture the success and failure message in the output. Example: $ ./bootvm.py --image ESC-2_3_0_8 --net network --format json --test-0 { "status" : "Success" , "vm_uuid" : "UUID" }
--user_rest_pass	Adds a user to access the Rest API. Format is username: password. This option can be repeated.
--user_portal_pass	Add a portal user. Format username: password. This option can be repeated.
--user_etsi_pass	Adds a user to access the ETSI REST API. Format username:password. This option can be repeated.

Arguments	Description
--no_vim_credentials	Use this argument to deploy ESC without passing the VIM credential. If this argument is used, following parameters will not be passed during the installation: • --os_auth_url • --os_username • --os_password • --os_tenant_name After the deployment is complete, the user can set these VIM credential through ESC's VIM/VIM User APIs (REST/Netconf). For more information on configuring through REST APIs and Netconf, see *Configuring VIM credentials after installing ESC* in the Post Installation Tasks chapter.
--etsi_startup	This argument is deprecated in ESC 4.4 and above, it is unavailable in future releases. The use of --etsi_startup shows an error message with the appropriate replacement argument to use. See --enable-etsi-http and --enable-etsi-https.

Cisco Elastic Services Controller Installer File Reference

File	Description
security_rules_file	The file contains the following: • Security rules to create a security group for the tenant. • Configurations to allow traffic for the tenant.
etc_hosts_file	The file contains one or more entries that you want to include in the /etc/hosts file.
esc_params_file	The file contains information to configure various parameters of ESC. For details on parameters that can be configured in the esc_params_file are described in table below.
host_mapping_file	The file contains information to map a network based on the hosts.

ESC Configuration Parameters

Using this file, you can configure various ESC parameters during the installation. The parameters that can be configured are shown in the table.

Below is an example configuration using this file:

```
openstack.endpoint=adminURL
affinity.filter=ServerGroupAffinity
```

Table 4: ESC Configuration Parameters

esc_param.conf	Type	Default Value	Description
~~default.vm_recovery_retries_max~~	Int	3	Number of recovery attempts allowed per VM.

openstack.endpoint	String	publicURL	The parameter to set up the keystone endpoint value of ESC. Options: adminURL, publicURL
			You can change the default value using CLI or REST services.
			Using CLI:
			`$ sudo escadm escmanager config set --key openstack.endpoint --value publicURL { "category": "OPENSTACK", "type": "STRING", "value": "publicURL", "key": "ENDPOINT" }`
			Using REST:
			`$ curl -X PUT http://172.16.0.1:8080/ESCManager/v0/config/openstack/endpoint/publicURL`

log.level	String	INFO	Level of logging. Options: INFO, Trace, DEBUG
affinity.filter	String	SameHostFilter	A constant string used to build PolicyEngine and initializing VM policy table. Options: SameHostFilter, ServerGroupAffinity
anti_affinity.filter	String	DifferentHostFilter	A constant string used to build PolicyEngine and initializing VM policy table. Options: DifferentHostFilter

Note ESC uses SameHostFilter and DifferentHostFilter for ESC policy engine by default but OpenStack may not configure those filters by default. You may need to add SameHostFilter and DifferentHostFilter to the following scheduler options in the **/etc/nova/nova.conf** file of the nova service in your OpenStack.

```
scheduler_default_filters = RetryFilter, AvailabilityZoneFilter,RamFilter, ComputeFilter,
ComputeCapabilitiesFilter,
ImagePropertiesFilter, ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter,
DifferentHostFilter, SameHostFilter
```

ServerGroupAntiAffinityFilter for Openstack

ESC adapts to use ServerGroupAntiAffinityFilter for Openstack.

REST

```
PUThttp://localhost:8080/ESCManager/v0/config/anti_affinity/filter/ServerGroupAntiAffinity

PUThttp://localhost:8080/ESCManager/v0/config/affinity/filter/ServerGroupAffinity
```

CLI

```
sudo escadm escmanager config set --key ANTI_AFFINITY.FILTER --value ServerGroupAntiAffinity
sudo escadm escmanager config set --key AFFINITY.FILTER --value ServerGroupAffinity
```

Important Points

ServerGroupAntiAffinityFilter from Openstack doesn't support inter-dep anti-affinit, scaling, and mix use of ServerGroup and default (SameHost/DifferentHost) filter. If you are using ServerGroupAntiAffinity filter, Intra vm group placement is not allowed. You can only use **<placement_group>** for the VM based placement policy, one VM per vm_group. You can not add a single vm group in two different placement_groups.

Cisco Elastic Services Controller Installer Arguments