Getting Started with Ansible

Created by:
Ron Wilhelmson
ron.wilhelmson@deltaww.com
Version:
Draft 1

Introduction

The process of deployment, setup, configuration, and general administration of switching systems can be automated with Ansible. Ansible is an open source automation environment and is a great tool to automate many of the switching platform configurations. It is very useful for reconfiguring switches, updating NOS with new releases, installing licenses, patches, and defining network configurations whether you have one switch or thousands of switches. The power is in the documentation within the playbooks and the ability to repeat with a simple command. This document will go over the basics of Ansible and how to get started using and creating playbooks for your network deployment projects.

Objective

The objective of this guide is to document the basic steps in getting started with Ansible and creating playbooks for the setup and configuration of Delta Network switches. Basic commands and module usage for Ansible are covered along with examples and additional references to help the user.

Pre-install Connectivity and Setup

Network and Systems required

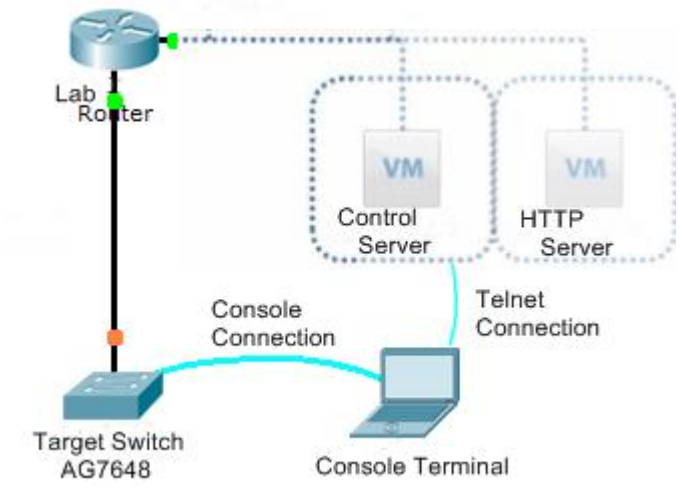
The basic systems required for building and running Ansible consist of the following:

Control Server – Linux server that runs Ansible and contains playbooks.

Target Switch – Switch that is to be configured and may be many.

Web Server – HTTP location where update, license, and install files are located.

Console Terminal – A PC running tera term and telnet to switch and control server



Network Diagram

Control server:

Ansible is configured on the control server. The Control Server, HTTP server and console terminal can be the same server and they do not need to be distinct boxes or VMs. They are covered here as separate systems for clarity.

You can configure Ansible (Ubuntu) via the PPA with the following commands:

```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install ansible
```

Verify Ansible is installed with:

```
$ ansible --version
```

Configuration should look something like the following:

```
ansible 2.5.0
  config file = /root/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.12 (default, Dec  4 2017, 14:50:18) [GCC 5.4.0 20160609]
```

Note: Please see reference section at the end of this document for additional details on installation of Ansible and details for additional linux versions.

Target Switch:

The target switch should be configured with the management port on the back of the switch connected to the local network and an IP assigned to the switch. A console connection should also be defined so that switch status can be monitored (see below).

The switch needs to have Open Network Install Environment (ONIE) configured and running. If there is no ONIE on the switch then please refer to the following:

https://github.com/DeltaProducts/SolutionCenter/blob/master/ONIE%20recovery%20from%20bootable%20USB.pdf

The console terminal should have the following prompt if ONIE is installed correctly:

```
ONIE:#
```

If ONIE is in discovery state it will try to look for NOS installers. To stop this you can use the command:

```
ONIE:# onie-dicovery-stop
```

WEB server:

The HTTP server can be a stand-alone web server that you can place files on such as Apache or you can create a simple http server on your control server. For Apache web server installation please refer to the following URL:

https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04

If you want to use a SimpleHTTPServer you can execute the following to create a working directory and start the http server:

```
$ mkdir /root/cumulus
$ cd /root/cumulus
$ python2 -m SimpleHTTPServer 80 &
```

Place your binary installation files, upgrade/patch files and license files via ftp/sftp/tftp/scp/etc in the working directory. It should be visible by a browser on the local network at URL:

http://[control server IP/Name]/

You should see similar files when connected to this URL with a browser:

```
cumulus-linux-3.5.0-bcm-amd64.bin
cumulus_license.txt
```

Console terminal:

Configure the console terminal connection to the switch as follows:

1. Connect the console port of the switch to a PC. Most switches come with a RJ45 console port. Use a RJ45-to-serial cable or an RJ45-to- USB cable to connect to a PC.
2. Use a terminal application; such as “Tera Term” to terminal connect. Configure the console port. Use these settings for the console port:
   - 115200 baud
   - No flow control
   - stop bit
   - No parity bits
   - 8 data bits

3. Connect MGMT port of the switch to the same segment as terminal station

Ansible Structure and Files

Ansible has two forms of execution, **`ansible`**, which is a single command line format, and **`ansible-playbook`**, which is a task execution format where a series of commands are executed. They both use a file structure and configuration files to gather details to execute the commands.

The following example directory structure shows what a simple set of Ansible files and playbooks may look like.

Ansible Structure

```
/root/ansible
├── ansible.cfg
├── cumulus-bin-install.yml
├── cumulus-dry-install.yml
├── cumulus-lic-install.yml
├── hosts
├── Readme.md
├── remote-onie-install.yml
└── scripts
    ├── roi.cfg
    ├── roi.sh
    └── update-host-key
```

Ansible Files

The following describes the primary files you will need to get your playbooks running.

`ansible.cfg`

The ansible.cfg file contains all of the configuration variables that can be set and Ansible will read this configuration file when it is initiated. There are many configuration option in Ansible and for initial playbook execution the standard configuration file will work without modification.

`hosts`

The hosts file is a reference file used to define the switches and systems that Ansible will be executing tasks on.

Example hosts file:

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups
#
# DPR Lab Agema Switch AG7648
#
[switches]
10.62.10.34
[http_server]
10.62.10.22
[control_server]
10.62.10.22
```

playbooks

Playbooks are written in the YAML language and are easy to read and understand after you work with them. A sample playbook is shown in the next section that executes a dry run image download from an http server to a target switch.

A basic playbook has the following sections defined:

`hosts` – defines the systems to pass commands to. In this case it is "`switches`"

`vars` – variable definitions that are referenced in the playbook

`tasks` – commands that will be executed by the playbook

`handlers` – secondary functions that are called by the tasks

Playbooks must be able to connect to the systems that are defined in the hosts file. In order to do this there are definitions for user accounts, `remote_user`, and `become` for the root account. Within the tasks there is also a `become_method` which has the option to run as sudo on the target system to execute root access commands.

Tasks and handlers are called by the `name` option and this `name` will be returned in the output as the playbook executes `tasks` and `handlers`. Within the tasks and handlers there are commands, such as `shell`, and modules, such as `wait_for`. There are many different commands and modules that can be used for different functions. Commands are built into Ansible while modules can be written and new ones are being added and created by Ansible users.

Note: Please see the references at the end of this document for additional details on commands and modules.

Ansible Playbook and Execution

The following example does a simple download of a new installation file on a cumulus switch by using the cumulus `onie-install` command in a shell. The shell command uses the –n option so the execution does nothing as a dry run. Ansible utilizes SSH to connect to other systems and execute the commands defined in the playbook tasks. It does this by creating the series of tasks and then using `sftp`, by default, to put the Ansible created commands in `.ansible/tmp` directory on the target switch. Once the tasks are created then Ansible issues an SSH commend to execute the temporary command file and then cleans up by deleting the temporary file.

Example Playbook: cumulus-dry-install.yml

```
---

- hosts: switches
  remote_user: cumulus
  become: yes
  gather_facts: yes

  vars:
    http_server: 10.62.10.22
    install_file: http://{{ http_server }}/cumulus-linux-3.5.0-bcm-amd64.bin

  tasks:
  - name: onie-nos-install Cumulus Linux NOS
    become_method: sudo
    shell: 'onie-install -n -f -i {{ install_file }}'
    tags: onie_nos_install
    notify:
        - wait for switch to come back up

  handlers:
  - name: wait for switch to come back up
    become: no
    local_action: wait_for host={{ inventory_hostname }}
                  connect_timeout=5
                  port=22
                  delay=20
                  timeout=2000
                  state=started
```

When running a playbook there are options for password passing and verbosity that are required and/or useful. The --ask-pass option will request the SSH password for the user ID being used to connect to the remote switch when the `remote_user: <user>` is used in the playbook. The --ask-sudo-pass option captures the password for sudo execution when `become_method: sudo` is used in a task. In most cases this password will be the same as the initial access user ID will become sudo when executing commands that require root access. The –v, -vv, -vvv, and –vvvv options can be very useful when running a playbook that fails as the increasing levels of verbosity display additional details on the execution.

The Ansible command for this example is run as follows:

```
ansible-playbook cumulus-dry-install.yml --ask-pass --ask-sudo-pass
```

Note: The password for the cumulus user in this case is `CumulusLinux!` and is entered at the SSH password prompt. At the SUDO password prompt you can just hit enter as it defaults then to the SSH user ID password being used.

Example Playbook Output: cumulus-dry-install.yml

```
root@DPR-LABVM-01:~/ansible# ansible-playbook cumulus-dry-install.yml --ask-pass --ask-sudo-pass
SSH password:
SUDO password[defaults to SSH password]:

PLAY [switches] ****************************************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [10.62.10.34]

TASK [onie-nos-install Cumulus Linux NOS] **************************************************************
changed: [10.62.10.34]

RUNNING HANDLER [wait for switch to come back up] ******************************************************
ok: [10.62.10.34 -> localhost]

PLAY RECAP *********************************************************************************************
10.62.10.34                : ok=3    changed=1    unreachable=0    failed=0

root@DPR-LABVM-01:~/ansible#
```

In this case the playbook executed 3 tasks OK, made 1 change and had no connections that were unreachable or any failures. This is the type of output that we want! If there are connections that were unreachable or failed executions then issuing the command again with the –vvv option would be recommended to help debug.

Example Playbooks and Projects

The following git repository has example playbooks created for this guide that may be useful:

https://github.com/Ron-delta/Getting-Started-with-Ansible

Additionally here are some examples specific to Cumulus Networks:

https://github.com/CumulusNetworks/cumulus-linux-ansible-modules

https://cumulusnetworks.com/blog/cumulus-linux-ansible-now-easier-ever/

SSH Host Keys – Issues and Updates

Because Ansible uses SSH as the connection protocol it therefore requires SSH keys for authentication to be properly in place. The following details may be useful in getting your connections to work correctly.

The first time the control server tries to connect to the switch may generate an error message in the playbook output containing a message similar to the following:

```
"Failed to connect to the host via ssh:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\r\n@    WARNING: REMOTE
HOST IDENTIFICATION HAS CHANGED!
@\r\n@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\r\n
```

First try to copy the host keys from the control server to the switch as follows:

```
root@DPR-LABVM-01:/etc/ansible# ssh-copy-id cumulus@10.62.10.34
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
any that are already installed

/usr/bin/ssh-copy-id: ERROR:
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERROR: @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
ERROR: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERROR: IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
ERROR: Someone could be eavesdropping on you right now (man-in-the-middle attack)!
ERROR: It is also possible that a host key has just been changed.
ERROR: The fingerprint for the ECDSA key sent by the remote host is
ERROR: SHA256:jxfw26hjERJ0NxURlRfj70redK/tQTSdq90gp2bXDjc.
ERROR: Please contact your system administrator.
ERROR: Add correct host key in /root/.ssh/known_hosts to get rid of this message.
ERROR: Offending ECDSA key in /root/.ssh/known_hosts:4
ERROR:   remove with:
ERROR:   ssh-keygen -f "/root/.ssh/known_hosts" -R 10.62.10.34
ERROR: ECDSA host key for 10.62.10.34 has changed and you have requested strict
checking.
ERROR: Host key verification failed.
```

If the above error occurs you need to generate new ssh keys for the new ID as follows:

```
root@DPR-LABVM-01:/etc/ansible# ssh-keygen -f "/root/.ssh/known_hosts" -R 10.62.10.34
# Host 10.62.10.34 found: line 4
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
```

Then the ssh-copy-id should be executed again to update the entry:

```
root@DPR-LABVM-01:/etc/ansible# ssh-copy-id cumulus@10.62.10.34
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '10.62.10.34 (10.62.10.34)' can't be established.
ECDSA key fingerprint is SHA256:jxfw26hjERJ0NxURlRfj70redK/tQTSdq90gp2bXDjc.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
```

```
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
cumulus@10.62.10.34's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'cumulus@10.62.10.34'"
and check to make sure that only the key(s) you wanted were added.

root@DPR-LABVM-01:/etc/ansible#
```

References:

Ansible User Guide
http://docs.ansible.com/ansible/devel/user_guide/intro_getting_started.html

Ansible Installation Guide

http://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Ansible ansible-playbook Guide:

https://docs.ansible.com/ansible/2.4/ansible-playbook.html

Ansible Commands

http://docs.ansible.com/ansible/latest/modules/list_of_commands_modules.html

Ansible Modules

http://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html

Tera Term Guide

https://ttssh2.osdn.jp/index.html.en

Open Network Install Environment (ONIE) Installation Guide

https://github.com/DeltaProducts/SolutionCenter/blob/master/ONIE%20recovery%20from%20bootable%20USB.pdf

Apache Web Server Setup Guide

https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04