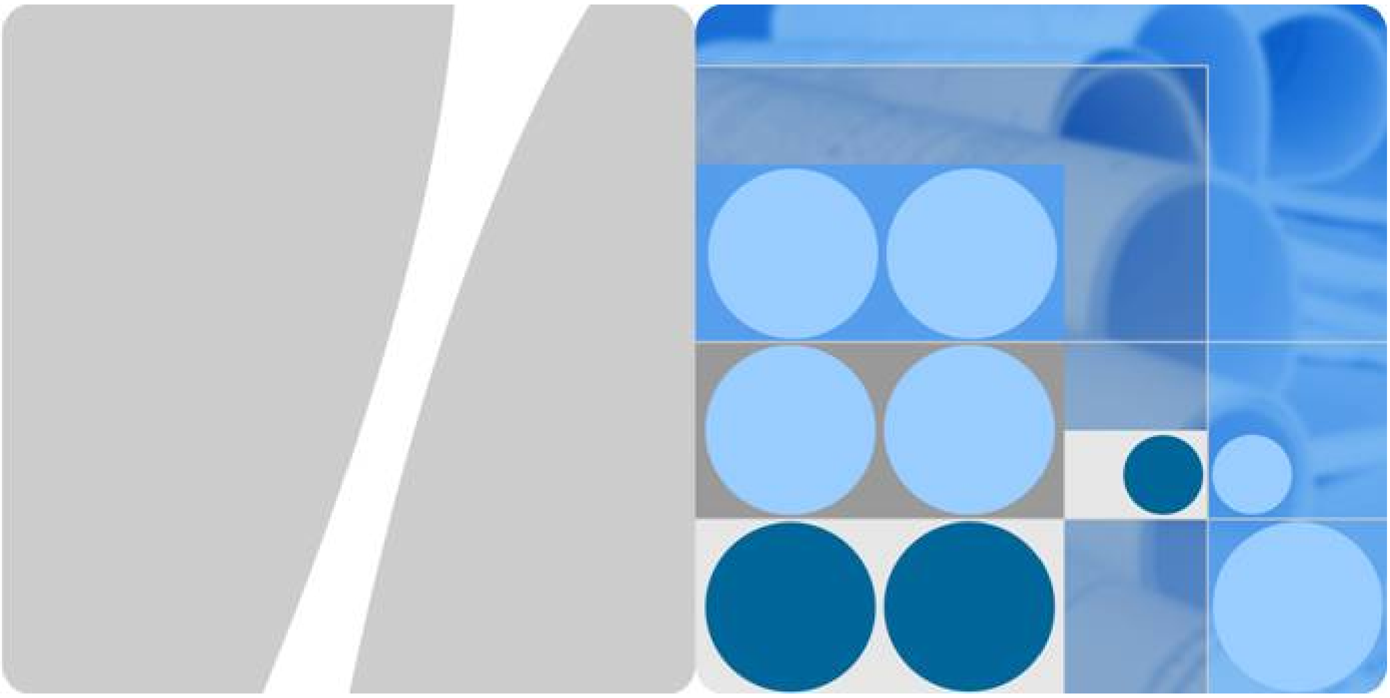


Automated Deployment of CloudEngine Series Switches Using Ansible

Issue 01

Date 2017-02-25

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions

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

About This Document

Overview

This document describes how to use Ansible to automatically deploy CloudEngine (CE for short) series switches, and provides examples for managing and configuring CE series switches using Ansible.

Intended Audience

This document is intended for network deployment personnel. Network deployment personnel must:

- Be familiar with the existing networking and configurations of related network elements (NEs).
- Have device maintenance experience and master device operation and maintenance (O&M) methods.
- Have experience in operating Linux servers.

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
⚠ DANGER	Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
⚠ WARNING	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
⚠ CAUTION	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

Symbol	Description
⚠ NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.
📖 NOTE	Calls attention to important information, best practices and tips. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

Change History

Issue	Release Date	Description
01	2017-02-25	Initial official release

Contents

1 Introduction

1.1 Overview

Ansible is an open-source IT automation tool and O&M tool. With CloudEngine modules, Ansible can implement automatic deployment and configuration of Huawei CE series data center switches. This document describes how to use Ansible to deploy and configure CE switches.

CloudEngine modules of Ansible are open-source application programs supported by the Ansible community. You can download CloudEngine modules from **https://github.com/HuaweiSwitch/CloudEngine-Ansible**. If you have problems during usage or want to obtain help from developers of CloudEngine modules, register for a GitHub account and submit issues.

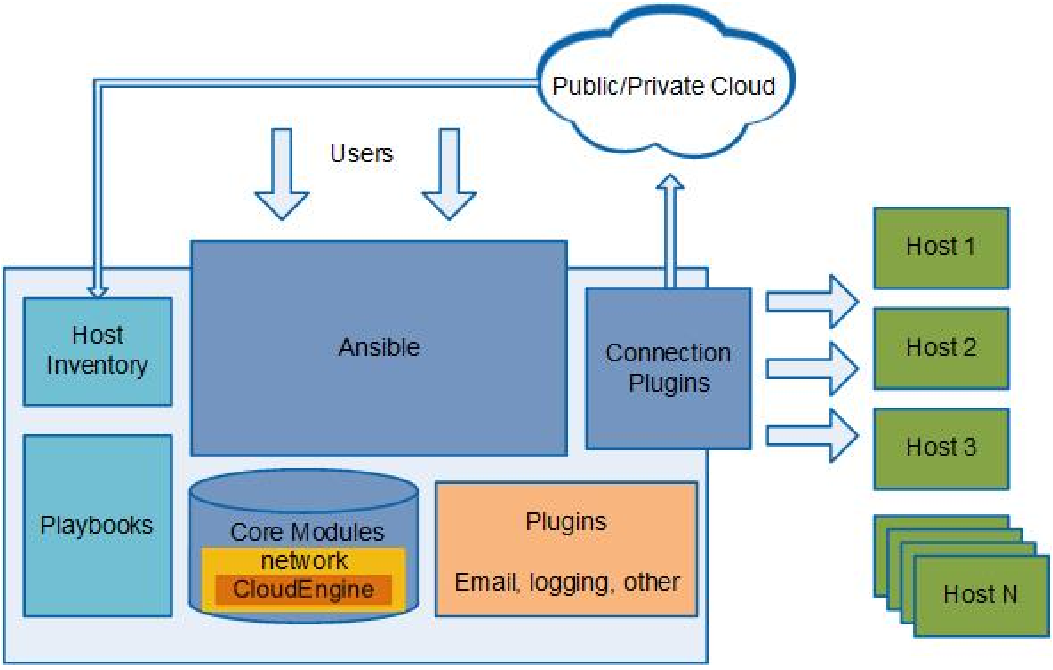
1.2 Ansible Overview

Ansible is an open-source automation O&M tool created by Michael DeHaan, the developer of automation tools Cobbler and Func. AnsibleWorks was established in 2012. Ansible is developed based on Python and integrates advantages of many O&M tools (Puppet, CFEngine, Chef, Func, and Fabric). Ansible implements functions such as batch system configuration, batch program deployment, and batch command execution. Ansible can be installed in platforms including Linux, BSD, and Mac OS X.

Ansible works based on modules and does not provide the batch deployment capability. Modules run on Ansible provide the batch deployment capability. Ansible only provides a framework including the following components:

- Connection Plugins: are connection plug-ins that communicate with monitored nodes.
- Host Inventory: specifies managed hosts.
- Modules: include core modules and user-defined modules.
- Plugins: implement functions such as logging and email.
- Playbooks: are Ansible's configuration, deployment, and orchestration languages.

Figure 1-1 Basic framework of Ansible



Ansible has the following characteristics:

- Low dependency: Only Python 2.6 or a later version need to be installed.
- Lightweight: No agent is required.
- Easy to read: Ansible's inventory hosts file uses the INI format and the playbook format is YAML. The two formats are simple and easy to understand.
- Support for multiple languages: You can choose a familiar language to write a module.

NOTE

To obtain more information about Ansible, visit **http://docs.ansible.com/**.

Basic Concepts

1. Playbooks

Ansible tasks are defined in playbooks. Multiple tasks can be defined in a playbook. Playbooks are automatically executed by Ansible. The Ansible server can run multiple tasks to manage multiple remote hosts.

Playbooks are Ansible's configuration, deployment, and orchestration languages. They can be described as a scheme containing commands to be executed by remote hosts or a collection of commands run by a group of IT programs. At a basic level, playbooks can be used to configure and deploy remote hosts. At an advanced level, playbooks can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way.

2. Inventory

The inventory defines hosts managed by Ansible. By default, managed nodes or hosts are defined in the inventory hosts file of Ansible. You can also define the inventory hosts file and manually specify the file location. Ansible can operate multiple hosts in the same group. The relationship between groups and hosts is defined by configuring the inventory hosts file. The default path of the inventory hosts file is **/etc/ansible/hosts**.

3. Modules

Modules are also called task plug-ins or library plug-ins, and can be executed in Ansible. Tasks in a playbook are executed by invoking modules. Modules can also be directly run using the **ansible** command.

Currently, Ansible supports Huawei CE switches and can implement automatic O&M and management of Huawei CE switches through CloudEngine modules. For the list of features supported by the CloudEngine Ansible library, visit **https://github.com/HuaweiSwitch/CloudEngine-Ansible/tree/master/library/docs**.

NOTE

For more information about Huawei CloudEngine Ansible library, visit **https://github.com/HuaweiSwitch/CloudEngine-Ansible**.

1.3 Ansible Tower Overview

Ansible Tower is a commercial product that manages the inventory through a graphical interface, controls users' access rights, and logs all jobs. Ansible Tower helps administrators manage complex network deployment. To obtain more information about Ansible Tower, visit **https://www.ansible.com/tower**.

Ansible Tower allows you to invoke CloudEngine modules and execute playbooks on a graphical interface to deploy and configure Huawei CE series switches.

2 Configuring and Managing CE Series Switches Using the Ansible Framework

2.1 Environment Preparation

Table 1 describes environment version information required for installing Ansible.

Table 2-1 Environment version information

Environment Category	Version
Operating System Type	Red Hat, Ubuntu, CentOS, OS X, and BSD in various versions
Python Version	Python 2.6 and later versions
Ansible framework version	Ansible v2.2.1.0-1

2.1.1 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 Create an SSH user on the CE switch.

# Create the SSH user **root001**.

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

Step 3 Enable the STelnet server function on the CE switch.

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

Step 4 Set the service type of the SSH user **root001** to all.

```
[HUAWEI] ssh user root001 service-type all
[*HUAWEI] commit
```

----End

2.1.2 Installing ncclient

After installing Ansible, you need to install ncclient. ncclient is a Python library that allows Ansible to remotely manage and configure devices using the Network Configuration Protocol (NETCONF).

The NETCONF application programming interface (API) is defined based on NETCONF. NETCONF uses a communication mechanism based on a remote procedure call (RPC). NETCONF uses the <rpc> and <rpc-reply> elements to provide NETCONF requests and responses independent of transport protocols, implementing device configuration and management.

Ansible CloudEngine modules use NETCONF to establish connections with CE switches and deliver configurations. When the Ansible server and a CE switch are establishing a NETCONF session, they must exchange their supported capability sets. They can perform the configuration delivery operation only after receiving the capability sets from each other.

The procedure of installing ncclient is as follows:

Step 1 Download ncclient from **https://github.com/ncclient/ncclient/releases**.

NOTE

CloudEngine modules support ncclient 0.5.3 and later versions.

Step 2 Upload the ncclient installation package **ncclient-0.5.3.zip** to the Ansible server.

Step 3 Decompress the ncclient installation package **ncclient-0.5.3.zip** and install ncclient.

1. Decompress the ncclient installation package.
```
# unzip ncclient-0.5.3.zip
```
2. Access the ncclient directory.
```
# cd ncclient-0.5.3/
```
3. Install ncclient.
```
# sudo python setup.py install
```

----End

2.1.3 Installing Ansible

After installing the Ansible framework successfully, you need to add the modules in the CloudEngine installation package to the corresponding modules of the Ansible framework, so that Ansible can implement automated management and maintenance of CE switches. The procedure of adding CloudEngine modules to the Ansible framework is as follows:

Step 1 Obtain the HuaweiSwitch CloudEngine installation package from **https://github.com/HuaweiSwitch/CloudEngine-Ansible/releases**. Take v0.1.0 as an example. Download the CloudEngine-Ansible-0.1.0.zip compressed package, upload it to the **/home** directory on the Ansible server, and decompress the package. You can view the **CloudEngine-Ansible-0.1.0** directory.

```
# ls
CloudEngine-Ansible-0.1.0.zip
# unzip CloudEngine-Ansible-0.1.0.zip
```

# After the installation package is decompressed, the **CloudEngine-Ansible-0.1.0** directory is generated.

```
# ls
CloudEngine-Ansible-0.1.0     # Directory generated after the installation
package is decompressed
```

Step 2 Query the directories where the Ansible framework and its modules are located.

# Query the directory where the Ansible framework is located through the file **network.py**. The directory where the **network.py** file is located is shown in the following command output.

```
# find / -name network.py   # Obtain the directory where the network.py file is
located on the Ansible server.
/usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/
module_utils/network.py
```

# Determine the directories where Ansible modules are located based on the directory where the **network.py** file is located.

- Directory of the Ansible public module
  ```
   /usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/
  module_utils    # Directory where the Ansible public module is located
  ```
- Directory of the Ansible service module
  ```
   /usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/
  modules    # Directory where the Ansible service module is located
  ```
- Directory of the Ansible config plug-in
  ```
  /usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/
  plugins/action    # Directory where the Ansible config plug-in is located
  ```
- Directory of the Ansible docs_fragments
  ```
  /usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/
  utils/module_docs_fragments    # Directory where the Ansible docs_fragments
  is located
  ```

Step 3 Create a directory in the Ansible framework to store CloudEngine service modules.

Access the directory where the Ansible service module is located and create the directory **core/network/cloudengine**.

```
# cd /usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/
modules     # Access the directory where the Ansible service module is located.
# mkdir core            # Create the core directory.
# cd core/              # Access the core directory.
# mkdir network         # Create the network directory.
# cd network/           # Access the network directory.
# mkdir cloudengine     # Create the cloudengine directory.
#
```

Step 4 Copy contents of CloudEngine modules to specified directories in the Ansible framework.

Access the directory where the CloudEngine installation package is located, and copy contents of each module in the CloudEngine installation package to specified directories in the Ansible framework respectively.

```
# cd /home/CloudEngine-Ansible-0.1.0   # Access the directory where the
CloudEngine installation package is located.
#
```

- Copy contents of the CloudEngine basic module to the directory of the Ansible public module.
  ```
  # cp -f ./module_utils/cloudengine.py /usr/local/lib/python2.7/dist-packages/
  ansible-2.2.1.0-py2.7.egg/ansible/module_utils
  #
  ```
- Copy contents of CloudEngine service modules to the directory of the Ansible CloudEngine service module.
  ```
  # cp -rf ./library/* /usr/local/lib/python2.7/dist-packages/ansible-2.2.1.0-
  py2.7.egg/ansible/modules/core/network/cloudengine
  #
  ```
- Copy contents of the CloudEngine config plug-in to the directory of the Ansible config plug-in.

```
# cp -f ./plugins/action/ce_config.py /usr/local/lib/python2.7/dist-packages/
ansible-2.2.1.0-py2.7.egg/ansible/plugins/action
#
```

- Copy contents of the CloudEngine docs_fragments file to the directory of the Ansible docs_fragments.

```
# cp -f ./utils/module_docs_fragments/cloudengine.py /usr/local/lib/python2.7/
dist-packages/ansible-2.2.1.0-py2.7.egg/ansible/utils/module_docs_fragments
#
```

Step 5 Check the Ansible version.

```
# ansible --version
ansible 2.2.1.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = Default w/o overrides
```

Verify basic functions.

```
# ansible -m ce_command -a 'host=10.10.10.10 port=12347 username=*** password=***
commands="display version"' localhost --connection local
```

NOTE

Set **host**, **port**, **username**, and **password** to switch information in the actual environment. If version information is correctly returned, the modules have been successfully installed.

----End

2.2 Configuration Procedure

2.2.1 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 saved 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.

Step 1 (Optional) Update the **/etc/hosts** file.

The **/etc/hosts** file contains IP addresses and corresponding host names. Updating the file is not mandatory, but facilitates 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    # 10.10.10.10 and ce12800-1 indicate the IP address of a
host and the corresponding host name respectively.
10.10.10.11 ce12800-2
```

After completing the configuration, you can run the **ping** command to check whether a configured host name takes effect.

```
# ping ce12800-1
```

Step 2 (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  roles
```

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

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

Step 3 Define hosts and host groups.

```
# vi /etc/ansible/hosts
[all:vars]
ansible_connection=local
ansible_ssh_user=root001
ansible_ssh_pass=root001
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 including:

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 preceding 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. During execution, Ansible 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

2.2.2 Creating a Playbook

Create a **ce-vlan.yml** playbook 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 file has **---** in the first row, indicating the beginning of the 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 device status. In the preceding playbook, its value is set to **no**, which means that the device 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 port configuration module of the CE switch.

Table 2-2 ce_vlan module parameter description

Parameter	Description
vlan_id	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.

Parameter	Description
host	Indicates the IP address of the switch. The preceding 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 preceding 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 preceding 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 preceding 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 2-3 ce_switchport module parameter description

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 preceding playbook uses the Ansible built-in variable {{ inventory_hostname }} to obtain the IP address of the switch.

Parameter	Description
username	Indicates the SSH user name used to log in to the switch. The preceding 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 preceding 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 preceding 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, see **https://github.com/HuaweiSwitch/CloudEngine-Ansible/tree/master/docs**.

In actual deployment, you are advised to set module parameter values to variables to adapt to different configurations on different devices. For details, see **http://docs.ansible.com/ansible/playbooks_variables.html**.

----End

NOTICE

Tab characters are not allowed in YAML files. All items of a list must have specific indentations for their levels in the hierarchy.

NOTE

For more information about playbooks, see **http://docs.ansible.com/ansible/playbooks.html**.

For more information about YAML files, see **http://docs.ansible.com/ansible/YAMLSyntax.html** and **http://yaml.org/**.

2.2.3 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 the 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
```

Fields in the preceding output information 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 numbers of successful tasks, configuration changes, host unreachable events, and failed tasks on each host.

Step 2 Check 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, see **http://docs.ansible.com/ansible/playbooks_templating.html**.

Create the **configs** directory under the directory where the playbook is saved. The file used to save the playbook execution result will be saved in this directory.

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

Step 2 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 3 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 4 In the **configs** directory, check the file that saves the VLAN information after the playbook is executed.

```
# cd /usr/huawei/ansible/configs
# cat vlan.json
[
    "1",
    "2",
    "100",
    "110",
    "200"
]
```

----End

3 Configuring and Managing CE Series Switches Using Ansible Tower

3.1 Environment Preparation

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 Create an SSH user on the CE switch.

# Create the SSH user **root001**.

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

Step 3 Enable the STelnet server function on the CE switch.

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

Step 4 Set the service type of the SSH user **root001** to all.

```
[HUAWEI] ssh user root001 service-type all
[*HUAWEI] commit
```

----End

3.1.2 Installing ncclient

After installing Ansible, you need to install ncclient. ncclient is a Python library that allows Ansible to remotely manage and configure devices using the Network Configuration Protocol (NETCONF).

The NETCONF application programming interface (API) is defined based on NETCONF. NETCONF uses a communication mechanism based on a remote procedure call (RPC). NETCONF uses the <rpc> and <rpc-reply> elements to provide NETCONF requests and responses independent of transport protocols, implementing device configuration and management.

Ansible CloudEngine modules use NETCONF to establish connections with CE switches and deliver configurations. When the Ansible server and a CE switch are establishing a NETCONF session, they must exchange their supported capability sets. They can perform the configuration delivery operation only after receiving the capability sets from each other.

The procedure of installing ncclient is as follows:

Step 1 Download ncclient from **https://github.com/ncclient/ncclient/releases**.

NOTE

CloudEngine modules support ncclient 0.5.3 and later versions.

Step 2 Upload the ncclient installation package **ncclient-0.5.3.zip** to the Ansible server.

Step 3 Decompress the ncclient installation package **ncclient-0.5.3.zip** and install ncclient.

1. Decompress the ncclient installation package.
```
# unzip ncclient-0.5.3.zip
```
2. Access the ncclient directory.
```
# cd ncclient-0.5.3/
```
3. Install ncclient.
```
# sudo python setup.py install
```

----End

3.1.3 Installing Ansible Tower

Ansible Tower can be installed using either a setup or setup-bundle package. The two installation packages differ in the following way:

- The setup package requires the server to connect to the Internet, whereas the setup-bundle package can be installed offline.
- The setup package can be installed on any Linux operating system, such as Ubuntu, Red Hat, SUSE, and CentOS. The setup-bundle package can only be installed on Red Hat Enterprise Linux (RHEL) or CentOS.

  The Ansible Tower setup installation package can be obtained at **https://releases.ansible.com/ansible-tower/setup-bundle/**.

  The Ansible Tower setup-bundle installation package can be obtained at **https://releases.ansible.com/ansible-tower/setup-bundle/**.

Install Ansible Tower according to the installation guide released on Ansible website **http://docs.ansible.com/ansible-tower/latest/html/installandreference/tower_install_wizard.html**.

After installing Ansible Tower successfully, you need to add the modules in the CloudEngine installation package to the corresponding Ansible Tower modules, so that Ansible Tower can implement automated management and maintenance of CE switches. To add modules in the CloudEngine installation package to Ansible Tower modules, follow steps 1 through 5 in section 2.1.3 "Installing Ansible."

3.2 Configuration Procedure

3.2.1 Adding a Playbook in a Specified Directory

Ansible Tower creates the **/var/lib/awx** directory automatically during installation. This directory is used to save Ansible Tower data. All Ansible Tower playbooks must be saved in folders under the **/var/lib/awx/projects** directory.

Perform the following steps to add a playbook to a specified directory:

Step 1 Access the **/var/lib/awx/projects** directory.

```
# cd /var/lib/awx/projects
```

Step 2 Create a folder.

```
# mkdir cloudengine
```

Step 3 Add a playbook in the **cloudengine** folder.

```
# cd cloudengine   # Access the folder.
# cat test_add_vlan.yml # Playbook added to the folder
---

- name: cloudengine vlan module test
  hosts: 10.134.48.55
  gather_facts: no
  connection: local

  tasks:

  - name: "Ensure a range of  vlans are not present on the switch"
    ce_vlan: vlan_range="20-21" state=present host={{inventory_hostname}}
port={{ansible_ssh_port}} username={{username}} password={{password}}
    register: data
#
```

----End

3.2.2 Deploying the Running Environment on Ansible Tower

Ansible Tower manages users, CE switches, and network configurations using projects. For more instructions on use of Ansible Tower, see the *Ansible Tower User Guide* at **http://docs.ansible.com/ansible-tower/latest/html/userguide/index.html#ug-start**.

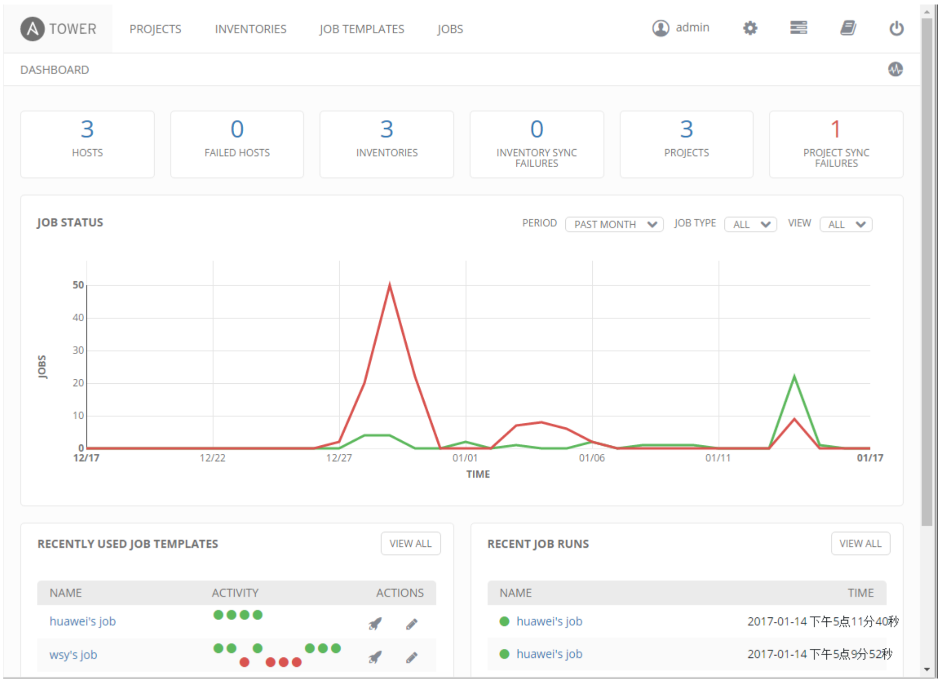
The roadmap of environment deployment on Ansible Tower is as follows:

1. Log in to Ansible Tower as the **admin** user.
2. Create an organization.
3. Create users and bind them to the organization.
4. Create a team and add users to the team.
5. Add credentials.
6. Create a project.
7. Add host information.

Perform the following steps to complete environment deployment on Ansible Tower:

Step 1 Log in as the **admin** user.

1. Enter the URL http://*Tower server IP address* in the address box on your browser to display the login page. Enter the **admin** user name and password to log in.
2. You will see the following page after login.



3	0	3	0	3	1
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

RECENTLY USED JOB TEMPLATES

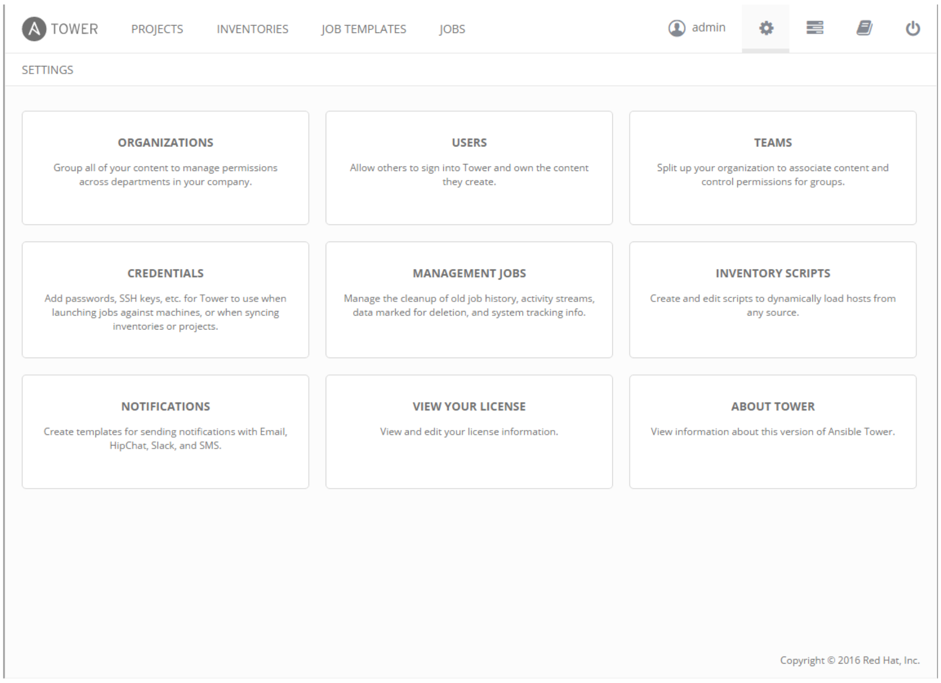
NAME	ACTIVITY	ACTIONS
huawei's job	●●●●	
wsy's job	●● ● ●●● ●●●●	

RECENT JOB RUNS

NAME	TIME
● huawei's job	2017-01-14 下午5点11分40秒
● huawei's job	2017-01-14 下午5点9分52秒

Step 2 Create an organization.

1. Click ⚙ to display the **SETTINGS** page.



ORGANIZATIONS
Group all of your content to manage permissions across departments in your company.

USERS
Allow others to sign into Tower and own the content they create.

TEAMS
Split up your organization to associate content and control permissions for groups.

CREDENTIALS
Add passwords, SSH keys, etc. for Tower to use when launching jobs against machines, or when syncing inventories or projects.

MANAGEMENT JOBS
Manage the cleanup of old job history, activity streams, data marked for deletion, and system tracking info.

INVENTORY SCRIPTS
Create and edit scripts to dynamically load hosts from any source.

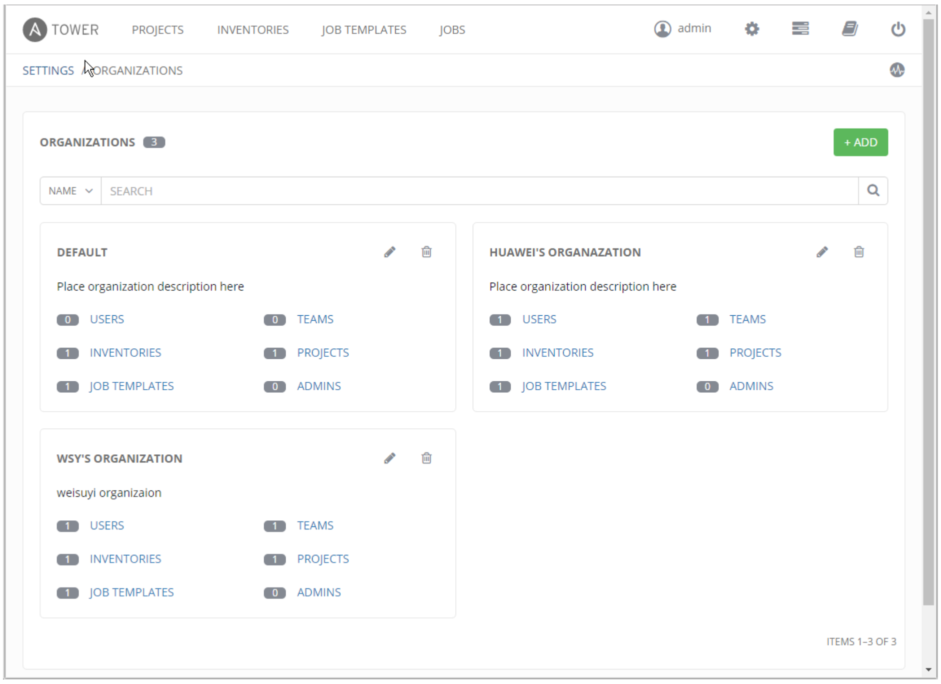
NOTIFICATIONS
Create templates for sending notifications with Email, HipChat, Slack, and SMS.

VIEW YOUR LICENSE
View and edit your license information.

ABOUT TOWER
View information about this version of Ansible Tower.

Copyright © 2016 Red Hat, Inc.

2. Click **ORGANIZATIONS** to display the **ORGANIZATIONS** page.



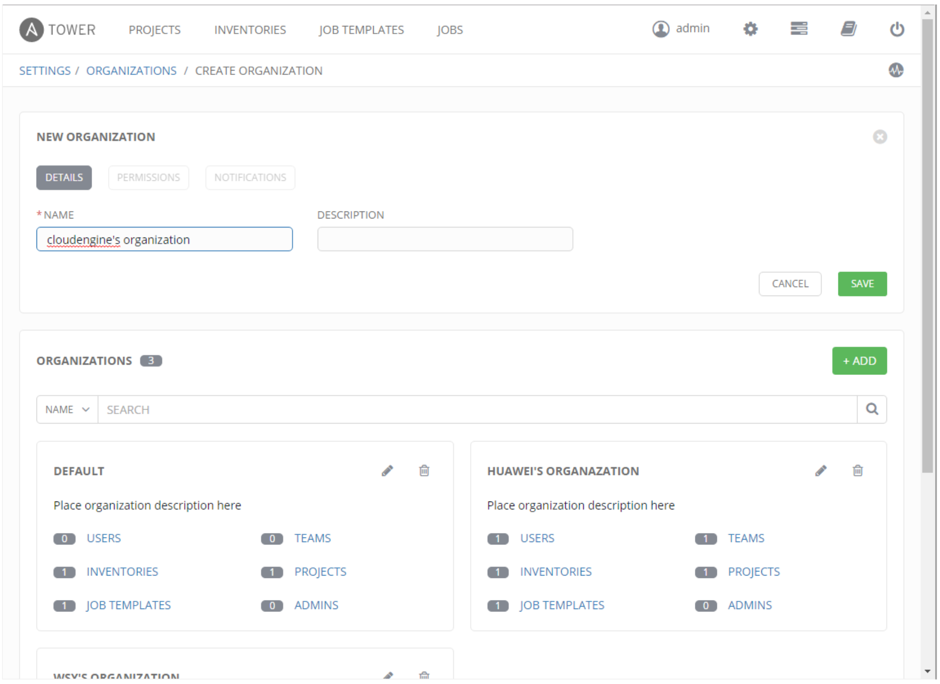
ORGANIZATIONS 3

DEFAULT
Place organization description here
0 USERS 0 TEAMS
1 INVENTORIES 1 PROJECTS
1 JOB TEMPLATES 0 ADMINS

HUAWEI'S ORGANAZATION
Place organization description here
1 USERS 1 TEAMS
1 INVENTORIES 1 PROJECTS
1 JOB TEMPLATES 0 ADMINS

WSY'S ORGANIZATION
weisuyi organizaion
1 USERS 1 TEAMS
1 INVENTORIES 1 PROJECTS
1 JOB TEMPLATES 0 ADMINS

3. Click + ADD to add an organization. In this example, the organization name is **cloudengine's organization**.



NEW ORGANIZATION

*NAME: cloudengine's organization

DESCRIPTION:

ORGANIZATIONS 3

DEFAULT
Place organization description here
0 USERS 0 TEAMS
1 INVENTORIES 1 PROJECTS
1 JOB TEMPLATES 0 ADMINS

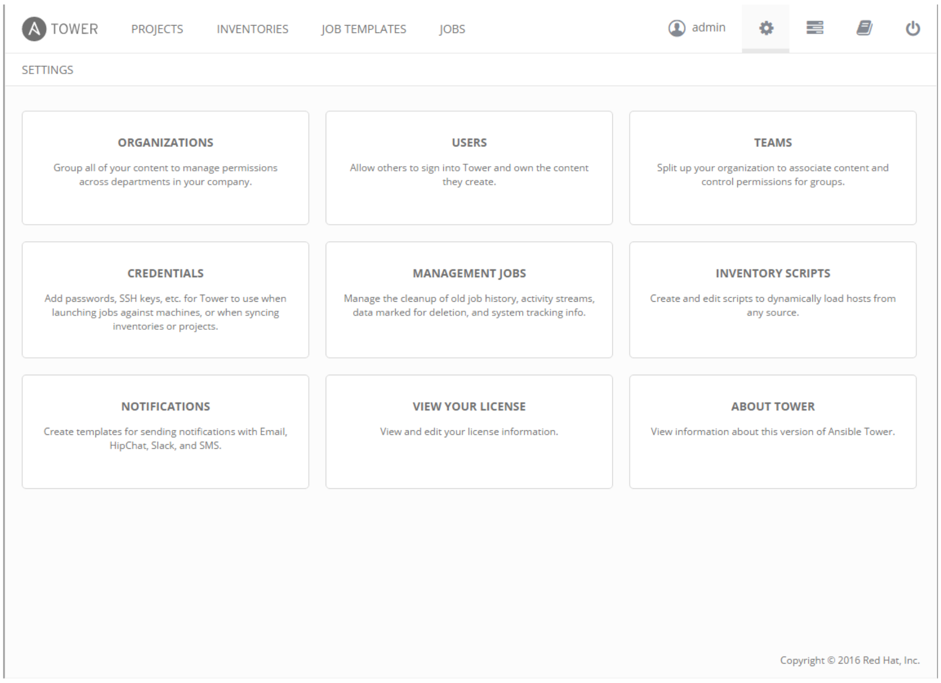
HUAWEI'S ORGANAZATION
Place organization description here
1 USERS 1 TEAMS
1 INVENTORIES 1 PROJECTS
1 JOB TEMPLATES 0 ADMINS

WSY'S ORGANIZATION

4. After entering the organization name, click SAVE.

Step 3 Create users and bind them to the organization.

1. Click ⚙ to display the **SETTINGS** page.



ORGANIZATIONS
Group all of your content to manage permissions across departments in your company.

USERS
Allow others to sign into Tower and own the content they create.

TEAMS
Split up your organization to associate content and control permissions for groups.

CREDENTIALS
Add passwords, SSH keys, etc. for Tower to use when launching jobs against machines, or when syncing inventories or projects.

MANAGEMENT JOBS
Manage the cleanup of old job history, activity streams, data marked for deletion, and system tracking info.

INVENTORY SCRIPTS
Create and edit scripts to dynamically load hosts from any source.

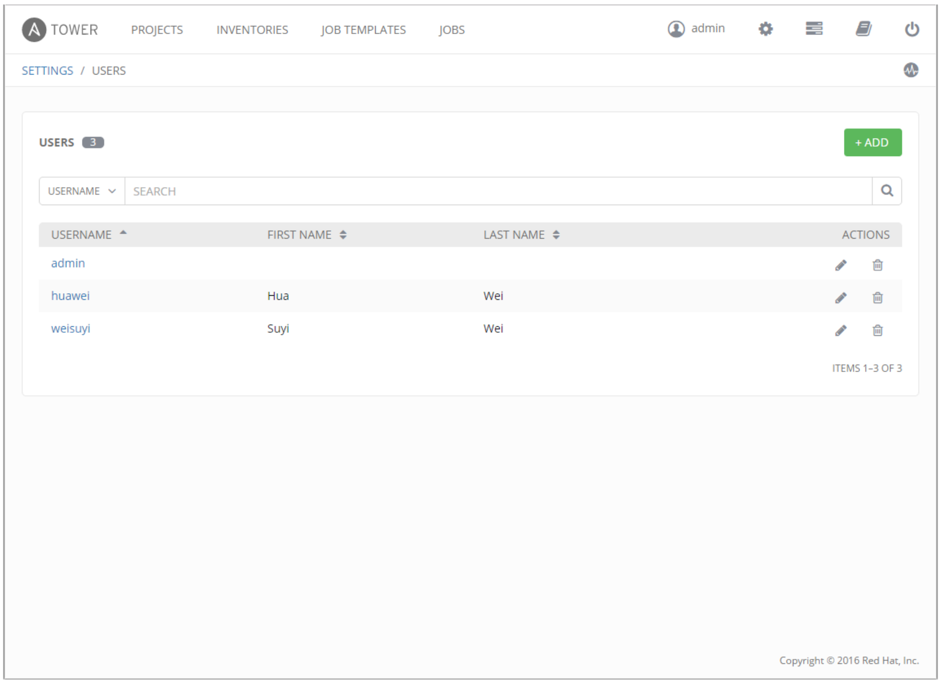
NOTIFICATIONS
Create templates for sending notifications with Email, HipChat, Slack, and SMS.

VIEW YOUR LICENSE
View and edit your license information.

ABOUT TOWER
View information about this version of Ansible Tower.

Copyright © 2016 Red Hat, Inc.

2. Click **USERS** to display the **USERS** page.



USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
huawei	Hua	Wei	
weisuyi	Suyi	Wei	

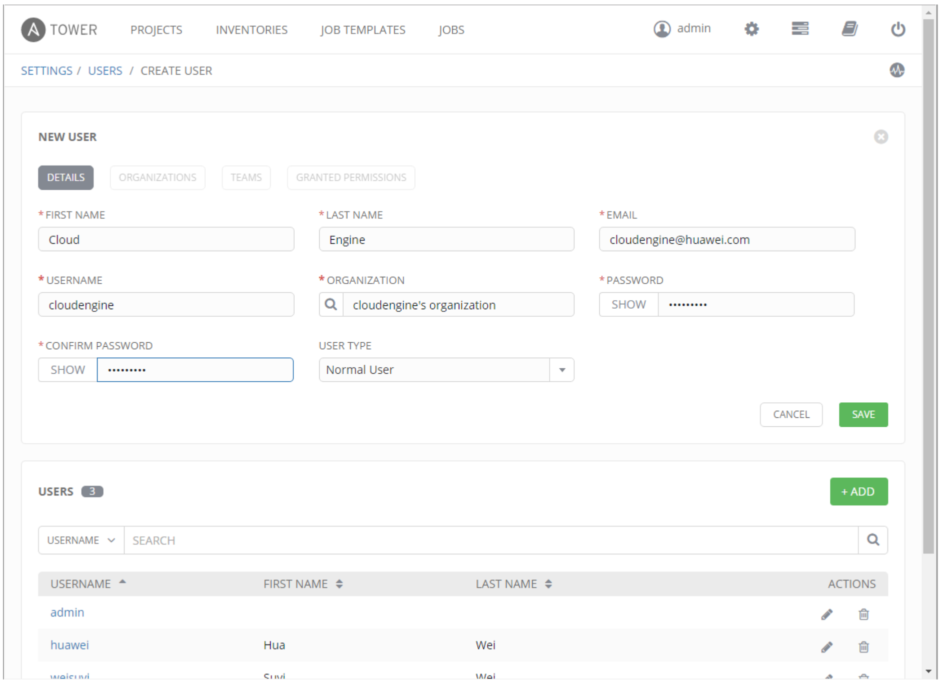
ITEMS 1-3 OF 3

Copyright © 2016 Red Hat, Inc.

3. Click + ADD to a user.

NOTE

Enter user information, including the user name, email address, organization name, and password. In this example, the user name, email address, and password are **cloudengine**, **cloudengine@huawei.com**, and **huawei@123**, respectively. The organization is **cloudengine's organization** you have created.



NEW USER

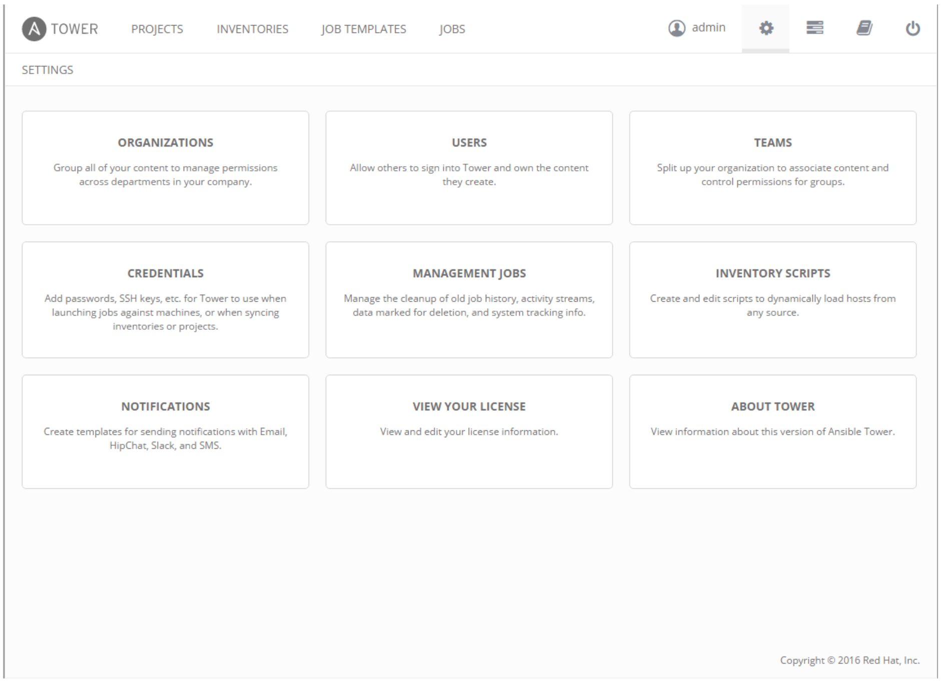
* FIRST NAME: Cloud	* LAST NAME: Engine	* EMAIL: cloudengine@huawei.com
* USERNAME: cloudengine	* ORGANIZATION: cloudengine's organization	* PASSWORD: •••••••••
* CONFIRM PASSWORD: •••••••••	USER TYPE: Normal User	

USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
huawei	Hua	Wei	
weisuvi	Suvi	Wei	

4. After setting user information, click SAVE.

Step 4 Create a team and add users to the team.

1. Click ⚙ to display the **SETTINGS** page.



ORGANIZATIONS
Group all of your content to manage permissions across departments in your company.

USERS
Allow others to sign into Tower and own the content they create.

TEAMS
Split up your organization to associate content and control permissions for groups.

CREDENTIALS
Add passwords, SSH keys, etc. for Tower to use when launching jobs against machines, or when syncing inventories or projects.

MANAGEMENT JOBS
Manage the cleanup of old job history, activity streams, data marked for deletion, and system tracking info.

INVENTORY SCRIPTS
Create and edit scripts to dynamically load hosts from any source.

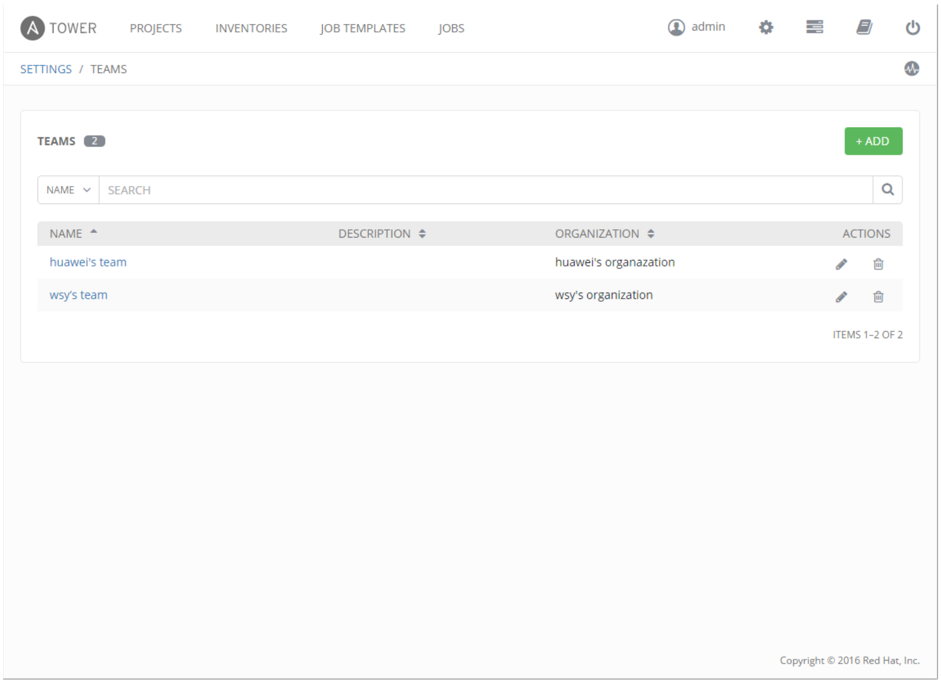
NOTIFICATIONS
Create templates for sending notifications with Email, HipChat, Slack, and SMS.

VIEW YOUR LICENSE
View and edit your license information.

ABOUT TOWER
View information about this version of Ansible Tower.

Copyright © 2016 Red Hat, Inc.

2. Click **TEAMS** to display the **TEAMs** page.



TOWER PROJECTS INVENTORIES JOB TEMPLATES JOBS admin

SETTINGS / TEAMS

TEAMS 2 + ADD

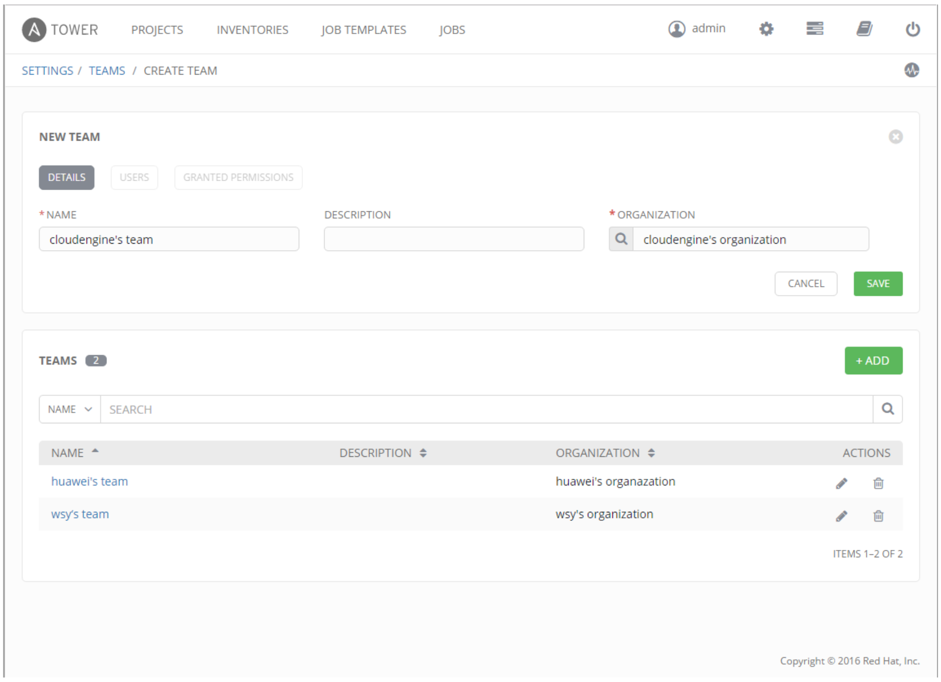
NAME SEARCH

NAME	DESCRIPTION	ORGANIZATION	ACTIONS
huawei's team		huawei's organazation	
wsy's team		wsy's organization	

ITEMS 1–2 OF 2

Copyright © 2016 Red Hat, Inc.

3. Click + ADD and enter the team name and organization name. In this example, the team name is **cloudengine's team**.



TOWER PROJECTS INVENTORIES JOB TEMPLATES JOBS admin

SETTINGS / TEAMS / CREATE TEAM

NEW TEAM

DETAILS USERS GRANTED PERMISSIONS

* NAME: cloudengine's team

DESCRIPTION:

* ORGANIZATION: cloudengine's organization

CANCEL SAVE

TEAMS 2 + ADD

NAME SEARCH

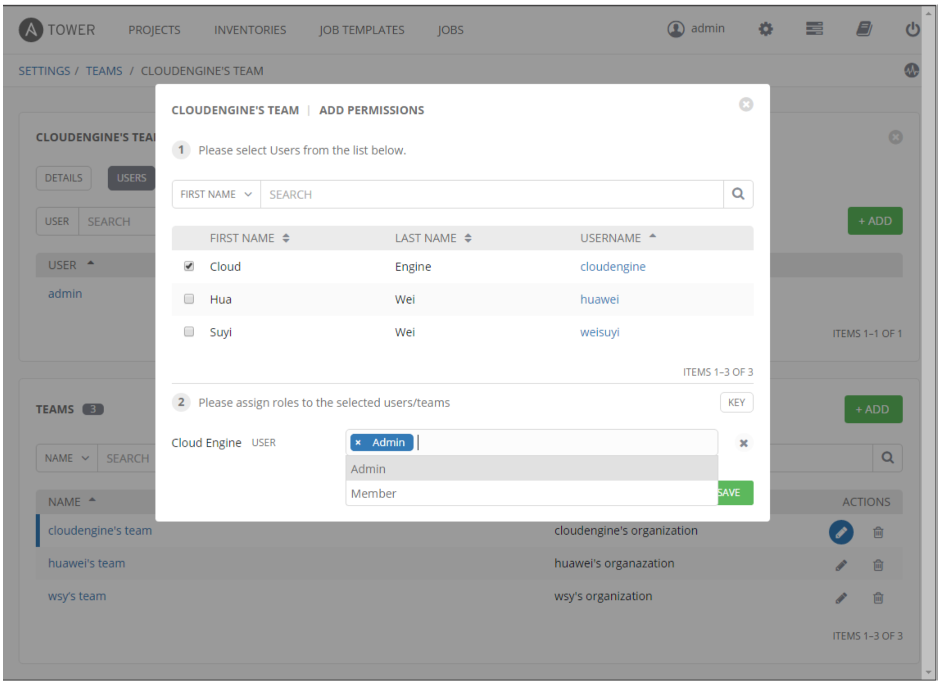
NAME	DESCRIPTION	ORGANIZATION	ACTIONS
huawei's team		huawei's organazation	
wsy's team		wsy's organization	

ITEMS 1–2 OF 2

Copyright © 2016 Red Hat, Inc.

4. After setting team information, click SAVE.

5. Click **USERS** under **CLOUDENGINE'S TEAM**, and then add + ADD to add a user to the team.



CLOUDENGINE'S TEAM | ADD PERMISSIONS

1 Please select Users from the list below.

FIRST NAME | SEARCH

	FIRST NAME	LAST NAME	USERNAME
☑	Cloud	Engine	cloudengine
□	Hua	Wei	huawei
□	Suyi	Wei	weisuyi

ITEMS 1–3 OF 3

2 Please assign roles to the selected users/teams

Cloud Engine USER: × Admin

Admin
Member

USER
admin

ITEMS 1–1 OF 1

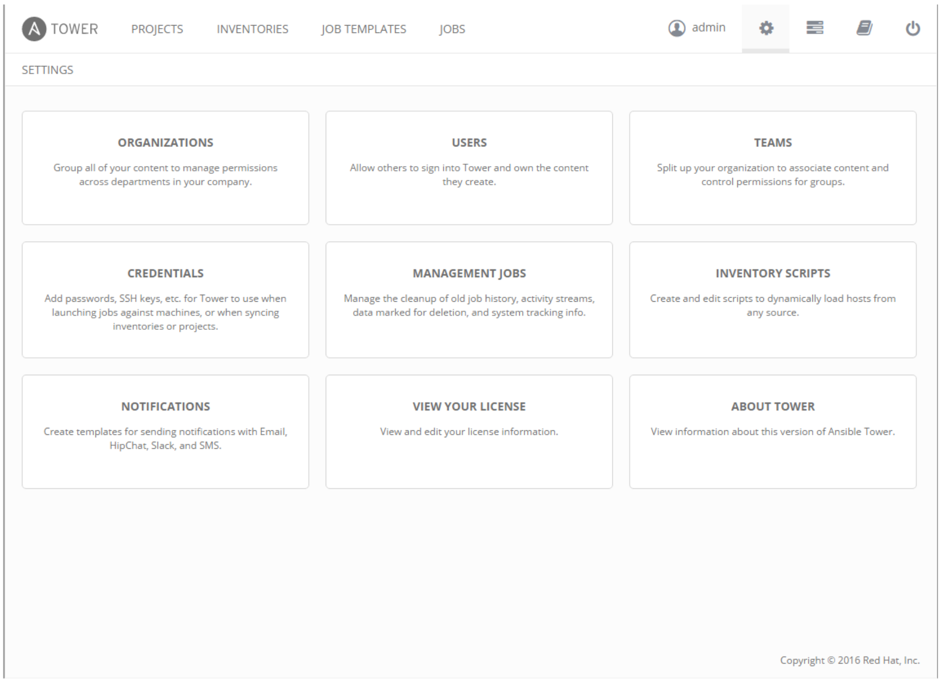
NAME		ACTIONS
cloudengine's team	cloudengine's organization	
huawei's team	huawei's organazation	
wsy's team	wsy's organization	

ITEMS 1–3 OF 3

6. Select a user, assign a role to the user, and click SAVE.

Step 5 Add credentials.

1. Click ⚙ to display the **SETTINGS** page.



ORGANIZATIONS
Group all of your content to manage permissions across departments in your company.

USERS
Allow others to sign into Tower and own the content they create.

TEAMS
Split up your organization to associate content and control permissions for groups.

CREDENTIALS
Add passwords, SSH keys, etc. for Tower to use when launching jobs against machines, or when syncing inventories or projects.

MANAGEMENT JOBS
Manage the cleanup of old job history, activity streams, data marked for deletion, and system tracking info.

INVENTORY SCRIPTS
Create and edit scripts to dynamically load hosts from any source.

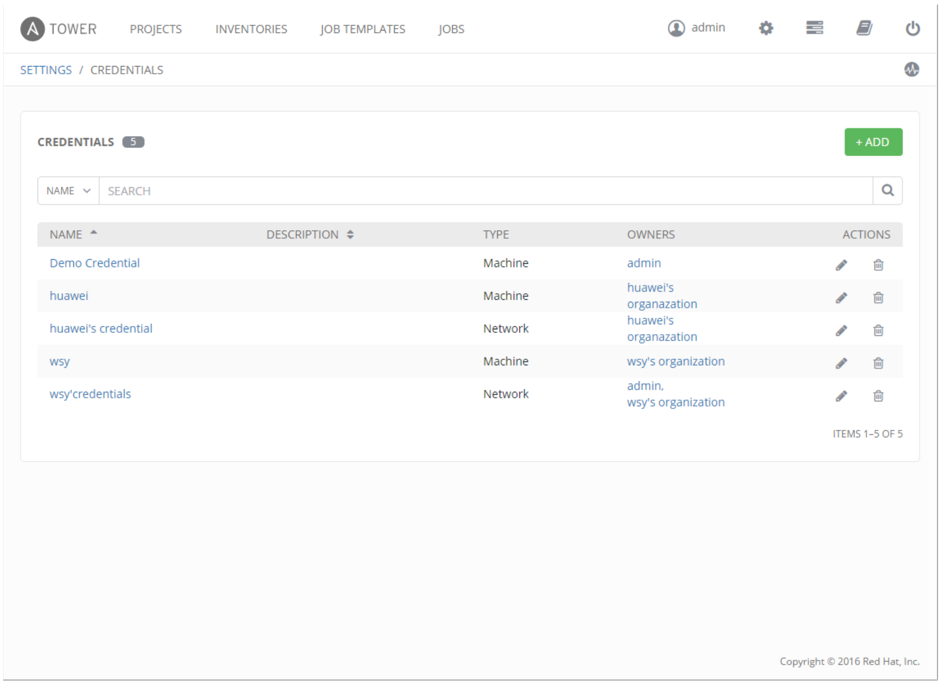
NOTIFICATIONS
Create templates for sending notifications with Email, HipChat, Slack, and SMS.

VIEW YOUR LICENSE
View and edit your license information.

ABOUT TOWER
View information about this version of Ansible Tower.

Copyright © 2016 Red Hat, Inc.

2. Click **CREDENTIALS** to display the **CREDENTIALS** page.



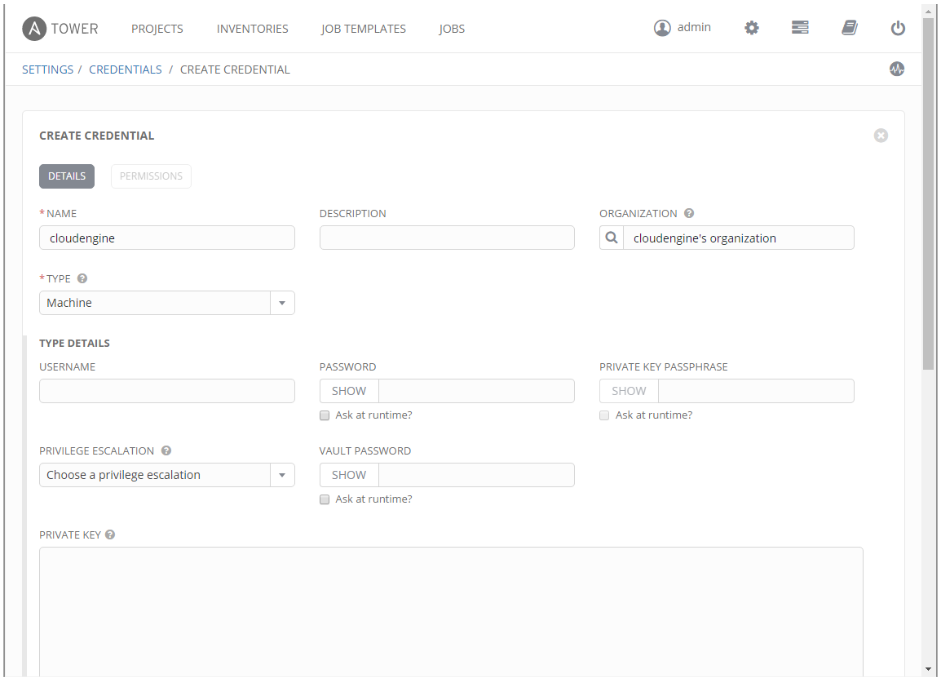
CREDENTIALS 5

NAME	DESCRIPTION	TYPE	OWNERS	ACTIONS
Demo Credential		Machine	admin	
huawei		Machine	huawei's organazation	
huawei's credential		Network	huawei's organazation	
wsy		Machine	wsy's organization	
wsy'credentials		Network	admin, wsy's organization	

ITEMS 1–5 OF 5

Copyright © 2016 Red Hat, Inc.

3. For a CE switch, you need to create machine and network credentials. Click + ADD to add a machine credential.



CREATE CREDENTIAL

*NAME: cloudengine

DESCRIPTION:

ORGANIZATION: cloudengine's organization

*TYPE: Machine

TYPE DETAILS

USERNAME:

PASSWORD: □ Ask at runtime?

PRIVATE KEY PASSPHRASE: □ Ask at runtime?

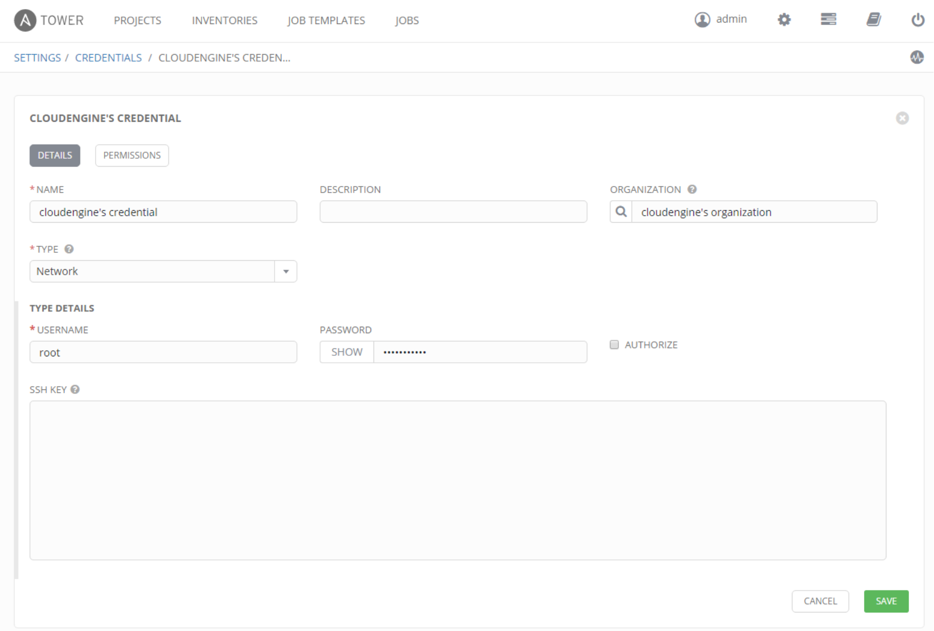
PRIVILEGE ESCALATION: Choose a privilege escalation

VAULT PASSWORD: □ Ask at runtime?

PRIVATE KEY:

4. Enter the credential name, organization, and type, and then click SAVE.

5. Click + ADD to add a network credential.



CLOUDENGINE'S CREDENTIAL

* NAME	DESCRIPTION	ORGANIZATION
cloudengine's credential		cloudengine's organization
* TYPE		
Network		

TYPE DETAILS

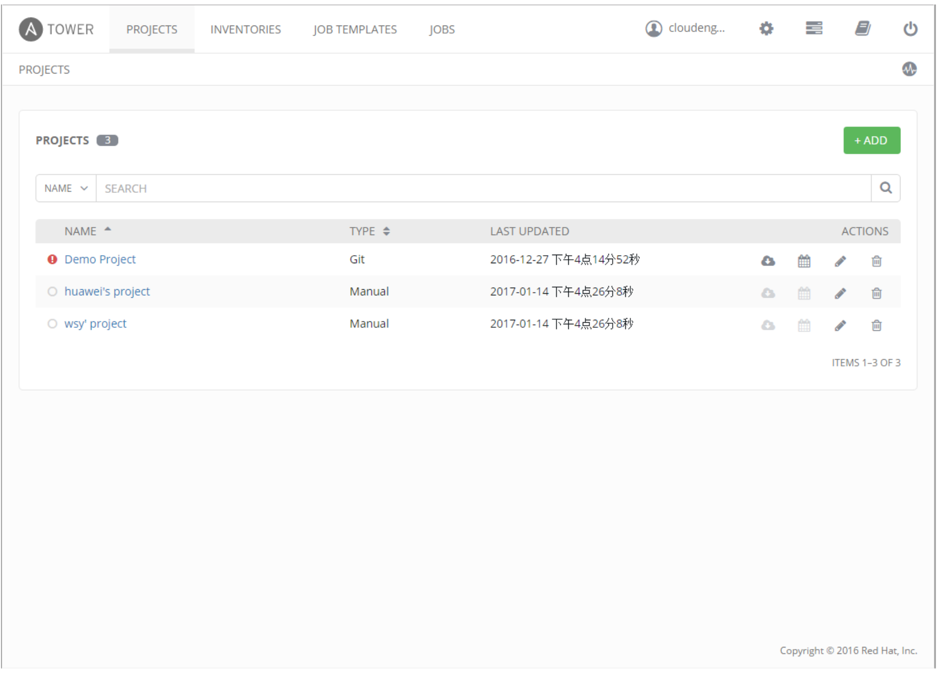
* USERNAME	PASSWORD	□ AUTHORIZE
root	SHOW ···········	

SSH KEY

6. Enter the credential name, organization, and type, as well as the user name and password for SSH login to the CE switch.

Step 6 Create a project.

1. Click **PROJICTS** to display the **PROJICTS** page.



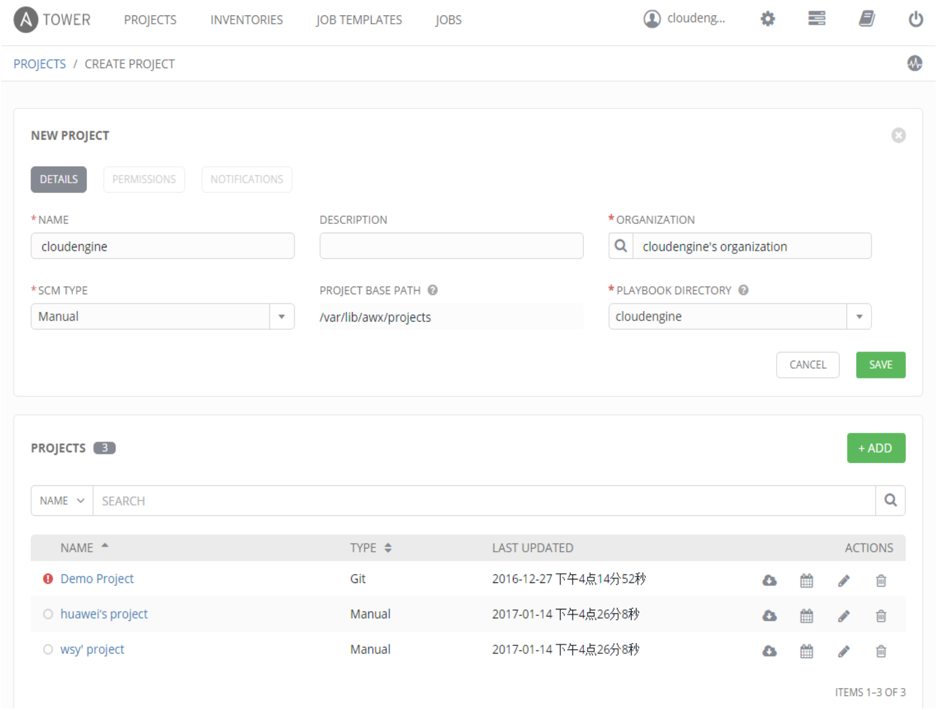
NAME	TYPE	LAST UPDATED	ACTIONS
Demo Project	Git	2016-12-27 下午4点14分52秒	
huawei's project	Manual	2017-01-14 下午4点26分8秒	
wsy' project	Manual	2017-01-14 下午4点26分8秒	

ITEMS 1–3 OF 3

2. Click + ADD to create a project.

NOTE

Enter the project name and organization, and set **SCM TYPE** to **Manual**. For **PLAYBOOK DIRECTORY**, select the playbook directory specified in section 3.2.1 "Adding a Playbook in a Specified Directory"



NEW PROJECT

*NAME: cloudengine

DESCRIPTION:

*ORGANIZATION: cloudengine's organization

*SCM TYPE: Manual

PROJECT BASE PATH: /var/lib/awx/projects

*PLAYBOOK DIRECTORY: cloudengine

PROJECTS 3

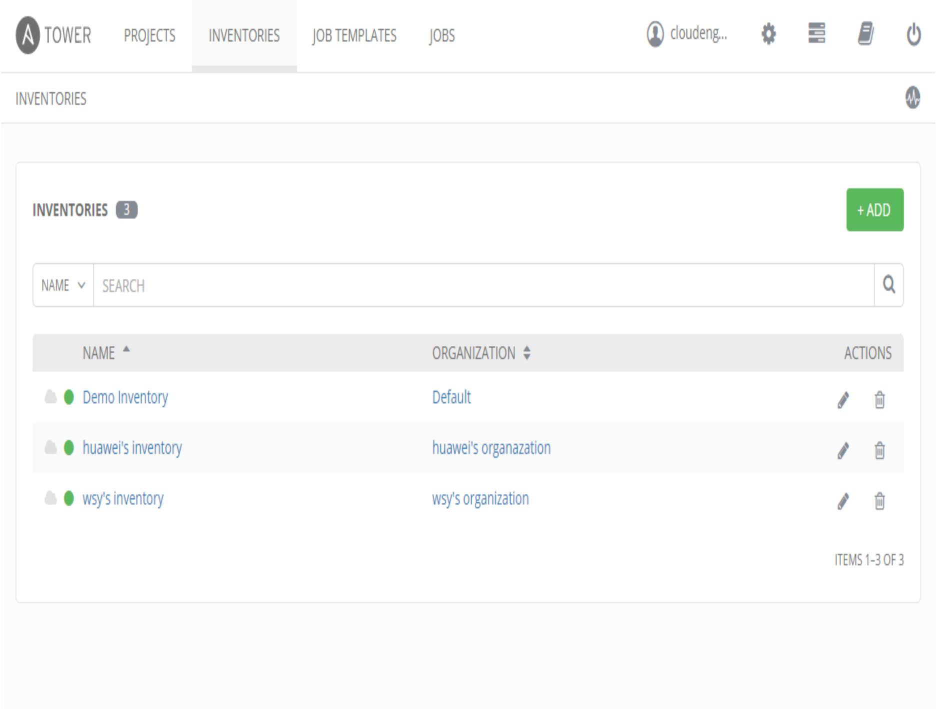
NAME	TYPE	LAST UPDATED	ACTIONS
Demo Project	Git	2016-12-27 下午4点14分52秒	
huawei's project	Manual	2017-01-14 下午4点26分8秒	
wsy' project	Manual	2017-01-14 下午4点26分8秒	

ITEMS 1-3 OF 3

3. Click SAVE.

Step 7 Add a host.

1. Click **INVENTORIES** to display the **INVENTORIES** page.

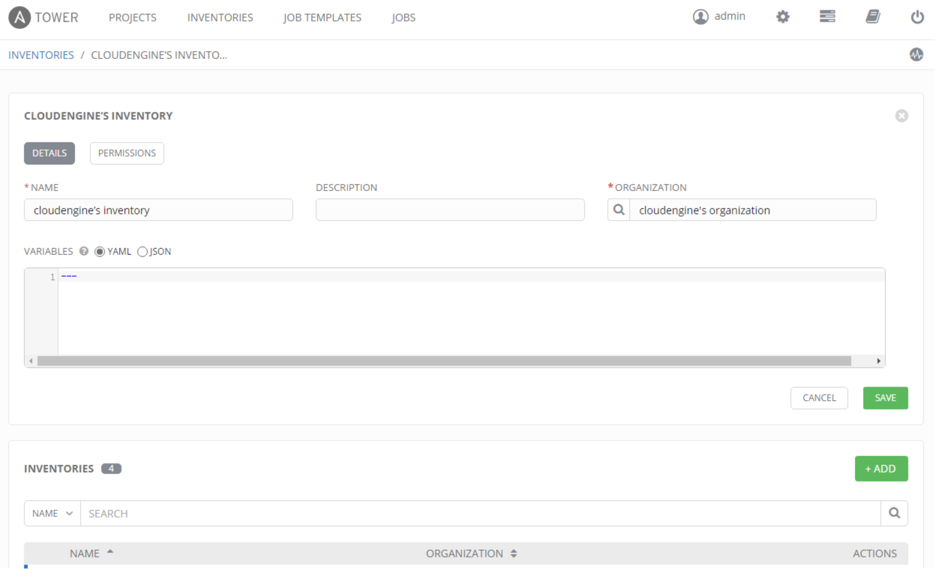


INVENTORIES 3

NAME	ORGANIZATION	ACTIONS
Demo Inventory	Default	
huawei's inventory	huawei's organazation	
wsy's inventory	wsy's organization	

ITEMS 1-3 OF 3

2. Click + ADD to create an inventory.



CLOUDENGINE'S INVENTORY

DETAILS | PERMISSIONS

*NAME: cloudengine's inventory

DESCRIPTION:

*ORGANIZATION: cloudengine's organization

VARIABLES ● YAML ○ JSON

```
---
```

CANCEL | SAVE

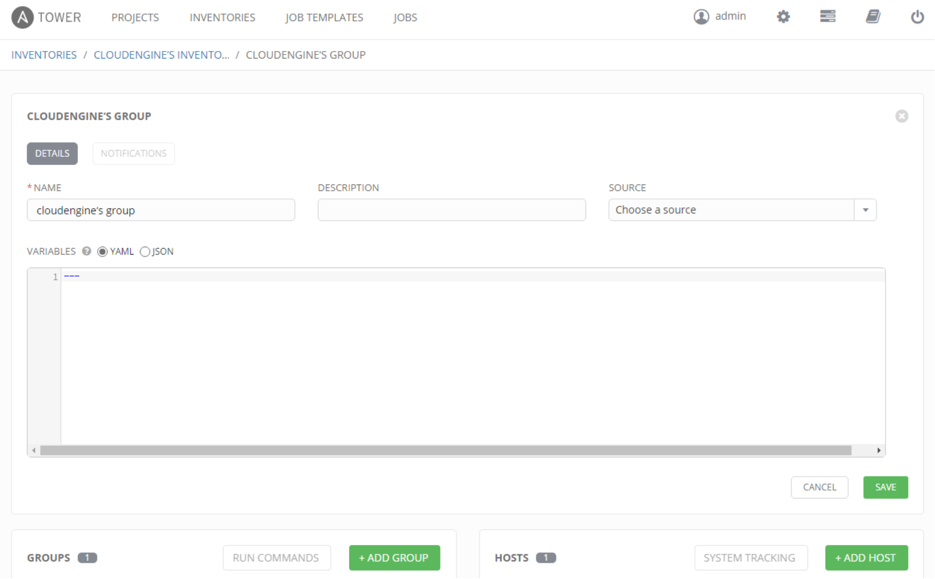
INVENTORIES 4 | + ADD

NAME | SEARCH

NAME | ORGANIZATION | ACTIONS

3. Click SAVE.

4. Click + ADD GROUP to add a group.


TOWER
PROJECTS
INVENTORIES
JOB TEMPLATES
JOBS
admin
INVENTORIES / CLOUDENGINE'S INVENTO... / CLOUDENGINE'S GROUP

CLOUDENGINE'S GROUP

DETAILS | NOTIFICATIONS

*NAME: cloudengine's group

DESCRIPTION:

SOURCE: Choose a source

VARIABLES ● YAML ○ JSON

```
---
```

CANCEL | SAVE

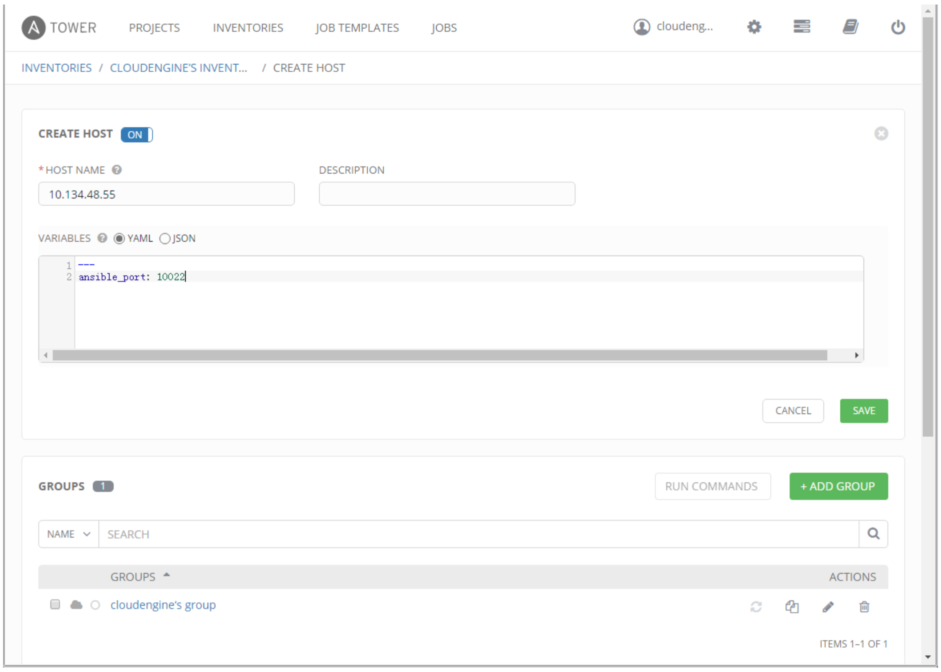
GROUPS 1 | RUN COMMANDS | + ADD GROUP

HOSTS 1 | SYSTEM TRACKING | + ADD HOST

5. Click + ADD HOST to add a host.

NOTE

Enter the host's IP address in **HOST NAME**, and enter variables of the host in **VARIABLES**. Set the ansible_port variable to the port number for SSH login to the host.



CREATE HOST ON

*HOST NAME: 10.134.48.55

DESCRIPTION:

VARIABLES ● YAML ○ JSON

```
---
ansible_port: 10022
```

GROUPS 1

□ cloudengine's group

6. Click SAVE.

----End

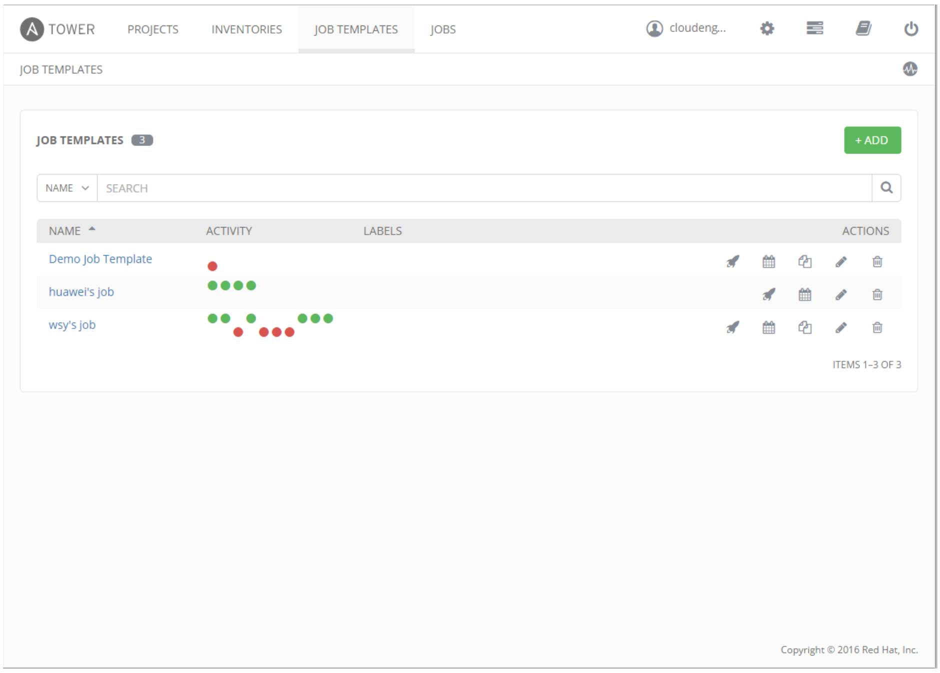
3.2.3 Running the Playbook

To run a playbook on Ansible Tower, create a job template, and specify the project to which the playbook belongs, the host where you want to run the playbook, and other associated information in the job template.

Perform the following steps to run a playbook on Ansible Tower.

Step 1 Create a job template.

1. Click **JOB TEMPLATES** to display the **JOB TEMPLATES** page.



NAME	ACTIVITY	LABELS	ACTIONS
Demo Job Template	●		
huawei's job	●●●●		
wsy's job	●● ● ● ●●● ●●●		

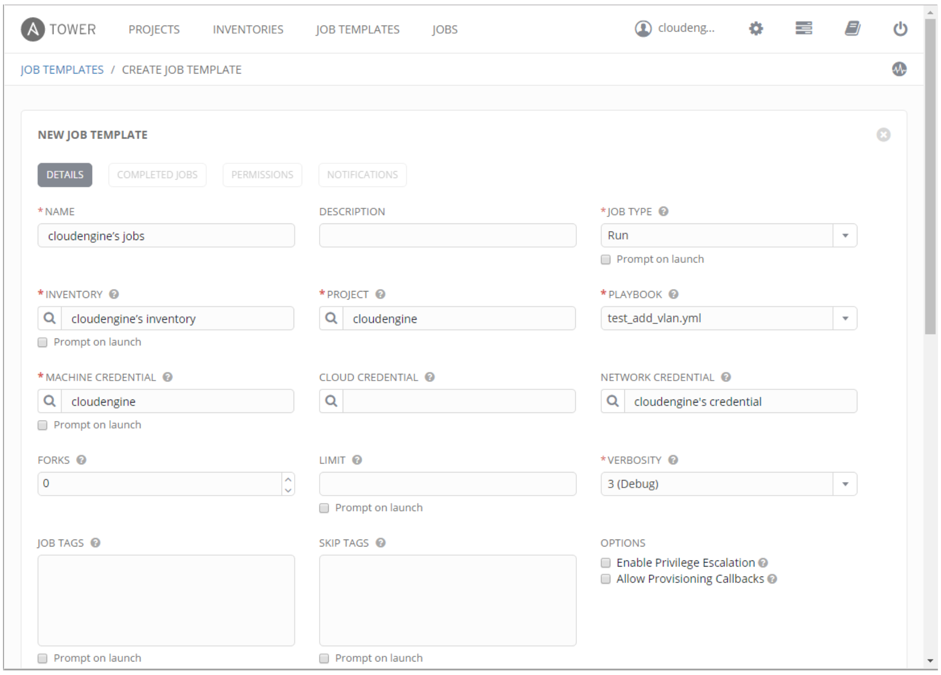
ITEMS 1-3 OF 3

Copyright © 2016 Red Hat, Inc.

2. Click + ADD to create a job template.

NOTE

Set **INVENTORY**, **PROJICT**, **PLAYBOOK**, **MACHINE CREDENTIAL**, and **NETWORK CREDENTIAL**, which are **cloudengine's inventory**, **cloudengine** project, .yml file in the selected project, **cloudengine**, and **cloudengine's credential**, respectively.



* NAME: cloudengine's jobs

DESCRIPTION:

* JOB TYPE: Run

□ Prompt on launch

* INVENTORY: cloudengine's inventory

□ Prompt on launch

* PROJECT: cloudengine

* PLAYBOOK: test_add_vlan.yml

* MACHINE CREDENTIAL: cloudengine

□ Prompt on launch

CLOUD CREDENTIAL:

NETWORK CREDENTIAL: cloudengine's credential

FORKS: 0

LIMIT:

□ Prompt on launch

* VERBOSITY: 3 (Debug)

JOB TAGS:

□ Prompt on launch

SKIP TAGS:

□ Prompt on launch

OPTIONS

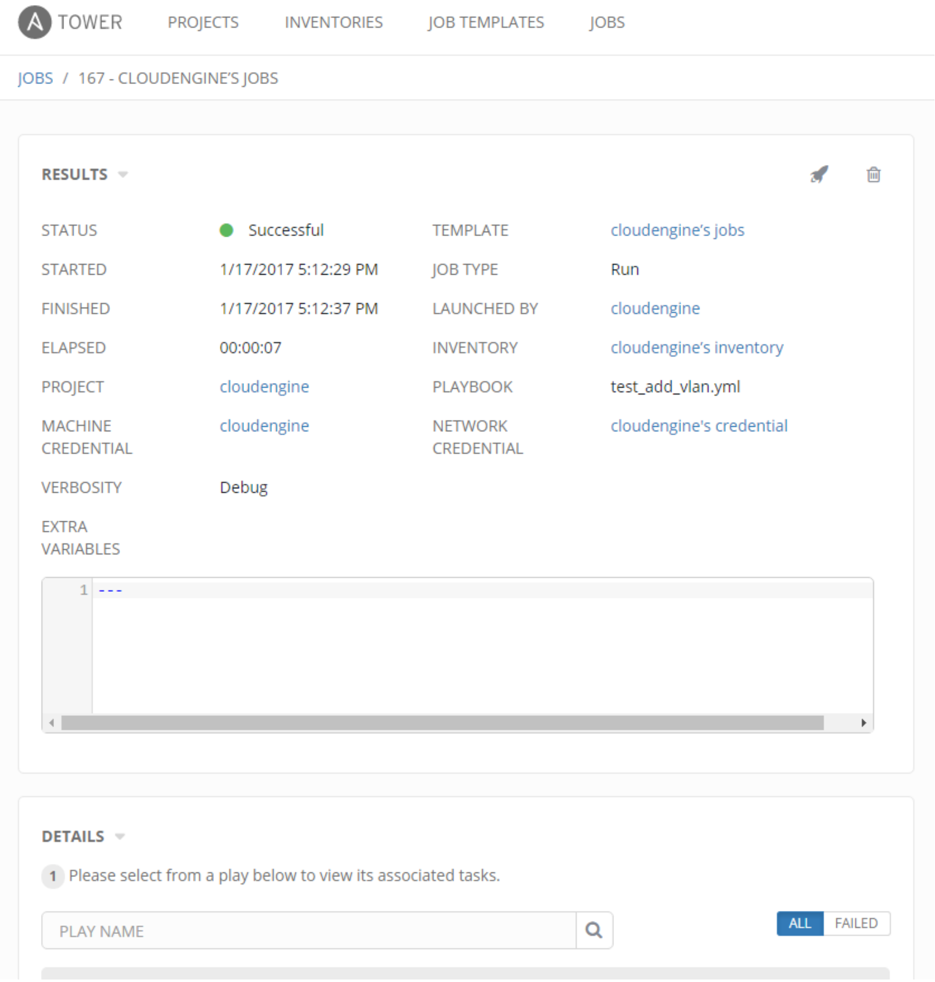
□ Enable Privilege Escalation

□ Allow Provisioning Callbacks

3. After setting job template information, click SAVE.

Step 2 Run the job.

1. Click 🚀 to run the job.
2. Check the playbook execution result.



TOWER PROJECTS INVENTORIES JOB TEMPLATES JOBS

JOBS / 167 - CLOUDENGINE'S JOBS

RESULTS

STATUS	● Successful	TEMPLATE	cloudengine's jobs
STARTED	1/17/2017 5:12:29 PM	JOB TYPE	Run
FINISHED	1/17/2017 5:12:37 PM	LAUNCHED BY	cloudengine
ELAPSED	00:00:07	INVENTORY	cloudengine's inventory
PROJECT	cloudengine	PLAYBOOK	test_add_vlan.yml
MACHINE CREDENTIAL	cloudengine	NETWORK CREDENTIAL	cloudengine's credential
VERBOSITY	Debug		
EXTRA VARIABLES			

```
---
```

DETAILS

1 Please select from a play below to view its associated tasks.

----End

4 Examples for Using Ansible

4.1 Automatic Data Collection

Networking Requirements

The Ansible server can log in to the switch using SSH and invoke related modules on the switch to collect device, VLAN, and interface information from the switch.

Figure 4-1 Using Ansible to check switch configuration

Switch
Ansible server

Configuration Roadmap

1. Create an SSH user on the switch.
2. Install Ansible and CloudEngine modules on the Ansible server, and add device information in the inventory hosts file.
3. Edit the playbook used to check device, VLAN, and interface on the switch.
4. Edit the **command.j2** file in the **templates** directory.
5. Run the playbook.

Procedure

Step 1 Create an SSH user on the switch.

# Generate a local key pair on the switch.

```
<HUAWEI> system-view
[~HUAWEI] rsa local-key-pair create
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
```

# Create an SSH user on the switch.

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

# Enable the STelnet server function on the switch.

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

# Set the service type of the SSH user **root001** to STelnet.

```
[HUAWEI] ssh user root001 service-type stelnet
[*HUAWEI] commit
```

Step 2 Add device information to the inventory hosts file on the Ansible server.

# Find the directory in which the inventory hosts file is located.

```
# ansible --version
ansible 2.2.0.0
  config file= /etc/ansible/ansible.cfg   # Ansible configuration file
  configured module search path = ['/Deafult']
#
# cat /etc/ansible/ansible.cfg
inventory      = /etc/ansible/hosts    # Inventory hosts file directory
library        = /usr/ansible/ANSIBLE-CODE/ansible/lib/ansible/modules/core/
network/cloudengine/
remote_tmp     = $HOME/.ansible/tmp
#local_tmp      = $HOME/.ansible/tmp
forks          = 5
poll_interval  = 15
sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
transport      = smart
remote_port    = 22
module_lang    = C
#module_set_locale = False
#
```

# Add device information to the inventory hosts file.

```
# cat /etc/ansible/hosts   // Modified inventory hosts file
[all:vars]
ansible_connection=local

[cloudengine]   // Host name
10.134.48.55 ansible_ssh_port=10023 username=huawei password=huaweiDC   // IP
address, port number, user name, and password for logging in to the switch
through SSH
#
```

Step 3 Edit the playbook used to check device, VLAN, and interface on the switch.

```
# cd /home/username   # Access the user directory.
# cat test-command.yml # Edited .yml playbook used to check device, VLAN, and
interface information on the switch
---

- name: cloudengine command test
  hosts: cloudengine
  connection: local
  gather_facts: no

  tasks:

  - name: "display device"
    ce_command: commands='display device' host={{inventory_hostname}}
port={{ansible_ssh_port}} username={{username}} password={{password}}
    register: data1

  - name: "display vlan"
    ce_command: commands='display vlan' host={{ inventory_hostname }}
port={{ansible_ssh_port}} username={{username}} password={{password}}
    register: date2

  - name: "display interface brief"
    ce_command: commands='display interface brief' host={{ inventory_hostname }}
port={{ansible_ssh_port}} username={{username}} password={{password}}
    register: data3

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

Step 4 Edit the **command.j2** file in the **templates** directory.

The **command.j2** file is used to deliver the output information displayed after playbook execution to the **command.json** file in the **configs** directory.

```
# cd /home/username/templates
# vi command.j2
{{ data1.stdout_lines | to_nice_json }}
{{ data2.stdout_lines | to_nice_json }}
{{ data3.stdout_lines | to_nice_json }}
#
```

Step 5 Run the playbook.

Invoke the **ansible-playbook** command to run the playbook. The output information includes the device, VLAN, and interface information on the switch. The output information and playbook execution result are recorded in the **command.json** command.

```
# cd /home/username    # Access the directory of the playbook.
# ansible-playbook test-command.yml    # Invoke the ansible-playbook command to
run the .yml playbook for displaying device information, VLAN, and interface, and
recording the output information in the command.json file.
# cd configs
# cat command.json   # Display output information and playbook execution result.
[
        [
            "Device status:",

"--------------------------------------------------------------------------------"
,
            "Slot  Card   Type                     Online   Power Register
Alarm     Primary        ",

"--------------------------------------------------------------------------------"
,
            "1     -      CE6850U-24S2Q-HI         Present  On    Registered
Normal    Master         ",
            "      FAN2   FAN-060A-F               Present  On    Registered
Normal    NA             ",
            "      PWR2   PAC-600WB-B              Present  On    Registered
Abnormal  NA             ",

"--------------------------------------------------------------------------------"
        ]
]
[
        [
            "The total number of vlans is : 7",

"--------------------------------------------------------------------------------"
,
            "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:40GE1/0/2(D)    10GE1/0/4(U)    10GE1/0/6(U)
10GE1/0/7(D)    ",
            "              10GE1/0/8(D)    10GE1/0/9(D)    10GE1/0/10(D)
10GE1/0/11(D)   ",
            "              10GE1/0/12(D)   10GE1/0/13(D)   10GE1/0/14(D)
```

```
10GE1/0/16(D)   ",
            "                10GE1/0/17(D)   10GE1/0/18(D)   10GE1/0/19(D)
10GE1/0/20(D)   ",
            "                10GE1/0/21(D)   10GE1/0/22(D)   10GE1/0/23(D)
10GE1/0/24(D)   ",
            "
20                                                                            ",
            "
21                                                                            ",
            "
22                                                                            ",
            "
30                                                                            ",
            " 100         TG:
10GE1/0/3(U)                                                    ",
            " 200         TG:
10GE1/0/5(U)                                                    ",

"                                                                                "
,
            "VID  Type     Status  Property  MAC-LRN STAT    BC  MC  UC
Description",

"--------------------------------------------------------------------------------"
,
            "   1 common   enable  default   enable  disable FWD FWD FWD VLAN
0001           ",
            "  20 common   enable  default   enable  disable FWD FWD FWD VLAN
0020           ",
            "  21 common   enable  default   enable  disable FWD FWD FWD VLAN
0021           ",
            "  22 common   enable  default   enable  disable FWD FWD FWD VLAN
0022           ",
            "  30 common   enable  default   enable  disable FWD FWD FWD VLAN
0030           ",
            " 100 common   enable  default   enable  disable FWD FWD FWD VLAN
0100           ",
            " 200 common   enable  default   enable  disable FWD FWD FWD VLAN
0200           "
        ]
]
[
        [
            "PHY: Physical",
            "*down: administratively down",
            "^down: standby",
            "(l): loopback",
            "(s): spoofing",
            "(b): BFD down",
            "(e): ETHOAM down",
            "(d): Dampening Suppressed",
            "(p): port alarm down",
            "(dl): DLDP down",
            "(c): CFM down",
            "InUti/OutUti: input utility rate/output utility rate",
            "Interface                  PHY      Protocol  InUti OutUti
inErrors  outErrors",
            "10GE1/0/1                  up       up        0.01%  0.01%
0          0",
            "10GE1/0/2                  up       up        0.01%  0.01%
0          0",
            "10GE1/0/3                  up       up        0.01%  0.01%
0          0",
            "10GE1/0/4                  up       up        0.01%  0.01%
0          0",
            "10GE1/0/5                  up       up        0.01%  0.01%
0          0",
            "10GE1/0/6                  up       up        0.01%  0.01%
0          0",
            "10GE1/0/7                    down     down         0%     0%
0          0",
            "10GE1/0/8                    down     down         0%     0%
0          0",
            "10GE1/0/9                    down     down         0%     0%
0          0",
            "10GE1/0/10                   down     down         0%     0%
0          0",
            "10GE1/0/11                   down     down         0%     0%
0          0",
            "10GE1/0/12                   down     down         0%     0%
0          0",
            "10GE1/0/13                   down     down         0%     0%
0          0",
            "10GE1/0/14                   down     down         0%     0%
0          0",
            "10GE1/0/15                   down     down         0%     0%
0          0",
            "10GE1/0/16                   down     down         0%     0%
0          0",
            "10GE1/0/17                   down     down         0%     0%
0          0",
            "10GE1/0/18                   down     down         0%     0%
0          0",
            "10GE1/0/19                   down     down         0%     0%
0          0",
            "10GE1/0/20                   down     down         0%     0%
0          0",
            "10GE1/0/21                   down     down         0%     0%
0          0",
            "10GE1/0/22                   down     down         0%     0%
0          0",
            "10GE1/0/23                   down     down         0%     0%
0          0",
            "10GE1/0/24                   down     down         0%     0%
0          0",
            "40GE1/0/2                    down     down         0%     0%
0          0",
            "Eth-Trunk1                   up       up        0.01%  0.01%
0          0",
            "  40GE1/0/1                  up       up        0.01%  0.01%
0          0",
            "LoopBack1                    up       up(s)        0%     0%
0          0",
            "MEth0/0/0                    up       up        0.01%  0.01%
0          0",
            "NULL0                        up       up(s)        0%     0%
0          0",
            "Nve1                         up       up           --     --
0          0",
            "Vlanif100                    up       up           --     --
0          0",
            "Vlanif200                    up       up           --     --
0          0"
        ]
]

#
```

----End

4.2 Automatic Configuration and Management of a Switch

Networking Requirements

The Ansible server can log in to the switch using SSH and invokes modules to deliver VLAN configuration to the switch.

Figure 4-2 Using Ansible to deliver configuration to a switch

Switch
Ansible server

Configuration Roadmap

1. Create an SSH user on the switch.
2. Add device information to the inventory hosts file on the Ansible server.
3. Edit the playbook used to deliver VLAN configuration.
4. Run the playbook.
5. Check VLAN information on the switch.

Procedure

Step 1 Create an SSH user on the switch.

# Generate a local key pair on the switch.

```
<HUAWEI> system-view
[~HUAWEI] rsa local-key-pair create
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
```

# Create an SSH user on the switch.

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

# Enable the STelnet server function on the switch.

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

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

```
[HUAWEI] ssh user root001 service-type all
[*HUAWEI] commit
```

Step 2 Add device information to the inventory hosts file on the Ansible server.

# Find the directory in which the inventory hosts file is located.

```
# ansible --version
ansible 2.3.0
  config file= /etc/ansible/ansible.cfg   # Ansible configuration file
  configured module search path = ['/Deafult']
#
# cat /etc/ansible/ansible.cfg
inventory      = /etc/ansible/hosts   # Inventory hosts file directory
library        = /usr/ansible/ANSIBLE-CODE/ansible/lib/ansible/modules/core/
network/cloudengine/
remote_tmp     = $HOME/.ansible/tmp
#local_tmp      = $HOME/.ansible/tmp
forks          = 5
poll_interval  = 15
sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
transport      = smart
remote_port    = 22
module_lang    = C
#module_set_locale = False
#
```

# Add device information to the inventory hosts file.

```
# cat /etc/ansible/hosts   # Modified inventory hosts file
[all:vars]
ansible_connection=local

[cloudengine]   # Host name
10.134.48.55 ansible_ssh_port=10023 username=huawei password=huaweiDC   # IP
address, port number, user name, and password for logging in to the switch
through SSH
#
```

Step 3 Edit the playbook used to deliver VLAN 20 and VLAN 21.

```
# cd /home/username   # Access the user directory.
# cat test_add_vlan.yml   # Edited .yml playbook for delivering VLANs
---

- name: cloudengine vlan module test
  hosts: cloudengine
  gather_facts: no
  connection: local

  tasks:

  - name: "Ensure a range of  vlans are not present on the switch"
    ce_vlan: vlan_range="20-21" state=present host={{inventory_hostname}}
port={{ansible_ssh_port}} username={{username}} password={{password}}
    register: data
#
```

Step 4 Run the playbook.

# Check information about VLAN 20 and VLAN 21 on the switch before running the playbook to deliver VLANs.

```
[~HUAWEI]display vlan 20
[~HUAWEI]
[~HUAWEI]display vlan 21
[~HUAWEI]
```

Invoke the **ansible-playbook** command to run the **test_add_vlan.yml** playbook.

```
# cd /home/username    # Access the user directory.
# ansible-playbook test_add_vlan.yml -vvv   # Invoke the ansible-playbook command
to run the .yml playbook to deliver VLANs.

PLAY [cloudengine vlan module test] ********************************************

```

```
TASK [Ensure a range of VLANs are not present on the switch] *******************
changed: [10.134.48.55]

PLAY RECAP *********************************************************************
10.134.48.55               : ok=1    changed=1    unreachable=0    failed=0   #
Running the playbook succeeded.

#
```

Step 5 Check the configuration on the switch.

# After the playbook is run successfully, check information about VLAN 20 and VLAN 30 on the switch.

```
[~HUAWEI]display vlan 20
--------------------------------------------------------------------------------
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
--------------------------------------------------------------------------------
  20

VID  Type     Status  Property  MAC-LRN STAT    BC  MC  UC  Description
--------------------------------------------------------------------------------
  20 common   enable  default   enable  disable FWD FWD FWD VLAN 0020
[~HUAWEI]
[~HUAWEI]display vlan 21
--------------------------------------------------------------------------------
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
--------------------------------------------------------------------------------
  21

VID  Type     Status  Property  MAC-LRN STAT    BC  MC  UC  Description
--------------------------------------------------------------------------------
  21 common   enable  default   enable  disable FWD FWD FWD VLAN 0021
[~HUAWEI]
```

----End

5 Appendix

For more cases about how to write Ansible YML scripts based on the switch command configuration, see **https://github.com/HuaweiSwitch/CloudEngine-Ansible/blob/master/docs/ce_config_to_yml.md**.