VERITAS™

Veritas InfoScale™ Chef Deployment Guide

Last updated: 2018-09-19

Table of Contents

Introduction

You can now deploy and configure the Veritas InfoScale product suite in your environment using Chef. Chef's powerful automation platform transforms infrastructure into code and allows you to automate various deployment and configuration operations performed in Veritas InfoScale.

Supported Chef Version

The procedure in this guide is supported with *Chef Client 12.5.1* and later versions.

Supported platforms

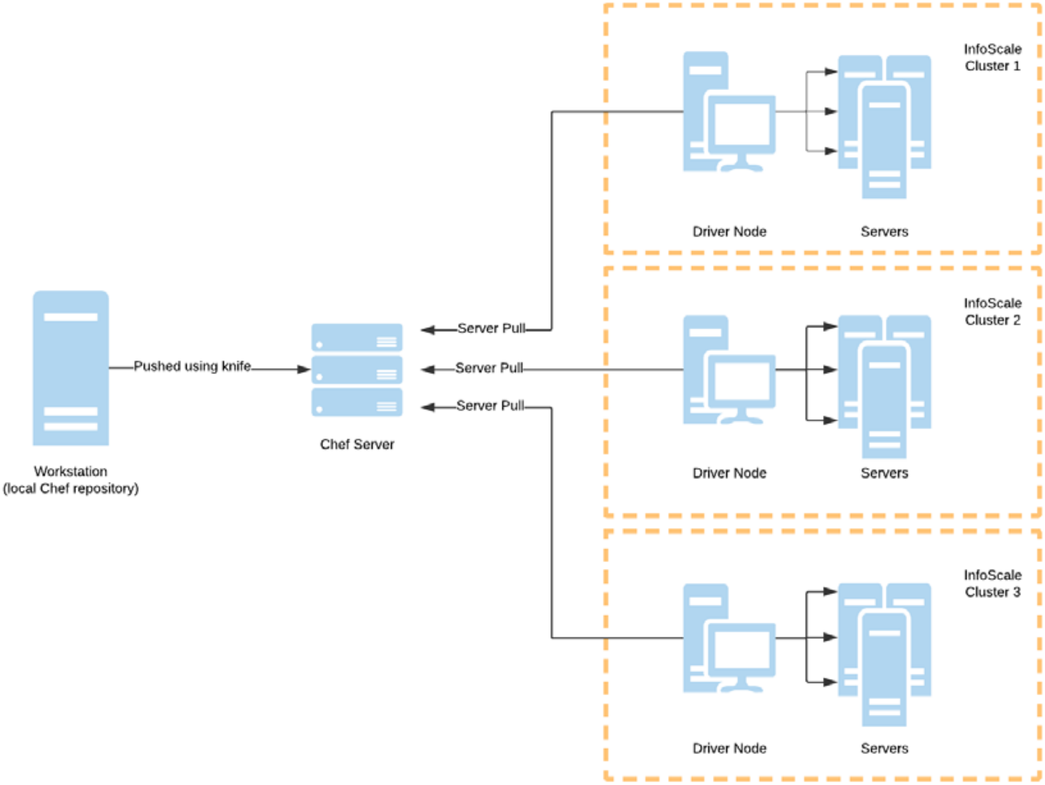
You can use the *veritas_infoscale* cookbook to perform operations on Veritas InfoScale 7.0 and later versions. The following platforms are supported by the *veritas_infoscale* cookbook:

- Linux
- Solaris
- Aix

Chef environment

Ensure that you have installed the Chef infrastructure in your environment. A typical Chef environment usually consists of the following components:

- **Workstation**: The Chef development kit is set up on this server. The workstation server is used to develop and test cookbooks and recipes.
- **Chef server**: The Chef server stores cookbooks, node policies, and node metadata, which are accessed and used by the Chef client on the Chef nodes.
- **Chef node**: A Chef node is used to run the Chef client. Each cluster must contain a single Chef node, which acts as a driver node for the cluster.
- **Knife**: A command-line tool that provides an interface between a local chef repository (workstation) and the Chef server.



Refer to Chef documentation for instructions on installing and setting up the Workstation, Chef server, Chef node, and Knife components in your environment.

Using Chef to install and configure InfoScale in a Linux, Solaris, or AIX environment

This guide provides information about using the veritas_infoscale cookbook. To perform any operation in Veritas InfoScale using Chef, you must perform the following high-level steps:

Prerequisites
Step 1: Downloading the Veritas InfoScale cookbook from the supermarket
Step 2: Specifying operations in a recipe file
Step 3: Creating roles and defining attributes
Step 4: Running the Chef-client on the Chef node

Note that you will require basic knowledge of the *JavaScript Object Notation (JSON)* format and *Chef* while performing the procedure in this guide.

Prerequisites

Ensure that the following prerequisites are met in your environment:

- Passwordless communication is established between all servers in a cluster.
  Tip: The user can use the *pl* utility to set up the ssh and rsh connections automatically.
- Veritas InfoScale installation files are downloaded on the chef node where chef-client is run.
- (Only for Veritas InfoScale version 7.4 and later) All required Veritas InfoScale license files (with a .slf extension) are accessible from the local server. Contact the Customer Care of your region to procure an applicable slf license key file. Refer to the following link for contact information of the Customer Care center for your region:
  https://www.veritas.com/content/support/en_US/contact-us.html
- The veritas_infoscale cookbook uses response files to perform all operations such as installation, upgrade, patch upgrade, and so on. Before using the veritas_infoscale cookbook, ensure that you have read and understood all prerequisites that are applicable while performing those operations with response files. For more information about using response files, refer to the following guides from the Veritas Documentation Library:
  - *Veritas InfoScale Installation Guide*
  - *Storage Foundation Configuration and Upgrade Guide*
  - *Cluster Server Configuration and Upgrade Guide*
  - *Storage Foundation and High Availability Configuration and Upgrade Guide*
  - *Storage Foundation Cluster File System High Availability Configuration and Upgrade Guide*
  - *Storage Foundation for Oracle RAC Configuration and Upgrade Guide*
  - *Storage Foundation for Sybase ASE CE Configuration and Upgrade Guide*
- Ensure that you read through and agree with the End User License Agreement in the PDF file (located in the installation media), before proceeding with any operation in Veritas InfoScale using Chef.

Step 1: Downloading the Veritas InfoScale cookbook from the supermarket

You can download the veritas_infoscale cookbook from the Chef supermarket to use its resources.

To download the Veritas InfoScale cookbook from the public Supermarket

1. To see a list of all community cookbooks available from Supermarket, run the following:

   ```
   knife cookbook site list
   ```

2. Search for the veritas_infoscale cookbook using the following command:

   ```
   knife cookbook site search veritas_infoscale
   ```

3. Download the veritas_infoscale cookbook using the following command:

```
   knife cookbook site download veritas_infoscale
```

The cookbook is downloaded in a tar.gz package. Use the following command to extract the package: *tar xvzf veritas_infoscale.tar.gz*

If you are in an air-gapped environment you can download the cookbook from the Chef Supermarket website and physically transfer the cookbook to your local network using removable storage. Alternatively, you can refer to the Chef documentation for steps on setting up a private supermarket in your local network.

Important: After you have downloaded and extracted the veritas_infoscale cookbook, ensure that you upload the cookbook from the current working directory in the chef repository to the Chef server. Use the following command:

```
knife cookbook upload veritas_infoscale
```

For more information about uploading cookbooks, see the Chef documentation.

Step 2: Specifying operations in a recipe file

Before you run the Chef client on a Chef node, you must specify the operations you want to perform in the recipe file you are using. The following resources are shipped in the veritas_infoscale cookbook:

- infoscale_deploy
- infoscale_configure

Depending on which operations you want to perform, enter the required values in the action block within the infoscale_deploy or infoscale_configure resource.

Value	**Resource name**	**Description**
Install	infoscale_deploy	Installs Veritas InfoScale products in your environment
Configure	infoscale_deploy	Configures the various components used by Veritas InfoScale products
fullupgrade	infoscale_deploy	Upgrades Veritas InfoScale products
patch_upgrade	infoscale_deploy	Upgrades Veritas InfoScale products to a particular patch
Uninstall	infoscale_deploy	Uninstalls Veritas InfoScale products
rollingupgrade	infoscale_deploy	Performs a rolling upgrade for highly available clusters
license	infoscale_deploy	Manages Veritas InfoScale licenses

security	infoscale_configure	Configures security in Veritas InfoScale environment
fencing	infoscale_configure	Configures I/O fencing to prevent data corruption in the event of a communication failure. Note that only disk-based, majority-based, and disabled-based fencing can be configured using Chef

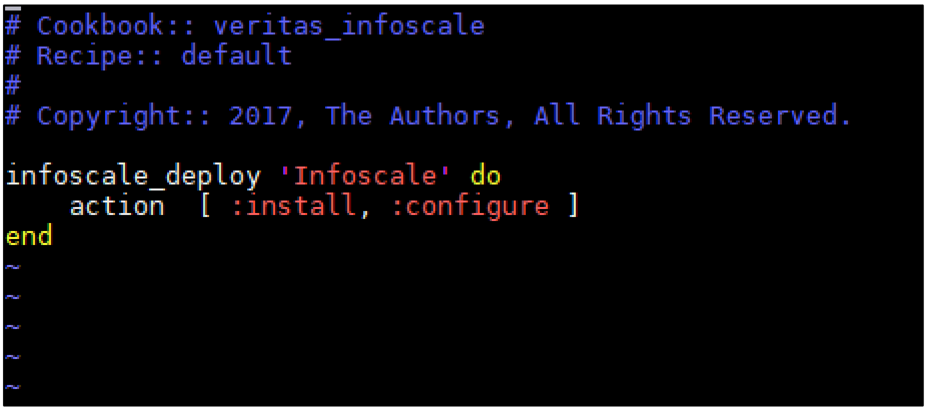
Syntax:

```
<resource_name> '<instance_name>' do
action [ :<value_1> :<value_2> :<value_n>]
end
```

Replace the following variables in the above syntax:

- *<resource_name>* is the name of the resource used by the operation.
- *<value_1>*, *<value_2>*, ... *<value_n>* are n number of values corresponding to the operations you want to perform. The operations will be performed in the sequence in which the values are entered.
- *<instance_name>* is the name of the resource instance. You can provide any instance name for the resource.

Because we will only be installing and configuring InfoScale, we will use the **infoscale_deploy** resource in our recipe, as follows:



```
# Cookbook:: veritas_infoscale
# Recipe:: default
#
# Copyright:: 2017, The Authors, All Rights Reserved.

infoscale_deploy 'Infoscale' do
    action  [ :install, :configure ]
end
~
~
~
~
~
```

Ensure that you upload the cookbook from the current working directory in the chef repository to the Chef server. Use the following command:

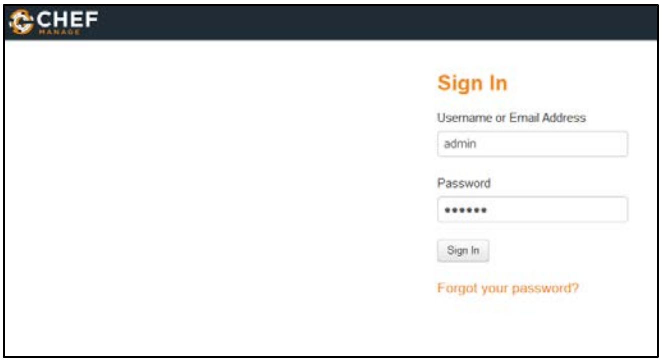
```
knife cookbook upload veritas_infoscale
```

Step 3: Creating roles and defining attributes

Chef roles can be used to perform operations to manage cluster configurations in a Veritas InfoScale environment. A separate role (specifying its run list and override attributes) must be created for each cluster configuration in your environment. When a role is run against a node, the configuration details of that node are compared against the attributes of the role, and then the contents of that role's run-list are applied to the node's configuration details.

To create a role for installation and configuration using the Chef Manage interface

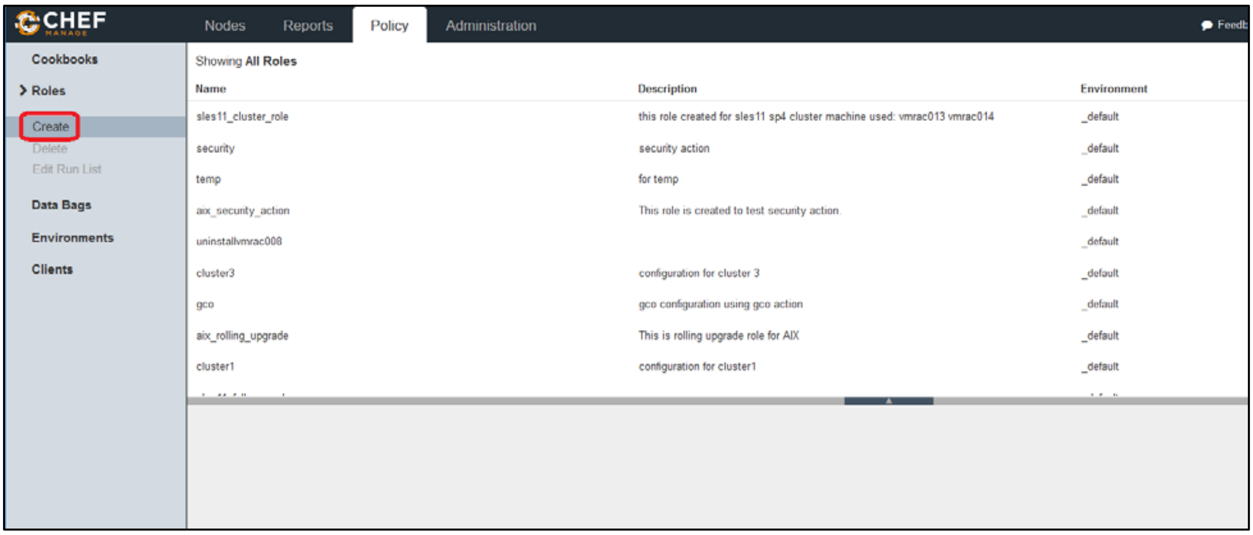
1. Log on to the Chef server with Administrator privileges.



Username or Email Address: admin

Password: ••••••

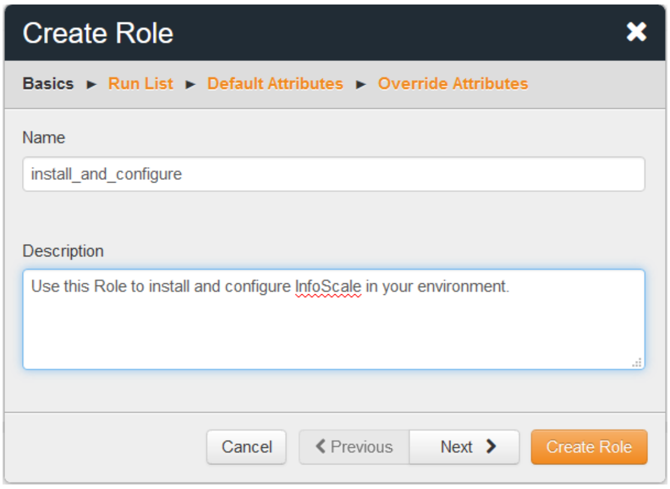
2. Click on the Policy tab at the top of the page.
3. Expand Role in the left pane and click Create. The Create Role dialog box appears.



Showing **All Roles**

Name	Description	Environment
sles11_cluster_role	this role created for sles11 sp4 cluster machine used: vmrac013 vmrac014	_default
security	security action	_default
temp	for temp	_default
aix_security_action	This role is created to test security action.	_default
uninstallvmrac008		_default
cluster3	configuration for cluster 3	_default
gco	gco configuration using gco action	_default
aix_rolling_upgrade	This is rolling upgrade role for AIX	_default
cluster1	configuration for cluster1	_default

4. Enter a name and description to help you easily identify the role.

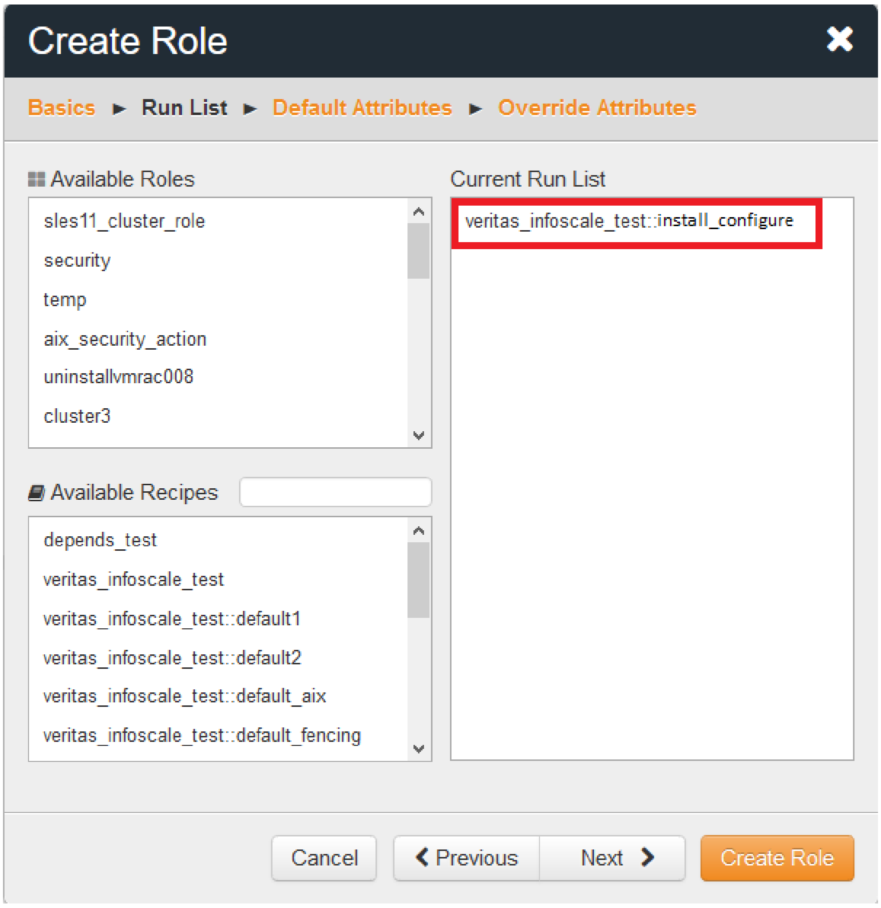


Create Role
Basics ► Run List ► Default Attributes ► Override Attributes
Cancel
Previous
Next
Create Role

Name: install_and_configure

Description: Use this Role to install and configure InfoScale in your environment.

5. Click Next.
6. Search for the recipe that you modified in Step 2: Specifying operations in a recipe file in the Available Recipes list.
7. Drag and drop the recipe to the Create Run List box so that the recipe is run whenever the Chef client is invoked using this role. Ensure that you have customized the default recipe file to perform the desired operations in your environment (see Step 2: Specifying operations in a recipe file).

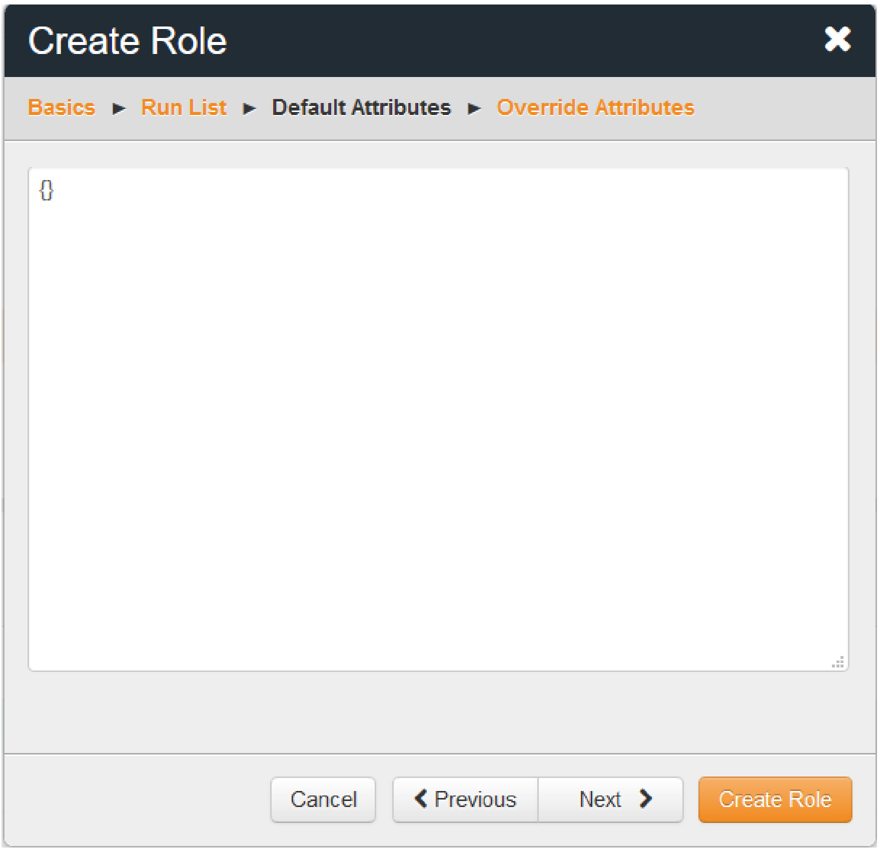


Create Role
Basics ► Run List ► Default Attributes ► Override Attributes
Available Roles
sles11_cluster_role
security
temp
aix_security_action
uninstallvmrac008
cluster3
Available Recipes
depends_test
veritas_infoscale_test
veritas_infoscale_test::default1
veritas_infoscale_test::default2
veritas_infoscale_test::default_aix
veritas_infoscale_test::default_fencing
Current Run List
Cancel
Previous
Next
Create Role

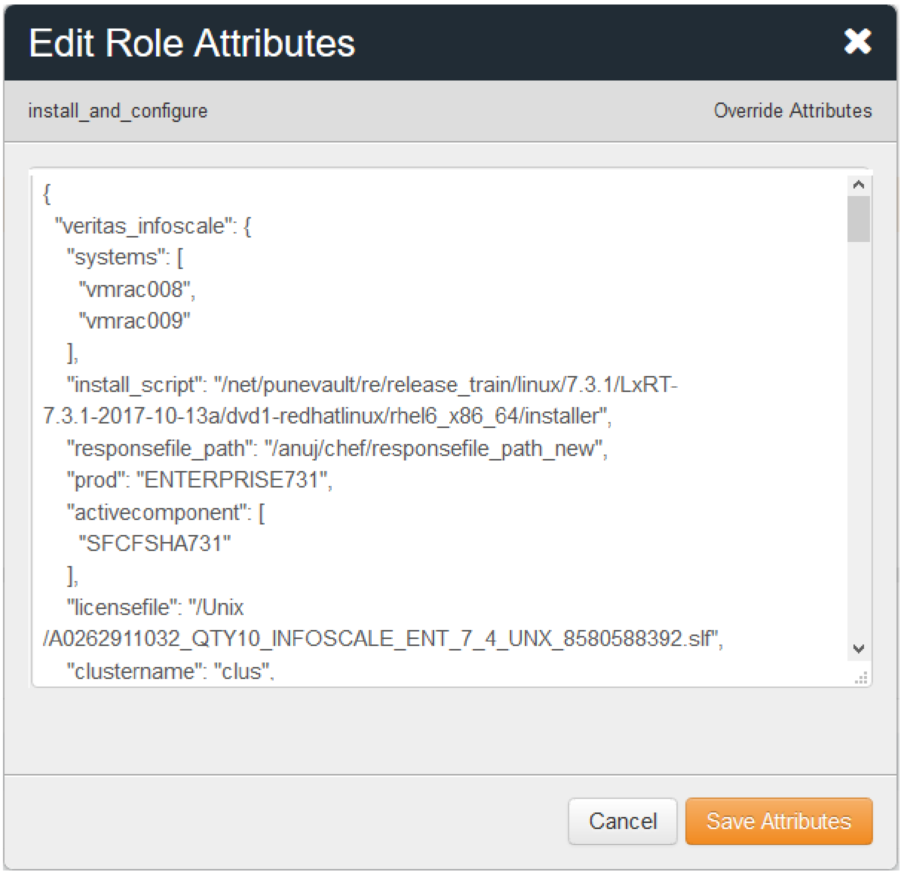
Current Run List: veritas_infoscale_test::install_configure

8. Click Next.
9. You do not need to enter default attributes in the role, because the default values for Veritas InfoScale

operations are pre-defined in the attribute file of the default.rb recipe.


Create Role
Basics ► Run List ► Default Attributes ► Override Attributes
{}
Cancel
Previous
Next
Create Role

10. Click Next.
11. Enter the override attributes for the role in JSON format. Refer to the [Attributes for installing and configuring InfoScale] section for a list of the attributes used in a Veritas InfoScale operation. You can also refer to [Examples of attributes in JSON format] for examples of attributes based on the operation you want to perform.



```
{
  "veritas_infoscale": {
    "systems": [
      "vmrac008",
      "vmrac009"
    ],
    "install_script": "/net/punevault/re/release_train/linux/7.3.1/LxRT-
7.3.1-2017-10-13a/dvd1-redhatlinux/rhel6_x86_64/installer",
    "responsefile_path": "/anuj/chef/responsefile_path_new",
    "prod": "ENTERPRISE731",
    "activecomponent": [
      "SFCFSHA731"
    ],
    "licensefile": "/Unix
/A0262911032_QTY10_INFOSCALE_ENT_7_4_UNX_8580588392.slf",
    "clustername": "clus",
```

12. Click Create Role.

Defining attributes in a role

The attributes can be of the following two types:

Optional: Optional attributes need not be defined by the user. The default values of optional attributes are pre-defined with default values in the attribute file of the default.rb recipe. However, you can override the default values by defining override type of attributes at the role-level.

Mandatory: All mandatory attributes required in an operation must be defined at the role-level as an override type of attribute.

Important: The veritas_infoscale cookbook uses response files to perform all operations such as installation, upgrade, patch upgrade, and so on. Before using the veritas_infoscale cookbook, ensure that you have read and understood all prerequisites that are applicable while performing those operations with response files. For more information about using response files, refer to the following guides:

- *Installation Guide*
- *Storage Foundation Configuration and Upgrade Guide*
- *Cluster Server Configuration and Upgrade Guide*
- *Storage Foundation and High Availability Configuration and Upgrade Guide*
- *Storage Foundation Cluster File System High Availability Configuration and Upgrade Guide*

- *Storage Foundation for Oracle RAC Configuration and Upgrade Guide*
- *Storage Foundation for Sybase ASE CE Configuration and Upgrade Guide*

Each cluster in the Chef-InfoScale environment consists of a Chef node. Veritas InfoScale replicates the configuration of the Chef node to other nodes in the cluster. In this way, we can use role attributes to define the configuration of the Chef node, and subsequently define the configuration of the cluster.

Attributes for installing and configuring InfoScale

The attributes listed in the below table are used while performing an installation and configuration of Veritas InfoScale products. Ensure that you specify values for all mandatory attributes in the role you are creating. Optional attributes which are not specified by the user will use their pre-defined default values.

Attribute	Dimension	Type	Description

systems	List	Mandatory	List of the systems on which the product is to be installed and configured.
			Syntax
			`"systems": ["<server_1>","<server_2>", ... , "<server_n>"]`
			`<server_1>`, *`<server_2>`*, ...*`<server_n>`* are the host names or IP addresses of n number of systems on which you want to perform the installation.
			Example
			`"systems": ["vmrac008","vmrac009"]`

install_script	Scalar	Mandatory	Path to where the Veritas InfoScale installation files are stored. For example, ***<temporary_location>*/installer**.
prod	Scalar	Mandatory	Specifies which Veritas InfoScale product and version you want to install. **Syntax** `<product_name><version_number>` *`<product_name>`* can be either ENTERPRISE. AVAILABILITY, STORAGE, or FOUNDATION. *`<version_number>`* is the version number of the Veritas InfoScale product. **Examples** ENTERPRISE74, AVAILABILITY731, STORAGE70, or FOUNDATION70
keyless	List	Mandatory (*Even though keyless is a*	List of the product keys for keyless installation that are to be registered on the systems

Attribute	Dimension	Type	Description
		mandatory attribtue, you must leave the keyless attribute empty if you are providing values for either the license or licensefile attribute.)	specified in the systems attribute. The value can be either ENTERPRISE, AVAILABILITY, STORAGE, or FOUNDATION.
license	List	Mandatory (*Even though license is a mandatory attribtue, you must leave the license attribute empty if you are providing values for either the keyless or licensefile attribute.*)	List of license keys to be registered on the systems specified in the systems attribute.
licensefile (only for versions 7.4 and later)	List	Mandatory (*Even though licensefile is a mandatory attribtue, you must leave the licensefile attribute empty if you are providing values for either the keyless or license attribute.*)	List of paths to the slf license file to be registered on the systems specified in the systems attribute. Ensure that the license files are accessible from the server where the installer is run.
clustername	Scalar	Mandatory	Defines the name of the cluster.

Attribute	Dimension	Type	Description
clusterid	Scalar	Optional (*If value not provided, we randomly generate number and assign it.*)	Defines an integer value between 0 and 65535 that uniquely identifies the cluster.

heartbeat_links_interface	List	Mandatory	Defines the NIC to be used for a private heartbeat link on each system. At least two LLT links are required per system (lltlink1 and lltlink2).
			Syntax 1:
			`"heartbeat_links_interface": {`
			`    "<server_1>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`    ],`
			`    "<server_2>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`    ],`
			`    "<server_3>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`    ],`
			`    "<server_n>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`

Attribute	Dimension	Type	Description

			`]`
			- *<server_1>, <server_2>, ... <server_n>* are the host names or IP addresses of n number of systems in the cluster.
			- *<NIC_1>, <NIC_2>, ... <NIC_4>* are the NICs that each private heartbeat link uses on that particular system. You must enter at least 2 entries per system, one for each heartbeat link.
			Syntax 2:
			`"heartbeat_links_interface": ["<NIC_1>","<NIC_2>","<NIC_3>","<NIC_4>"]`
			- *<NIC_1>, <NIC_2>, ... <NIC_4>* are the NICs that each private heartbeat link uses on all systems in the cluster. You must enter at least 2 entries per system, one for each heartbeat link.

lopri_link_interface	List	Optional	Lists the NIC interfaces to be configured for low priority heartbeat links. Typically, lopri_link_interface is used on a public network link to provide an additional layer of communication. If you use different media speed for the private NICs, you can configure the NICs with lesser speed as low-priority links to enhance LLT performance. For example, lltlinklowpri1, lltlinklowpri2, and so on. You must enclose the system name within double quotes. It is recommended to configure at least one low priority link in the cluster. **Syntax 1**: `"lopri_link_interface": {`

Attribute	Dimension	Type	Description

			`    "<server_1>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`     ],`
			`    "<server_2>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`     ],`
			`    "<server_3>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`     ],`
			`     "<server_n>": [`
			`      "<NIC_1>",`
			`      "<NIC_2>",`
			`      "<NIC_3>",`
			`      "<NIC_4>"`
			`     ]`
			• *<server_1>*, *<server_2>*, ... *<server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<NIC_1>*, *<NIC_2>*, ... *<NIC_4>* are the NICs that each private low priority heartbeat link uses on that particular system.
			Syntax 2:
			`"lopri_link_interface":`
			`["<NIC_1>","<NIC_2>","<NIC_3>","<NIC_4>`

Attribute	Dimension	Type	Description
			`"]` • *<NIC_1>, <NIC_2>, ... <NIC_4>* are the NICs that each private low priority heartbeat link uses on all systems in the cluster.
lltoverudp	Scalar	Optional	Specifies whether to configure heartbeat link using LLT over UDP. Default value is 0.

heartbeat_udplink_port	Hash (two-dimensional array)	Mandatory (*This attribute is mandatory if LLT is configured over UDP.*)	Lists the UDP port number (16-bit integer value) that each heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the UDP port number for each of the heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_udplink_port": {`
			`"<server_1>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_2>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_3>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`

Attribute	Dimension	Type	Description

			`"<server_n>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`]`
			• *<server_1>*, *<server_2>*, ... *<server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<port_1>*, *<port_2>*, ... *<port_4>* are the port numbers that each heartbeat link uses on that particular system.

heartbeat_udplinklowpri_port	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over UDP.*)	Lists the UDP port number (16-bit integer value) that each low priority heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the UDP port number for each of the low priority heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_udplinklowpri_port": {`
			`"<server_1>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_2>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`

Attribute	Dimension	Type	Description

			`],`
			`"<server_3>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_n>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`]`
			• *<server_1>*, *<server_2>*, ... *<server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<port_1>*, *<port_2>*, ... *<port_4>* are the port numbers that each low priority heartbeat link uses on that particular system.

heartbeat_udplink_address	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over UDP.*)	Lists the IP address (IPv4 or IPv6) that each heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the IP addresses for each of the heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_udplink_address": {`
			`"<server_1>": [`
			`"<IP_address_1>",`
			`"<IP_address_2>",`
			`"<IP_address_3>",`
			`"<IP_address_4>"`

Attribute	Dimension	Type	Description

			`],`
			`"<server_2>": [`
			`"<IP_address_1>",`
			`"<IP_address_2>",`
			`"<IP_address_3>",`
			`"<IP_address_4>"`
			`],`
			`"<server_3>": [`
			`"<IP_address_1>",`
			`"<IP_address_2>",`
			`"<IP_address_3>",`
			`"<IP_address_4>"`
			`],`
			`"<server_n>": [`
			`"<IP_address_1>",`
			`"<IP_address_2>",`
			`"<IP_address_3>",`
			`"<IP_address_4>"`
			`]`
			• *<server_1>*, *<server_2>*, ... *<server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<IP_address_1>*, *<IP_address_2>*, ... *<IP_address_4>* are the IP addresses that each heartbeat link uses on that particular system.

heartbeat_udplink_netmask	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over UDP.*)	Lists the netmask (prefix for IPv6) that each heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the netmask for each of the heartbeat links are specified in a hash (two dimensional array) in JSON.

<table>
<tr><th>Attribute</th><th>Dimension</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>

Syntax:

```
"heartbeat_udplink_netmask": {
   "<server_1>": [
     "<netmask_1>",
     "<netmask_2>",
     "<netmask_3>",
     "<netmask_4>"
    ],
 "<server_2>": [
     "<netmask_1>",
     "<netmask_2>",
     "<netmask_3>",
     "<netmask_4>"
    ],
 "<server_3>": [
     "<netmask_1>",
     "<netmask_2>",
     "<netmask_3>",
     "<netmask_4>"
    ],
"<server_n>": [
    "<netmask_1>",
    "<netmask_2>",
    "<netmask_3>",
    "<netmask_4>"
    ]
```

- *<server_1>*, *<server_2>*, ... *<server_n>* are the host names or IP addresses of n number of systems in the cluster.
- *<netmask_1>*, *<netmask_2>*, ... *<netmask_4>* are the netmasks that each heartbeat link uses on that particular system.

</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Dimension</th><th>Type</th><th>Description</th></tr>
<tr><td>heartbeat_udplinklowpri_address</td><td>Hash (two-dimensional array)</td><td>Optional (<i>This attribute is mandatory if LLT is configured over UDP</i>.)</td><td>

Lists the IP address (IPv4 or IPv6) that each low priority heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the IP addresses for each of the low priority heartbeat links are specified in a hash (two dimensional array) in JSON.

Syntax:

```
"heartbeat_udplinklowpri_address": {
   "<server_1>": [
      "<IP_address_1>",
      "<IP_address_2>",
      "<IP_address_3>",
      "<IP_address_4>"
    ],
   "<server_2>": [
      "<IP_address_1>",
      "<IP_address_2>",
      "<IP_address_3>",
      "<IP_address_4>"
    ],
   "<server_3>": [
      "<IP_address_1>",
      "<IP_address_2>",
      "<IP_address_3>",
      "<IP_address_4>"
    ],
    "<server_n>": [
      "<IP_address_1>",
      "<IP_address_2>",
      "<IP_address_3>",
      "<IP_address_4>"
    ]
```

- *<server_1>*, *<server_2>*, ... *<server_n>* are the host names or IP

</td></tr>
</table>

Attribute	Dimension	Type	Description
			addresses of n number of systems in the cluster. • *<IP_address_1>*, *<IP_address_2>*, ... *<IP_address_4>* are the IP addresses that each low priority heartbeat link uses on that particular system.
heartbeat_udplinklowpri_netmask	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over UDP*.)	Lists the netmask (prefix for IPv6) that each low priority heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the netmask for each of the low priority heartbeat links are specified in a hash (two dimensional array) in JSON. **Syntax**: "heartbeat_udplinklowpri_netmask": { "<server_1>": ["<netmask_1>", "<netmask_2>", "<netmask_3>", "<netmask_4>"], "<server_2>": ["<netmask_1>", "<netmask_2>", "<netmask_3>", "<netmask_4>"], "<server_3>": ["<netmask_1>", "<netmask_2>", "<netmask_3>", "<netmask_4>"], "<server_n>": ["<netmask_1>",

Attribute	Dimension	Type	Description

			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`]`
			- *`<server_1>, <server_2>, ... <server_n>`* are the host names or IP addresses of n number of systems in the cluster.
			- <netmask_1>, <netmask_2>, ... <netmask_4> are the netmasks that each low priority heartbeat link uses on that particular system.

lltoverrdma	Scalar	Mandatory	Indicates whether to configure heartbeat link using LLT over Remote Direct Memory Access (RDMA).

heartbeat_rdmalink_address	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over RDMA.*)	Lists the IP address (IPv4 or IPv6) that each heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the IP addresses for each of the heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_rdmalink_address": {`
			`  "<server_1>": [`
			`    "<IP_address_1>",`
			`    "<IP_address_2>",`
			`    "<IP_address_3>",`
			`    "<IP_address_4>"`
			`  ],`
			`  "<server_2>": [`
			`    "<IP_address_1>",`
			`    "<IP_address_2>",`
			`    "<IP_address_3>",`

Attribute	Dimension	Type	Description

			`    "<IP_address_4>"`
			`  ],`
			`  "<server_3>": [`
			`    "<IP_address_1>",`
			`    "<IP_address_2>",`
			`    "<IP_address_3>",`
			`    "<IP_address_4>"`
			`  ],`
			`  "<server_n>": [`
			`    "<IP_address_1>",`
			`    "<IP_address_2>",`
			`    "<IP_address_3>",`
			`    "<IP_address_4>"`
			`  ]`
			• *`<server_1>, <server_2>, ... <server_n>`* are the host names or IP addresses of n number of systems in the cluster.
			• *`<IP_address_1>, <IP_address_2>, ... <IP_address_4>`* are the IP addresses that each heartbeat link uses on that particular system.

heartbeat_rdmalink_netmask	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over RDMA.*)	Lists the netmask (prefix for IPv6) that each heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the netmask for each of the heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_rdmalink_netmask": {`
			`   "<server_1>": [`
			`     "<netmask_1>",`
			`     "<netmask_2>",`

Attribute	Dimension	Type	Description

			`"<netmask_3>",`
			`"<netmask_4>"`
			`],`
			`"<server_2>": [`
			`"<netmask_1>",`
			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`],`
			`"<server_3>": [`
			`"<netmask_1>",`
			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`],`
			`"<server_n>": [`
			`"<netmask_1>",`
			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`]`
			• *<server_1>, <server_2>, ... <server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<netmask_1>, <netmask_2>, ... <netmask_4>* are the netmasks that each heartbeat link uses on that particular system.

heartbeat_rdmalink_port	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over RDMA*.)	Lists the RDMA port number (16-bit integer value) that each heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the RDMA port number (16-bit integer value) for each of the heartbeat

Attribute	Dimension	Type	Description

			links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_rdmalink_port": {`
			`"<server_1>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_2>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_3>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_n>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`]`
			• *<server_1>, <server_2>, ... <server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<port_1>, <port_2>, ... <port_4>* are the port numbers that each

Attribute	Dimension	Type	Description
			heartbeat link uses on that particular system.
heartbeat_rdmalinklowpri_address	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over RDMA*.)	Lists the IP address (IPv4 or IPv6) that each low priority heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the IP addresses for each of the low priority heartbeat links are specified in a hash (two dimensional array) in JSON. **Syntax:** "heartbeat_rdmalinklowpri_address": { "<server_1>": ["<IP_address_1>", "<IP_address_2>", "<IP_address_3>", "<IP_address_4>"], "<server_2>": ["<IP_address_1>", "<IP_address_2>", "<IP_address_3>", "<IP_address_4>"], "<server_3>": ["<IP_address_1>", "<IP_address_2>", "<IP_address_3>", "<IP_address_4>"], "<server_n>": ["<IP_address_1>", "<IP_address_2>", "<IP_address_3>", "<IP_address_4>"]

Attribute	Dimension	Type	Description
			`<server_1>, <server_2>, ... <server_n>` are the host names or IP addresses of n number of systems in the cluster. *`<IP_address_1>, <IP_address_2>, ... <IP_address_4>`* are the IP addresses that each low priority heartbeat link uses on that particular system.

heartbeat_rdmalinklowpri_netmask	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over RDMA.*)	Lists the netmask (prefix for IPv6) that each low priority heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the netmask for each of the low priority heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_rdmalinklowpri_netmask": {`
			`"<server_1>": [`
			`"<netmask_1>",`
			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`],`
			`"<server_2>": [`
			`"<netmask_1>",`
			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`],`
			`"<server_3>": [`
			`"<netmask_1>",`
			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`],`
			`"<server_n>": [`
			`"<netmask_1>",`

Attribute	Dimension	Type	Description

			`"<netmask_2>",`
			`"<netmask_3>",`
			`"<netmask_4>"`
			`]`
			• *`<server_1>, <server_2>, ... <server_n>`* are the host names or IP addresses of n number of systems in the cluster.
			• *`<netmask_1>, <netmask_2>, ... <netmask_4>`* are the netmasks that each low priority heartbeat link uses on that particular system.

heartbeat_rdmalinklowpri_port	Hash (two-dimensional array)	Optional (*This attribute is mandatory if LLT is configured over RDMA*.)	Lists the RDMA port number (16-bit integer value) that each low priority heartbeat link uses on each of the nodes in the cluster. The system name of each of the nodes and the RDMA port number for each of the low priority heartbeat links are specified in a hash (two dimensional array) in JSON.
			Syntax:
			`"heartbeat_rdmalinklowpri_port": {`
			`  "<server_1>": [`
			`       "<port_1>",`
			`       "<port_2>",`
			`       "<port_3>",`
			`       "<port_4>"`
			`    ],`
			`  "<server_2>": [`
			`      "<port_1>",`
			`      "<port_2>",`
			`      "<port_3>",`
			`      "<port_4>"`
			`    ],`
			`  "<server_3>": [`

Attribute	Dimension	Type	Description

			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`],`
			`"<server_n>": [`
			`"<port_1>",`
			`"<port_2>",`
			`"<port_3>",`
			`"<port_4>"`
			`]`
			• *<server_1>, <server_2>, ... <server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<port_1>, <port_2>, ... <port_4>* are the port numbers that each low priority heartbeat link uses on that particular system.

userenpw	List	Optional	List of encoded passwords for users. The value in the list can be "Administrators Operators Guests" **Note:** The order of the values for the userenpw list must match the order of the values in the username list.
username	List	Optional	Lists the names of users.
userpriv	List	Optional	Lists the privileges to be granted to the users. **Note:** The order of the values for the userpriv list must match the order of the values in the username list.
eat_security	Scalar	Optional	Specifies whether the cluster should be enabled with the secure mode or not. By default, this

Attribute	Dimension	Type	Description
			value is set to 1, which means that security will be enabled in the cluster.
eat_security_fips	Scalar	Optional	Specifies whether to enable or disable security with FIPS mode on a running VCS cluster. By default, this value is set to 0, which means that security with FIPS mode will be disabled in the cluster.
keyfile	Scalar	Optional	Location of the SSH key file that is used by the Chef node to communicate with other nodes in the cluster.
logpath	Scalar	Optional	Specifies the location where the log files are to be stored. If no value is defined the default location is **/opt/VRTS/install/logs**.
noipc	Scalar	Optional	Disables the installer from making outbound networking calls to Veritas Services and Operations Readiness Tool (SORT) in order to automatically obtain patch and release information updates. (0: sends outbound networking calls to SORT, 1: does not send outbound networking calls to SORT). The default value is 0.
patch_path	List (*maximum 5 patches*)	Optional	Defines the path of a patch level release to be integrated with a base or a maintenance level release in order for multiple releases to be simultaneously installed.
responsefile_path	Scalar	Optional	Temporary location where you want to store the response file created during installation. By default, the response file is stored in the home directory of the user. You can override the default path by specifying a different path for this

Attribute	Dimension	Type	Description
			attribute. Note that after successful installation is complete the response file is moved to **/opt/VRTS/install/chef**.
rsh	Scalar	Optional	Specifies that RSH protocol must be used instead of SSH protocol while communicating between systems. A boolean value of 1 indicates that RSH will be used and a boolean value of 0 indicates that SSH will be used. This is an optional attribute and if no value is provided SSH protocol is used by default.
tmppath	Scalar	Optional	Specifies the location where temporary files and dependent RPMs are stored during the install. The default location is **/var/tmp**.
uploadlogs	Scalar	Optional	Specifies whether you want to upload logs to the Veritas website. The value 1 indicates that the installation logs are uploaded to the Veritas website. The value 0 indicates that the installation logs are not uploaded to the Veritas website. By default this value is set to 1.
activecomponent	List	Mandatory	Specifies the component for operations like precheck, configure, addnode, install, and configure (together). The value can be either of the following: - VCS<VersionNumber> for VCS component. - SF<VersionNumber> for SF component. - SVS<VersionNumber> for SVS component. - SFRAC<VersionNumber> for SF Oracle RAC component. - SFSYBASECE<VersionNumber> for

			SFSYBASECE component. - SFCFSHA<VersionNumber> for SFCFSHA component. - SFHA<VersionNumber> for SFHA component.
fencing_option	Scalar	Optional	Specifies the I/O fencing configuration mode. The default value is 1. • 1: Disk-based fencing • 2: Majority-based fencing • 3: Disabled-based I/O fencing
fencingenabled	Scalar	Optional	Specifies whether to enable fencing in your Veritas product. Valid values are 0 or 1. The default value is 0.
fencing_auto_refresh_reg	Scalar	Optional	Enable the auto refresh of coordination points variable in case registration keys are missing on any of CP servers. The default value is 0.
fencing_config_cpagent	Scalar	Optional	Enter '1' or '0' depending upon whether you want to configure the Coordination Point agent using the installer or not. Enter "0" if you do not want to configure the Coordination Point agent using the installer. Enter "1" if you want to use the installer to configure the Coordination Point agent. The default value is 1.
fencing_cpagent_monitor_freq	Scalar	Optional	Specifies the frequency at which the Coordination Point Agent monitors for any changes to the Coordinator Disk Group constitution. **Note:** Coordination Point Agent can also monitor changes to the Coordinator Disk Group constitution such as a disk being accidently deleted from the Coordinator Disk Group. The frequency of this detailed monitoring can be tuned with the LevelTwoMonitorFreq attribute. For example, if you set this attribute to 5, the agent will monitor the Coordinator Disk

			Group constitution every five monitor cycles. If LevelTwoMonitorFreq attribute is not set, the agent will not monitor any changes to the Coordinator Disk Group. 0 means not to monitor the Coordinator Disk Group constitution. The default value is 5
fencing_cpagentgrp	Scalar	Optional	Name of the service group which will have the Coordination Point agent resource as part of it. The default value is "vxfen". **Note:** This field is obsolete if the fencing_config_cpagent field is given a value of '0'.
fencing_dgname	Scalar	Optional *(Mandatory if fencingenabled is 1 and fencing_option is set to 1)*	Specifies the disk group for I/O fencing. **Note:** You must define the fencing_dgname variable to use an existing disk group. If you want to create a new disk group, you must use both the fencing_dgname variable and the fencing_newdg_disks variable.
fencing_newdg_disks	List	Optional *(Mandatory if fencingenabled is 1 and fencing_option is set to 1)*	Specifies the disks to use to create a new disk group for I/O fencing. **Note:** You must define the fencing_dgname variable to use an existing disk group. If you want to create a new disk group, you must use both the fencing_dgname variable and the fencing_newdg_disks variable.
allowcomms	Scalar	Optional	Indicates whether to start LLT and GAB when you set up a single-node cluster. The value can be 0 (do not start) or 1 (start). By default, this value is set to 1.
defaultaccess	Scalar	Optional	Specifies whether to grant read access to all. By default this value is set to 0.
donotreconfigurefencing	Scalar	Optional	Specifies whether you want to re-configure fencing. Note that any customizations made to the configuration will be deleted and the default

			configuration will be re-installed. By default this value is set to 1.
donotreconfigurevcs	Scalar	Optional	Specifies whether you want to re-configure VCS. Note that any customizations made to the configuration will be deleted and the default configuration will be re-installed. By default this value is set to 1.
secusrgrps	List	Optional	Specifies the user groups which will receive read access to the cluster.
vm_no_open_vols	Scalar	Optional	Specifies whether to ask the user if there are any open volumes (when vxconfigd is not enabled). Such prompts are asked during uninstallations. (1: affirms there are no open volumes on the system). By default value is set to 0.
csgnetmask	Scalar	Optional (*This attribute must be specified if (Cluster Service Group) CSG is used.*)	Defines the netmask of the virtual IP address for the cluster.

csgnic	Scalar	Optional (*This attribute must be specified if (Cluster Service Group) CSG is used.*)	Defines the NIC device to use on a system.
			Syntax 1:
			`"csgnic": {`
			`    "<server_1>": [`
			`      "<NIC>"`
			`    ],`
			`    "<server_2>": [`
			`      "<NIC>"`
			`    ],`
			`    "<server_3>": [`
			`      "<NIC>"`
			`    ],`
			`    "<server_n>": [`

			`      "<NIC>"`
			`     ]`
			• *<server_1>, <server_2>, ... <server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• <*NIC*> is the NIC that is used on that particular system.
			Syntax 2:
			`"csgnic": "<NIC>"`
			• <*NIC*> is the NIC that is used on all the systems.

csgvip	Scalar	Optional (*This attribute must be specified if (Cluster Service Group) CSG is used*.)	Defines the virtual IP address for the cluster.
gconetmask	Scalar	Optional (*This attribute must be specified if GCO is used*.)	Defines the Netmask of the virtual IP address that the Global Cluster Option uses.

gconic	Scalar	Optional (*This attribute must be specified if GCO is used*.)	Specifies the NIC for the virtual IP that the Global Cluster Option uses. You can enter all as a system value if the same NIC is used on all systems.
			Syntax 1:
			`"gconic": {`
			`     "<server_1>": [`
			`        "<NIC>"`
			`      ],`
			`     "<server_2>": [`
			`        "<NIC>"`
			`      ],`
			`     "<server_3>": [`

			`"<NIC>"`
			`],`
			`"<server_n>": [`
			`"<NIC>"`
			`]`
			• *<server_1>, <server_2>, ... <server_n>* are the host names or IP addresses of n number of systems in the cluster.
			• *<NIC>* is the NIC that is used on that particular system.
			Syntax 2:
			`"gconic": "<NIC>"`
			• *<NIC>* is the NIC that is used on all the systems..

gcovip	Scalar	Optional (*This attribute must be specified if GCO is used*.)	Defines the virtual IP address that the Global Cluster Option uses.
smtprecp	List	Optional (*This attribute must be specified if SMTP protocol is used*.)	List of full email addresses (example: user@example.com) of SMTP recipients.
smtprsev	List	Optional (*This attribute must be specified if SMTP protocol is used*.)	Defines the minimum severity level of messages (Information, Warning, Error, SevereError) that listed SMTP recipients are to receive. Note that the ordering of severity levels must match that of the addresses of SMTP recipients.
smtpserver	Scalar	Optional (*This attribute must be specified if SMTP protocol is used*.)	Defines the domain-based host name (for example smtp.example.com) of the SMTP server to be used for web notification.
snmpcons	List	Optional (*This attribute must be*	List of SNMP console system names.

		specified if SNMP protocol is used.)	
snmpcsev	List	Optional (*This attribute must be specified if SNMP protocol is used*.)	Defines the minimum severity level of messages (Information, Warning, Error, SevereError) that listed SNMP consoles are to receive. Note that the ordering of severity levels must match that of the SNMP console system names.
snmpport	Scalar	Optional (*This attribute must be specified if SNMP protocol is used*.)	Defines the SNMP trap daemon port (default is 162).

Step 4: Running the Chef-client on the Chef node

A Chef node is a node, which is managed by Chef, and is used to run the Chef-client. We recommend that you run the Chef client from a Chef node that is part of the cluster on which you are performing the operation. Use the following command to run the Chef-client on the Chef node:

```
chef-client -r "role[<name_of_the_role>]"
```

<name_of_the_role> is the name of the role that you have created in Step 3: Creating roles and defining attributes.

Example:

```
chef-client -r "role[install_and_configure]"
```

Note: Ensure that you have added a dependency on the veritas_infoscale cookbook in your cookbook's metadata.rb, as follows:

```
depends 'veritas_infoscale'
```

Example of attributes used for installation and configuration

The following is an example of attributes entered (in JSON format) in a role for installing and configuring InfoScale:

```
{
  "veritas_infoscale": {
    "systems": [
      "vm1",
      "vm2"
    ],
    "install_script": "/mnt/7.2/dvd1-redhatlinux/rhel7_x86_64/installer ",
    "prod": "ENTERPRISE72",
    "activecomponent": [
      "SFHA72"
    ],
    "keyless": "ENTERPRISE",
    "license": "xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-xxxx-x",
    "clustername": "clustertest",
    "heartbeat_links_interface": {
      "vm1": [
        "eth1",
        “eth2”
      ],
      "vm2": [
        "eth1",
        “eth2”
      ]
    },
    "lopri_link_interface": {
      "vm1": [
        "eth0"
      ],
      "vm2": [
        "eth0"
      ]
    },
    "username": [
      "user1",
      "user2"
    ],
    "userenpw": [
      "xxxxxxxxxxx",
      "xxxxxxxxxxx"
    ],
    "userpriv": "Administrators",
    "smtpserver": "xxxxxxxxx.example.com",
    "smtprecp": [
      "John.Johnson@example.com"
    ],
    "smtprsev": [
      "Information"
    ],
    "fencingenabled": "1",
    "fencing_dgname": "vxfendg",
    "fencing_option": "1",
  }
}
```

About Veritas Technologies LLC

Veritas Technologies LLC enables organizations to harness the power of their information, with solutions designed to serve the world's largest and most complex heterogeneous environments. Veritas works with 86 percent of Fortune 500 companies today, improving data availability and revealing insights to drive competitive advantage.

For specific country offices and contact numbers, please visit our website.

Veritas World Headquarters

500 East Middlefield Road

Mountain View, CA 94043

+1 (650) 933 1000

www.veritas.com