By
Nitin Khullar
Software Consultant & Trainer

AGENDA

- Overflow of Configuration Management
- Introduction of Ansible
- Ansible Architecture
- Let's get started with Ansible
- Ansible Authentication & Authorization
- Let's start with Ansible Adhoc commands
- Let's write Ansible Inventory
- Let's write Ansible Playbook
- Working with popular modules in Ansible
- Deep Dive into Ansible Playbooks
- Working with Ansible Variables
- Working with Ansible Template
- Working with Ansible Handlers
- Roles in Ansible
- Ansible Galaxy

Overflow of Configuration Management

- Configuration management is a process for maintaining computer systems, servers, and software in a desired, consistent state. It's a way to make sure that a system performs as it's expected to as changes are made over time.
- Managing IT system configurations involves defining a system's desired state - like server configuration—then building and maintaining those systems. Closely related to configuration assessments and drift analyses, configuration management uses both to identify systems to update, reconfigure, or patch.
- Ex : Ansible, Chef, Puppet

Introduction of Ansible

- Ansible is a simple configuration management and IT automation engine for multi-tier deployments.
- It automates both cloud and on-premise provisioning & configuration.
- It automates cloud provisioning. Rather than managing one system at a time, Ansible uses a model that inter-relates the entire IT infrastructure and enables you to manage everything using something called an Infrastructure as Code (IAC) approach.

Why Ansible

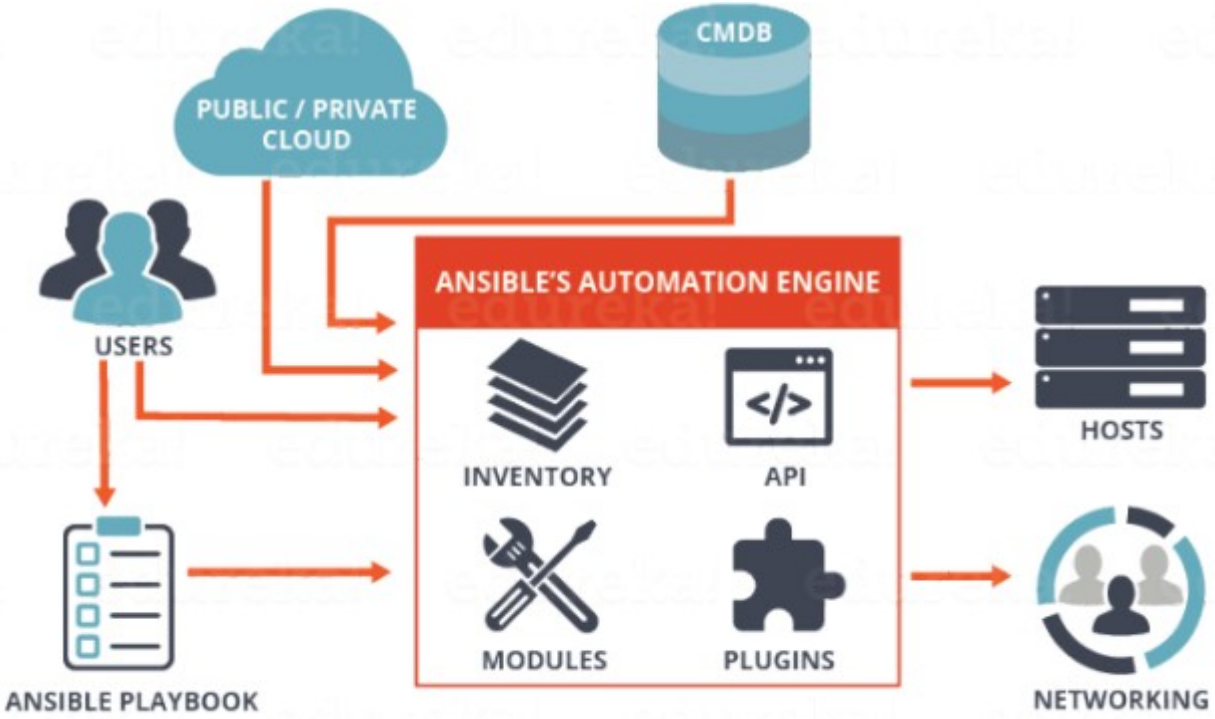
- Ansible is secure and agentless.
- It relies on OpenSSH and the code is written in YAML format.
- Ansible nodes are run on Unix systems but they can be used to configure changes across Unix as well as Windows systems.

Ansible Use Cases

- Provisioning
- Configuration Management
- Application Deployment
- Continuous Delivery
- Security & Compliance
- Orchestration

Reference : https://www.ansible.com/use-cases

Ansible Architecture



Ansible Architecture

- From the diagram above, the Ansible Orchestration Engine interacts with user who is writing the Ansible Playbook in order to execute the Ansible Orchestration Engine and interacting along with the services of public/private cloud and Configuration Management Database.
- Refer https://docs.ansible.com/ansible/2.9/dev_guide/overview_architecture.html

Let's get started with Ansible

- Installation – For this demo, we would need 3 Servers (1 Controller Node & 2 Managed Nodes) with RedHat & Python Installed.
- Setup password less authentication between Controller Nodes & Managed node.
- Enable SSH communication (port 22 should be white-listed)
- Enable EPEL repository on Redhat. Reference - https://www.tecmint.com/install-epel-repo-on-rhel-8/
- Install Ansible using sudo yum install ansible –y on Controller Node.

Ansible Authentication & Authorization

- Once Ansible has been installed, next step would be to make your managed nodes accessible from controller node without password i.e. copy content of your key (id_rsa.pub) from Controller to all managed nodes (in /home/ec2-user/.ssh/authorized_keys) and ensure its permissions are set to 600
- Once keys are set, you should be able to login from controller node to managed node

Let's start with Ansible Adhoc commands

- An Ansible ad hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable.
- ad hoc commands are great for tasks you repeat rarely.
- For example, if you want to power off all the machines in your lab for Christmas vacation, you could execute a quick one-liner in Ansible without writing a playbook. An ad hoc command looks like this:

LET'S START WITH ANSIBLE ADHOC COMMANDS

- ansible **[**pattern**]** -m **[**module**]** -a "[module options]"

Use Cases of Adhoc Commands

- ad hoc tasks can be used to reboot servers, copy files, manage packages and users, and much more. You can use any Ansible module in an ad hoc task. ad hoc tasks, like playbooks, use a declarative model, calculating and executing the actions required to reach a specified final state. They achieve a form of idempotence by checking the current state before they begin and doing nothing unless the current state is different from the specified final state.

Let's start with Ansible Adhoc commands

- Reference : https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html
- ansible webserver --list-hosts –i myinventoy
- Ansible –m ping all or ansible –m ping webserver -- i /home/ec2-user/inventory/myservers
- ansible 192.168.1.4 -a "/sbin/reboot" -f 10 -u ec2-user -become --i /home/ec2-user/inventory/myservers
- Ansible all -m command -a 'mkdir /home/ec2-user/test'
- Ansible all -m command -a 'uptime'

Let's write Ansible Inventory

- The Inventory is a description of the nodes that can be accessed by Ansible.
- By default, the Inventory is described by a configuration file whose default location is /etc/ansible/hosts
- The configuration file lists either the IP addresses or hostname of each node that is accessible by Ansible
- Every host is assigned to a group such as "web servers", "app server", "db server", etc
- The inventory file can be in one of many formats depending on your Ansible environment and plugins. Common formats include INI and YAML.

EAMPLE OF AN INVENTORY FILE

- [webserver]
- server1.example.com
- server2.example.com

- [appserver]

- 182.71.21.12
- 187.12.4.9

- [dbserver]
- db1.example.com
- db2.example.com

Let's write Ansible Playbook

- ansible-playbook command is used to run playbooks
- This command is executed on controller node and the name of playbook to be run is passed as argument
- Ex : ansible-playbook install-httpd.yml
- ansible-playbook --syntax-check install-httpd.yml can be used to check for syntax error prior executing a playbook.
- ansible-playbook -C install-httpd.yml can be used for executing a dry-run.

Let's write Ansible Playbook

```
- name: Install and start Apache HTTPD
  hosts: webservers
  become: true
  tasks:
   - name: httpd package is present
     yum:
       name: httpd
       state: present

   - name: correct index.html is present
     copy:
       src: /home/ec2-user/index.html
       dest: /var/www/html/index.html

   - name: httpd is started
     service:
      name: httpd
      state: started
      enabled: true
```

Working with popular modules in Ansible

- Yum / dnf - There is a module for most popular package managers, such as YUM, DNF and APT, to enable you to install any package on a system. Functionality depends entirely on the package manager, but usually these modules can install, upgrade, downgrade, remove, and list packages.

- Service - After installing a package, you need a module to start it. The service module enables you to start, stop, and reload installed packages

- Copy - The copy module copies a file from the local or remote machine to a location on the remote machine

Working with popular modules in Ansible

- File - The file module manages the file and its properties.
  - It sets attributes of files, symlinks, or directories.
  - It also removes files, symlinks, or directories.
- Command - Ansible Command module is used to execute commands on a remote node. The Command module, is used mostly to run simple Linux commands on a remote node/server which is part of a host group or Stand alone server mentioned in the host group.
- Ping - The Ansible ping request checks up on the remote host. This module specifically checks for: Whether the remote host is up and accessible

- Reference - https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html#

Deep Dive into Ansible Playbooks

- Refer to playbooks that will be shared with you all

Working with Ansible Variables

- Ansible uses variables to manage differences between systems. With Ansible, you can execute tasks and playbooks on multiple different systems with a single command. To represent the variations among those different systems, you can create variables with standard YAML syntax, including lists and dictionaries. You can define these variables in your playbooks, in your inventory, in re-usable files or roles, or at the command line. You can also create variables during a playbook run by registering the return value or values of a task as a new variable.

- Reference - https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Working with Ansible Template

- Templates allow you to create new files on the nodes using predefined models based on the Jinja2 templating system. Ansible templates are typically saved as .tpl files and support the use of variables, loops, and conditional expressions.

- Templates are commonly used to configure services based on variable values that can be set up on the playbook itself, in included variable files, or obtained via facts. This enables you to create more versatile setups that adapt behavior based on dynamic information.

Working with Ansible Handlers

- Sometimes you want a task to run only when a change is made on a machine.
- For example, you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged.
- Ansible uses handlers to address this use case. Handlers are tasks that only run when notified. Each handler should have a globally unique name.

Roles in Ansible

- Roles are a way to group multiple tasks together into one container to do the automation in very effective manner with clean directory structures.
- Roles are set of tasks and additional files for a certain role which allow you to break up the configurations.
- It can be easily reuse the codes by anyone if the role is suitable to someone.
- It can be easily modify and will reduce the syntax errors.
- Reference - https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

Ansible Galaxy

- Ansible Galaxy is a role and collection manager for Ansible
- Ansible Galaxy hosts Ansible roles and collections created by the community. Instead of rewriting them from scratch, you can install it on your computer using the Ansible Galaxy command-line tool and use them on your playbooks.
- Reference - https://docs.ansible.com/ansible/latest/galaxy/user_guide.html