Lenovo Networking

User Guide

for Ansible 2.7

Lenovo™

Note: Before using this information and the product it supports, read the general information in the *Safety information and Environmental Notices* and *User Guide* documents on the Lenovo *Documentation* CD, and the *Warranty Information* document that comes with the product.

Second Edition (December 2018)

Contents

Ansible for Lenovo Networking

Ansible is an open-source automation and orchestration framework. It is a simple and easy to use tool to help you with configuration management, application deployment, and the automation of tasks that involve servers, switches, and other network elements.

This guide help you set up an Ansible environment to control and configure topologies that involve Lenovo switches.

Following are the major sections covered in this guide:

- Ansible Overview - a short introduction to Ansible
- Creating and Executing Ansible Playbooks - information on how to create and execute an Ansible playbook
- Basic Networking Scenarios - examples of basic network tasks where Ansible and Lenovo modules can help make such tasks much easier
- Lenovo Modules for Ansible - general information about the Ansible modules provided by Lenovo for Ansible version 2.7
- Lenovo Ansible Modules Overview - the list of modules provided by Lenovo for Ansible
- Configuring a Clos Network on Lenovo CNOS using Ansible - Clos network example configuration using Lenovo switches and Ansible roles and playbooks
- Configuring a Single Layer vLAG on Lenovo CNOS using Ansible - single layer vLAG configuration using Lenovo switches and Ansible roles and playbooks
- Configuring a Multiple Layer vLAG on Lenovo CNOS using Ansible - multiple layer vLAG configuration using Lenovo switches and Ansible roles and playbooks
- Configuring Telemetry on Lenovo CNOS using Ansible - network telemetry configuration using Lenovo switches and Ansible roles and playbooks

Ansible Overview

Ansible is an open source software that automates software provisioning, configuration management, and application deployment. Ansible connects via SSH, remote PowerShell or via other remote APIs.

For more details about Ansible, visit ansible.com.

What is Ansible and how does it work?

Ansible is an open-source automation and orchestration framework. It is a simple and easy to use tool to help you with configuration management, application deployment, and the automation of tasks that involve servers, switches, and other network elements.

Tasks can be configured to run in a sequence on multiple different network devices. For example a software update can be scheduled for a group of devices. Ansible can upgrade the devices one after the other to ensure the group is always accessible.

Ansible uses an agentless architecture, which does not require the installation and background execution of daemons - processes that handle requests for services and are inactive until called upon. Instead Ansible uses OpenSSH to communicate with other devices on the network.

Ansible establishes connections with all the devices specified in its inventory file, executes commands on those devices, and the returns their results. Commands are executed through the use of small programs called modules. These modules can be run in a specific order that is configured in files called playbooks.

Ansible Installation

Ansible manages the network and its components using OpenSSH. It needs to be installed only on a single server, from which it can manage the whole network. Ansible communicates with other devices without needing to install or run software on those machines.

For details on how to install Ansible and also to learn about any prerequisites it may have, please consult the official Ansible documentation.

Securing Sensitive Data

If securing sensitive data is a concern, Ansible provides a means for keeping sensitive data such as passwords or keys in encrypted files, rather than as plain text in your playbooks or roles. Encryption of sensitive data is provided through Ansible's Vault feature. Please refer to Ansible's Vault page for more information on use of this feature.

Configuration File

The Ansible configuration file - *ansible.cfg* - contains settings like the default timeouts, port numbers, and other parameters. The configuration file is located at the following address:

```
/etc/ansible/ansible.cfg
```

To leave the configuration file intact, create a new file in a local directory. The new configuration file can be edited and a reference to it must be provided in the ansible.cfg file.

Inventory

The inventory contains the list of hosts managed by Ansible. Hosts in the inventory are generally arranged into groups and the actions performed by Ansible are carried out on all the hosts in a group simultaneously. The Ansible host file is usually found at the following address:

```
/etc/ansible/hosts
```

Ansible can make use of multiple inventory files at the same time.

In the inventory file, variables can be assigned to each host to be used at a later time in playbooks. Variables can be also applied to a host group.

Modules

Modules are used by Ansible to do its work. They are Python scripts that get executed on the network devices that Ansible manages. Multiple modules are usually executed as part of playbooks. Single modules can be run using the **`ansible`** command. After being executed on a device, a module returns information to Ansible in JSON format.

Modules can have arguments assigned to them.

Playbooks

Playbooks are used by Ansible for the configuration, deployment, and orchestration of remote devices. They are like a set of instructions that tell Ansible how to perform specific actions. For example, playbooks can describe the steps that Ansible needs to take to do a rolling update for multiple devices.

Playbooks can declare configurations. They can also be used to orchestrate the steps of any manually ordered process, even if different steps require to jump from one device to another in a specific sequence. Playbooks can launch tasks both synchronously and asynchronously.

Playbooks are written in YML format.

Tasks

Every playbook contains a lists of tasks. Ansible goes through a playbook and executes tasks in the specified order, one after the other. It runs a task on all devices that match the host pattern assigned to the task. After completing the task, Ansible moves to the next task in the playbook.

During the execution of a playbook, all hosts are going to get the same task directives. The purpose of a playbook is to map a selection of hosts to a selection of tasks, while the purpose of a task is to execute a module, usually with very specific arguments. Variables can be used in arguments assigned to modules.

Templates

Templates are like mathematical functions. Functions require some form of initial data that is processed by the function and then a result is generated. Templates work in a similar way. During the execution of a playbook, Ansible substitutes the variables in a template and outputs a command set file that can be used to configure different network elements.

The concept of a template is introduced in Ansible as a module. Templates for Ansible are made using Jinja2 - a popular Python template engine. For more information about Jinja2, see http://jinja.pocoo.org/.

Variables

Variables are place holders for which their value may or not be known. The variable name is used to reference the value stored. This separation between the variable name and value allows the variable name to be used independently of the exact value it represents.

Variable names should be letters, numbers, and underscores. They should always start with a letter.

Variables can be assigned to hosts in the inventory file. They can also be used in a playbook and can be defined in other files, such as roles or templates. All variables are further defined in the following file, where values can be provided for them:

```
/vars/main.yml
```

Handlers

Handlers perform different post-deployment tasks, such as the restart of a device. They are triggered tasks that are executed only if they are notified as such by other devices through the use of notifications. For example, you can set up a handler to restart a switch when its configuration file changes. The switch detects that its configuration has been modified and it notifies the handler of this event. The handler is triggered and Ansible executes the tasks associated with it.

Ansible ensures that the tasks are performed only if the handler is triggered. For example, Ansible restarts the switch only if the switch configuration file has actually changed.

Handlers can also be configured to listen to specific events. This allows handlers to be triggered without being directly notified of an event.

Roles

Roles are methods of automatically loading specific variable files, tasks, and handlers based on a predetermined file structure. Roles are assigned to different hosts. When a host is declared to have a specific role, Ansible knows what actions to perform on that host based on the configuration of its assigned role.

Roles are the best way to organize playbooks. Grouping content by role also allows roles to be easy to share with other users.

Creating and Executing Ansible Playbooks

Ansible offers various approaches to the automation of configuration management tasks. The most popular method is the use of roles and Ansible Galaxy is created as a repository of roles.

Lenovo provides many reusable roles and modules that any developer can utilize to create his automation tasks. Though the use of roles created by somebody else saves time, those roles might not prove to be the best ones to utilize for your specific scenario. Roles found on Ansbile Galaxy are free to be modified to meet your particular needs.

Although it is not required, it is recommended to use the Python Programming Language and the YML Editor during development to reduce the time spent on the syntax of roles.

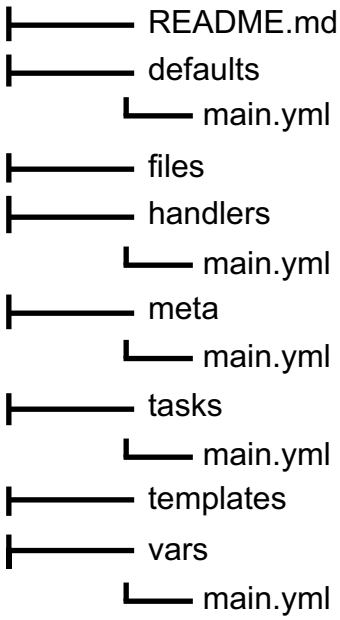
Creating a Role

When creating a new role, the first step is to build its directory structure. Since version 1.4.2 Ansible provides a tool to create the base directory structure. The tool is called ansible-galaxy and below is an example on how to use it:

```
$ ansible-galaxy init levono.config

lenovo.config was created successfully
```

The command creates the following directory structure:



The Directory Structure of Roles

The directory structure of a role consists of the following:

- defaults
- vars
- files
- handlers
- meta
- templates
- tasks

defaults

Within the *defaults* directory there is a *main.yml* file that contains the default variables used by a role. For the cnos role there is only one default variable called `cnos_version`.

```
cnos_version: "<supported CNOS version>"
```

If you do not use variables in a role, you are not required to create the *defaults* directory.

vars

The *vars* and *defaults* directories both hold variables, the variables stored in the *vars* directory have a higher priority. Variables with a higher priority are more difficult to overwrite than variables with a lower priority. The variables stored in the *defaults* directory have the lowest priority available, meaning they are easy to overwrite.

Inside the *vars* directory there is a *main.yml* file that contains the variables that you define. You can also define variables in a playbook. The following is an example of the content of the *main.yml* file from the *vars* directory:

```
vlag_1tier_leaf_data:
  - {username: <username>, password: <password>}

vlag_1tier_leaf_switch1_data:
  - {username: <username>, password: <password>, stp_mode1: disable,
port_range1: "17,18,29,30", portchannel_interface_number1: 1001,
portchannel_mode1: active, slot_chassis_number1: 1/48,
switchport_mode1: trunk}
```

If you do not use variables in a role, you are not required to create the *vars* directory.

files

The *files* directory stores the files that need to be added to the device that is provisioned and do not required any modifications. Usually, files stored in this directory are references by copy tasks.

If you do not use such files in a role, you are not required to create the *files* directory.

handlers

The *handlers* directory holds handlers that usually contain the targets of notify directives and are in most cases associated with services. For example, when creating a role that configures a network switch, the *main.yml* file in the *handlers* directory might have an entry that overwrites the startup configuration of the switch with its current running configuration and then restarts the device.

meta

The *meta* directory stores the metadata of a role. The *main.yml* file of the *meta* directory holds metadata attributes such as the author of the role, the supported platforms, and the role's dependencies. For the most part, this file is commented out by default.

The roles provided by Lenovo contain information belonging to the organization.

templates

The *templates* directory stores files in a similar way to the files directory, except that the files held can be modified as they are added to the devices that are provisioned.

Modification to these files are done using the Jinja2 templating language. Most software configuration files become templates.

Lenovo provides a variety of templates to suit different scenarios. The following is an example of a template. Variables used in the example need to be defined in the *main.yml* file in the *vars* directory.

```
#Common commands for Tier1 leaf nodes

#STP configuration
spanning-tree mode {{item.stp_mode1}}

#LACP configuration
interface ethernet {{item.slot_chassis_number1}}
channel-group {{item.portchannel_interface_number1}} mode
{{item.portchannel_mode1}}
exit

#VLAN configuration
interface port-channel {{item.potchannel_interface_number1}}
switchport mode {{item.switchport_mode1}}
exit

interface ethernet {{item.slot_chassis_number2}}
switchport mode {{item.switchport_mode1}}
exit
```

tasks

The *tasks* directory holds various Ansible playbooks to install, configure, and run software. The majority of the Ansible activity is stored in this directory. The following is an example where a Command Line Interface (CLI) template is used to achieve a set of configuration tasks. Variables used in this example marked with the entry `with_items` need to be defined in the *main.yml* file of the *vars* directory.

```
#This file contains VLAG Tier1 leaf configurations tasks

- name: Replace VLAG Tier1 Leaf CLI template with value for switch1
  template: src=vlag_1tier_leaf_common.j2
dest=./commands/vlag_1tier_leaf_switch1_commands.txt
  with_items: "{{vlag_1tier_leaf_switch1_data}}"

- name: Replace VLAG Tier1 Leaf CLI template with value for switch2
  template: src=vlag_1tier_leaf_common.j2
dest=./commands/vlag_1tier_leaf_switch2_commands.txt
  with_items: "{{vlag_1tier_leaf_switch2_data}}"

- name: Applying CLI template on VLAG Tier1 Leaf Switch1
  cnos_conditional_template: host={{ inventory_hostname }}
condition={{ hostvars[inventory_hostname]['condition']}}
commandfile=./commands/vlag_1tier_leaf_switch1_commands.txt
outputfile=./results/vlag_1tier_leaf_switch1_output.txt
  with_items: "{{vlag_1tier_leaf_data}}"

- name: Applying CLI template on VLAG Tier1 Leaf Switch2
  cnos_conditional_template: host={{ inventory_hostname }}
condition={{ hostvars[inventory_hostname]['condition']}}
commandfile=./commands/vlag_1tier_leaf_switch2_commands.txt
outputfile=./results/vlag_1tier_leaf_switch2_output.txt
  with_items: "{{vlag_1tier_leaf_data}}"

- name: Replace VLAG Tier1 Leaf CLI template with value
  template: src=vlag_1tier_leaf_show.j2
dest=./commands/vlag_1tier_leaf_show.txt
  with_items: "{{vlag_1tier_leaf_data}}"

- name: Applying CLI template on  VLAG Tier1 Leaf Switches
  cnos_template: host={{ inventory_hostname }}
commandfile=./commands/vlag_1tier_leaf_show.txt
outputfile=./results/vlag_1tier_leaf_show_output.txt
  with_items: "{{vlag_1tier_leaf_data}}"

# Completed file
```

Executing a Playbook

To execute an Ansible playbook, use the following steps:

1. Install Ansible on a Ubuntu or Red Hat Enterprise Linux (RHEL) server.

   For details on how to install Ansible, see the official Ansible documentation.

2. Install all the required libraries to ensure the installation is successful.

3. Check that you can access the Ansible playbook from any directory of the machine you are running Ansible on. This can be tested using the following command:

   ```
   $ ansible-playbook
   ```

   If Ansible returns the following message, then Ansible is not correctly installed.

   ```
   command not found
   ```

4. Create a test directory and unzip the Lenovo Ansible roles and solutions. This contains a *library* directory and a *dictionary* directory. In this example the test directory is called *test*.

   ```
   $ mkdir test
   ```

5. Check if the Python environment variable `PYTHONPATH` is already configured. If the echo command returns a result, then reconfigure the `PYTHONPATH` variable to *<installation directory>/library*.

   ```
   echo $PYTHONPATH
   ```

   ```
   unset PYTHONPATH
   PYTHONPATH = <installation directory>/library
   export PYTHONPATH
   ```

6. Update the hosts file found under *etc/ansible/hosts* with the appropriate network device information as specified in the *<lenovo-role>_hosts* file.

   Ansible keeps track of all network elements that it manages through a hosts file. Before the execution of a playbook, the hosts file must be set up.

   Open the */etc/ansible/hosts* file with root privileges. Most of the file is commented out by using **#**. You can also comment out the entries you add by using **#**. You need to copy the content of the hosts file for the role into the */etc/ansible/hosts* file. The hosts file for the role is located in the main directory of the solution.

For example, for the single layer vLAG Ansible solution, the host information specified in the *vlag_1tier_leaf_hosts* and *vlag_1tier_spine_hosts* files needs to be copied and added to the hosts file found under the *etc/ansible/hosts* directory.

```
[vlag_1tier_leaf]
10.240.178.74   username=<username> password=<password>
deviceType=g8272_cnos condition=leaf_switch1
10.240.178.75   username=<username> password=<password>
deviceType=g8272_cnos condition=leaf_switch2
```

```
[vlag_1tier_spine]
10.240.178.76   username=<username> password=<password>
deviceType=g8272_cnos peerip=10.240.178.77
10.240.178.77   username=<username> password=<password>
deviceType=g8272_cnos peerip=10.240.178.76
```

Note: You need to change the IP addresses, including the vLAG peer IP addresses, to fit your specific topology. You also need to change the `<username>` and `<password>` to the appropriate values used to log into the specific network devices.

7. Execute the playbook by using the following command:

```
$ ansible-playbook <myPlaybook.yml> -vvv
```

-vvv is an optional verbos command that helps identify what is happening during playbook execution. The playbook for each role of the multiple layer vLAG configuration solution is located in the main directory of the solution.

8. Check the screen prompt for the result of the command. For more details, consult the *results* directory.

Basic Networking Scenarios

Some of the most basic network configuration and automation tasks you may need to perform include the manual configuration of network devices, firmware image upgrades, back up device configurations, and debugging operations. In such scenarios Ansible and the Lenovo modules can help you out.

1. **Initial device configuration**

   Newly added network devices need their IP addresses manually configured for Ansible to be able to manage them.

   After this step, Ansible can take care of further configurations by using CLI templates and run time substitutions of template variables with the specific values. This ensures a more robust planning and a more precise and faster parallel execution.

2. **Deploying configurations across multiple devices**

   In scenarios where a configuration change needs to be applied across multiple network devices, the new setting can be applied smoothly through Ansible. This can be achieved by using one of the appropriate Lenovo modules for Ansible.

3. **Workflow automation**

   Multiple network configurations can be combined to create custom specific workflows by leveraging one or more playbooks. These can be used to integrate workflows that include system and application changes.

   In such scenarios Ansible proves to be very useful.

4. **Firmware image upgrades**

   Once or twice a year the firmware images of the network devices under your administration need to be upgraded to the latest version.

   When new enhancements and bug fixes are available for a Lenovo switch, this task can be achieved by copying the newer firmware image from a remote server using a file transfer protocol, such as SFTP or SCP. You can leverage the Lenovo modules or playbooks for Ansible to help you out with the upgrade process.

5. **Backup and rollback of switch configurations**

   Lenovo provides modules and sample roles for Ansible that can be leveraged to help you when backing up startup and running switch configurations to remote servers or when restoring previously saved configurations.

6. **Testing and troubleshooting**

   Ansible can be used in network automation and verification tests. Modules intended for testing that include the specific test steps can be written using the Python programming language. When the execution of such modules result in a failure, the logs generated in the *results* directory can be analyzed to determine the cause of the failure.

   The data generated in the logs can used for troubleshooting purposes or it can be gathered for use in other tasks defined further down in the playbook.

As an alternative when executing playbooks that is very helpful for debugging, is the addition of **-v** when running a playbook. This is an optional verbos command that helps identify what is happening during playbook execution by displaying the data that is being returned.

```
$ ansible-playbook <myPlaybook.yml> -v
```

In short, Ansible can perform specific operational and configuration tasks on network devices running Lenovo Cloud Network Operating System (CNOS) or Lenovo Enterprise Network Operating System (ENOS). Such tasks include installing and upgrading the firmware image, deploying new devices in the network, perform configuration changes, retrieving information from Lenovo switches, and resetting, reloading, or shutting down network devices managed by Ansible.

Lenovo Modules for Ansible

Modules are the basic blocks of Ansible. They are commonly used in playbooks or roles to achieve a variety of tasks, but they can also be executed as single actions directly on remote devices using the **`ansible`** command.

This section describes how to use Lenovo's Ansible modules only for versions 2.7 of Ansible. In Ansible 2.7, the Lenovo modules are integrated into the Ansible product. There is no need to download the Lenovo modules or roles as was required for Ansible versions prior to 2.5.

All modules and roles provided by Lenovo for Ansible are prefixed with `cnos_` and `enos_` indicating the switch operating system for which they are supported. These prefixes stands for Lenovo Cloud Network Operating System (CNOS) and Lenovo Enterprise Network Operating System (ENOS). Detailed description of the modules can be found on the Ansible website:

- CNOS - https://docs.ansible.com/ansible/list_of_network_modules.html#cnos
- ENOS - https://docs.ansible.com/ansible/list_of_network_modules.html#enos

Lenovo's Ansible modules are scripts written in the Python programming language and they do the interfacing to the switch through the use of SSH and CLI commands. These modules do not typically require any modifications.

Example roles are described in the documentation of each module. These need modifications to meet the requirements of the network environment in which they are going to be deployed. The sample files are written using industry standard formats: Jinja2 (.j2) - a popular Python template engine, and YAML (.yml) syntax. All the roles are documented using industry standard markdown markup language (.md).

Installation

Install Ansible version 2.7 on a supported operating system (RedHat Linux, Ubuntu etc.) per the instructions provide by Ansible at the following link: https://docs.ansible.com/ansible/intro_installation.html

Supported network devices for Ansible 2.7

Lenovo's modules and roles for Ansible are supported on the following switches running Lenovo Cloud Network Operating System (CNOS) or Lenovo Enterprise Network Operating System (ENOS):

Lenovo NOS Version	Switch Model	Ansible "DeviceType" to use
CNOS 10.8 or later	Lenovo RackSwitch G8272	g8272_cnos
	Lenovo RackSwitch G8296	g8296_cnos
	Lenovo RackSwitch G8332	g8332_cnos
	Lenovo ThinkSystem NE1032 RackSwitch	NE1032
	Lenovo ThinkSystem NE1032T RackSwitch	NE1032T
	Lenovo ThinkSystem NE1072T RackSwitch	NE1072T
	Lenovo ThinkSystem NE2572 RackSwitch	NE2572
	Lenovo ThinkSystem NE10032 RackSwitch	NE10032
	Lenovo ThinkSystem NE0152T RackSwitch	NE0152T
ENOS 8.4 or later	Lenovo RackSwitch G7028	Not Applicable (N/A)
	Lenovo RackSwitch G7052	
	Lenovo RackSwitch G8052	
	Lenovo RackSwitch G8124E	
	Lenovo RackSwitch G8264	
	Lenovo RackSwitch G8264CS	
	Lenovo RackSwitch G8272	
	Lenovo RackSwitch G8296	
	Lenovo RackSwitch G8332	
	Lenovo Flex System Fabric CN4093 10Gb Converged Scalable Switch	
	Lenovo Flex System Fabric EN4093R 10Gb Scalable Switch	

Lenovo NOS Version	Switch Model	Ansible "DeviceType" to use
ENOS 8.4 or later	Lenovo Flex System Fabric SI4093 System Interconnect Module	Not Applicable (N/A)
	Lenovo Flex System SI4091 10Gb System Interconnect Module	
	Lenovo Flex System Interconnect Fabric	
	Lenovo ThinkSystem NE2552E Flex Switch	

Lenovo Ansible Modules Overview

Lenovo has developed several Ansible modules using the Python Programming Language. These modules can be used by network administrators to perform different configuration, maintenance and troubleshooting operations.

Notes:

- As a reference for creating playbooks with Lenovo CNOS or ENOS Network modules, refer to sample playbooks available at the following location:

  `<ansible-install-folder>/test/integration/targets`

- Unit testing code has been introduced for Lenovo CNOS Network modules at the following location:

  `<ansible-install-folder>/test/units/modules/network/cnos/`

Lenovo Network Modules List

Lenovo provides the following Ansible modules:

- cnos_facts - Collects facts on devices running Lenovo CNOS
- cnos_showrun - Collects the current running configuration on devices running Lenovo CNOS
- cnos_image - Performs firmware upgrade/download from a remote server on devices running Lenovo CNOS
- cnos_save - Saves the running configuration as the startup configuration on devices running Lenovo CNOS
- cnos_backup - Backs up the current running or startup configuration to a remote server on devices running Lenovo CNOS
- cnos_rollback - Rolls back the running or startup configuration from a remote server on devices running Lenovo CNOS
- cnos_reload - Performs switch restart on devices running Lenovo CNOS
- cnos_factory - Resets the switch's startup configuration to default (factory) on devices running Lenovo CNOS
- cnos_command - Executes a single command on devices running Lenovo CNOS
- cnos_conditional_command - Executes a single command based on condition on devices running Lenovo CNOS
- cnos_template - Manages switch configuration using templates on devices running Lenovo CNOS
- cnos_conditional_template - Manages switch configuration using templates based on condition on devices running Lenovo CNOS
- cnos_config - Manages Lenovo CNOS configuration sections
- cnos_interface - Manages interface configuration on devices running Lenovo CNOS
- cnos_portchannel - Manages portchannel (port aggregation) configuration on devices running Lenovo CNOS

- cnos_vlan - Manages VLAN resources and attributes on devices running Lenovo CNOS
- cnos_vlag - Manages vLAG resources and attributes on devices running Lenovo CNOS
- cnos_bgp - Manages BGP resources and attributes on devices running Lenovo CNOS
- cnos_restapi - Performs REST API operations from a remote server on devices running Lenovo CNOS
- enos_facts - Collects facts on devices running Lenovo ENOS
- enos_command - Runs commands on remote devices running Lenovo ENOS
- enos_config - Manages Lenovo ENOS configuration sections

Changes in Lenovo Network Modules for Ansible 2.7

Starting with Ansible 2.7, the following changes have been made:

- Support for CLI Keyword Syntax changes introduced in Lenovo Cloud Network Operating System 10.7
- Support for Ansible Persistent Connections framework which allows SSH connection to stay active across multiple Ansible Tasks and speeds up the execution of Network modules

Modules Error Codes for CNOS

When an error occurs during the execution of playbook using Lenovo modules, Ansible returns an error code. These errors codes are listed in the following table:

Error Code	Error Code Description
0	Success
1	Failure
101	Device Response timed out
102	Command not supported - Use CLI command
103	Invalid Context
104	Command value not supported as of now - Use VLAN ID only
105	Invalid interface range
106	Please provide Enable Password
110	Invalid protocol option
111	The value is not integer
112	The value is not float
113	Value is not in range
114	Range value is not integer

Error Code	Error Code Description
115	Value is not in options
116	The value is not long
117	Range value is not long
118	The value cannot be empty
119	The value is not string
120	The value is not matching
121	The value is not IPv4 address
122	The value is not IPv6 address
130	Invalid access map name
131	Invalid VLAN dot1q tag
132	Invalid VLAN filter value
133	Invalid VLAN range value
134	Invalid VLAN ID
135	Invalid VLAN access map action
136	Invalid VLAN access map name
137	Invalid access list
138	Invalid VLAN access map parameter
139	Invalid VLAN name
140	Invalid VLAN flood value
141	Invalid VLAN state value
142	Invalid VLAN last-member-query-interval
143	Invalid Querier IP address
144	Invalid Querier timeout
145	Invalid query interval
146	Invalid VLAN query-max-response-time
147	Invalid VLAN robustness variable
148	Invalid VLAN startup query count
149	Invalid VLAN startup query interval
150	Invalid VLAN snooping version
151	Invalid VLAN ethernet interface
152	Invalid VLAN port tag number

Error Code	Error Code Description
153	Invalid mrouter option
154	Invalid VLAN option
160	Invalid vLAG auto recovery value
161	Invalid vLAG config consistency value
162	Invalid vLAG port aggregation number
163	Invalid vLAG priority value
164	Invalid vLAG startup delay value
165	Invalid vLAG tier ID
166	Invalid vLAG instance option
167	Invalid vLAG keep alive attempts
168	Invalid vLAG keep alive interval
169	Invalid vLAG retry interval
170	Invalid vLAG peer IP VRF value
171	Invalid vLAG health check options
172	Invalid vLAG option
176	Invalid BGP AS number
177	Invalid routing protocol option
178	Invalid BGP address family
179	Invalid AS path options
180	Invalid BGP med options
181	Invalid best path option
182	Invalid BGP local count number
183	Cluster ID has to either IP or AS number
184	Invalid confederation identifier
185	Invalid confederation peer AS value
186	Invalid confederation option
187	Invalid state path relay value
188	Invalid max AS limit as value
189	Invalid neighbor IP address or neighbor AS number
190	Invalid router ID
191	Invalid BGP keep alive interval

Error Code	Error Code Description
192	Invalid BGP hold time
193	Invalid BGP option
194	Invalid BGP address family option
195	Invalid BGP address family redistribution option
196	Invalid BGP address family route map name
197	Invalid next hop critical delay
198	Invalid next hop non critical delay
199	Invalid multipath number value
200	Invalid aggregation group mode
201	Invalid aggregation group number
202	Invalid BFD access VLAN
203	Invalid CFD switchport mode
204	Invalid trunk option
205	Invalid BFD option
206	Invalid portchannel description
207	Invalid portchannel duplex option
208	Invalid flow control option state
209	Invalid flow control option
210	Invalid LACP port priority
211	Invalid LACP timeout options
212	Invalid LACP command options
213	Invalid LLDP TLV option
214	Invalid LLDP option
215	Invalid load interval delay
216	Invalid load interval counter number
217	Invalid load interval option
218	Invalid MAC access group name
219	Invalid MAC address
220	Invalid microburst threshold value
221	Invalid MTU value
222	Invalid service instance value

Error Code	Error Code Description
223	Invalid service policy name
224	Invalid service policy options
225	Invalid interface speed value
226	Invalid storm control level value
227	Invalid storm control option
228	Invalid portchannel dot1q tag
229	Invalid VRRP ID value
230	Invalid VRRP options
231	Invalid portchannel source interface option
232	Invalid portchannel load balance options
233	Invalid portchannel configuration attribute
234	Invalid BFD interval value
235	Invalid BFD minrx value
236	Invalid BFD multiplier value
237	Invalid key chain value
238	Invalid key name option
239	Invalid key ID value
240	Invalid key option
241	Invalid authentication option
242	Invalid destination IP
243	Invalid source IP
244	Invalid IP option
245	Invalid access group option
246	Invalid access group name
247	Invalid ARP MAC address value
248	Invalid ARP timeout value
249	Invalid ARP option
250	Invalid DHCP request option
251	Invalid DHCP client option
252	Invalid DHCP relay IP address
253	Invalid DHCP option

Error Code	Error Code Description
254	Invalid OSPF option
255	Invalid OSPF ID IP address value
256	Invalid IP router option
257	Invalid spanning tree bpdufilter options
258	Invalid spanning tree bpduguard options
259	Invalid spanning tree cost options
260	Invalid spanning tree guard options
261	Invalid spanning tree link-type options
262	Invalid spanning tree link-type options
263	Invalid spanning tree options
264	Port priority in increments of 32 is required
265	Invalid spanning tree VLAN options
266	Invalid IPv6 option
267	Invalid IPv6 neighbor IP address
268	Invalid IPv6 neighbor MAC address
269	Invalid IPv6 DHCP option
270	Invalid IPv6 DHCP relay address option
271	Invalid IPv6 ethernet option
272	Invalid IPv6 VLAN option
273	Invalid IPv6 link local option
274	Invalid IPv6 DHCP option
275	Invalid IPv6 address
276	Invalid IPv6 address option
277	Invalid BFD neighbor options
278	Invalid secondary option
289	Invalid portchannel IPv4 address
290	Invalid max path options
291	Invalid distance local route value
292	Invalid distance internal AS value
293	Invalid distance external AS value
294	Invalid BGP reachability half life

Error Code	Error Code Description
295	Invalid BGP dampening parameter
296	Invalid BGP aggregate prefix value
297	Invalid BGP aggregate prefix option
298	Invalid BGP address family route map name
299	Invalid BGP net IP mask value
300	Invalid BGP net IP prefix value
301	Invalid BGP neighbor configuration option
302	Invalid BGP neighbor weight value
303	Invalid neighbor update source option
304	Invalid ethernet slot/chassis number
305	Invalid loopback interface number
306	Invalid VLAN ID
307	Invalid number of hops
308	Invalid neighbor keep alive interval
309	Invalid neighbor timer hold time
310	Invalid neighbor password
311	Invalid max peer limit
312	Invalid local AS number
313	Invalid maximum hop count
314	Invalid neighbor description
315	Invalid neighbor connect timer value
316	Invalid neighbor address family option
317	Invalid neighbor address family option
318	Invalid route map name
319	Invalid route map
320	Invalid name of a prefix list
321	Invalid filter incoming option
322	Invalid AS path access list name
323	Invalid filter route option

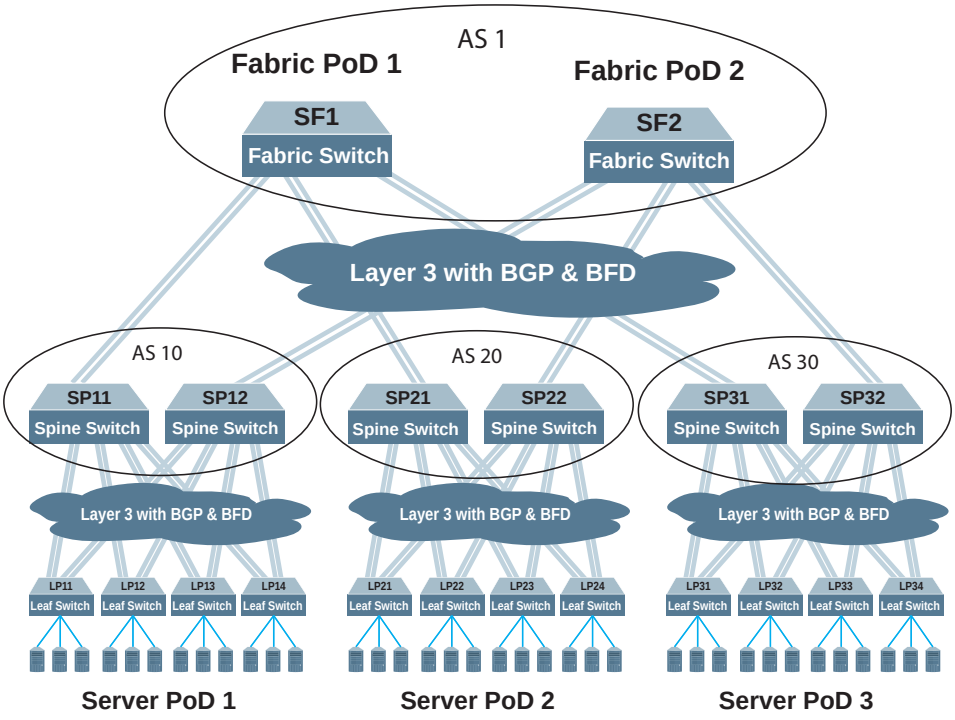
Error Code	Error Code Description
324	Invalid route map name
325	Invalid number of occurrences of AS number
326	Invalid prefix limit

Configuring a Clos Network on Lenovo CNOS using Ansible

A Clos network is a type of non-blocking, multistage switching architecture that reduces the number of ports required in an interconnected fabric. Instead of a hierarchically oversubscribed system of clusters, a Clos network turns your configuration into a high-performance network.

Clos networks are named after Bell Labs researcher Charles Clos, who determined that throughput is increased in a switching array or fabric if the switches are organized in a leaf-spine hierarchy.

In a leaf-spine topology, a series of leaf switches that form the access layer are fully meshed to a series of spine switches. The figure below shows a sample leaf-spine topology with a three-stage multi-Point of Delivery (PoD) Layer 3 Clos network.



Each fabric switch (SF1 and SF2) serves as a fabric PoD that connects to one of the spine switches in each server PoD using BGP and Bidirectional Forwarding Detection (BFD). The spine switches each connect to four leaf switches using BGP and BFD. Using this kind of configuration, you can add capacity to your network by adding another server PoD instead of reconfiguring your existing infrastructure.

Solution Deployment for Ansible 2.7

Installation

The two roles and example playbooks for this solution can be installed from Ansible Galaxy.

- *spine*: https://galaxy.ansible.com/lenovo/cnos-clos-spine/
- *leaf*: https://galaxy.ansible.com/lenovo/cnos-clos-leaf/

Usage

Once installed, there are example playbooks and hosts file for each role within the solution under the *tests* directory:

- *cnos-clos-leaf.yml*
- *cnos-clos-spine.yml*
- *cnos-clos-leaf-hosts*
- *cnos-clos-spine-hosts*

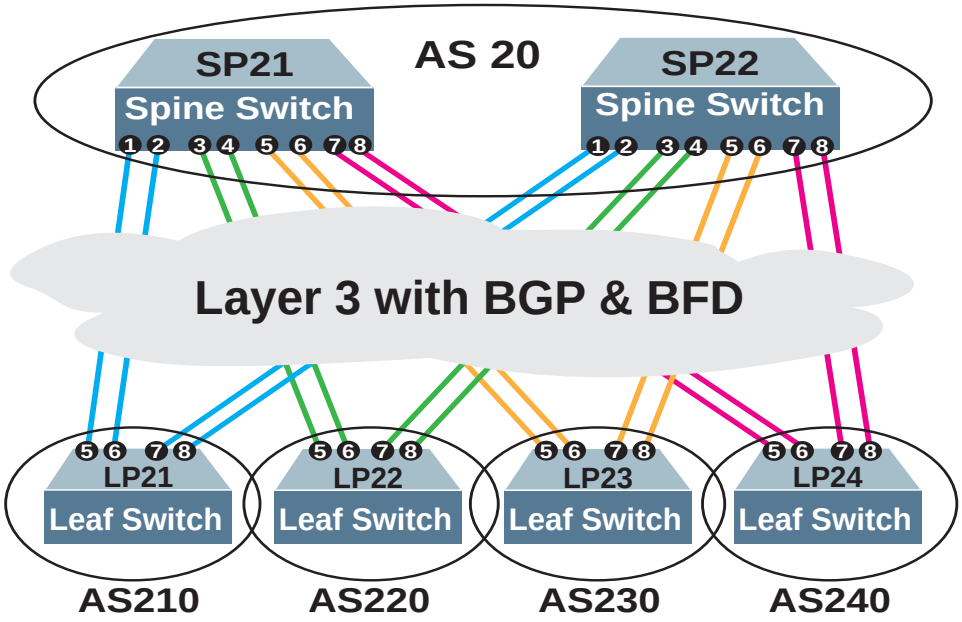
The files required for the role are under the following directories:

- *tasks* - the *main.yml* file in this folder contains all the tasks that need to be executed by the playbook;
- *templates* - contains all the CLI templates that are executed on the devices. Typically there are multiple templates for a role. One is the CLI template and the others consist of the **show** commands relating to the CLI template;
- *vars* - contains all the variables and example values specified in both templates and tasks. You need to edit the *main.yml* file to specify the value for each variable for you environment.

Clos Network BGP Configuration Example

In this configuration example, only the configuration of spine and leaf switches is provided. From the spine and leaf tiers of the network topology, a single switch configuration is covered:

- for the Spine tier: spine switch SP21
- for the Leaf tier: leaf switch LP21



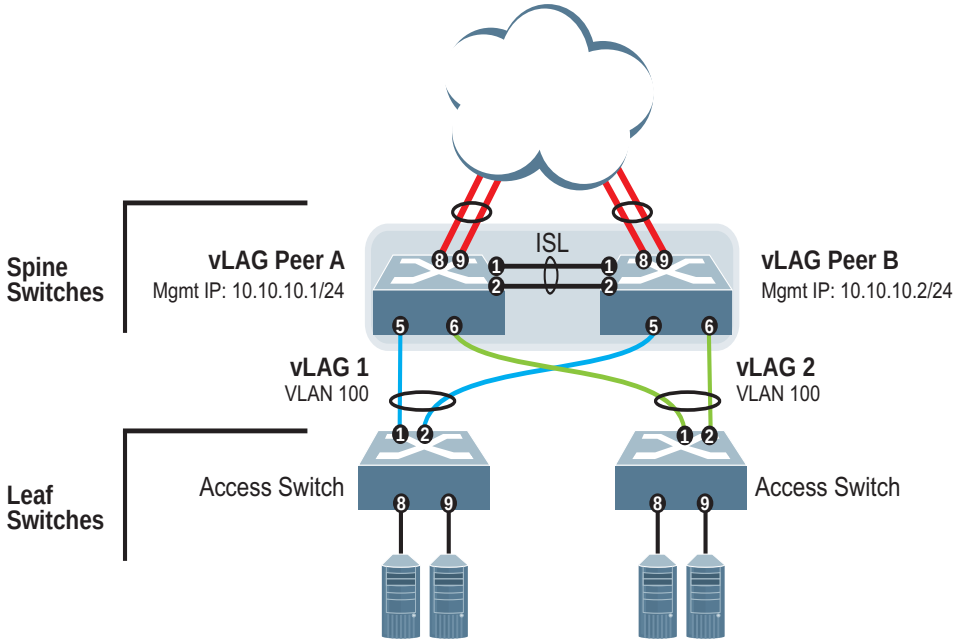
The configurations for the rest of the switches in each tier are similar to the those presented. Only the IP addresses of the switch interfaces, the neighbor addresses and AS numbers of the BGP peers are different.

Autonomous System (AS) membership is as follows:

- AS 20 includes:
  - SP21
  - SP22
- AS 210 includes LP21
- AS 220 includes LP22
- AS 230 includes LP23
- AS 240 includes LP24

Configuring a Single Layer vLAG on Lenovo CNOS using Ansible

The following is an example configuration where two vLAG peers are used for aggregating traffic from downstream devices.



Each access switch is connected to both vLAG peers. On each access switch, the ports connecting to the vLAG peers are configured as members of a LACP LAG. The vLAG peer switches share a dedicated ISL for synchronizing vLAG information. On the individual vLAG peers, each port leading to a specific access switch (and part of the access switch's port LAG) is configured as a vLAG.

In the example configuration, only the configuration for vLAG A on vLAG Peer 1 is shown. vLAG Peer B and vLAG 2 are configured in a similar fashion.

Solution Deployment for Ansible 2.7

Installation

The two roles and example playbooks for this solution can be installed from Ansible Galaxy.

- *spine*: https://galaxy.ansible.com/lenovo/cnos-vlag-1tier-spine/
- *leaf*: https://galaxy.ansible.com/lenovo/cnos-vlag-1tier-leaf/

Usage

Once installed, there are example playbooks and hosts file for each role within the solution under the *tests* directory:

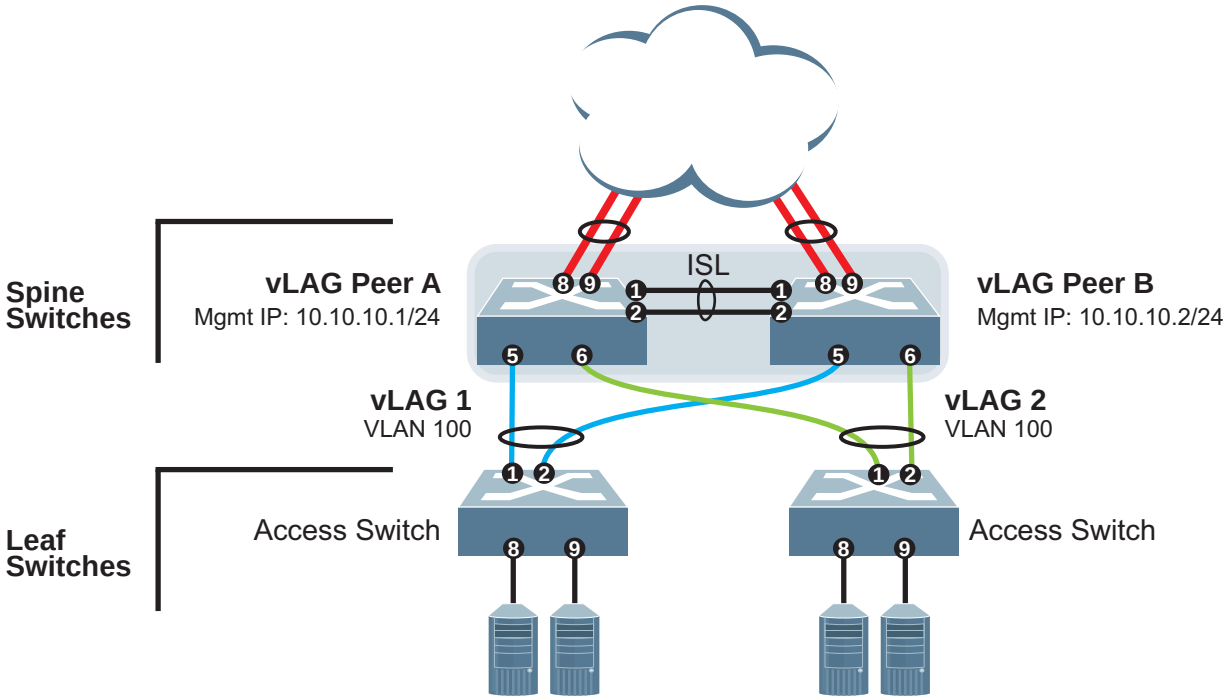
- *cnos-vlag-1tier-leaf.yml*
- *cnos-vlag-1tier-spine.yml*
- *cnos-vlag-1tier-leaf-hosts*
- *cnos-vlag-1tier-spine-hosts*

The files required for the role are under the following directories:

- *tasks* - the *main.yml* file in this folder contains all the tasks that need to be executed by the playbook;
- *templates* - contains all the CLI templates that are executed on the devices. Typically there are multiple templates for a role. One is the CLI template and the others consist of the **show** commands relating to the CLI template;
- *vars* - contains all the variables and example values specified in both templates and tasks. You need to edit the *main.yml* file to specify the value for each variable for you environment.

Single Layer vLAG Configuration Example

In this example, only the configuration for vLAG 1 on vLAG Peer A is shown. vLAG Peer B and vLAG 2 are configured in a similar fashion.

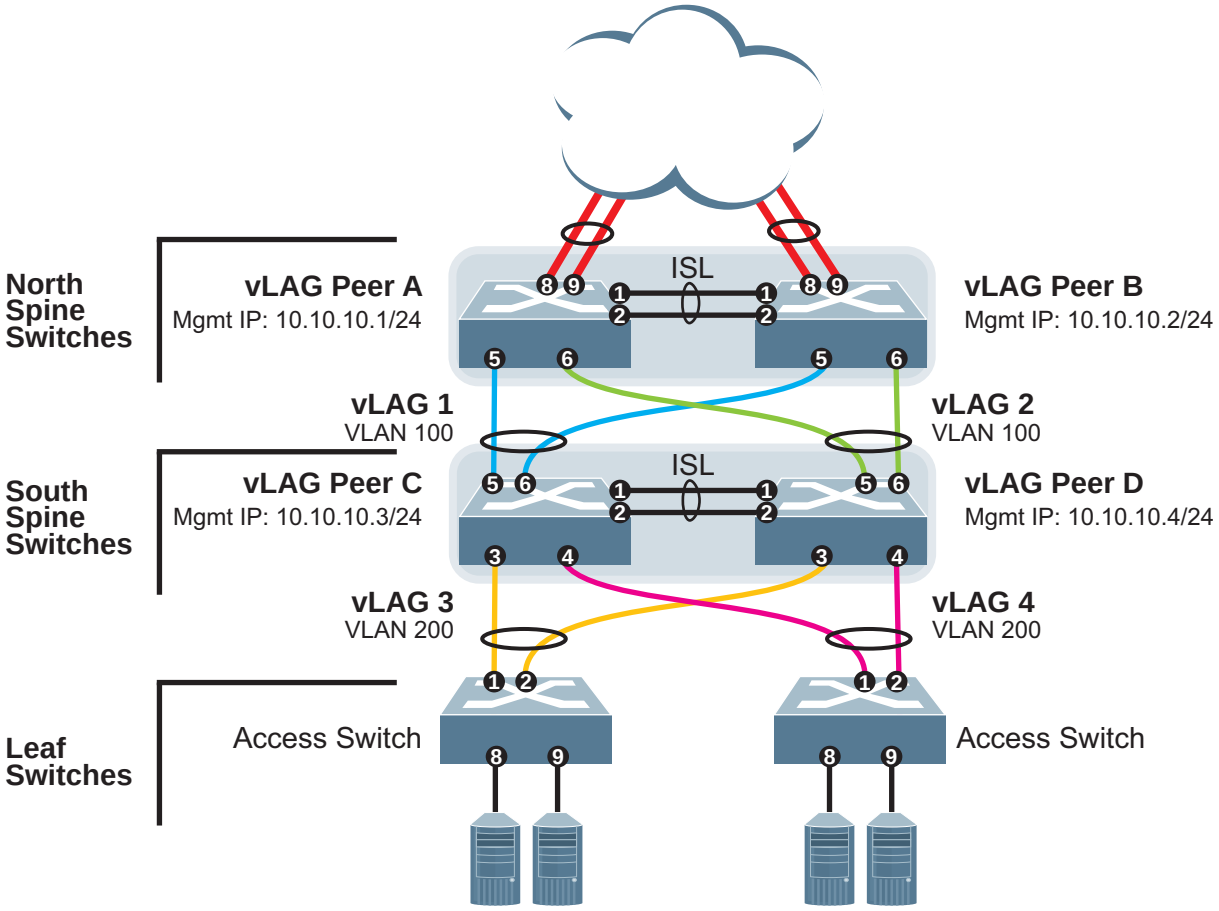


Consider the following vLAG information before starting the configuration:

- the vLAG ISL consists of a LAG encompassing ethernet ports 1/1 and 1/2 on vLAG peers A and B
- vLAG 1 connects the first access switch to ethernet ports 1/5 on vLAG peers A and B
- vLAG 2 connects the second access switch to ethernet ports 1/6 on vLAG peers A and B
- the vLAG peers are referred to as Spine Switches and the access switches are referred to as Leaf Switches

Configuring a Multiple Layer vLAG on Lenovo CNOS using Ansible

The following is an example configuration where several vLAG peers are used for aggregating traffic from downstream devices in a two layer vLAG topology.



vLAG peers A and B share a dedicated ISL for synchronizing vLAG information. The vLAG peers are connected to the downstream vLAG switch C through vLAG 1 and to vLAG switch D through vLAG 2.

vLAG peers C and D share a dedicated ISL for synchronizing vLAG information. The vLAG peers are connected to the upstream vLAG switch A through vLAG 1 and to vLAG switch B through vLAG 2. The vLAG peers are also connected to the downstream access switches through vLAGs 3 and 4.

Solution Deployment for Ansible 2.7

Installation

The two roles and example playbooks for this solution can be installed from Ansible Galaxy.

- *spine*: https://galaxy.ansible.com/lenovo/cnos-vlag-2tier-spine/
- *leaf*: https://galaxy.ansible.com/lenovo/cnos-vlag-2tier-leaf/

Usage

Once installed, there are example playbooks and hosts file for each role within the solution under the *tests* directory:

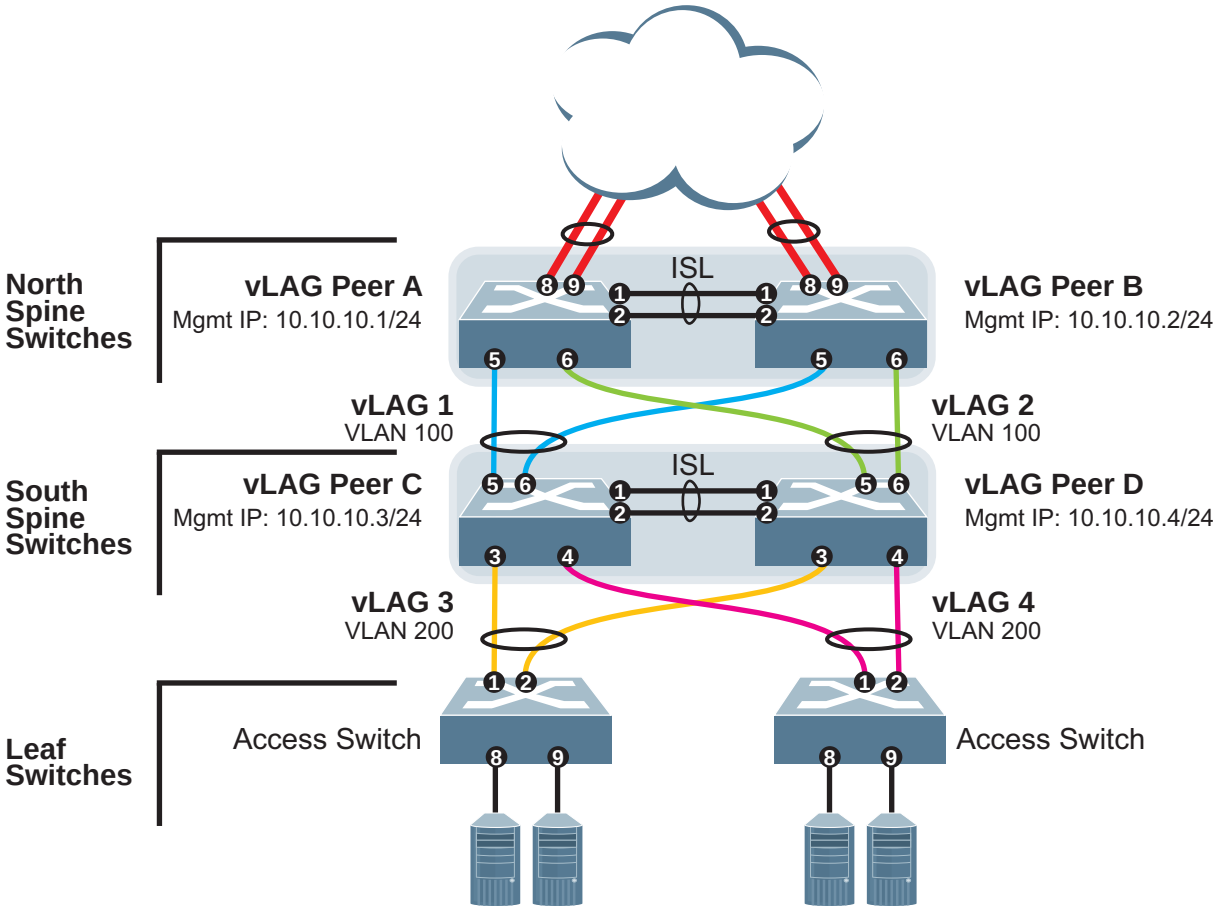
- *cnos-vlag-2tier-leaf.yml*
- *cnos-vlag-2tier-spine.yml*
- *cnos-vlag-2tier-leaf-hosts*
- *cnos-vlag-2tier-spine-hosts*

The files required for the role are under the following directories:

- *tasks* - the *main.yml* file in this folder contains all the tasks that need to be executed by the playbook;
- *templates* - contains all the CLI templates that are executed on the devices. Typically there are multiple templates for a role. One is the CLI template and the others consist of the **show** commands relating to the CLI template;
- *vars* - contains all the variables and example values specified in both templates and tasks. You need to edit the *main.yml* file to specify the value for each variable for you environment.

Multiple Layer vLAG Configuration Example

In this example, only the configurations for vLAG 1 on vLAG Peer A and for vLAG 3 on vLAG Peer C are shown. vLAG Peers B and D and all other vLAGs are configured in a similar fashion.



Consider the following vLAG information before starting the configuration:

- the vLAG ISL consists of a LAG encompassing ethernet ports 1/1 and 1/2 on vLAG peers A and B and on vLAG peers C and D, respectively
- vLAG 1 connects vLAG peer C through ethernet port 1/5 to vLAG peer A on ethernet port 1/5, and through ethernet port 1/6 to vLAG peer B on ethernet port 1/5
- vLAG 2 connects vLAG peer D through ethernet port 1/5 to vLAG peer A on ethernet port 1/6, and through ethernet port 1/6 to vLAG peer B on ethernet port 1/6
- vLAG 3 connects the first access switch through ethernet port 1/1 to vLAG peer C on ethernet port 1/3, and through ethernet port 1/2 to vLAG peer D on ethernet port 1/3

- vLAG 4 connects the second access switch through ethernet port 1/1 to vLAG peer C on ethernet port 1/4, and through ethernet port 1/2 to vLAG peer D on ethernet port 1/4
- vLAG peers A and B are referred to as North Spine Switches, vLAG peers C and D are referred to as South Spine Switches, and the access switches are referred to as Leaf Switches

Configuring Telemetry on Lenovo CNOS using Ansible

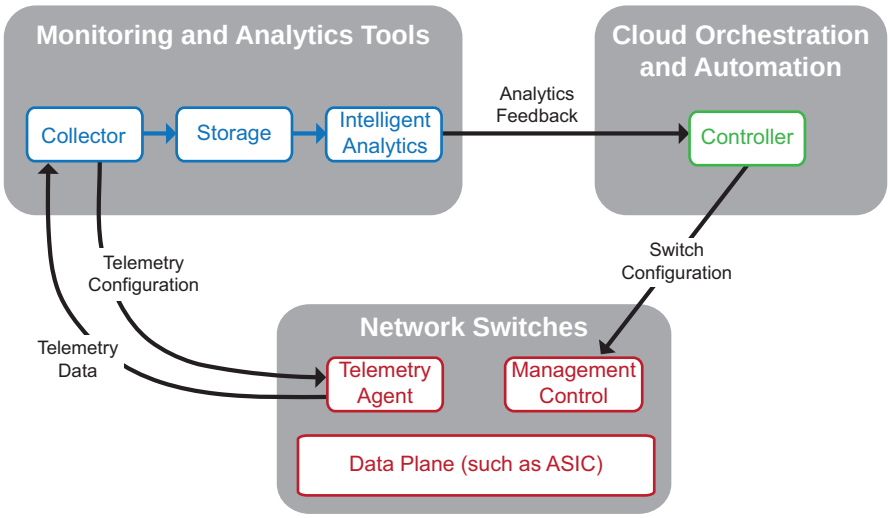
Network devices, such as switches and routers, can be monitored by using network telemetry and providing the collected data to software controllers, which can analyze it. In Lenovo Cloud Network Operating System (CNOS), telemetry is used for pro-active congestion monitoring.

Telemetry enables the continuos monitoring of network devices to detect potential congestion problems, preferably before they happen. This type of monitoring operations address the following scenarios:

- *Long-term congestion*: Packets are discarded by the switch when an ethernet port is close to its line rate or when flows are back pressured for a long time interval due to insufficient buffer space.
- *Microbursts*: Short peaks in data traffic that manifest as a sudden increase in the number of packets transmitted over millisecond-level time frames can potentially overwhelm network buffers an cause packet loss and backpressure. Since microbursts last for very short time periods, they are not detected by traditional methods, such as SNMP or port statistics. In cloud environments, microbursts are even more difficult to detect because of the increased complexity and reduced visibility of the cloud network.

In a complete cloud and data center ecosystem, the ultimate goal is to make sure applications can run in an efficient and reliable fashion. To accomplish this, applications use computing, storage, and network resources as accessories to this goal. Having end-to-end visible information about application performance becomes a critical aspect on modern networks. Telemetry provides real-time monitoring and reporting of how virtual and physical networks are being used.

The figure below shows the key components of a typical telemetry solution:



where:

- The **Collector** efficiently collets, normalizes, and transforms data. This data is used to create different views and help solve various telemetry use cases;

- This data is sent to a time-series resource indexing and metric **Storage** service, which provides a scalable means of storing short-term and long-term data;
- **Intelligent Analytics** trigger actions based on defined rules against sampled or event data collected from the network;
- The analytics are sent to the **Cloud Orchestration and Automation Controller**. The controller sends the best switch configuration to the Management Control agent on the switch;
- The **Telemetry Configuration** tells the **Telemetry Agent** which subset of all supported data types must be sent to the **Collector**;
- The **Telemetry Agent** sends the **Telemetry Data** back to the **Collector**, which sends the **Telemetry Configuration** back to the **Telemetry Agent** on the switch;

CNOS has a telemetry agent that is validated using the open source Ganglia monitoring application. Any application that supports the REST architecture and is capable of exchanging JSON messages over HTTP or HTTPS can be used to interact with the CNOS telemetry agent. The CNOS telemetry agent is built into CNOS and runs on the switch, whereas the telemetry controllers that interact with it run on external systems.

Any external application that uses a standard REST client can interact with the CNOS telemetry agent using the CNOS REST API. For more information about the REST functions supported by the CNOS telemetry agent, see the *Lenovo Network REST API Programming Guide for Lenovo Cloud Network Operating System*.

Solution Deployment for Ansible 2.7

Installation

The role and example playbook for this solution can be installed from Lenovo Github for Ansible.

https://github.com/lenovo/ansible-role-cnos-telemetry/

Note: The role and example playbook are not posted on Ansible Galaxy.

Requirements

To configure telemetry on a CNOS switch using Ansible, the following requirements must be met:

- Ansible 2.7 is installed (for details, consult the official Ansible documentation)
- Lenovo network switch with CNOS 10.5 or later
- REST API feature is enabled on the switch
- SSH connection to the switch (SSH must be enabled on the switch)

Usage

The following are mandatory inventory variables:

Variable	Choice	Description
username		Specifies the username used to log onto the switch.
password		Specifies the password used to log onto the switch.
hostname		Searches the hosts file at */etc/ansible/hosts* and identifies the IP address of the switch on which the role is going to be applied.
deviceType	• g8272_cnos • g8296_cnos	Specifies the type of device from where the configuration is backed up.

The values of the variables used need to be modified to fit the specific scenario in which you are deploying the solution. To change the values of the variables, visit the *vars* directory of each role and edit the *main.yml* file located there. The values stored in this file are used by Ansible when the template is executed.

The syntax for variables in the *main.yml* file is the following:

```
<template variable>:<value>
```

Notes:

- You need to replace the `<value>` field with the value that suits your topology;
- The `<template variable>` fields are taken from the template and it is recommended that they are left unchanged.

Variables for `cnos_restapi` in the context of `cnos_telemetry` configuration:

Variable	Choice	Description
urlpath		Specifies the URL path of the REST API.
use_ssl	• False • True	Specifies the transport layer used by the REST API: • False - HTTP plain-text communication over port 8090 • True - HTTPS encrypted communication
method		The HTTP method of the REST API request.
jsoninp		Input JSON dictionary. It is used by POST and PUT methods to input request parameters.

Following are the REST APIs and their corresponding parameters used by the `cnos_telemetry` role:

URL Path	Method	Description
/nos/api/cfg/telemetry/bst/feature	PUT	Configures the BST feature for the telemetry report.
/nos/api/cfg/telemetry/bst/tracking	PUT	Configures the tracking of BST realms.
/nos/api/cfg/telemetry/bst/threshold	PUT	Configures the BST threshold for a BST realm.
/nos/api/info/telemetry/bst/congestion-drop-counters	POST	Configures the BST congestion drop statistics report.

Following are the `jsoninp` parameters for the `/nos/api/cfg/telemetry/bst/feature` REST API:

Variable	Values	Description
bst-enable	• 0 • 1	Configures the status of BST: • 0: disable • 1: enable
collection-interval	1-600 (seconds)	Configures the time interval between consecutive heartbeat messages. These allow collectors to learn about the switches present in the network.
send-async-reports	• 0 • 1	Configures the asynchronous collection of buffer statistics: • 0: enable • 1: disable
trigger-rate-limit	1-5	Configures the maximum number of trigger reports for the configured time interval.
trigger-rate-limit-interval	• 0 • 10-600 (seconds)	Configures the time interval when trigger reports are rate limited. 0: trigger reports are not rate limited
send-snapshot-on-trigger	• 0 • 1	Configures the type of buffer statistics included in trigger reports sent by the Agent: • 0: trigger reports contain buffer statistics only for the counter for which the trigger was raised • 1: trigger reports contain buffer statistics for all configured realms
async-full-report	• 0 • 1	Configures whether full BST reports are asynchronously sent to the Collector: • 0: disable • 1: enable

Following are the `jsoninp` parameters for the `/nos/api/cfg/telemetry/bst/tracking` REST API:

Variable	Values	Description
track-egress-port-service-pool	• 0 • 1	Configures the tracking of egress port service pool statistics: • 0: disable • 1: enable
track-egress-uc-queue	• 0 • 1	Configures the tracking of egress unicast queue statistics: • 0: disable • 1: enable
track-egress-rqe-queue	• 0 • 1	Configures the tracking of egress RQE queue statistics: • 0: disable • 1: enable
track-egress-cpu-queue	• 0 • 1	Configures the tracking of egress CPU queue buffers: • 0: disable • 1: enable
track-ingress-port-service-pool	• 0 • 1	Configures the tracking of ingress port service pool statistics: • 0: disable • 1: enable
track-ingress-service-pool	• 0 • 1	Configures the tracking of ingress service pool statistics: • 0: disable • 1: enable
track-egress-mc-queue	• 0 • 1	Configures the tracking of egress multicast queue statistics: • 0: disable • 1: enable
track-peak-stats	• 0 • 1	Configures whether the switch tracks current buffer usage or peak buffer usage: • 0: track current buffer usage • 1: track peak buffer usage

Variable	Values	Description
track-ingress-port-priority-group	• 0 • 1	Configures the tracking of ingress port priority group statistics: • 0: disable • 1: enable
track-egress-service-pool	• 0 • 1	Configures the tracking of egress service pool statistics: • 0: disable • 1: enable
track-device	• 0 • 1	Configures the tracking of device statistics: • 0: disable • 1: enable

Following are the `jsoninp` parameters for the `/nos/api/cfg/telemetry/bst/threshold` REST API:

Realm	Index 1	Index 2	Values in the Realm Report
ingress-port-priority-group	interface	priority-group	um-share-threshold
ingress-port-service-pool	interface	service-pool	um-share-threshold
ingress-service-pool	service-pool		um-share-threshold
egress-port-service-pool	interface	service-pool	um-share-threshold, uc-share-threshold
egress-service-pool	service-pool		um-share-threshold, mc-share-threshold
egress-rqe-queue	queue		rqe-threshold
egress-cpu-queue	queue		cpu-threshold
egress-mc-queue	queue		mc-threshold
egress-uc-queue	queue		uc-threshold
device			threshold

Following are the permitted values for the `jsoninp` indexes for the `/nos/api/cfg/telemetry/bst/threshold` REST API:

Index	Values	Description
rqe-threshold	1-100	The RQE queue threshold value as a percentage.
cpu-threshold	1-100	The CPU queue threshold value as a percentage.
uc-threshold	1-100	The unicast queue threshold value as a percentage.
mc-threshold	1-100	The multicast queue threshold value as a percentage.
queue	integer	The queue member with respect to the realm.
service-pool	integer (0-1)	The service pool.
priority-group	integer (0-7)	The priority group.
interface	string	The ethernet port of the device.

Following are the `jsoninp` parameters for the `/nos/api/info/telemetry/bst/congestion-drop-counters` REST API:

Variable	Values	Description
request-type	• top-drops • top-port-queue-drops • port-drops • port-queue-drops	Configures the request of specific sets of drop statistics: • top-drops: drop statistics for switch ports suffering maximum congestion • top-port-queue-drops: drop statistics for pairs of switch ports and queues suffering maximum congestion • port-drops: drop statistics for switch ports • port-queue-drops: drop statistics for pairs of switch ports and queues
collection-interval	• 0 • 10-3600 (seconds)	Configures the time interval for the collection of congestion drop statistics. 0: disables the collection of congestion drop statistics
request-param	Dictionary (see below)	Request parameters for congestion drop statistics.

`request-param` is a dictionary containing the following:

Variable	Values	Description
count	integer	The number of requested records.
interface-list	list	The list of switch interfaces.
queue-type	• all • ucast • mcast	The type of queues: • all: both unicast and multicast queues • ucast: only unicast queues • mcast: only multicast queues
queue-list	list	The list of queue IDs.

Telemetry Role Template

The `cnos_telemetry` role uses the following `cnos_template`:

```
feature telemetry
telemetry controller ip {{item.controllerip}}
port {{item.controllerport}} vrf {{item.vrf}}
telemetry heartbeat enabled interval {{item.hbinterval}}
```

where:

Variable	Values	Description
controllerip	IP address	Specifies the controller IP address.
controllerport	TCP port number (integer)	Specifies the listening TCP port for the controller.
vrf	• default • management	Specifies on what VRF instance is the controller configured.
hbinterval	1-600 (seconds)	Specifies the time interval between consecutive heartbeat messages.

If you want to disable heartbeat messages, use the following template:

```
feature telemetry
telemetry controller ip {{item.controllerip}}
port {{item.controllerport}} vrf {{item.vrf}}
telemetry heartbeat disable
```

Dependencies:

- `username.iptables` - Configures the firewall to block all ports except those needed for HTTP (port 8090) or HTTPS (port 443) server and the SSH server
- `/etc/ansible/hosts` - You must edit the *hosts* file located here with the device information of the switches

Ansible keeps track of all network elements that it manages through a *hosts* file. Before the execution of a playbook, the *hosts* file must be configured.

Open the *hosts* file at `/etc/ansible/hosts` with root privileges. Most of the file is commented out by using #. You can also comment out the entries you add by using #. You need to copy the content of the *hosts* file for the role into the `/etc/ansible/hostsfile`. For example:

```
[cnos_restapi_sample]
10.241.107.39   username=<username> password=<password>
deviceType=g8272_cnos
10.241.107.40   username=<username> password=<password>
deviceType=g8272_cnos
```

Note: The IP addresses need to be changed to fit your specific topology. You also need to replace `<username>` and `<password>` with the appropiate values used to log onto the specific Lenovo switch.

Telemetry Configuration Example

To execute an Ansible playbook, use the following command:

```
$ ansible-playbook cnos_telemetry_sample.yml -vvv
```

`-vvv` is an optional verbos command that helps identify what is happening during playbook execution. The playbook for each role is located in the main directory of the solution.

Following is an example of a telemetry configuration using Ansible:

```
- name: Module to configure telemetry reports
   hosts: cnos_telemetry_sample
   gather_facts: no
   connection: local
   roles:
    - cnos_telemetry_sample
```

Appendix A. Getting help and technical assistance

If you need help, service, or technical assistance or just want more information about Lenovo products, you will find a wide variety of sources available from Lenovo to assist you.

Use this information to obtain additional information about Lenovo and Lenovo products, and determine what to do if you experience a problem with your Lenovo system or optional device.

Before you call, make sure that you have taken these steps to try to solve the problem yourself.

If you believe that you require warranty service for your Lenovo product and you have purchased the plug-in through the “Lenovo Networking Bundle for vRealize”, the service technicians will be able to assist you more efficiently if you prepare before you call.

- Go to the Lenovo Support portal to check for information to help you solve the problem.
- Gather the following information to provide to the service technician. This data will help the service technician quickly provide a solution to your problem and ensure that you receive the level of service for which you might have contracted.
  - Pertinent information such as error messages and logs
- Start the process of determining a solution to your problem by making the pertinent information available to the service technicians. The service technicians can start working on your solution as soon as you have completed and submitted an Electronic Service Request.

You can solve many problems without outside assistance by following the troubleshooting procedures that Lenovo provides in the online help or in the Lenovo product documentation. The Lenovo product documentation also describes the diagnostic tests that you can perform. The documentation for most systems, operating systems, and programs contains troubleshooting procedures and explanations of error messages and error codes. If you suspect a software problem, see the documentation for the operating system or program.

Appendix B. Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area.

Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service.

Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
1009 Think Place - Building One
Morrisville, NC 27560
U.S.A.

Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties.

Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary.

Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

Lenovo, the Lenovo logo, Flex System, System x, NeXtScale System, ThinkSystem and X-Architecture are trademarks of Lenovo in the United States, other countries, or both.

Ansible is a trademark of Red Hat, Inc.

Other company, product, or service names may be trademarks or service marks of others.