Ansible adhoc command

- An *ad hoc command* is a way of executing a single Ansible task quickly, one that you do not need to save to run again later.
- They are simple, online operations that can be run without writing a playbook.
- You could use another ad hoc command to efficiently restart a service on many different machines, or to ensure that a particular software package is up-to-date.
- Ad hoc commands are used to execute quick one liners.
- They are useful for non-routine tasks.
- Execute them using the ansible command.
- Arguments are require double quotes and are space delimited
- Commands are executed as the user running Ansible.

Ad Hoc Command

 # ansible [-i inventory] host-pattern -m module [-a 'module arguments']

- The *host-pattern* argument is used to specify the managed hosts on which the ad hoc command should be run.
- The **-i** option to specify a different inventory location to use than the default in the current Ansible configuration file.
- The **-a** option takes a list of those arguments as a quoted string.
- The **-m** option takes as an argument the name of the *module* that Ansible should run on the targeted hosts.
- One of the simplest ad hoc commands uses the ping module. This module does not do an ICMP ping, but checks to see if you can run Python-based modules on managed hosts.
- *$ ansible all -m ping*

Working with Ad Hoc command

- Modules are the tools that ad hoc commands use to accomplish tasks. Ansible provides hundreds of modules which do different things.
- The **ansible-doc -l** command lists all modules installed on a system.
- The **ansible-doc** command will give you to view the documentation of particular modules by name, and find information about what arguments the modules take as options.
- *$ ansible-doc ping*

Ansible Module

- Modules are essentially tools for particular tasks.
- Modules can take, and usually do take, parameters.
- Module return JSON.
- Run modules from the command line or within a playbook.
- Ansible ships with a significant amount of modules by default.
- Custom module can be written by using python.

Module Category	Module	Description
Files Module	copy	Copy a local file to the managed host
	file	Set permissions and other properties of files
	lineinfile	Ensure a particular line is or is not in a file
	synchronize	Synchronize content using rsync
Software Package Module	package	Manage packages using autodetected package manager native to the operating system
	yum	Manage packages using the YUM package manager
	apt	Manage packages using the APT package manager
	dnf	Manage packages using the DNF package manager
	gem	Manage Ruby gems
	pip	Manage Python packages from PyPI
System modules	firewalld	Manage arbitrary ports and services using firewalld
	reboot	Reboot a machine
	service	Manage services
	user	Add, remove, and manage user accounts
Net Tools modules	get_url	Download files over HTTP, HTTPS, or FTP
	nmcli	Manage networking
	uri	Interact with web services

$ ansible server1 -m user -a 'name=user1 uid=4000 state=present'

Running Arbitrary Commands on Managed Hosts

- The command module allows administrators to run arbitrary commands on the command line of managed hosts.
- The command module allows administrators to quickly execute remote commands on managed hosts.
- These commands are not processed by the shell on the managed hosts. As such, they cannot access shell environment variables or perform shell operations, such as redirection and piping.
- For situations where commands require shell processing, administrators can use the shell module.
- Ansible then executes the command remotely on the managed hosts. Unlike the command module, the commands are processed through a shell on the

managed hosts. Therefore, shell environment variables are accessible and shell operations such as redirection and piping are also available for use.

- The command to be run is specified as an argument to the module using the **-**a option.
- The -o option can be used for view the output in single line
- *$ ansible db-servers -m command -a /usr/bin/hostname*
- *$ ansible db-servers -m command -a /usr/bin/hostname*
- *$ ansible websrv –m shell -a "ls -l /tmp"*
- Both command and shell modules require a working Python installation on the managed host.

Note: If an ad hoc command does not specify which module to use with the -m option, Red Hat Ansible Engine uses the command module by default.

- A third module, is raw module, can run commands directly using the remote shell, bypassing the module subsystem.
- This is useful when managing systems that cannot have Python installed.

CONFIGURATION FILE DIRECTIVES	COMMAND-LINE OPTION	Description
inventory	-i	Specified the inventory file
remote_user	-u	Specified the remote user
become	-b, --become	Become sudo privileges
become_method	--become-method	Specifies how to escalate privileges.
become_user	--become-user	Specifies which user to escalate to, but the default is root.
become_ask_pass	--ask-become-pass, -K	Ask sudo password

.

Examples:

$ pwd
$ ansible-doc –l | wc –l
$ ansible all –list-hosts
$ cat inventory/inv.ini
$ ansible -i inv.ini all -m ping
$ ansible -i inv.ini all -m setup
$ ansible db-servers -m setup | head –n 20
$ ansible websrv -m copy -a "src=/home/ansadmin/sec.txt dest=/tmp"
$ ansible websrv -m copy -a "src=/home/ansadmin/sec.txt dest=/tmp mode=0644 owner=ansadmin group=ansadmin"
$ ansible office -m lineinfile -a"dest=/tmp/secret.txt line='This Line is added by ansible controller\n' " -b
$ ansible websrv –m copy –a "content='Hello this ansible engine' dest=/etc/motd"
$ ansible websrv –m command –a "cat /etc/motd"

$ ansible websrv –m file –a “path=/home/ansadmin/demo state=directory owner=ansadmin group=ansadmin mode=0755”
$ ansible websrv –m user –a “name=tom”
$ ansible webserver –m yum –a “name httpd state=present”
$ ansible webserver –m service –a “name=httpd state=started enable=yes
$ absible websrv -m fetch -a "src=ansible_node_path dest=/ansible_eng._path"
$ absible websrv -m fetch -a "src=ansible_node_path dest=/ansible_eng._path flat=yes"
$ absible websrv -m fetch -a "src=ansible_node_path dest=/home/ansadmin/newdir/{{inventory_hostname}}_file.txt flat=yes"

```
Identifying the push model:
$ ANSIBLE_KEEP_REMOTE_FILES=1 ansible all –m shell –a “uptime”
$ tree .ansible
$ cd .ansible/tmp
$ ls
$ cd ansible-tmp-****
$ ls
$ cat filename.py
```