Ansible - Automate for Everyone

- Ansible is a configuration management system used to configure, automate, monitor, and troubleshoot devices in large networks;
- It's written in Python and is **OpenSource under GPL Licence;**
- There is a control node and many managed nodes (network devices we manage with Ansible). **Windows OS is not supported as a control node at this moment;**
- Ansible uses an agentless architecture. No application is installed on the managed nodes and no daemon, process or agent is running;
- Ansible uses SSH for device configuration. Any system that can be configured using SSH can also be configured using Ansible;
- If managing Windows it users native PowerShell remote support instead of SSH;

Ansible Components

- Inventory is a **list of managed nodes**. An inventory file is also sometimes called a “hostfile”. The inventory can specify information like IP address, hostname or domain-name for each managed node. Default location: */etc/ansible/hosts*
  An inventory file can be formatted as an INI or YAML file.
- Modules are the **units of code ansible executes**. Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device or to installing, configuring and starting a specific server like for example Apache2 on a Linux distribution.
- Tasks are **units of action** in Ansible. You can execute a single task once with an ad-hoc command or multiple tasks in a playbook.
- Playbooks are ordered lists of tasks. We can run those tasks in that order repeatedly.
- Playbooks are written in YAML and are easy to read, write, share and understand.

Ansible Ad-Hoc Commands

- Ad-Hoc commands can be used to do quick and simple things like checking the logs, checking if a process is running, or installing a package on a list of servers
- Playbooks are used for big deployments, orchestration or system configuration
- We use the ansible command to run an ad-hoc command and the ansible-playbook command to run a playbook.
- Both ad-hoc commands and playbooks use modules to perform different tasks. Modules are units of code that do the actual work in Ansible.

Examples:

ansible 192.168.0.121 -m shell -a 'uname -a' -u andrei -k

Linux ubuntu-server2 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux

ansible 192.168.122.11 -m raw -a 'sh version' -u u1 -k | grep 'Software'

Cisco IOS Software, Linux Software (I86BI_LINUX-ADVENTERPRISEK9-M), Version 15.4(1)T, DEVELOPMENT TEST SOFTWARE

Ansible Ad-Hoc Commands

Syntax:

ansible <hosts> [-m <module_name>] -a <"arguments"> -u <username> [-k] [-K] [--become]

- command
- shell
- raw
- setup
- file
- copy
- apt
- user
- service

Ansible Playbooks

- Playbooks are Ansible's configuration, deployment, and orchestration language.
- Playbooks are used to run single or multiple tasks. They offer advanced functionality that can't be obtained with Ansible Ad-Hoc commands. Playbooks are very well suited to deploying complex applications;
- Playbooks are YAML configuration files for tasks;
- A playbook is a YAML file that contains one or more plays. Plays are dictionaries and are unordered;
- Each play contains one or more tasks. A task is a single action that will be executed. Tasks are represented as lists so orders matters;
- ansible-playbook command is used to run a playbook;
- Registers are used to capture the output of a single task into a variable;

Ansible Playbooks

- Playbooks are Ansible's configuration, deployment, and orchestration language.
- Playbooks are used to run single or multiple tasks. They offer advanced functionality that can't be obtained with Ansible Ad-Hoc commands.
- Playbooks are YAML configuration files for tasks;
- A playbook is a YAML file that contains one or more plays.
- Each play contains one or more tasks. A task is a single action that will be executed. ansible-playbook command is used to run a playbook;
- Registers are used to capture the output of a single task into a variable;

Behavioral Inventory Parameters

The following variables control how Ansible interacts with remote hosts.

- ansible_host - the name of the host to connect to
- ansible_port - the ssh port number, if not 22
- ansible_user - the default ssh username to use.
- ansible_ssh_pass - the ssh password to use
- ansible_ssh_private_key_file - private key file used by ssh
- ansible_become - equivalent to ansible_sudo or ansible_su, allows to force privilege escalation
- ansible_become_method - allows to set privilege escalation method
- ansible_become_user - equivalent to ansible_sudo_user or ansible_su_user, allows to set the user you become through privilege escalation
- ansible_become_pass - equivalent to ansible_sudo_pass or ansible_su_pass, allows you to set the privilege escalation password
- ansible_network_os - new in Ansible 2.5, used with network_cli of netconf

Ansible Vault

- The vault feature can encrypt any data used by Ansible. The vault feature can also encrypt arbitrary files, even binary files;
- Never write/use clear-text password in configuration files!

Setting up the Vault feature:

1. **Create an encrypted vault file**

   ansible-vault create vault.yaml

   New Vault password:

   Confirm New Vault password:

2. **Edit the file**

   device_name_from_inventory: <my_sudo_password_for_user_on_device>

3. **Include the vaulted variables into the inventory file**

   router1 ansible_host=192.168.122.10 ansible_become_pass="{{ router1_become_pass }}"

4. **Run the playbook:**

   ansible-playbook -i ./inventory playbook.yml --ask-vault-pass -e@./vault.yaml