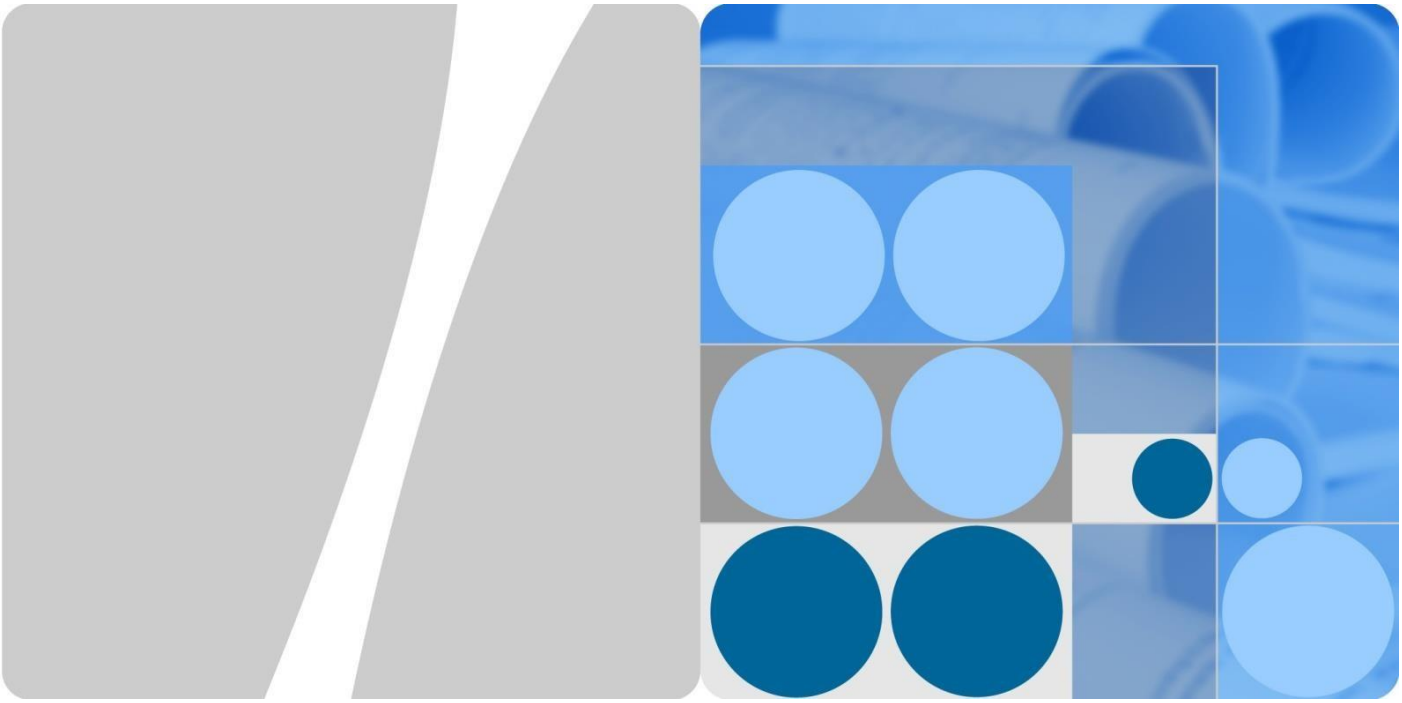


Ansible Automation Technology White Paper

Issue 01

Date 2018-01-20

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Tel: 4008302118

Contents

Figures

Ansible Automation Technology White Paper

Keywords:

Ansible, automation

Abstract:

Ansible enables automated management of CE switches during data center O&M.

1 Background and Pain Points

The rapid development of the data center poses an urgent requirement for automated service deployment and O&M to address challenges such as insufficient O&M manpower, low maintenance efficiency, and long service delivery time.

Currently, zero touch provisioning (ZTP) can only complete initial configuration and does not provide the batch modification function.

Currently, popular automated O&M tools include chef, SaltStack, and Ansible. Table 1-1 compares these tools.

Table 1-1 Comparison between automated O&M tools

	Language	Whether an Agent is Required
Ansible	Python	No
chef	ruby	Yes
SaltStack	python	Yes

Different from its equivalents, Ansible does not require an agent on the device to which Ansible connects. Therefore, the Ansible version is decoupled from the device software version, which is why Ansible is popular among customers.

Figure 1-2 Ansible architecture

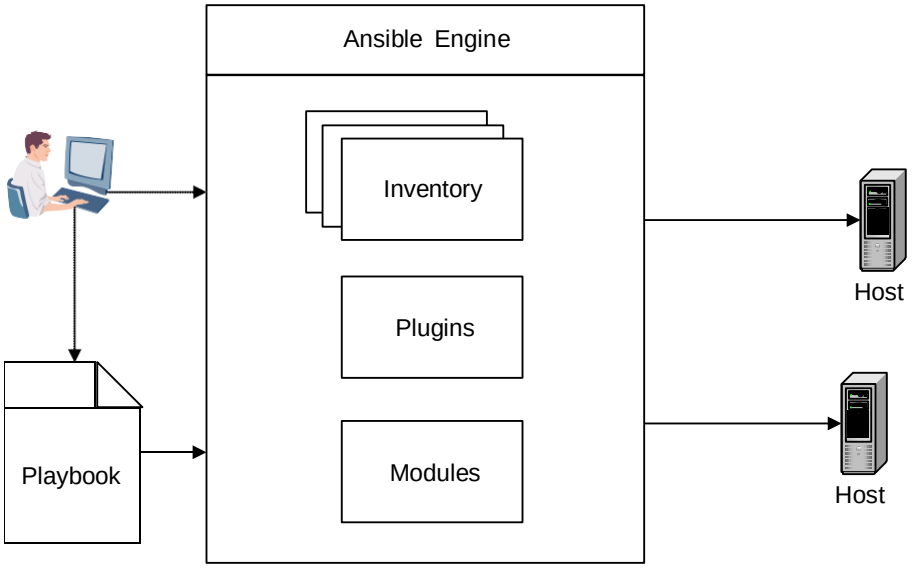
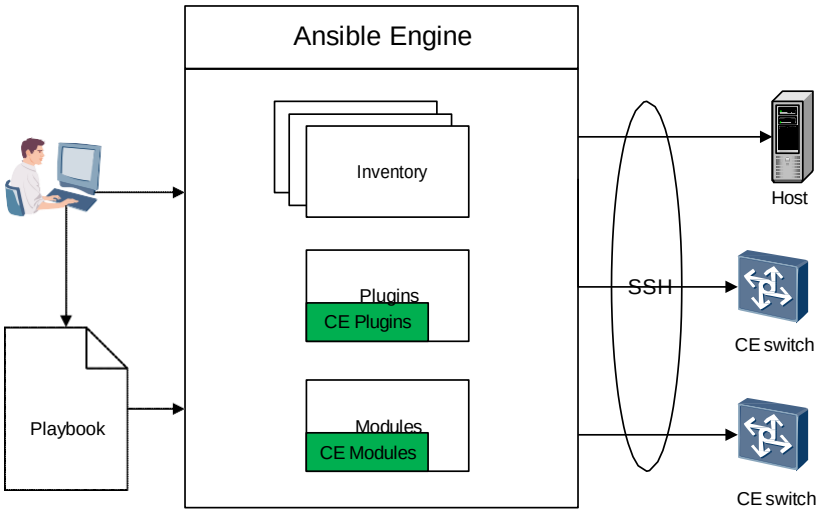


Table 1-2 Description of Ansible modules

Module	Description
Playbook	Is responsible for service orchestration based on modules.
Inventory	Defines managed objects, such as the device IP address and alias.
Modules	Indicates service interfaces, for example, for deploying a database or uninstalling software.
Plugins	Provides a framework for extending functions, such as NETCONF connection and email notification triggering.

Huawei CloudFabric open ecosystem is developed to eliminate interoperability problems between cloud platforms, management tools, and network devices from different vendors and thereby improve capabilities for integrated deployment and maturity of data center network solutions. Huawei has expanded the Ansible platform in order to manage CE switches on this platform.

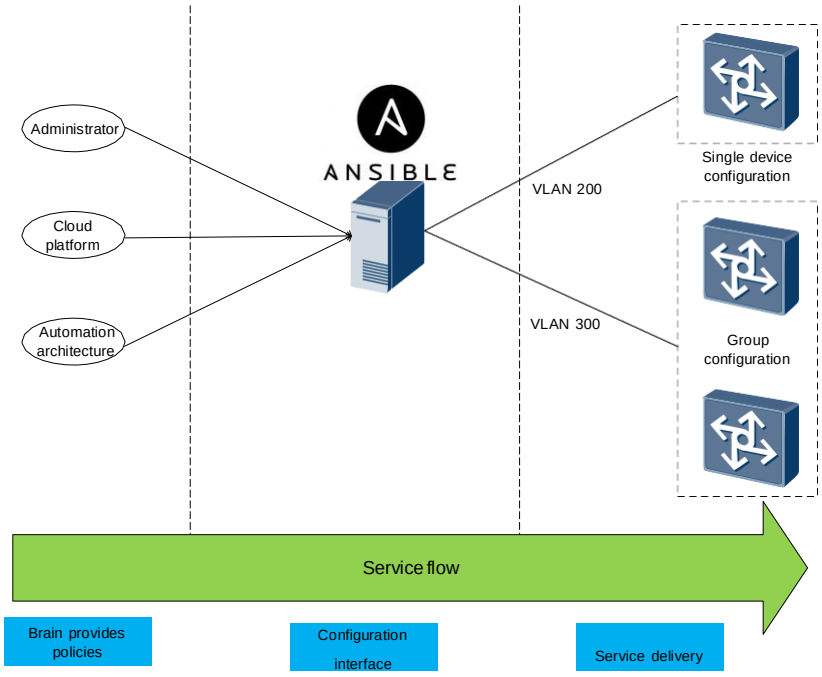
Figure 1-3 Ansible architecture expanded with CE plugins and modules



Ansible provides a network automation platform to make network management to be as simple, efficient, and plug-in-free as system and application management. Huawei CloudFabric solution integrates with Ansible to allow automated network O&M and management to be more secure, efficient, and reliable.

In terms of the tool positioning, Ansible is a configuration management tool without a "brain". Figure 1-4 shows typical application of Ansible.

Figure 1-4 Typical application of Ansible

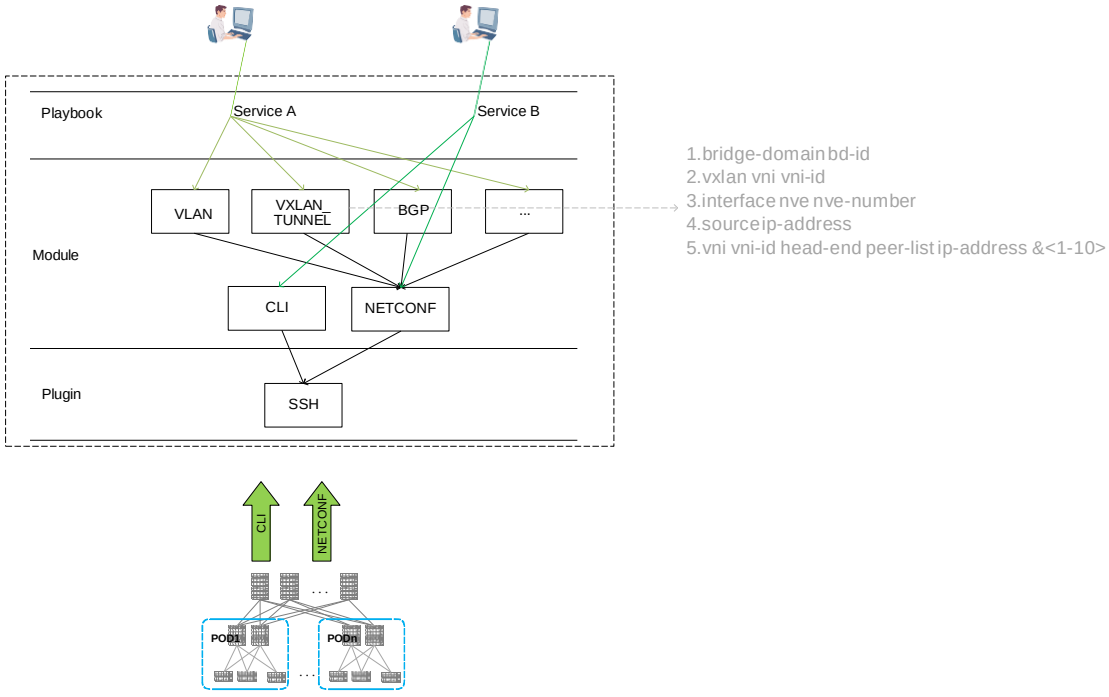


Service policies can be managed by the administrator, cloud platform, and automation framework, depending on the actual application scenario.

2 Ansible Solution

2.1 Introduction to the Solution for Connecting Ansible and CE Switches

Figure 2-1 Connecting Ansible to CE switches



Huawei provides multiple types of service modules that function as APIs for playbooks to invoke. These modules further encapsulate the CLI or NETCONF interfaces for typical services (for example, encapsulates the six CLI commands for configuring a basic VXLAN Layer 2 gateway into an API), which reduces the development workload.

For the modules that are provided by Huawei, visit https://github.com/HuaweiSwitch/CloudEngine-Ansible/tree/master/library.

Table 2-1 provides the features and functions for which modules are provided.

Table 2-1 Features and functions for which modules are provided

Basic Features	L2 and L3 Features	ACL and Overlay	Status Query Functions
Command/Configure	Interface (L2, L3)	ACL	Function for querying device information, such as the model and software version.
AAA	DLDP	VXLAN	Interface status query function
NTP	VLAN	EVPN	Function of collecting statistics about transmitted and received packets on an interface.
SNMP	Eth-Trunk		
Syslog	M-LAG		
Rollback	STP		
Reboot	Static Route		
CopyFile	OSPF		
Setting the system software package for next startup	BGP		

NOTE

The features listed in Table 2-1 are used to complete basic and common functions, and not all parameters can be delivered through Ansible.

Customers can invoke several modules in playbooks to deploy services, such as service A in Figure 2-1.

If existing modules cannot meet customers' requirements, the customers can invoke the modules that provide CLI or NETCONF interfaces in playbooks to develop their own playbooks or modules, such as service B in Figure 2-1.

The following section describes how to use Huawei Ansible modules.

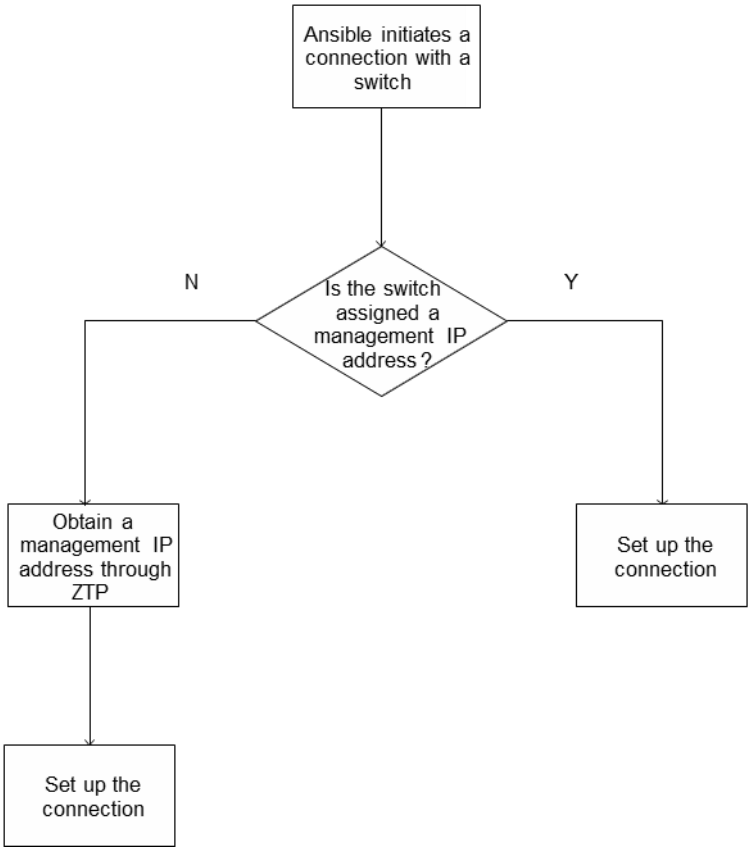
2.2 Using Huawei Ansible Modules

If a switch has been installed on the network, you can use the management IP address to remotely log in to the switch, without the need to use ZTP.

If a switch is just delivered to the site and has not been installed, you need to use Ansible to automatically deploy services by following the procedure provided in this section.

Figure 2-2 shows the process of setting up a connection between Ansible and a switch.

Figure 2-2 Process of setting up a connection between Ansible and a switch



2.2.1 Installing Ansible

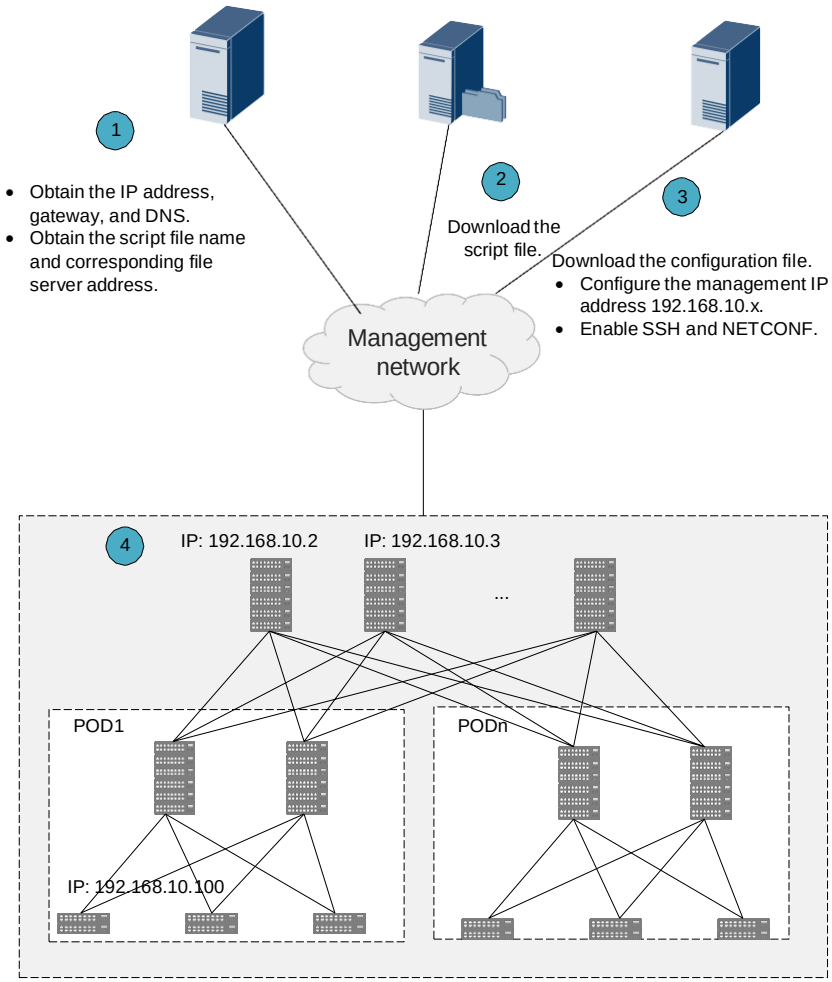
Huawei Ansible modules are released on GitHub at https://github.com/HuaweiSwitch/CloudEngine-Ansible.

For details about the installation procedure, see the **install.sh** file.

2.2.2 Using ZTP to Deploy Devices

After Ansible is installed, it can manage a device only when the device IP address is reachable. If a large number of devices need to be deployed and configured, ZTP is a desired tool to use because configuration using the serial port is inefficient. Figure 2-3 shows the process of deploying devices using ZTP

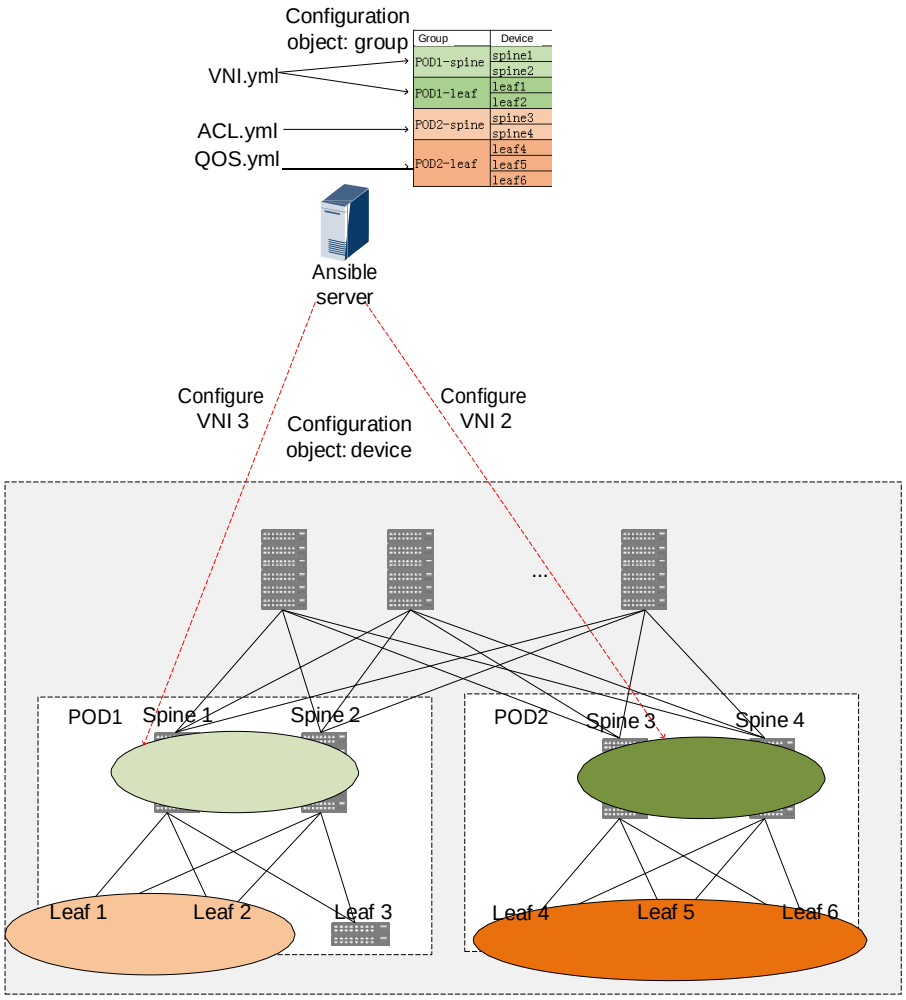
Figure 2-3 Process of deploying devices using ZTP



Steps 1 and 2 are standard ZTP steps, and step 3 is a simplified one, which only involves configuring the management IP address and enabling SSH and NETCONF. After step 4 is complete, each device has an independent management IP address and the NETCONF and SSH connections are enabled.

2.2.3 Configuring Services in Batches

Figure 2-4 Batch configuration of services



Divide configuration objects into groups. Table 2-2 provides group information.

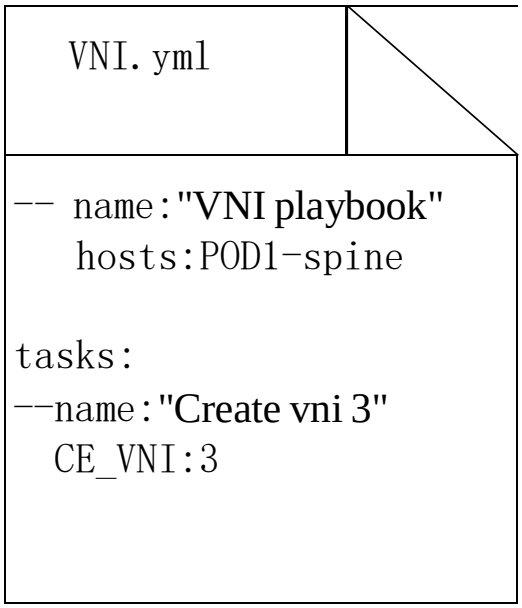
Table 2-2 Groups of configuration objects

Group Name	Configuration Objects	Description
POD1-spine	IP addresses of spine devices in POD1	The modification for group POD1-spine takes effect on spine 1 and spine 2.
POD1-leaf	IP addresses of leaf devices in POD1	The modification for group POD1-leaf takes effect on leaf 1, leaf 2, and leaf 3.
POD2-spine	IP addresses of spine devices in POD2	The modification for group POD2-spine takes effect on spine 3 and spine 4.

Group Name	Configuration Objects	Description
POD2-leaf	IP addresses of leaf devices in POD2	The modification for group POD2-leaf takes effect on leaf 4, leaf 5, and leaf 6.

Compile yml files based on service requirements to invoke required modules. Figure 2-5 provides an example yml file.

Figure 2-5 Example yml file



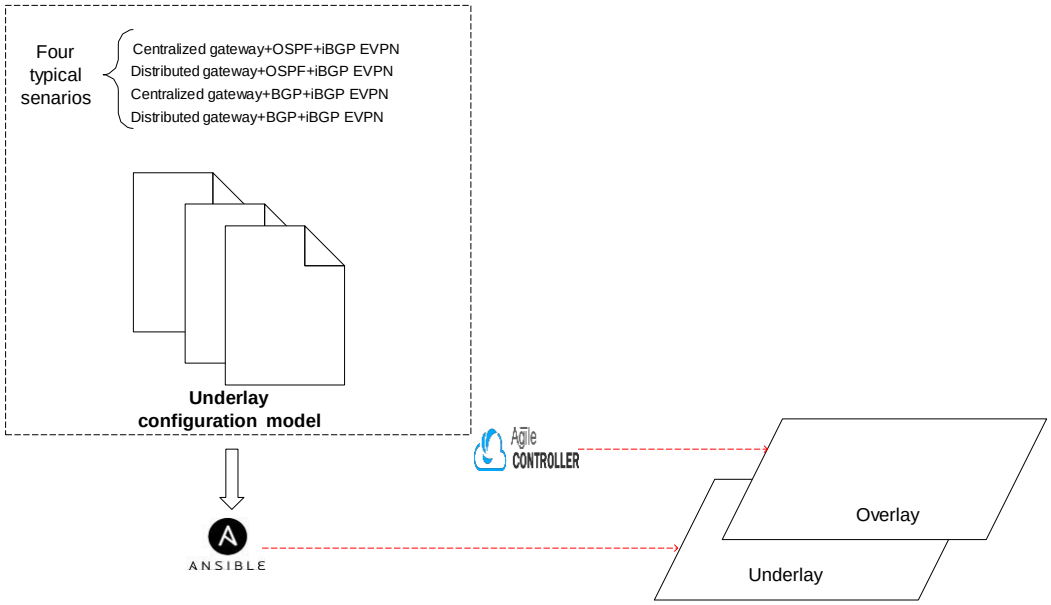
After the **VNI.yml** script is executed, Ansible will automatically create VNI 3 on spine 1 and spine 2.

2.3 Application Scenarios Recommended by Huawei

2.3.1 Standard Templates

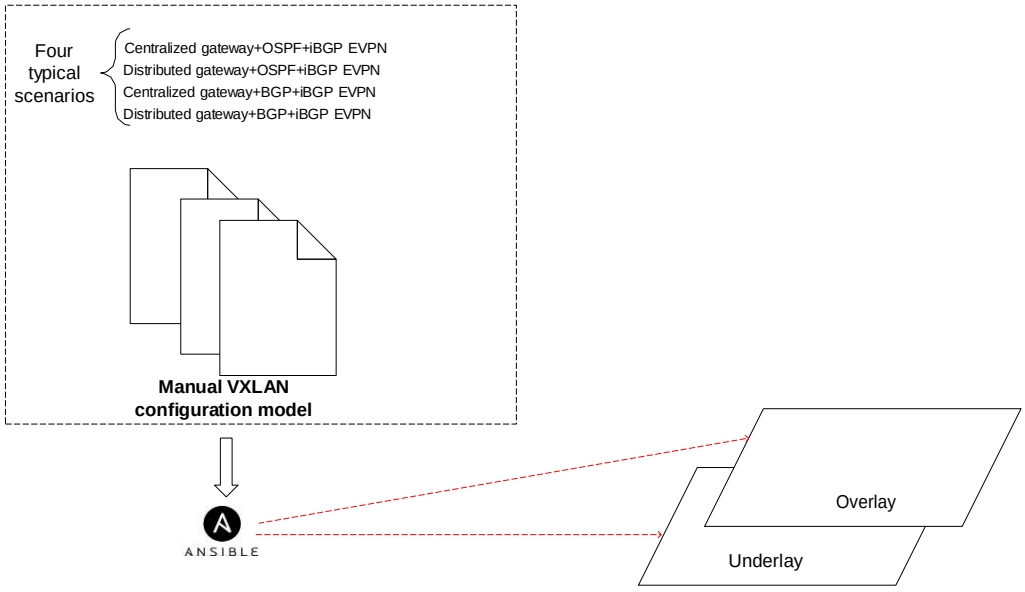
Based on the characteristics of Ansible and years of deployment experience, Huawei developed multiple deployment templates to help customers quickly deploy services, improving efficiency in initial service deployment.

2.3.1.1 Template for DCN3.0 Delivery



In the DCN3.0 delivery scenario, Ansible is used to configure the underlay network.

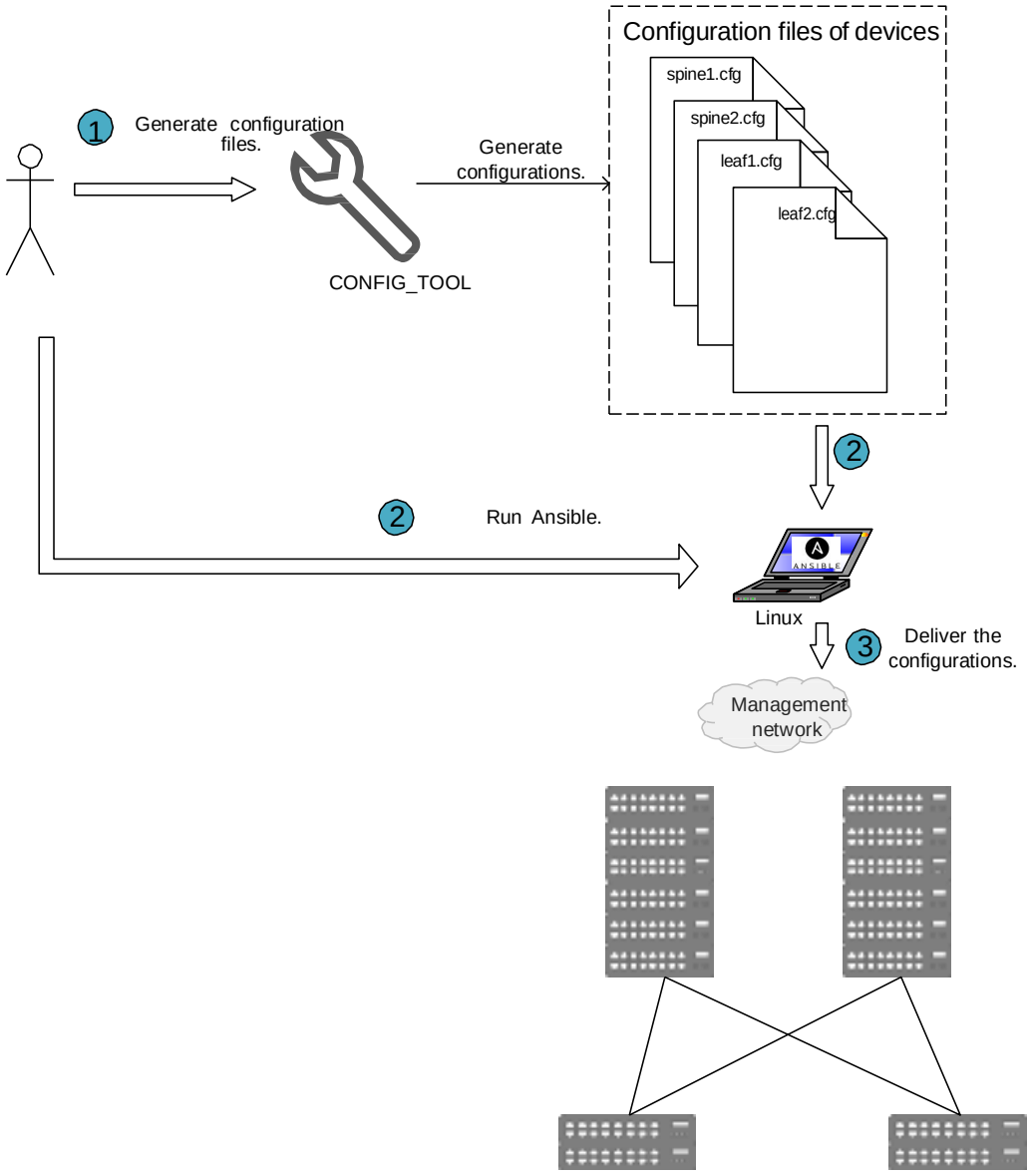
2.3.1.2 Template for Manual VXLAN Delivery



If no Agile Controller is deployed, Ansible templates cover the configurations of both underlay and overlay networks.

2.3.1.3 Process of Using a Template to Configure Overlay and Underlay Networks

Figure 2-6 Process of using a template

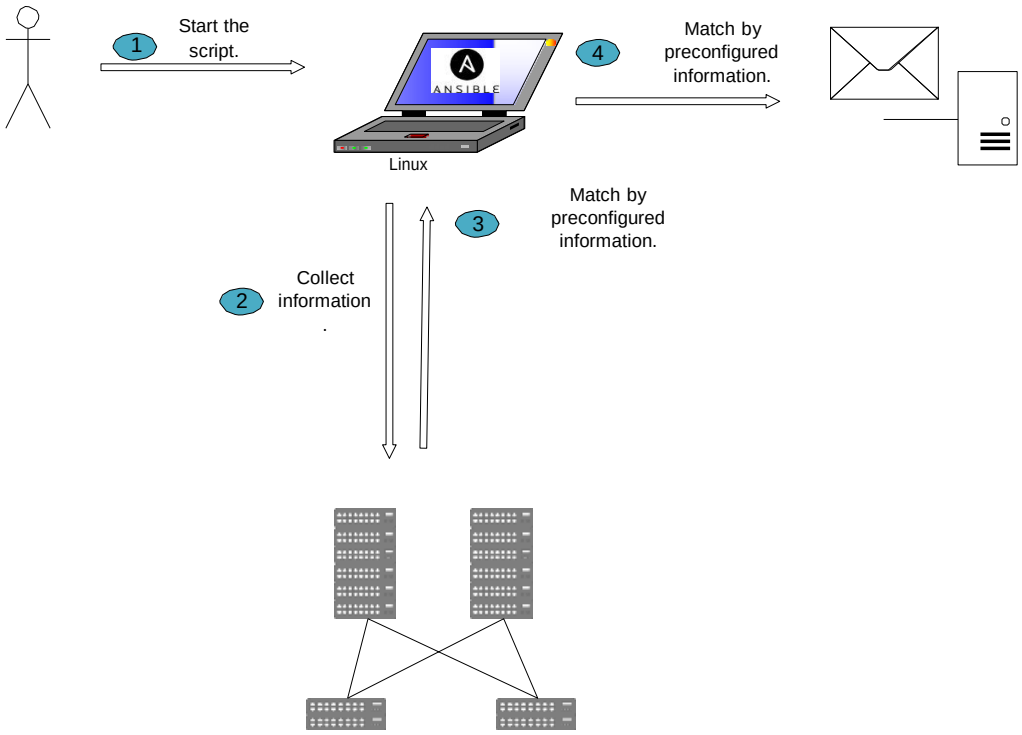


The process of using a template is as follows:

1. Use tools provided by Huawei to generate device configurations based on the typical scenario.
2. Copy the configurations to Ansible, and run Ansible.
3. Ansible updates configurations for devices.

2.3.2 Using Ansible to Perform O&M

Figure 2-7 Process of using Ansible to perform O&M



The administrator can set concerned information, such as the MAC address table size and optical module power, in playbooks provided by Ansible. The playbooks then periodically collect the information from devices and process it. If an exception is detected, Ansible triggers an email notification.

3 Method of Using Ansible

3.1 Preparing the Environment

Table 3-1 lists environment version information required for installing Ansible.

Table 3-1 Environment version information

Environment Category	Version
Operating system type	Debian, Red Hat, Ubuntu, CentOS in various versions
Python version	Python 2.7 or later
Ansible framework version	Ansible v2.4.2.0 or later

3.1.1 Configuring a CE Switch

Ansible establishes a connection with a CE switch using SSH. Therefore, you need to configure an SSH login user on the CE switch.

The procedure of configuring an SSH user on the CE switch is as follows:

Step 1 Generate a local key pair on the CE switch.

```
<HUAWEI> system-view
[~HUAWEI] rsa local-key-pair create    // Generate the local RSA host and server
key pairs.
The key name will be: HUAWEI
The range of public key size is (2048~2048).  NOTE:
Key pair generation will take a short while.
[*HUAWEI] commit
```

Step 2 Configure the SSH user login interface.

```
[HUAWEI] user-interface vty 8 13
[HUAWEI-ui-vty13] authentication-mode aaa
[*HUAWEI-ui-vty13] commit
[*HUAWEI-ui-vty13] protocol inbound ssh
[*HUAWEI-ui-vty13] commit
```

Step 3 Create an SSH user on the CE switch.

```
# Create an SSH user root0001. [HUAWEI]
aaa
[HUAWEI-aaa] local-user root0001 password irreversible-cipher Root_$0001    //Configure a local user name and password.
[*HUAWEI-aaa] local-user root0001 level 3    // Set the local user level to 3.
[*HUAWEI-aaa] local-user root0001 service-type ssh    // Configure the VTY user interface to support the SSH protocol.
[*HUAWEI-aaa] quit
[*HUAWEI] ssh user root0001 authentication-type password    // Set the authentication mode for the SSH user root001 to password authentication.
[*HUAWEI] commit
```

Step 4 Enable the STelnet server function on the CE switch.

```
[HUAWEI] stelent server enable
[*HUAWEI] commit
```

Step 5 Set the service type of the SSH user root001 to **all**.

```
[HUAWEI] ssh user root0001 service-type all
[*HUAWEI] commit
```

----End

3.1.2 Installing Ansible

The following uses ubuntu 14 as an example to describe how to install Ansible and use CE modules. The procedure varies in other operating systems because different Ansible versions are used.

Command to install Ansible:

```
#sudo pip install -v ansible==2.4.2.0
```

Huawei Ansible modules have been integrated into the Ansible mainline. You can directly use the Huawei modules so long as you have Ansible v2.4.2.0 or a later version installed.

3.2 Configuration Procedure

3.2.1 Creating the Inventory Hosts File

When managing a large-scale network, network administrators need to manage hosts running different services. Network devices can be considered as hosts for Ansible. Information of these hosts is saved in Ansible's inventory hosts file. Ansible's inventory hosts file is a static file in INI format and is stored in the **/etc/ansible/hosts** directory by default.

NOTE

You can specify the directory using the **ANSIBLE_HOSTS** environment variable or using the **-i** parameter when running Ansible and Ansible-playbook.

The **/etc/hosts** file contains IP addresses and corresponding host names. Updating the file is not mandatory, but you can modify the file to facilitate host IP address maintenance. For example, you can add the following host information to the **/etc/hosts** file.

```
# vi /etc/hosts
127.0.0.1 localhost

10.10.10.10 ce12800-1    # Indicates the IP address and name of a host.
10.10.10.11 ce12800-2
```

After completing the configuration, you can run the **ping** command to check whether the configured host names take effect.

```
# ping ce12800-1
```

Step 1 (Optional) Create the inventory hosts file.

By default, the inventory hosts file is created during Ansible installation and is located in the **/etc/ansible/hosts** directory.

```
# ls /etc/ansible/
ansible.cfg  hosts
```

If the inventory hosts file does not exist, you can create it manually.

```
# touch /etc/ansible/hosts
```

Step 2 Define hosts and host groups.

```
# vi /etc/ansible/hosts
[all:vars]
ansible_connection=local
ansible_ssh_user=root0001
ansible_ssh_pass=Root_$0001
ansible_ssh_port=22

[spine]
ce12800-1        # Add a host to a host group using the host name.
ce12800-2

[leaf]
10.10.10.12      # Add a host to a host group using the IP address.
10.10.10.13
```

The value in brackets ([ ]) indicates a host group name. Host group names are used to classify systems, facilitating management of different systems.

In the host group **all**, **vars** indicates that the group defines variables. The variables are:

- **ansible_connection**: specifies the host connection type.
- **ansible_ssh_user**: specifies the user name of the connected host. The value must be the same as that configured on the host.
- **ansible_ssh_pass**: specifies the password corresponding to a host user name. The value must be the same as that configured on the host.
- **ansible_ssh_ssh**: specifies the SSH port number. The default value is 22. The value must be the same as that configured on the host.

  The **vi /etc/ansible/hosts** command output shows that two hosts are defined in the host group **spine**. The two hosts are added to the host group using their host names ce12800-1 and ce12800-2 respectively. After Ansible is executed, it will automatically convert the host names into IP addresses based on the configuration in the **/etc/hosts** file.

  Two hosts are defined in the host group **leaf** using their IP addresses 10.10.10.12 and 10.10.10.13.

----End

3.2.2 Creating a Playbook

Create a playbook named **ce-vlan.yml** and save it in your working directory. The following procedure uses **/usr/huawei/ansible** as the working directory.

Step 1 Create a playbook.

```
# touch /usr/huawei/ansible/ce-vlan.yml
```

Step 2 Edit the playbook.

You can edit the **ce-vlan.yml** file using an editor, such as vi, vim, or gedit, or copy the content edited in another file to the **ce-vlan.yml** file. The following is the content of the **ce-vlan.yml** file.

```
---

- name: "sample playbook"
  gather_facts: no
  hosts: spine

  tasks:
  - name: "Create vlan 100"
    ce_vlan: vlan_id=100 state=present host={{ inventory_hostname }}
username={{ ansible_ssh_user }} password={{ ansible_ssh_pass }}
port={{ ansible_ssh_port }}

  - name: "Add interface to vlan 100"
    ce_switchport: interface=10ge2/0/10 mode=access access_vlan=100 state=present
host={{ inventory_hostname }} username={{ ansible_ssh_user }}
password={{ ansible_ssh_pass }} port={{ ansible_ssh_port }}
```

All YAML files have --- in the first row, indicating the beginning of a file.

Each Ansible YAML file starts with a list, in which each item is a key-value pair. These key-value pairs form a dictionary. All items of the list start with a hyphen and a space, and have the same indentation.

The preceding playbook defines two tasks. The first task is to create VLAN 100, and the second task is to add 10GE interface 2/0/10 to VLAN 100 in access mode.

Parameters in the playbook are described as follows:

- **gather_facts**: indicates whether to collect switch status. In the example playbook, its value is **no**, which means that the switch status will not be collected.
- **name**: indicates the description of the playbook.
- **tasks**: indicates that the following content is about Ansible tasks.
- **name** under **tasks**: indicates the name or description of a specific task.
- **ce_vlan**: indicates the VLAN configuration module of the CE switch.
- **ce_switchport**: indicates the switchport configuration module of the CE switch.

Table 3-2 Description of ce_vlan module parameters

Parameter	Description
vlan_id	Indicates the VLAN ID.
state	Indicates whether a configuration should exist on the switch. • **present**: The configuration should exist on the switch. • **absent**: The configuration should not exist on the switch.
host	Indicates the IP address of the switch. The example playbook uses the Ansible built-in variable **{{ inventory_hostname }}** to obtain the IP address of the switch.
username	Indicates the SSH user name used to log in to the switch. The example playbook uses the Ansible built-in variable **{{ ansible_ssh_user }}** to obtain the SSH user name. The user name must be the same as that configured on the switch.
password	Indicates the SSH user password used to log in to the switch. The example playbook uses the Ansible built-in variable **{{ ansible_ssh_pass }}** to obtain the SSH user password. The password must be the same as that configured on the switch.
port	Indicates the SSH port number used to log in to the switch. The example playbook uses the Ansible built-in variable **{{ ansible_ssh_port }}** to obtain the SSH port number. The port number must be the same as that configured on the switch.

Table 3-3 Description of ce_switchport module parameters

Parameter	Description
interface	Indicates the full name of an interface.
mode	Indicates the interface type.
access_vlan	Indicates the VLAN ID for the access interface.
state	Indicates whether a configuration should exist on the switch. • present: The configuration should exist on • the switch. • absent: The configuration should not exist • on the switch.
host	Indicates the IP address of the switch. The example playbook uses the Ansible built-in variable

Parameter	Description
	{{ inventory_hostname }} to obtain the IP address of the switch.
username	Indicates the SSH user name used to log in to the switch. The example playbook uses the Ansible built-in variable **{{ ansible_ssh_user }}** to obtain the SSH user name. The user name must be the same as that configured on the switch.
password	Indicates the SSH user password used to log in to the switch. The example playbook uses the Ansible built-in variable **{{ ansible_ssh_pass }}** to obtain the SSH user password. The password must be the same as that configured on the switch.
port	Indicates the SSH port number used to log in to the switch. The example playbook uses the Ansible built-in variable **{{ ansible_ssh_port }}** to obtain the SSH port number. The port number must be the same as that configured on the switch.

NOTE

For the switch functions and features supported by the CloudEngine Ansible library and description of related parameters, visit *https://github.com/HuaweiSwitch/CloudEngine-Ansible/tree/master/docs*.

3.2.3 Running a Playbook

Before running a playbook, ensure the following:

1. (Optional) The host names have been configured correctly in **/etc/hosts**.
2. The inventory hosts file has been configured correctly.
3. The playbook has been created.

Perform the following steps to run the playbook:

Step 1 Run a playbook.

```
# cd /usr/huawei/ansible
# ansible-playbook ce-vlan.yml

PLAY [sample playbook] *********************************************************

TASK [Create vlan 100] *********************************************************
changed: [ce12800-1]
changed: [ce12800-2]

TASK [Add interface to vlan 100] ***********************************************
changed: [ce12800-1]
changed: [ce12800-2]

PLAY RECAP *********************************************************************
ce12800-1                  : ok=2    changed=2    unreachable=0    failed=0
ce12800-2                  : ok=2    changed=2    unreachable=0    failed=0
```

The fields in the command output are described as follows:

- **PLAY**: indicates the running playbook. The name of the playbook defined in the **ce-vlan.yml** file is included in the brackets.
- **TASK**: indicates the ongoing task. The task name defined in the playbook is included in the brackets. The result of each task is displayed in real time. In this example, **changed: [ce12800-1]** under a task indicates that the task has been executed correctly on the specified host and configuration of the host has changed.
- **PLAY RECAP**: indicates the playbook execution result, including the number of successful tasks, configuration changes, host unreachable events, and failed tasks on each host.

Step 2 Verify configurations on the CE switches.

After running the playbook, log in to the CE switches to check whether the configurations of the switches are consistent with the playbook execution result.

```
<HUAWEI>display vlan
The total number of vlans is : 2
--------------------------------------------------------------------------------
U: Up;        D: Down;        TG: Tagged;        UT: Untagged;
MP: Vlan-mapping;             ST: Vlan-stacking;
#: ProtocolTransparent-vlan;    *: Management-vlan;
MAC-LRN: MAC-address learning;  STAT: Statistic;
BC: Broadcast; MC: Multicast;   UC: Unknown-unicast;
FWD: Forward;  DSD: Discard;
--------------------------------------------------------------------------------

VID         Ports
-------------------------------------------------------------------------------- 1
            UT:Eth-Trunk100(D) 10GE2/0/0(D)    10GE2/0/1(D)    10GE2/0/2(D)
              10GE2/0/3(D)    10GE2/0/4(D)    10GE2/0/5(D)    10GE2/0/6(D)
              10GE2/0/7(D)    10GE2/0/8(D)    10GE2/0/9(D)    10GE2/0/11(D)
              10GE2/0/12(D)   10GE2/0/13(D)   10GE2/0/14(D)   10GE2/0/15(D)
              10GE2/0/16(D)   10GE2/0/18(D)   10GE2/0/19(D)   10GE2/0/20(D)
              10GE2/0/21(D)   10GE2/0/22(D)   10GE2/0/23(D)   10GE2/0/24(D)
              10GE2/0/26(D)   10GE2/0/27(D)   10GE2/0/28(D)   10GE2/0/29(D)
              10GE2/0/30(D)   10GE2/0/31(D)   10GE2/0/32(D)   10GE2/0/33(D)
              10GE2/0/34(D)   10GE2/0/35(D)   10GE2/0/36(D)   10GE2/0/37(D)
              10GE2/0/38(D)   10GE2/0/39(D)   10GE2/0/40(D)   10GE2/0/41(D)
              10GE2/0/42(D)   10GE2/0/43(D)   10GE2/0/44(D)   10GE2/0/45(D)
              10GE2/0/46(D)   10GE2/0/47(D)
 100         UT:10GE2/0/10(D)  //The interface has been added to VLAN 100.
```

----End

The playbook execution result is displayed on the server. To save the execution result in a file, perform the following steps:

Step 1 Create the **templates** directory under the directory where the playbook is saved, and add a **vlan.j2** file in the directory. The **vlan.j2** file will save the VLAN information of the hosts after the playbook is executed.

```
# cd /usr/huawei/ansible
# mkdir templates        # Create the templates directory.
# cd templates
```

```
# vi vlan.j2
{{ data.end_state_vlans_list | to_nice_json}}   # end_state_vlans_list is the function
indicating the Playbook execution result.
#
```

NOTE

For more information about playbook templates, visit http://docs.ansible.com/ansible/playbooks_templating.html.

Step 2 Create the **configs** directory under the directory where the playbook is saved. The file recording the playbook execution result will be saved in this directory.

```
# cd /usr/huawei/ansible
# mkdir configs          # Create the configs directory.
```

Step 3 Add a task to write the playbook execution result function in a file.

Use the vi editor to edit the **ce-vlan.yml** file. The file content is as follows after the task is added:

```
---

- name: "sample playbook"
  gather_facts: no
  hosts: spine

  tasks:
  - name: "Create vlan 200"
    ce_vlan: vlan_id=200 state=present host={{ inventory_hostname }}
username={{ ansible_ssh_user }} password={{ ansible_ssh_pass }}
port={{ ansible_ssh_port }}

  - name: "collection data to file"
    template: src=vlan.j2 dest=configs/vlan.json
```

Step 4 Execute the **ce-vlan.yml** file.

```
# cd /usr/huawei/ansible
# ansible-playbook ce-vlan.yml

PLAY [sample playbook] ***********************************************************

TASK [create vlan] ***************************************************************
changed: [ce12800-1]
changed: [ce12800-2]

TASK [collection data to file] ***************************************************
changed: [ce12800-1]
changed: [ce12800-2]

PLAY RECAP ***********************************************************************
ce12800-1                  : ok=2    changed=2    unreachable=0    failed=0
ce12800-2                  : ok=2    changed=2    unreachable=0    failed=0
```

Step 5 In the **configs** directory, check the file that stores the VLAN information after the playbook is executed.

```
# cd /usr/huawei/ansible/configs
# cat vlan.json
```

```
[
   "1",
   "2",
   "100",
   "110",
   "200"
]
```

----End

4 Application Constraints and Model Requirements

The constraints of using Ansible are as follows:

- Ansible can only run on Linux.
- Software version dependency:

Software	Version
Python	2.7
Ansible	2.4.2.0 or later

  The host where Ansible runs needs to be installed with an operating system version that Ansible supports, including Debian, Ubuntu, and Red Hat.

- Recommended CE switch models:

  All CE switch models are supported.

5 Summary

Ansible does not require an agent on the device to which Ansible connects, so Ansible is decoupled from the device software. In addition, Ansible has powerful community and software expansion capabilities, which allow customers to easily integrate Ansible in their own environments.

The integration of Huawei CE switches and Ansible facilitates switch management using existing IT O&M capabilities. After the integration, configurations can be modified in a centralized manner in batches, greatly improving the automated deployment and O&M capabilities. In addition, typical configuration templates are provided to reduce the workload of initial service deployment.