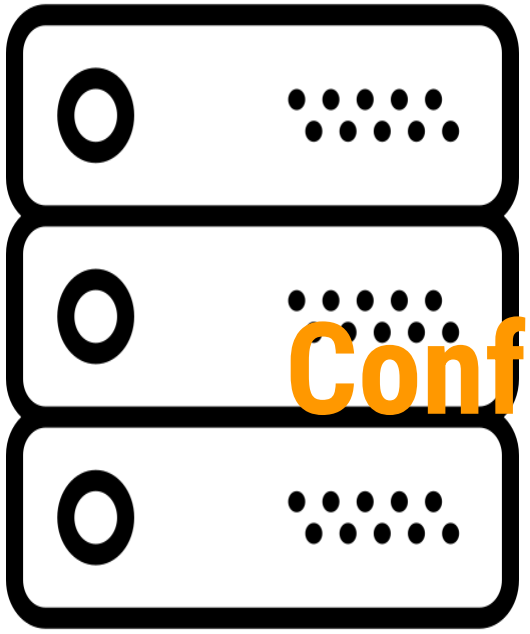
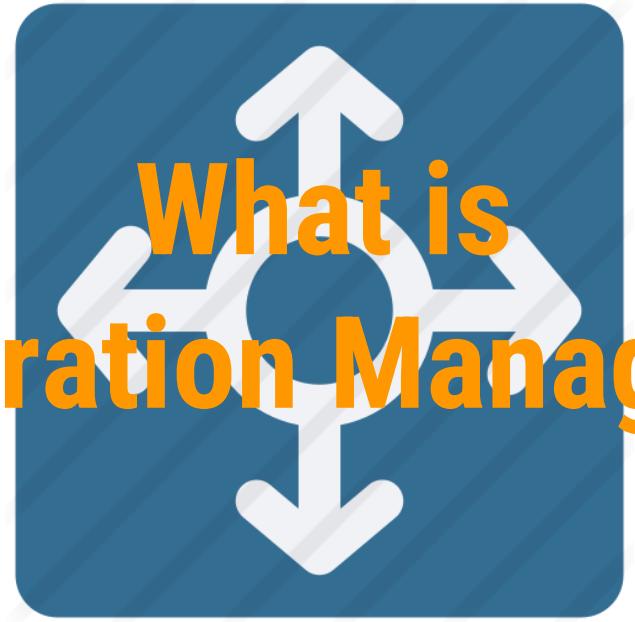
Ansible Advanced

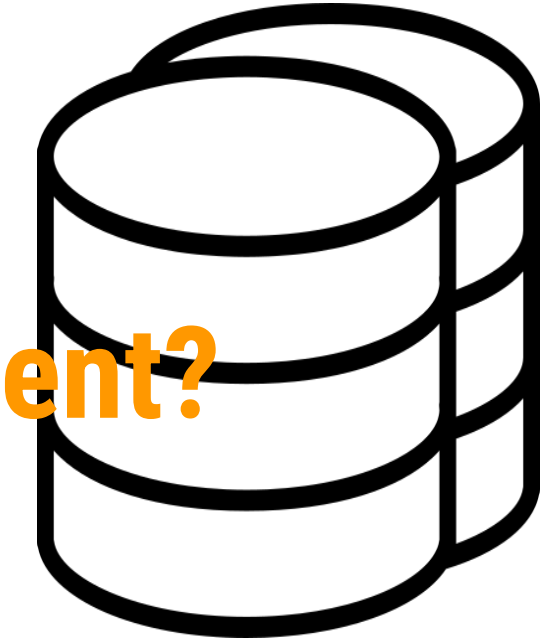
Traditional Datacenter



Servers



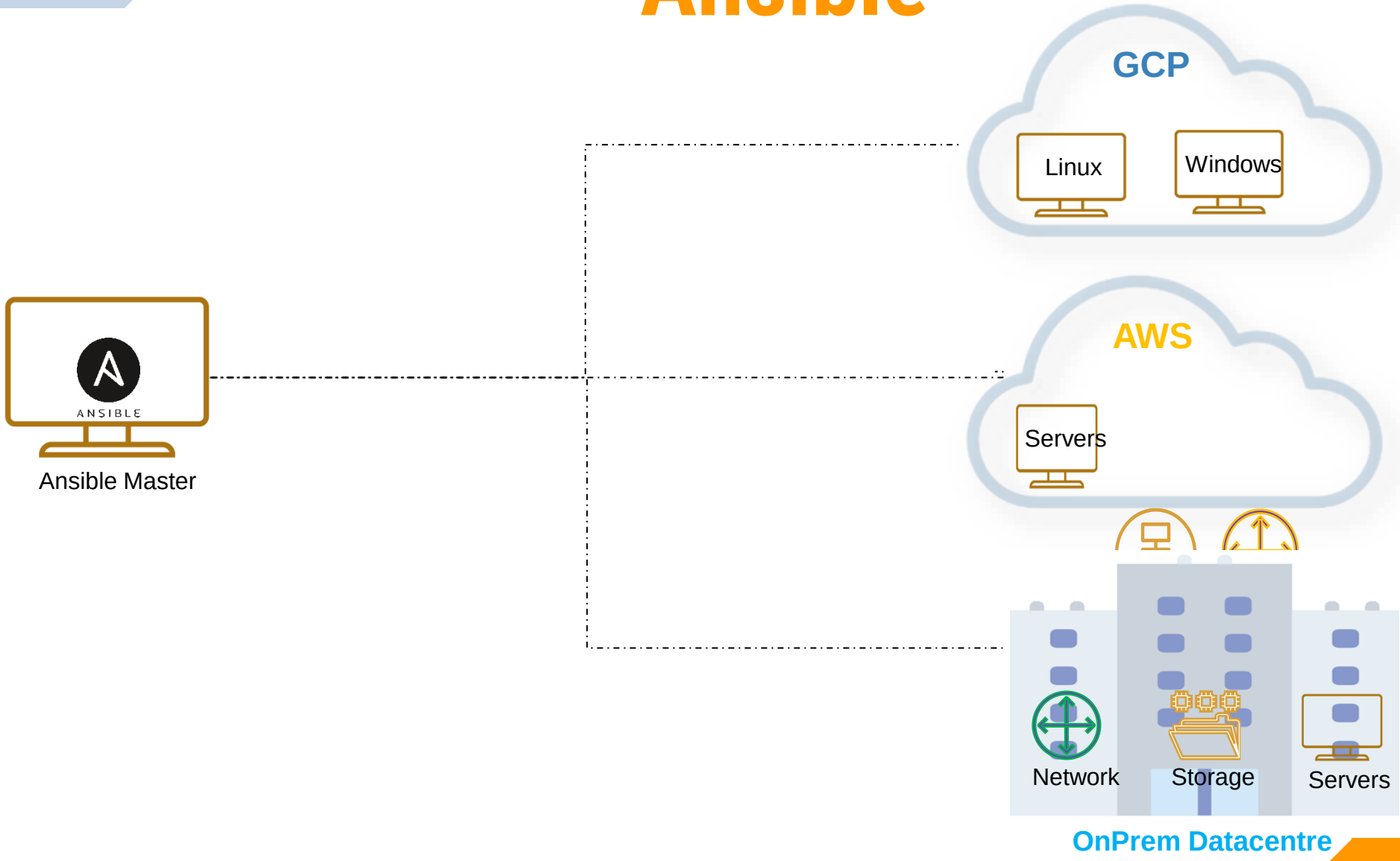
Network



Storage

What is Configuration Management?

Ansible

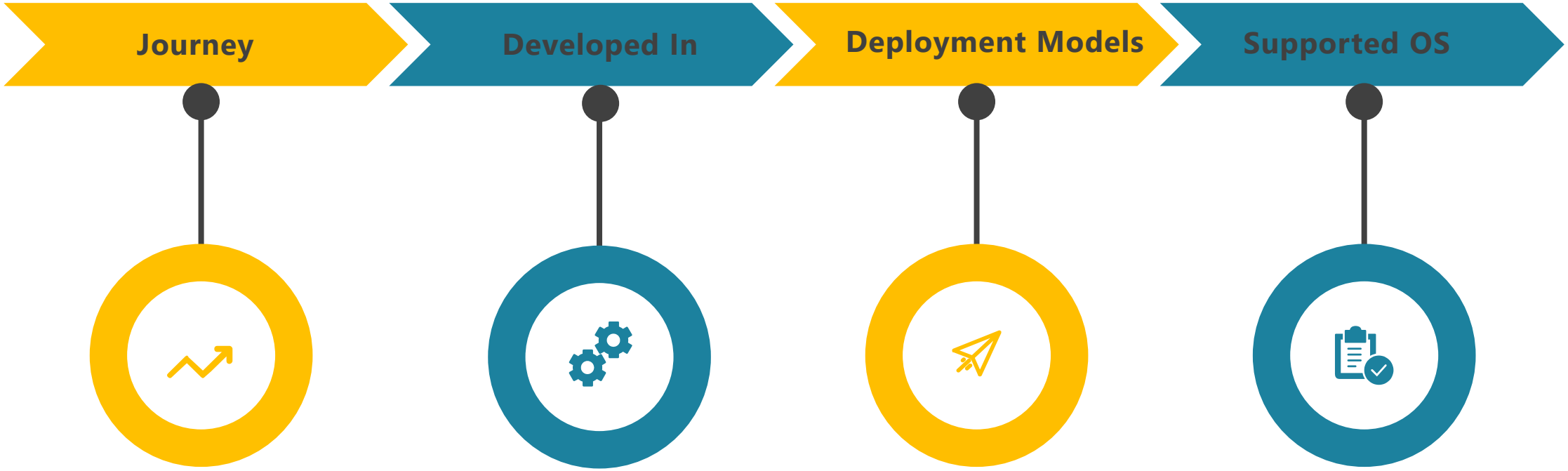


Ansible

Ansible is an easy-to-use IT Automation, Configuration Management & Orchestration Software for System Administrators & DevOps Engineers.

- Founded in Feb, 2012
- First commercial product release in 2012
- Multiple in-built functional modules
- Multiple Community Members
- 40,000+ Users
- 50,000+ Nodes managed in the largest deployments
- Support for Red Hat, CentOS, Ubuntu, Oracle Linux, MAC, OS, Solaris 10/11, Windows.
- Ansible Controller node Supported on Linux variants only

Ansible Introduction



Michael DeHaan – Feb 2012

Redhat Acquisition – Oct 2015

Python

Require Python as pre-requisite

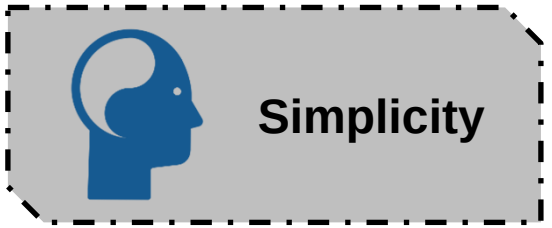
Push Based Architecture

Agentless

Master Node – Linux

Managed Node – Unix, Linux and Windows

Why Ansible?



Current IT Automation State

Manually Configure: Literally logging into every node to configure it.

Golden Images: Creating a single copy of a node's software and replicating that across nodes.

Custom One-off Scripts: Custom code written to address a specific, tactical problem.

Software Packages: Typically all or nothing approach.

Current IT Automation State

- **Manually Configure:**
  - Difficult to scale.
  - Impossible, for all intents and purposes, to maintain consistency from node-to-node.

Current IT Automation State

- **Golden Images:**
  - Need separate images for different deployment environments, e.g. development, QA, production, or different geo locations.
  - As number of images multiply it becomes very difficult to keep track and keep consistent.
  - Since they're monolithic copies, golden images are rigid and thus difficult to update as the business needs change.

Current IT Automation State

- **Custom One-off Scripts:**
  - No leverage – effort typically cannot be reused for different applications or deployments.
  - Brittle – as needs change, often the entire script must be re-written.
  - Difficult to maintain when the original author leaves the organization.

Current IT Automation State

- **Software Packages:**
  - These packages typically require that all resources be placed under management – cannot selectively adopt and scale automation.
  - As a result, longer deployments times.
  - Dated technology developed before virtualization and cloud computing – lacks responsiveness to changing requirements.

Why Configuration Management?

- To provide optimized level of automated way to configure Applications and Software's inside your system.
- Enable you to Discover, Provision, Configure and Manage the systems.
- Developers should be able to use a single command to build and test software in minutes or even in seconds.
- Maintaining configuration state of all systems simultaneously should be easy.
- Login into every client machine for CM tasks should not be mandate.
- It should be easy to maintain desired state as per policy

Why Orchestration with Ansible?

- A single tool for deployment and Configuration management
- Easy to manage and use
- Compatible with all major cloud service providers
- Can Orchestrate Infrastructure and Software both

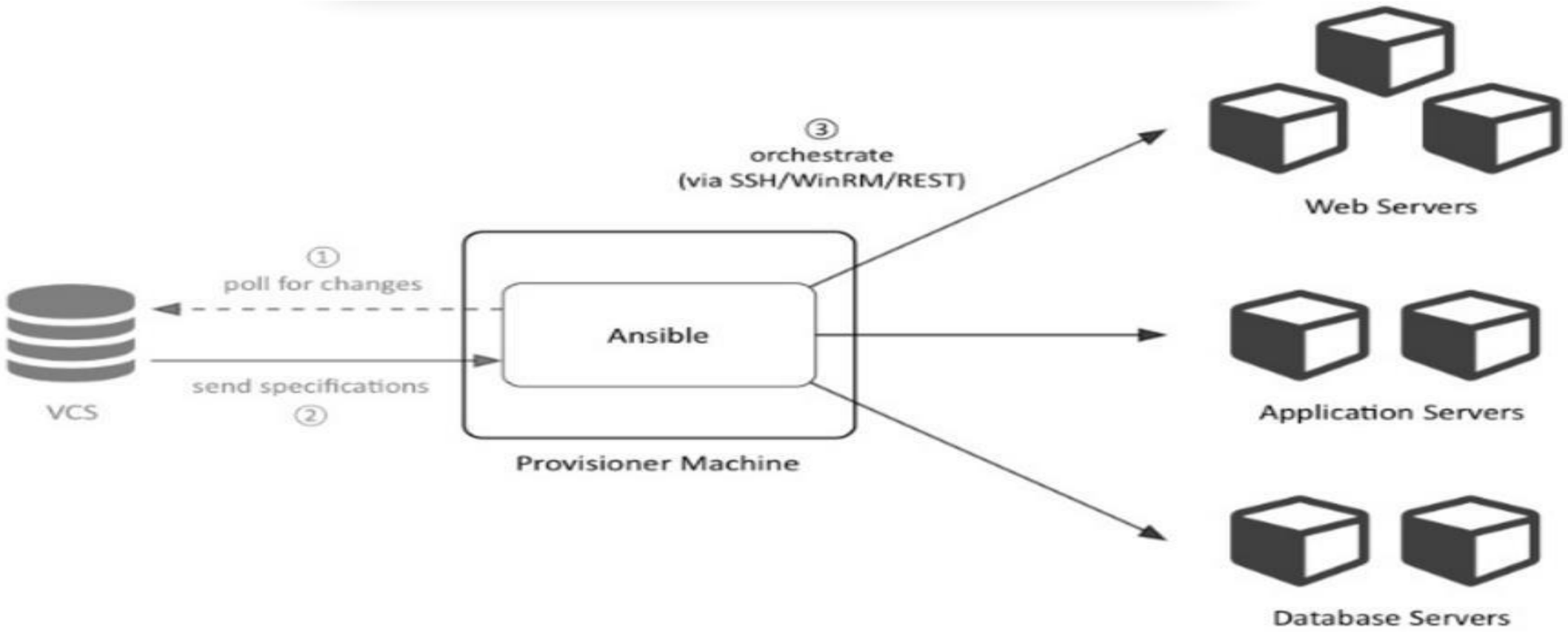
Ansible Components

- Ansible consists of **Agentless Model** and majorly have two parts:

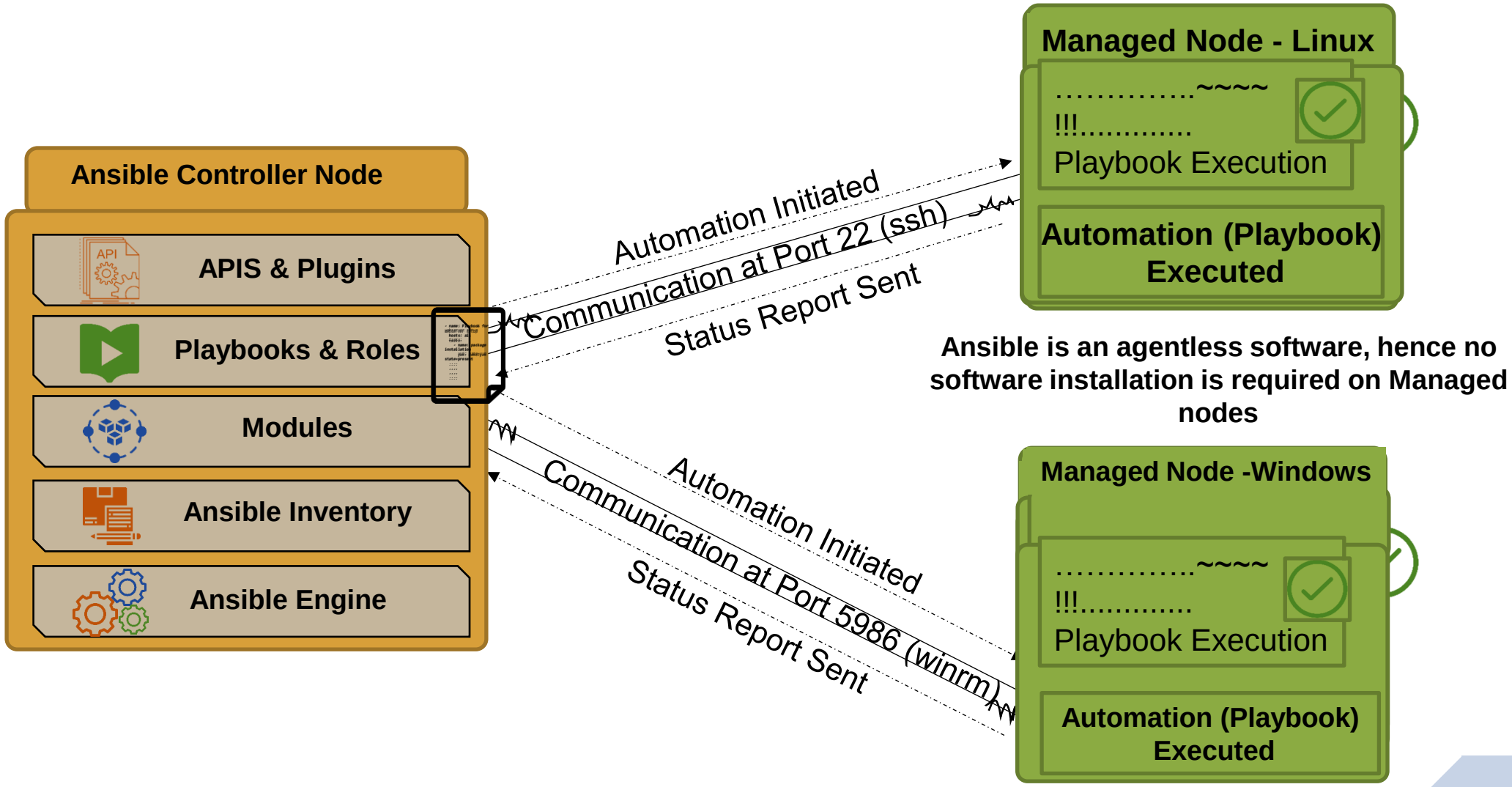
  - **Controller / Master**: The central configuration server where we will have our all configurations stored.
  - **Managed Nodes / Clients**: All clients getting configured from Ansible Master.

- **Note**:
- Ansible Master can be run from any Linux machine (Windows not supported) with Python 2 (version 2.7) or Python 3 (versions 3.5 and higher) installed.
- On the managed nodes, you need a way to communicate, which is normally SSH. By default this uses SFTP. If that's not available, you can switch to SCP in ansible.cfg. You also need Python 2 (version 2.6 or later) or Python 3 (version 3.5 or later).

Dataflow



Ansible Architecture



Ansible and its Peers

Many tools available in Market. Few things to consider, before selecting any tool:

- Configuration Management vs Orchestration
- Mutable Infrastructure vs Immutable Infrastructure
- Procedural vs Declarative
- Client/Server Architecture vs Client-Only Architecture

Ansible and its Peers

	Chef	Puppet	Ansible	SaltStack	CloudFormation	Terraform
Code	Open source	Open source	Open source	Open source	Closed source	Open source
Cloud	All	All	All	All	AWS only	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable	Immutable
Language	Procedural	Declarative	Declarative	Declarative	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client-Only	Client/Server	Client-Only	Client-Only

Knowledge Checks

- What is Configuration Management?
- List a few available configuration Management tools.
- What are the Advantages of Ansible?
- Explain Data flow of Ansible.

Ansible Installation

Installation of Ansible

- The Ansible **master** is the machine that controls the infrastructure and dictates policies for the servers it manages.
- Currently Ansible can be run from any machine with Python 2.6 or 2.7 installed (Windows isn't supported for the control machine).
- This includes Red Hat, Ubuntu, Debian, CentOS, OS X, any of the BSDs, and so on.

Lab1: Installation of Ansible

- To install the Ansible Master, we need to install EPEL repository package:
  - Ansible Repository

  http://fedoraproject.org/wiki/EPEL

  Note: If internet connectivity is there just do:

  - wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

- Pre-installation
- Assign a hostname to your machine(Master) and make that name persist across reboot.

Lab1: Installation of Ansible

- yum install ansible
- rpm –qa | grep –i ansible
- ansible --version
- Default Configuration file is /etc/ansible/ansible.cfg
- Default Inventory file is /etc/ansible/hosts

Ansible Master Configuration

- Edit /etc/ansible/ansible.cfg for any Master Configurations
- Default options are fine
- All parameters can be overridden in ansible-playbook or with command line flags.

Special Shortcut: cat /etc/ansible/ansible.cfg | grep "^\["

Ansible Master Configuration

- Ansible comes with a default Ansible configuration file which can be customized by changing Ansible configuration parameters.
- **/etc/ansible/ansible.cfg** – by default base configuration file location.
- **~/.ansible.cfg** - user specific configuration file, this configuration file will be used by Ansible if Ansible is executed by logged in user.
- **./ansible.cfg** - the precedence will be given to this file, if Ansible run is executed from the directory path where ansible.cfg file is present.
- **ANSIBLE_CONFIG** - configuration file location defined by an environment variable.

Lab2: Working with Ansible.cfg

- Run Ansible --version command and check the outcome of “config file”.

- Copy /etc/ansible/ansible.cfg file into your home directory as below:
  - cp /etc/ansible/ansible.cfg ~/.ansible.cfg
- Run Ansible --version command and check the outcome of “config file”.

- Switch to /tmp and Copy /etc/ansible/ansible.cfg file into your Current directory as below:
  - cp /etc/ansible/ansible.cfg /tmp/ansible.cfg
- Run Ansible --version command and check the outcome of “config file”.

- Export ansible Config file into variable **ANSIBLE_CONFIG** as below:
  - cp /etc/ansible/ansible.cfg /ansible.cfg
  - export ANSIBLE_CONFIG=“/ansible.cfg”
- Run Ansible --version command and check the outcome of “config file”.

Ansible Authentication

- As Ansible is using SSH by default during communication, this communication connection supports both:

- **Password Based Authentication:** Password based authentication is acceptable if your environment is small and easily manageable. But it become very difficult to work with password-based authentications once you scale your environment. Password based authentication is only useful in Engineering Labs or Test Labs or Playbook creations tests.

- **Key Based Authentication:** Key based Authentication is adopted in Enterprise Environments. Here we create one generic user and amend the keys of the generic user in Managed Nodes for Key Based - Password less Authentication. This is a onetime task and can be used with any number of servers.

Ansible Inventory

- Ansible Inventory is a text-based list of individual servers and/or group of multiple servers.
- By default the Ansible Inventory location is **"/etc/ansible/hosts"**.
- You may have multiple Inventory files.
- Ansible Inventory can have Host Name or IP Address or Combination of both.
- Ansible provides the flexibility to pull inventory from Dynamic or Cloud sources with the help of scripts.
- You can specify a different inventory file using the -i <path> option on the command line.

Ansible Fundamentals

Case Study - 1

You need to manage a user, .

You care specifically about:

- his existence
- his primary group
- his home directory

Case Study - 1

- Tools built into most distro’s that can help:

- useradd
- usermod
- groupadd
- groupmod
- mkdir
- chmod
- chgrp
- chown

Case Study - 1

- Platform idiosyncrasies:
  - Does this box have ‘useradd’ or ‘adduser’?
  - What was that flag again?
  - What is difference between ‘–l’ and ‘–L’?
  - What does ‘–r’ means
  - Recursive
  - Remove read privileges
  - System user

- If I run this command again, what will it do?

Case Study - 1

- You could do something like this:

```
#!/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
```

Case Study - 1

What About?

- Robust error checking?
- Solaris and Windows support?
- Robust logging of changes?
- Readable code?
- What if need to create in 1000+ Servers?

Case Study – 1

Ansible Way of Configuration Management:

```
tasks:
 - name: Creating Gagandeep User
    user: name=gagandeep comment="Gagandeep Singh"  state=present

tasks:
 - name: Creating Singh Group
    group: name=singh state=present
```

Ansible Way: Maintaining State

- You(Even Ansible can do it on cloud) provision a node.
- Ansible configures it.
- Ansible maintains the desired state when needed.

Note: You make to sure the state is configured as per environment requirements.

Ansible : Infrastructure as Code

- Descriptive
- Straightforward
- Transparent

```
[root@gagan]# cat ntp.yml
---
# This is my Host section
- hosts: localhost
# This is my Task section
  tasks:
  - name: NTP Installation
    yum: name=ntp state=present
  - name: NTP Service
    service: name=ntpd state=started enabled=yes
```

Ansible : Idempotency

- Ansible enforces in an idempotent way.
- The property of certain operations in mathematics or computer science is that they can be applied multiple times without further changing the result beyond the initial application.
- Able to be applied multiple times with the same outcome.

Ansible Terminology

- **Controller/Master:** The Ansible master is the machine that controls the infrastructure and dictates policies for the servers it manages. It operates both as a repository for configuration data and as the control center that initiates remote commands and ensures the state of your other machines.

- **Managed/Agent Nodes :** The servers that Ansible configures are called Clients/Nodes.

- **Ansible Inventory:** Ansible Inventory represents which machines it should manage using a very simple INI file that puts all of your managed machines in groups of your own choosing.

- **Ansible Adhoc-tasks:** Ansible uses adhoc requests to confirm simple and small tasks on any server right a way without login into the client. The best example is to check the Alive Status for whole managed inventory.

Ansible Terminology

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.

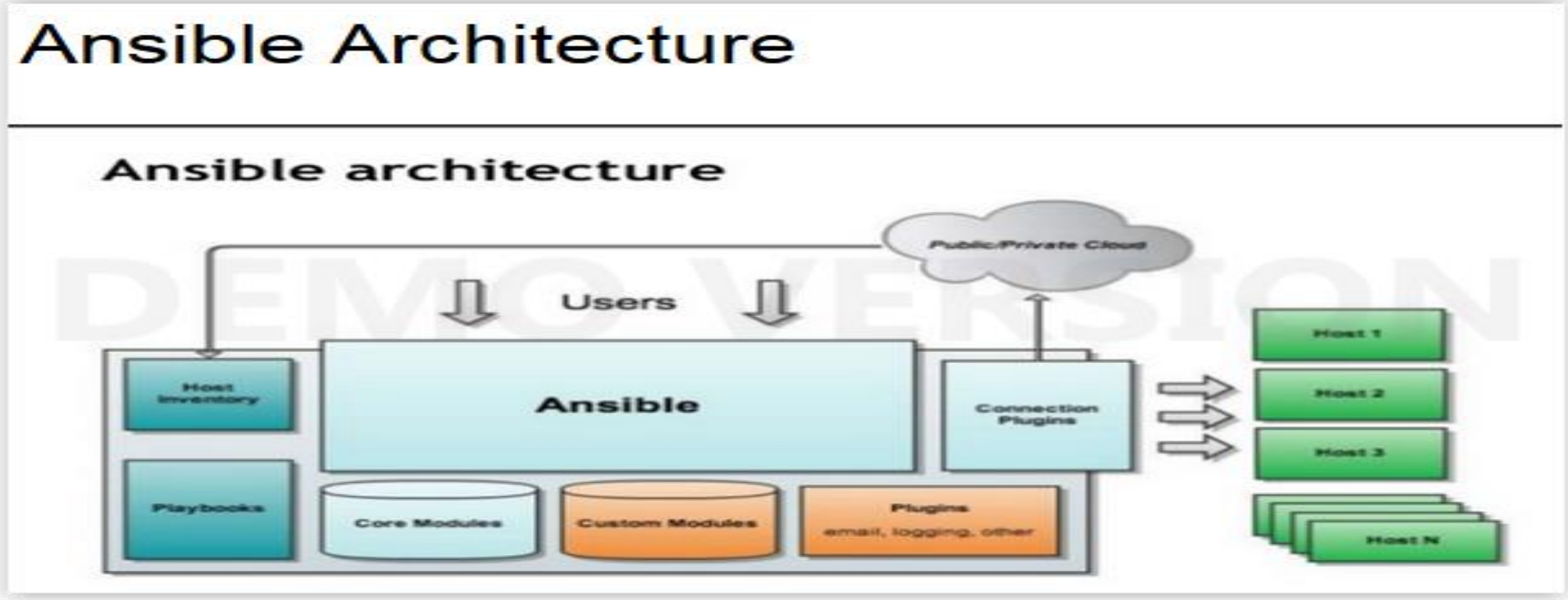
Ansible Terminology

- **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 Communications



Ansible Communication test

- Communication checks with password authentications:

```
[root@gagan-controller ~]#ansible demo -m ping --ask-pass
SSH password:
centos-managed | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Lab 3: Ansible configuration

- Edit Client IP and Name entry in /etc/hosts file for Name to IP Mapping:

  172.31.19.138 gagan-client // /etc/hosts
- Put client name entry in in /etc/ansible/hosts file (under unmanaged section/top) , so that it can be managed with ansible:

  gagan-client // /etc/ansible/hosts
- Run ansible commands to check if client host is manageable with ansible or not:

  ansible gagan-client –m ping --ask-pass // ask-pass as we have not provided password in file
- Provide username and password in /etc/ansible/hosts file and run command without password:

  gagan-client ansible_user=USER ansible_password=PASSWORD // /etc/ansible/hosts
- Generate SSH keypair and copy keypair in remote machine for passwordless connection:

  ssh-keygen –t rsa // press enter thrice after this – no passphrase

  ssh-copy-id gagan-client // hit enter and provide password for next machine

  ansible gagan-client –m ping

  ansible-doc -l

Ansible Modules

- Modules are the Basic Building Block of Ansible.
- These are the readymade tools to perform various tasks and operations on “Managed Nodes”.
- Modules can be used with Ansible Ad-Hoc Remote Executions and/or Playbooks as a core building blocks.
- Ansible Ships with multiple in-built Modules (approx. 2000+ in Ansible 2.7).
- Can be used for Standalone servers, Virtual Machines and for any Public/Private Cloud Instances.

Ansible Modules

- Two types of Ansible Modules: Core Modules & Custom Modules.
- Robust Module Documentation on website.
- Command line utility on Module information and usage.
- CLI utility ansible-doc on “Controller Node”.
- Execute ansible-doc -l to list all available Modules.
- Execute ansible-doc <module-name> to find all details about Modules.
- For GUI refer **“http://docs.ansible.com/ansible/modules_by_category.html”**

Ansible Adhoc Execution

- Easy to learn ad-hoc command line utility - “ansible”.
- Quick On-demand tasks on “Managed Nodes”.
- 1 to 1 approach, single ad-hoc command is used to perform single operation.
- Multiple ad-hoc operations require multiple “ansible” ad-hoc run.
- Ad-hoc execution syntax.
- Ansible Ping Communication Test with “Managed Nodes”.
- Various real time examples with ad-hoc execution.

Ansible Adhoc Execution

- Let’s try executing a remote command, before that make sure you have out an entry of the host in “/etc/ansible/hosts”

- Connect to the master and type:

ansible <host-name/IP> -m ping --ask-pass

ansible "*" -m ping --ask-pass

- First argument = target client
- Second argument = function to execute
- Other arguments = params for the function

Ansible Adhoc Execution

Adding Username and Connection method in “/etc/ansible/hosts”:

You can specify each host for specific connection type/port and connection username:

- ansible_host : The name of the host to connect to, if different from the alias you wish to give
- ansible_port: The ssh port number, if not 22
- ansible_user: The default ssh user name to use.
- ansible_ssh_pass: The ssh password to use (never store this variable in plain text; always use a vault. See Variables and Vaults)
- ansible_ssh_private_key_file: Private key file used by ssh. Useful if using multiple keys and you don’t want to use SSH agent.

user20-client ansible_connection=ssh ansible_user=centos

Ansible Adhoc Execution

- There are a bunch of predefined :

«**Ad-Hocmodules**»

«**execution modules**»

«**Ad-Hocmodules**»: ansible all/"Client or Group“ -a “<adhoc-command>" --ask-pass

«**execution modules**» ansible all/"Client or Group“ -m <module_name> -a <arguments>

- Note: To list all Ansible Modules run below command:

ansible-doc -l

Ansible Adhoc Execution

- For example, executing a shell commands:

ansible 192.168.74.51 -a "ls -l /tmp" --ask-pass

ansible 192.168.74.51 -a "uname -a" --ask-pass

ansible 192.168.74.51 -a "cat /etc/redhat-release" --ask-pass

ansible 192.168.74.51 -a 'service ntpd status' --ask-pass

ansible “*” --list-hosts // Its’ll show all the hosts that’ll effect with the command