Tech Mahindra

Ansible Overview

Disclaimer

- The source for some of the images and material in this slide pack are Google, http://docs.ansible.com, https://www.ansible.com/. This material is to be used for Learning purpose only and for internal consumption

Objectives

- At the end of this module you will be able to understand:
  - Introduction to Configuration Management
  - Introduction to Ansible Tool
  - Features and Requirements of Ansible
  - Basic Terminologies used in Ansible
  - Sample Use Cases in Ansible
  - Ansible for Network Automation

Tech Mahindra

Ansible Overview

Configuration Management

What is Configuration Management?

Tech Mahindra

- Term applies to both Application/Software Engineers and System Administrators

- System Administrators can:
  - Discover
  - Provision
  - Perform configuration
  - Deployment
  - Maintenance
  - Other repetitive tasks such as patch management
- Through Simple Codes

- Automation through use of tools ensures consistency, simplicity, uniformity through one time efforts to write the codes to Configure & Manage Infrastructure

Why Ansible?

- Example of a typical Use Case
- Need to Manage a User

- Specifically about :
  - Existence
  - Primary Group that he belongs to
  - His Home Directory

- Actions may be to:
  - Create/Modify a User profile
  - Assign Primary Group and related Policies etc.

Why Ansible?

- Typical Script

```
#!/bin/sh
USER=$1 ; GROUP=$2 ; HOME=$3
if [ 0 -ne $(getent passwd $USER > /dev/null)$? ]
then useradd $USER -home $HOME -gid $GROUP -n ; fi
OLDGID=`getent passwd $USER | awk -F: '{print $4}'`
OLDGROUP=`getent group $OLDGID | awk -F: '{print $1}'`
OLDHOME=`getent passwd $USER | awk -F: '{print $6}'`
if [ "$GROUP" != "$OLDGID" ] && ["$GROUP" != "$OLDGROUP" ]
then usermod -gid $GROUP $USER; fi
if [ "$HOME" != "$OLDHOME" ]
then usermod -home $HOME $USER; fi
```

- Ansible Way

```
tasks:
 - name: Creating Yogesh Raheja User
   user: name=yogeshraheja comment="Yogesh Raheja"  state=present
```

```
tasks:
 - name: Creating Raheja Group
   group: name=raheja state=present
```

Why Ansible?

- Simplicity
  - Easy to understand and automate
  - No specialized skills required
  - Tasks are executed in order and no special training is required
  - Intuitive
- Powerful
  - Can handle infrastructure, network as well as services offered
  - Can handle orchestration of the complete lifecycle of application and environment setup
  - Agentless and relies on industry standard communication using SSH

Configuration Management Tools

- Chef
- Puppet
- Ansible
- CFEngine
- Juju
- Rudder
- Salt
- Vagrant
- Bcfg2
- Smartfrof
- Palletops

What is Ansible?

- It is powerful IT automation Opensource Software for System Administrators
- Founded in Feb 2012 and released in 2012
- Supports Red Hat, CentOS, Ubuntu, MAC, Solaris …Support for Windows is limited
- Ansible Controller Node is supported on Linux variants
- Ansible Inc. that commercially supported Ansible was acquired by Red Hat
- Ansible Tower is an Enterprise Product

Features of Ansible

Tech Mahindra

- Ansible is open source, powerful automation software for configuring, managing and deploying software applications on the nodes without downtime

- Ansible written in Python and easily extendible
- Python (2.6 or later) needs to be installed on the remote nodes to perform it's action.

- It can be used for Network Automation

- There are Five Key Aspects related to Ansible
  - Control
  - Managed Nodes
  - Inventory
  - Modules
  - Tasks

Features of Ansible

Tech Mahindra

- Ansible categorizes as follows:
  - The **controlling machine**, where Ansible is installed
  - **Nodes** are managed by the controlling machine over SSH.
  - The location of nodes are specified by controlling machine through its inventory.
- Ansible is agent-less - No need of any agent installation on remote nodes, no background daemons or programs are executed for Ansible
- Ansible is scalable and can easily handle few to 100's of nodes from a single system over SSH connection
- Multiple commands for a deployment, can be handled by building playbooks in YAML format

Ansible Terminologies

- Control Node
  - Machine where Ansible is installed
  - Can run commands and playbooks invoking /usr/bin/ansible or /usr/bin/ansible-playbook
  - Controls Infrastructure & dictates Policies
  - Operates repository for configuration data
  - Initiates remote commands & ensures state of other machines

- Managed Node
  - Network Devices (and/or servers) managed by Ansible
  - Controller/master configures these agents/nodes
  - Sometimes referred to as “hosts”
  - Ansible is not installed on managed nodes

Ansible Terminologies

Tech Mahindra

- Inventory
  - List of things that you need to automate for e.g. Managed Nodes. Also called “hostfile”
  - Can be static file or dynamic one written to pull a list from external source
  - May contain IP Addresses for each managed node
  - Represents servers managed using .INI files
  - Organizes managed nodes, as well as creating and nesting groups for easier management
  - Machines can be grouped as required

- Modules
  - Units of code that ansible executes
  - Each Module has a particular use
  - There are almost 450 Ansible provided modules that can automate nearly every part of the environment
  - Standard Structure
  - Module : directive1=value directive2=value
  - For e.g.
  - Administers specific type of database
  - Manage VLAN interface on specific device

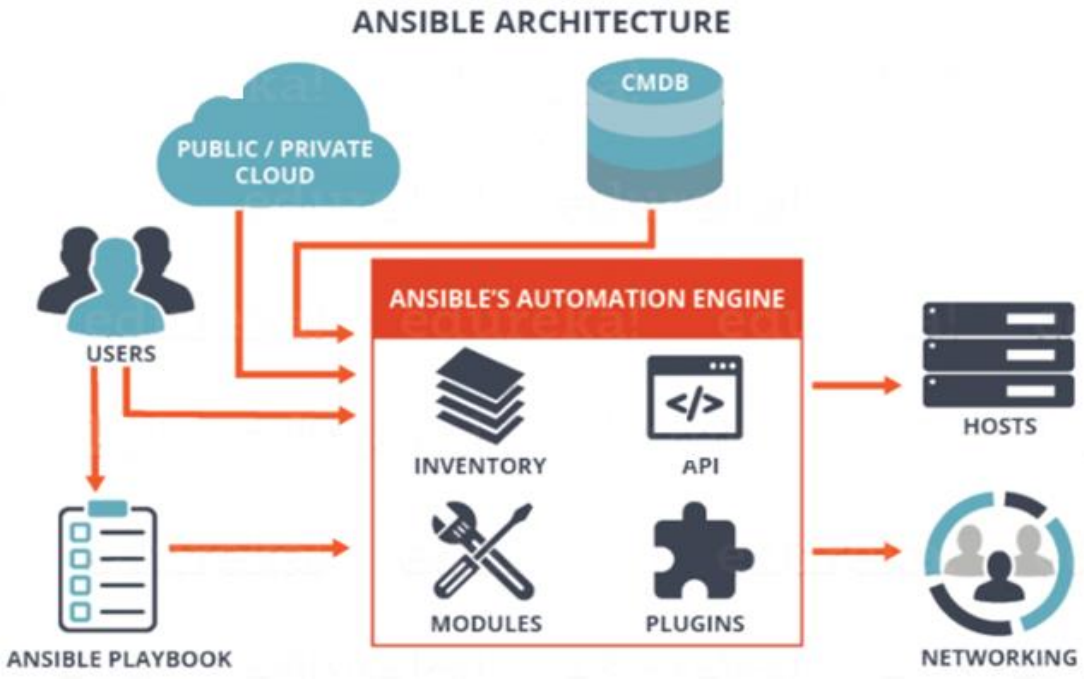
Ansible Terminologies

- Tasks
  - You can execute a single task once with an ad-hoc command
  - Simple small tasks that can be executed without logging in to client
  - for e.g.: GetAlive status of server

- Playbooks
  - Ordered list of tasks for repeated execution
  - Can include variables as well as tasks
  - Written in YAML

- Variables
  - Allow you to alter how commands, etc. run
  - Can be used in many different ways
    - Playbooks
    - Files
    - Inventories (group_vars, host_vars)
    - Command Line
    - Discovered Variables (facts)
    - Ansible Tower

Ansible Architecture



The Ansible Automation engine consists of:

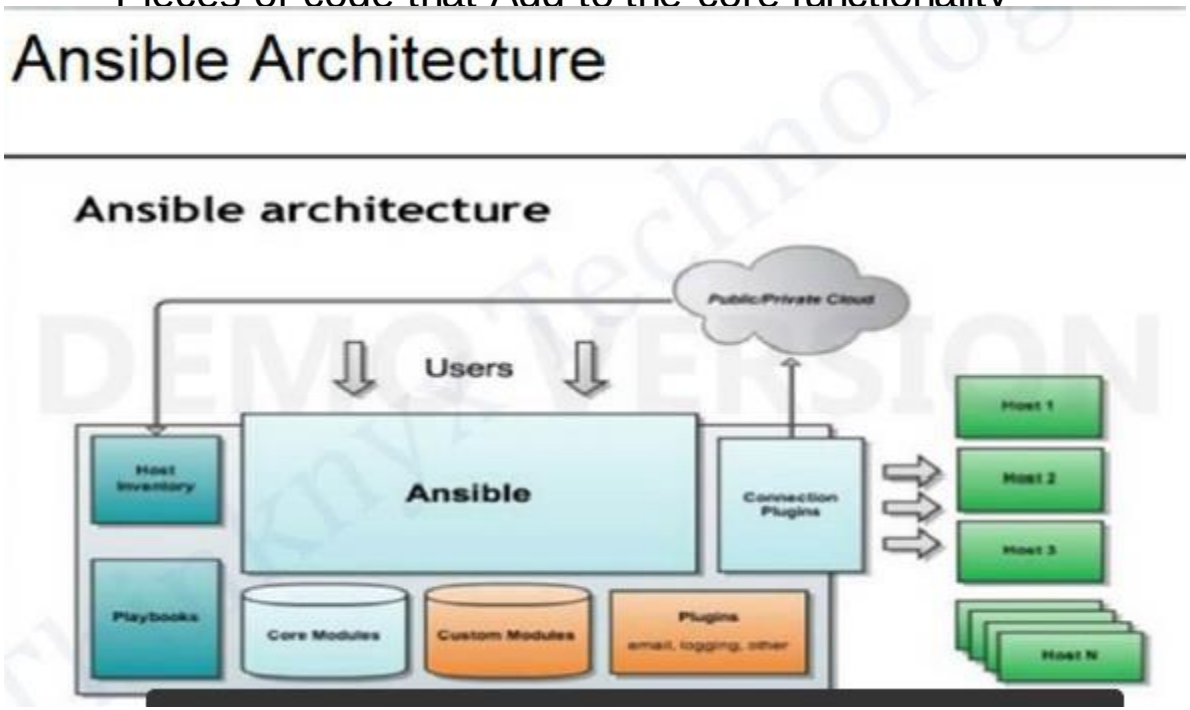
1. Inventories
2. API's
3. Modules
4. Plugin

There are a few more components in Ansible Architecture

1. Networking
2. Hosts
3. Playbook
4. CMDB
5. Cloud

Ansible Architecture : contd.

- Modules
  - Connects to nodes, pushes out “Ansible” modules, executes them over SSH and removes them when finished
  - Library of modules can reside on any machine
- Plugins:
  - Pieces of code that Add to the core functionality



Ansible Architecture : contd.

Tech Mahindra

- Inventories:
  - Represents which machines it manages using a very simple INI file
  - An inventory file looks like

```
[webservers]
www1.example.com
www2.example.com

[dbservers]
Db0.example.ccom
Db1.example.com
```

- After hosts are listed, variables can be assigned to them in simple text files (in sub directory called group_vars or host_vars

- Dynamic inventory can be used to pull your inventory from data sources such as EC2, Rackspace etc.

Ansible Architecture : contd

- Playbooks
  - Orchestrate groups of infrastructure topology with detailed control
  - This is where automation steps in
  - Simple playbook
  - ---

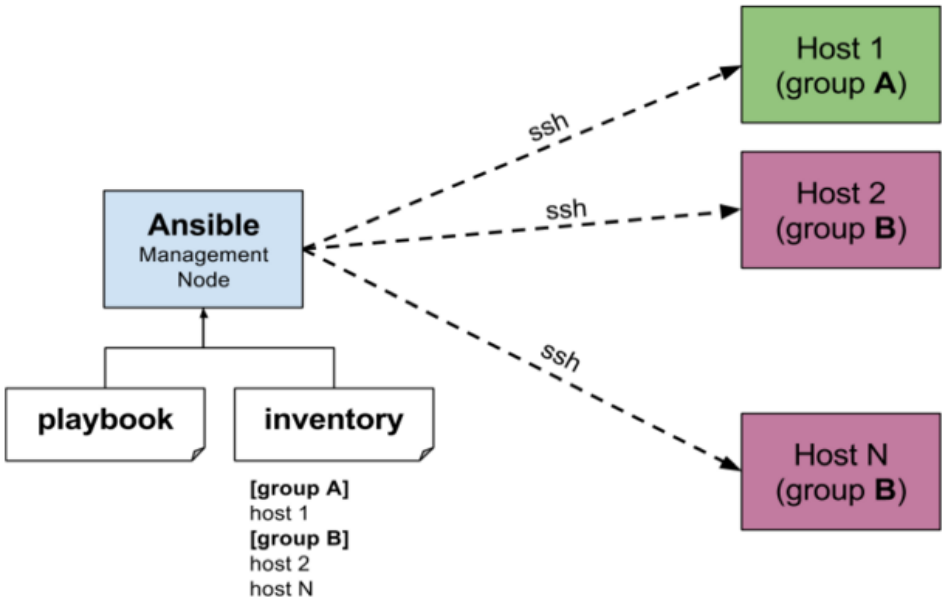
```
- hosts: webservers
serial: 5 # update 5 machines at a time
 roles:
 - common
 - webapp

- hosts: content_servers
roles:
- common
- content
```

Ansible Architecture : contd

- APIs
  - Ansible modules can be written in any language that returns JSON
  - Inventory can plugin to any datasource by writing a program that speaks to the datasource and returns JSON
  - APIs can be written in python for extending Ansible connection types, callbacks etc.

How Ansible works?

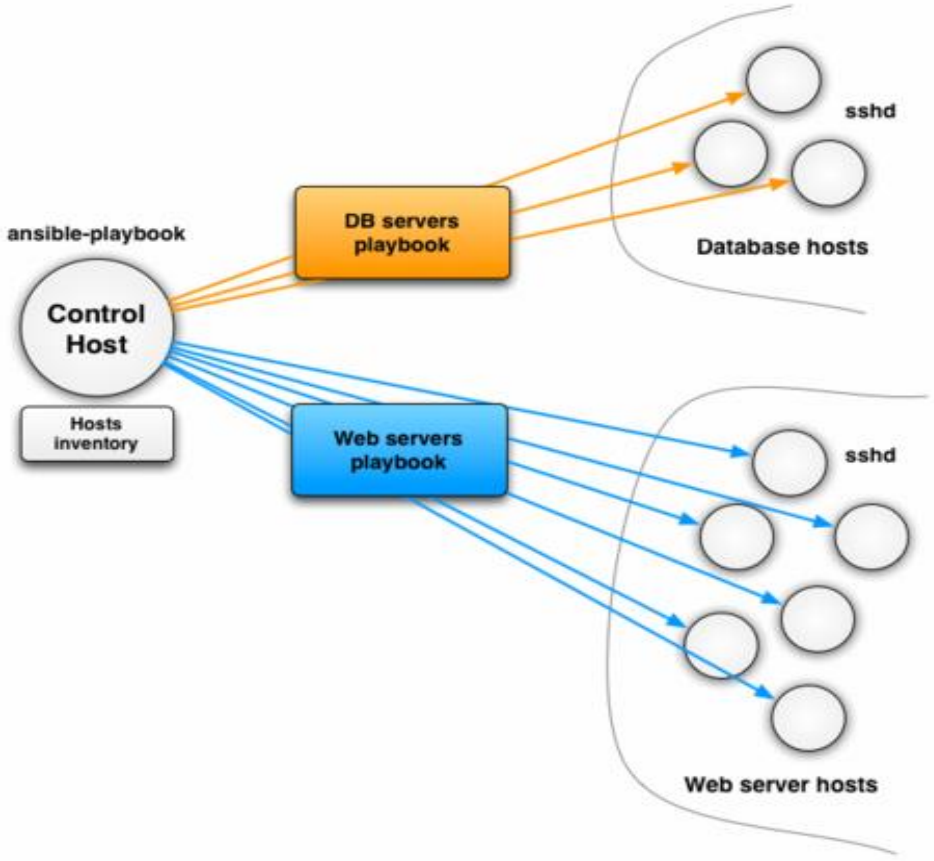



```
[root@ansible ~]#
[root@ansible ~]# cd /etc/ansible/
[root@ansible ansible]#
[root@ansible ansible]# ls
ansible.cfg  hosts
[root@ansible ansible]#
[root@ansible ansible]#
```

Ansible has a default inventory file used to define which servers it will be managing. Ex. **/etc/ansible/hosts**.

- Ansible works by configuring client Host machines from a computer with Ansible components installed and configured in Ansible Management node.
- It communicates over **SSH** channels to retrieve information from remote machines, issue commands, and copy files etc
- Configuration files are in the **YAML** data serialization format.
- Host **Inventory** contains the IP addresses of all Hosts.
- Ansible can interact with clients through either command line tools or through its configuration scripts called **Playbooks**.

Provisioning with Ansible



- Ansible is Push Based , in few cases Pull based as well.
- Ansible takes advantage of a **hosts inventory file**. It contains a list of machine addresses arranged by groups
- Here we have 2 groups: database servers and web servers. The inventory file would simple list the IP addresses and/or host names for each one.

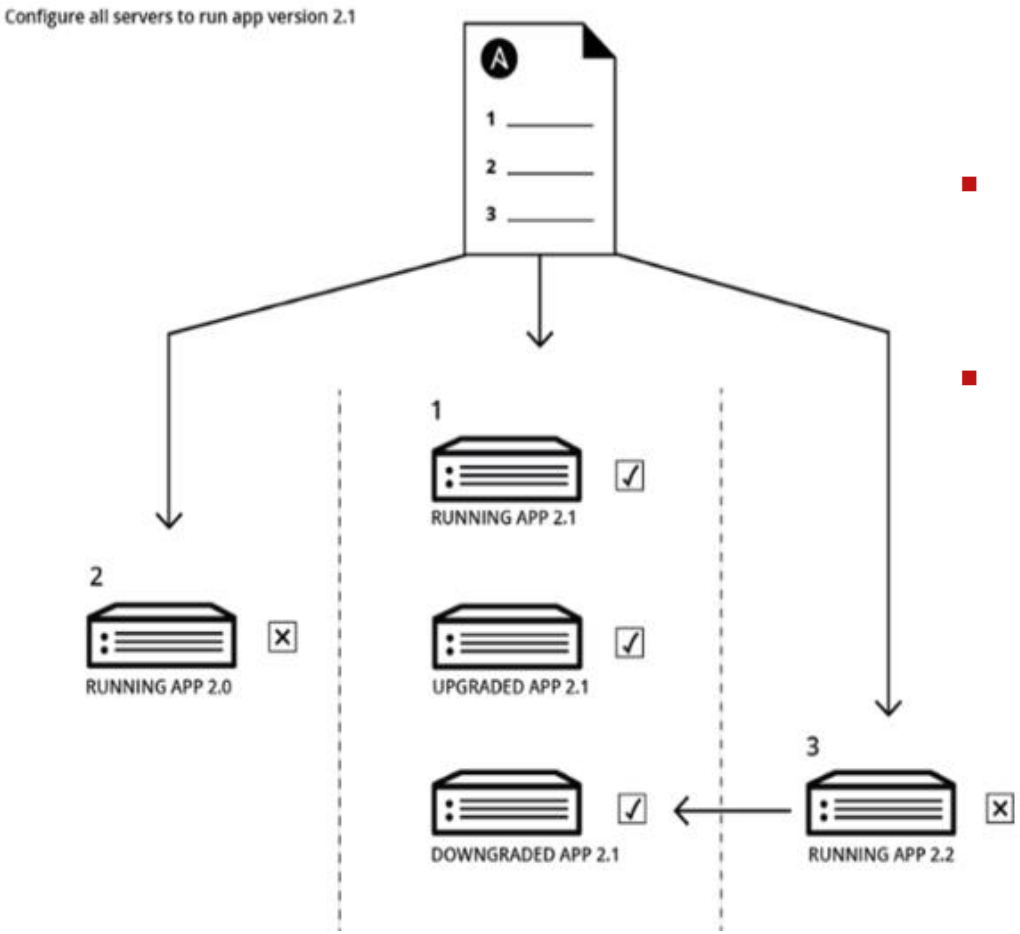
Orchestration with Ansible

Tech Mahindra

- **Complex Orchestration – Simple solutions**
  - Today's IT brings complex deployments and complex challenges.
  - Need to deal with clustered applications, multiple datacenters, public, private and hybrid clouds and applications with complex dependencies.
  - Here Ansible will help which can orchestrate complex tasks simply.

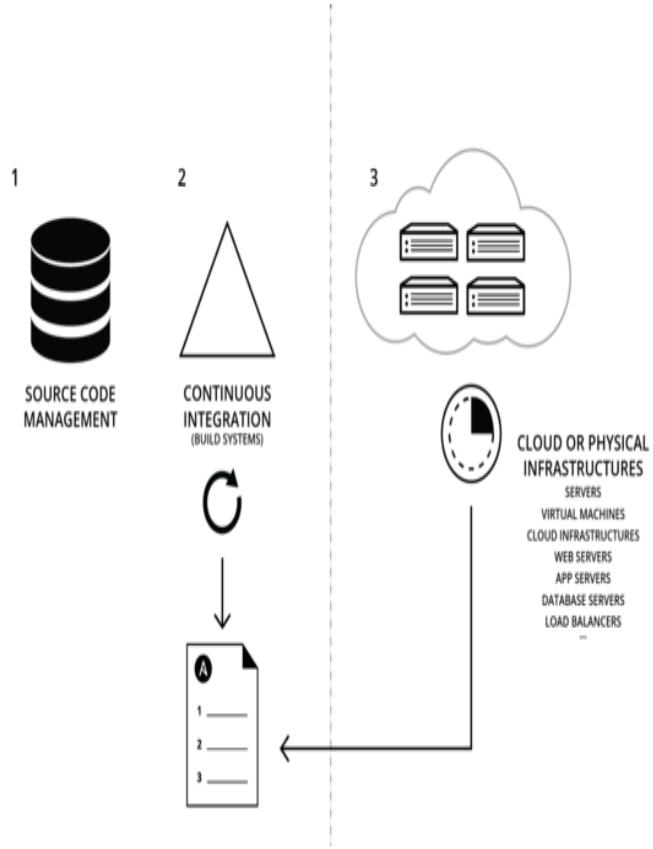
- **Ex. Complex IT orchestrations example like OpenStack.**

Configuration Management with Ansible



- Ansible is the simplest solution for configuration management.
- Ansible configurations are simple data descriptions of infrastructure
- With password or SSH key Ansible can manage systems without any agent software installation

Continuous Delivery with Ansible



- **Rolling updates and zero downtime Ansible provides**
- true multi-tier,
- multi-step orchestration.
- Fine grained control over operations
- Batchwise updates to servers while working with load balancers, monitoring systems etc.
- **Call your Play**
  - **Playbooks can be created as per need**
  - **Playbooks can**
    - **select hosts, assign tasks or roles**
  - **Identify the order in which playbooks/tasks are executed**

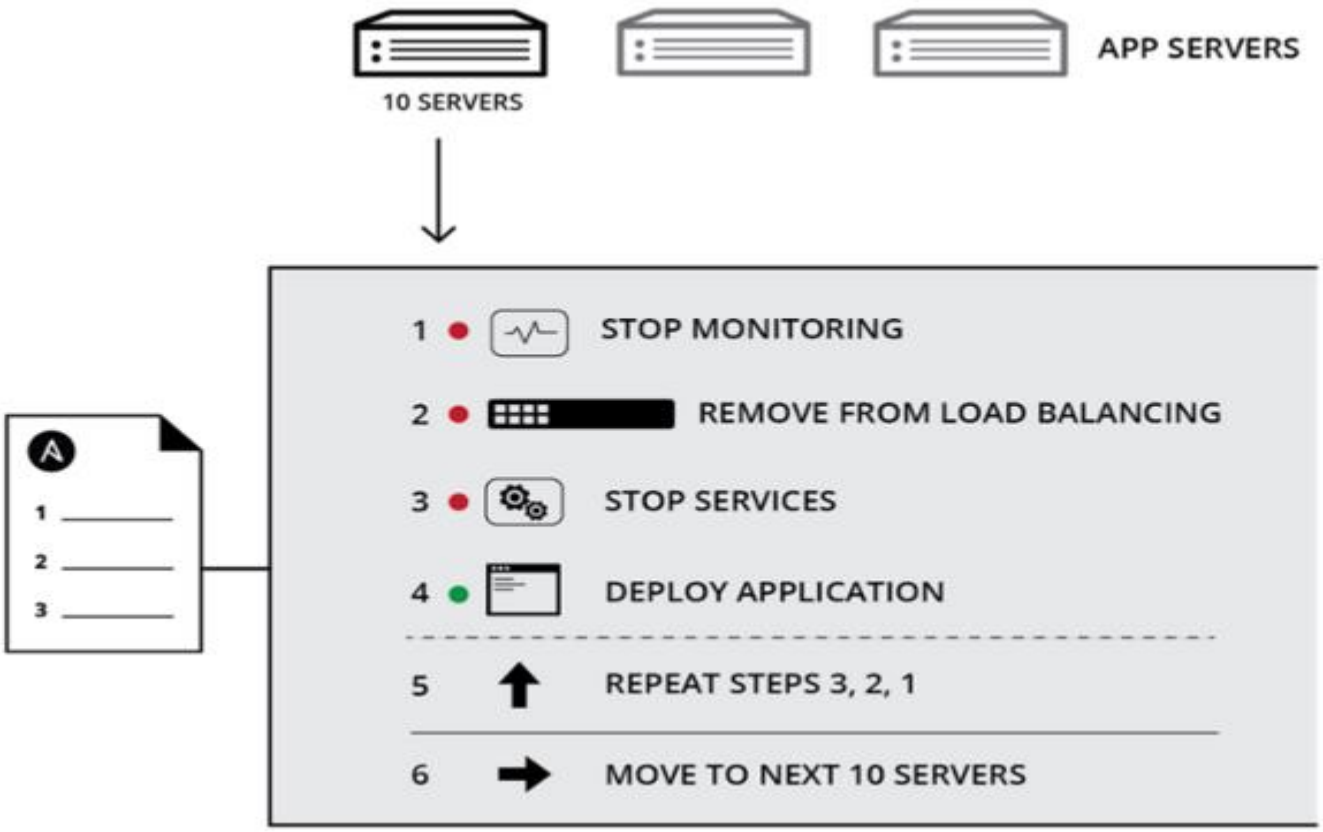
Security and Compliance with Ansible

Tech Mahindra

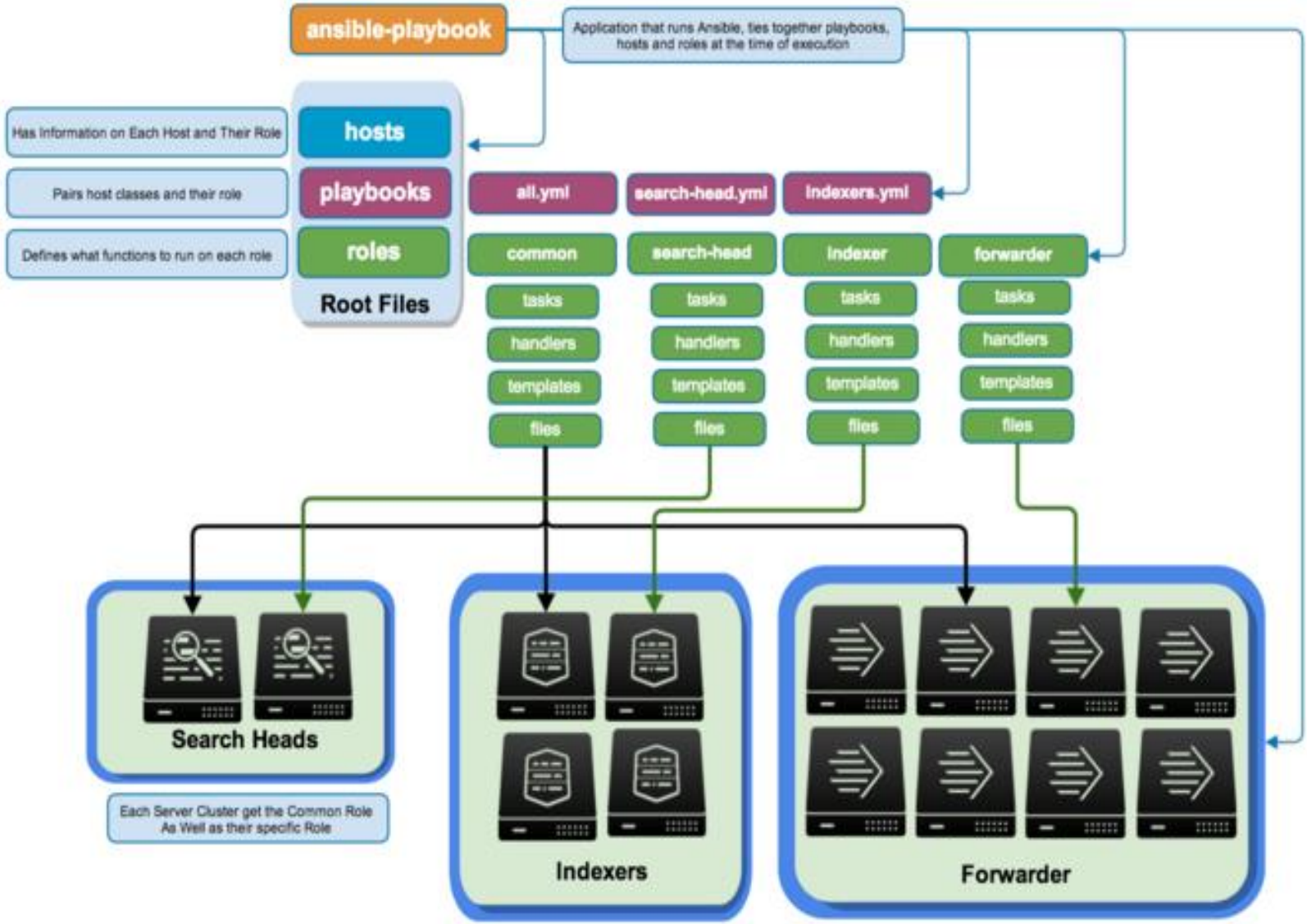
- Ansible allows you to simply define systems for security.

- Ansible's Playbook syntax allows to define secure any part of our system,
  - Like
    - setting firewall rules,
    - locking down users and groups,
    - applying custom security policies

Automated Deployment with Ansible



Ansible deployment



Tech Mahindra

Ansible Overview

Installation

Basics of Installation

- Enable ‘Extras’ and ‘Optional’ yum repos
- Ansible by default manages machines over SSH protocol
- It needs to be installed on only one machine (controller)
- It executes modules by running them on clients/managed nodes as required
- No running software/commands are left on the clients/managed nodes
- Any upgrade to Ansible needs to be run only on one machine
- Single controller can manage huge no of managed nodes remotely

Pre-requisites for installation

- Controller Machine
  - Python 2.6 or above or Python 3.5 or above needs to be installed
  - Any flavor of Unix such as Red Hat, Cent OS, Ubuntu etc.
  - Windows is not supported on control machine

- Managed Nodes
  - Python 2.6 or above or Python 3.5 or above needs to be installed
  - Communication using ssh

Installation Procedure

Tech Mahindra

- Installation on Control Machine using dnf or yum
- On RHEL or CentOS
  - $ sudo yum install ansible
- RPMS for RHEL 7 are available in Ansible Engine repository
- To enable the Ansible Engine repository
- $ sudo subscription-manager repos --enable rhel-7-server-ansible-2.6-rpms
- To install from the source clone git ansible repository
  - $ git clone https://github.com/ansible/ansible.git --recursive
  - $ cd ./ansible
- Using Bash
  - $ source ./hacking/env-setup

Tech Mahindra

Ansible Overview

Network Automation

Ansible for Network Automation

Tech Mahindra

- Network Modules do not run on managed nodes
- Network Devices cannot work with python while Ansible is designed using python
- Network Modules execute on control node, where ansible or ansible-playbook run
- Network Modules use control node to create Backup Files
- Can support multiple communication protocols as they run on control node
- Communication protocol XML over SSH or CLI over SSH depends on platform or purpose of the module
- Communication Protocol is set with ***ansible_connection*** variable

Ansible for Network Automation

- Persistent Network connection is used over local from Ansible 2.6 onwards
- In case of persistent connection, host and credentials are defined only once
- Network Platform is considered as a set of network devices with common operating system, managed by a collection of modules
- Modules for a specific network platform share a prefix, For e.g.
  - Arista: eos_
  - Cisco : ios_, iossxr_, nxos_
  - Juniper : junos_
  - VyOS: vyos_
- All modules with same network platform share certain requirements

Privilege Escalation

Tech Mahindra

- Few network Platforms allow certain functions/commands to be executed by a user with certain privileges
  - for e.g. super user in unix (sudo)
  - “enable” mode in network devices

- Ansible provides support for privilege escalation for network devices that support it

- Ansible Parameter used for privilege escalation if network platform that supports privilege escalation
  - **Become: yes with become_method: enable**

- For network devices use
  - **Connnection: network_cli or connection: httpapi** with become and become_method as above

Privilege Escalation : contd.

- When using network_cli to connect ansible to network devices, a group_vars file:
  - Ansible_connection: network_cli
  - Ansible_network_os: ios
  - Ansible_become: yes
  - Ansible_become_method: enable

- Earlier versions of ansible (2.5 or earlier), some network platforms support privilege escalation but not network_cli or httpapi connections
- For this use case code will look like:
  - Ansible_connection: local
  - Ansible_network_os: eos

```
# provider settings
    authorize: yes
    auth_pass: “{{ secret_auth_pass”}}
    port: 80
    transport: eapi
    use_ssl: no
```

Tech Mahindra

Ansible Overview

Ansible Language & Getting Started

Ansible Language Basics

- Playbooks contain Plays, Plays contain tasks, Tasks call modules
- Tasks run Sequentially
- Handlers are triggered by tasks and are run once , at the end of plays

- Static Inventory for e.g.

[web]

web-1.example.com

Web-2 example.com

[db]

db-a.example.com

db-b.example.com

Advanced Playbook Capabilities

Tech Mahindra

- Ansible has many different ways to alter how Playbooks run
- With_items, failed_when, changed_when, until etc.
- Refer to http://docs.ansible.com/ansible/playbooks_special_topics.html
- Ansible Roles are a special kind of playbook
- Fully self contained with tasks, variables, configuration templates as well as supporting files
- Refer to http://docs.ansible.com/ansible/playbooks_roles.html
- Ansible Galaxy has been created to contain user created Ansible Roles

How to Use Ansible?

- You can directly automate by running the module from command line
- Adhoc: ansible <inventory>-m

- Run an Ansible Playbook from command line
- Playbooks:ansible-playbook

- Use Automation Framework provided by Ansible Tower
- It is an Enterprise product supported by Red Hat Linux.

Ad-Hoc Commands

- Two methods
  - Directly run commands from command line

```
[justin@quickstart ~]$ ansible web -a /bin/date
172.31.12.202 | SUCCESS | rc=0 >>
Fri Apr 29 15:32:46 EDT 2016

172.31.12.200 | SUCCESS | rc=0 >>
Fri Apr 29 15:32:46 EDT 2016
```

  - Run module from the command line
  - For e.g.

```
ansible <inventory> <options>
ansible web -a /bin/date
ansible web -m ping
ansible web -m yum -a "name=openssl state=latest"
```

Running Playbooks

- Used in case of repetitive and complex tasks
- Playbooks are run against selected inventories from command line
- Each Task within playbook is run sequentially

```
[justin@quickstart playbooks]$ ansible-playbook my-playbook.yml

PLAY [Set up the demo users and files] ****************************************

TASK [setup] ******************************************************************
ok: [172.31.12.202]
ok: [172.31.12.200]

TASK [Install httpd] **********************************************************
changed: [172.31.12.202]
changed: [172.31.12.200]

TASK [Write apache config file] ***********************************************
changed: [172.31.12.202]
changed: [172.31.12.200]

TASK [Start httpd] ************************************************************
changed: [172.31.12.202]
changed: [172.31.12.200]

RUNNING HANDLER [restart apache] **********************************************
changed: [172.31.12.200]
changed: [172.31.12.202]

PLAY RECAP ********************************************************************
172.31.12.200              : ok=5    changed=4    unreachable=0    failed=0
172.31.12.202              : ok=5    changed=4    unreachable=0    failed=0
```

Running Playbooks

- Output at the end after same playbook is run again

```
[justin@quickstart playbooks]$ ansible-playbook my-playbook.yml

PLAY [Set up the demo users and files] ****************************************

TASK [setup] ******************************************************************
ok: [172.31.12.202]
ok: [172.31.12.200]

TASK [Install httpd] **********************************************************
ok: [172.31.12.202]
ok: [172.31.12.200]

TASK [Write apache config file] ***********************************************
ok: [172.31.12.202]
ok: [172.31.12.200]

TASK [Start httpd] ************************************************************
ok: [172.31.12.202]
ok: [172.31.12.200]

PLAY RECAP ********************************************************************
172.31.12.200              : ok=4    changed=0    unreachable=0    failed=0
172.31.12.202              : ok=4    changed=0    unreachable=0    failed=0
```

Additional Features/Utilities

- Check mode
  - Allows you to check the execution of the tasks/playbooks before they are executed in the actual environment
  - Dry Run for the commands/playbooks
  - Validates playbooks/commands before they are run on target systems
  - There is no change in state of the machines after the command/playbook is executed
- Ansible Tower
  - Red hat Enterprise Product
  - Offers ease and support to use of Ansible
  - Easy Access and view to playbooks and state of the systems post execution
- Ansible Galaxy
  - Contains many user defined and vendor provided roles
  - Easy to Adapt/modify
  - Easy to execute

Tech Mahindra

Ansible Overview

Ansible Playbooks

Ansible playbooks

- Playbooks are sequence of tasks you want to execute on your devices maintained in your inventory file in groups
- They are expressed in YAML
- Each playbook can have a play and each play can have one or more tasks

- Tasks are executed sequentially

```
- name: example playbook
  hosts: junos
  tasks:
  - tasks you want to automate
```

Playbook in Ansible

```
---
- hosts: lb
  remote_user: root
  roles:
  - haproxy

- hosts: www
  remote_user: vagrant
  sudo: yes
  roles:
  - nginx
  - php
  - wordpress

- hosts: db
  remote_user: root
  roles:
  - mysql
  - mysqladmin
```

Which servers to configure

Which user to login as

What to run – tasks/roles

Whether to run tasks with admin privileges

Play

Ex of Playbook

Tech Mahindra

- **Tasks**

  Let's create a simple play which will check and install http packages on the remote host. Here, this play will run the [**task**] named "**Install Apache httpd**" on [**all**] hosts listed in the hosts inventory file.

  ## PLAYBOOK TO INSTALL AND CONFIGURE APACHE HTTP ON CENTOS

  - host: all
  tasks:
  - name: Install Apache httpd
  yum: pkg=httpd state=installed

```
---
## PLAYBOOK TO INSTALL AND CONFIGURE APACHE HTTP ON CENTOS

- hosts: all
  tasks:
   - name: Install Apache httpd
     yum: pkg=httpd state=installed
~
```

Run the playbook

- Run the playbook. you should see the following sample output as shown below:
  # ansible-playbook main.yml

```
[root@ansible http-playbook]#
[root@ansible http-playbook]# ansible-playbook main.yml
 [WARNING]: The version of gmp you have installed has a known issue regarding
timing vulnerabilities when used with pycrypto. If possible, you should update
it (i.e. yum update gmp).


PLAY [all] *********************************************************************

GATHERING FACTS ****************************************************************
ok: [host1]

TASK: [Install Apache httpd] ***************************************************
changed: [host1]

PLAY RECAP *********************************************************************
host1                      : ok=2    changed=1    unreachable=0    failed=0

[root@ansible http-playbook]#
[root@ansible http-playbook]#
```

Run Playbook

1. Create an inventory file and define your devices:

```
[ios]
router1 ansible_host=192.168.0.201
switch1 ansible_host=192.168.0.203
[junos]
router2 ansible_host=192.168.0.130
```

Sample inventory file: 2 groups, total of 3 network devices defined

2. Create a playbook:

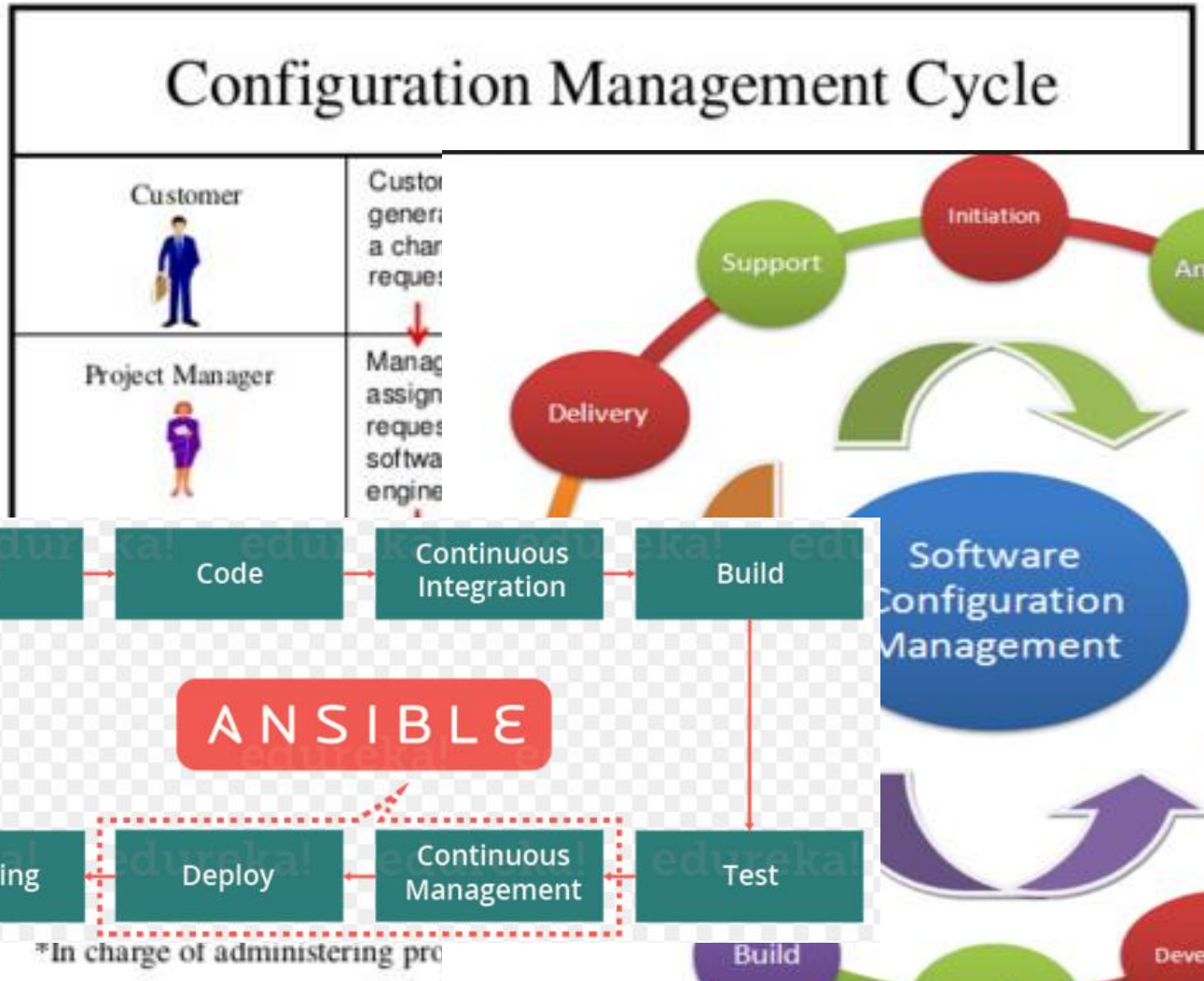
```
- name: ping Google's DNS from the device
  hosts: ios
  tasks:
   - ios_command:
       commands: ping 8.8.8.8
  ..... <omitted>
```

Sample playbook: Include plays and tasks to execute

3. Run the playbook: **ansible-playbook *myPlaybook.yml***

Who uses Ansible

Configuration management cycle



Prerequisite for Ansible

- **Ansible Tower has the following setup requirements:**
  http://docs.ansible.com/ansible-tower/2.2.2/html/installandreference/requirements_refguide.html

- **Ansible installation information**
  http://docs.ansible.com/intro_installation.html

- **Ansible Documentation**
  http://docs.ansible.com/

- **Pre-requsite to start Ansible Learning –**
  - Linux platform.
  - Configuration Management of Linux OS
  - Basic Linux Adminstration
  - Programming Knowledge to write modules.
  - With Ansible, basic YAML knowledge

Summary

- You can now understand:
  - Need for Configuration Management
  - Ansible Tool Architecture
  - Features and Requirements for Ansible
  - Basic Terminologies used in Ansible
  - Sample Use Cases in Ansible

Tech Mahindra

Ansible Terminologies

- **Playbooks:** A structured way to put all of the defines tasks for your application or your whole setup.
- **Modules:** In built functions which executes at the backend to perform underlined tasks in Ansible.
- ***yml files:** describe a set of desired states that a system needs to be in, for example "apache needs to be installed and running".
- **Ansible Tower:** Ansible Tower by Red Hat helps is a web-based solution that makes Ansible even more easy to use for IT teams of all kinds. It's designed to be the hub for all of your automation tasks.

- **Ansible Galaxy:** Ansible Galaxy is a free site for finding, downloading, and sharing community developed roles. Downloading roles from Galaxy is a great way to jumpstart your automation projects..
- **Ansible for Unix/Linux:** The Ansible master communicates and manage Unix/Linux Clients using SSH by default.
- **Ansible for Windows:** Starting in version 1.7, Ansible also contains support for managing Windows machines. This uses native PowerShell remoting, rather than SSH. and uses the "winrm" Python module to talk to remote hosts.

Ansible Commands

Tech Mahindra

- **Ansible**: Ansible is an extra-simple tool/framework/API for doing 'remote things' over SSH. This is the adhoc command that allows for a 'single task playbook' run.
- **Ansible-doc**: Ansible-doc displays information on modules installed in Ansible libraries. It displays a terse listing of modules and their short descriptions provides a printout of their DOCUMENTATION strings, and it can create a short "snippet" which can be pasted into a playbook..
- **Ansible-playbook:** Ansible playbooks are a configuration and multinode deployment system. Ansible-playbook is the tool used to run them.
- **Ansible-pull**: ansible-pull to set up a remote copy of ansible on each managed node, each set to run via cron and update playbook source via a source repository. This inverts the default push architecture of ansible into a pull architecture, which has near-limitless scaling potential.

- **Ansible-vault:** ansible-vault can encrypt any structured data file used by Ansible.
- **Ansible-galaxy**: Ansible Galaxy is a shared repository for Ansible roles. Th ansible-galaxy command can be used to manage these roles, or for creating skeleton framework for roles you'd like to upload to Galaxy.
- **Ansible-console**: Ansible console is a REPL that allows for running ad-hoc tasks against a chosen inventory (based on dominis' ansible-shell).

Tech Mahindra

Additional Resources

- Network Automation with Ansible, report by Jason Edelman (free, login required)

  https://www.oreilly.com/learning/network-automation-with-ansible

- Up and Running with Ansible (free eBook)

  https://ipfs.io/ipfs/QmTJaLdhUW6jTdXGFoqv7wZe5KguBi5F2u4ihBdrUMVPhw

- Ivan Pepeljak's Blog http://ipspace.net

- Learn Linux: video training from safaribooksonline.com or lynda.com or pluralsight.com