Ansible ad-hoc Commands

Ad-hoc commands are one of the simplest ways of using Ansible. These are used when you want to issue some commands on a server or bunch of servers. The ad-hoc commands are not stored for future use, but it represents a fast way to interact with the desired servers.

The Ansible ad-hoc command uses the **/usr/bin/ansible** command-line tool to automate a single task on one or more managed nodes. The Ad-hoc commands are quick and easy, but they are not re-usable. The Ad-hoc commands demonstrate the simplicity and power of Ansible.

Syntax

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

Explanation

Hosts: It can be an entry in the inventory file. For specifying all hosts in the inventory, use all or "*".

module_name: It is an optional parameter. There are hundreds of modules available in the Ansible, such as **shell, yum, apt, file,** and **copy**. By default, it is the **command**.

Arguments: We should pass values that are required by the module. It can change according to the module used.

Username: It specifies the user account in which Ansible can execute commands.

Become: It's an optional parameter specified when we want to run operations that need sudo privilege. By default, it becomes false.

1. Parallelism and shell commands

You can reboot your company server in 12 parallel forks at the same time. For this, you need to set up the SSHagent for connection.

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
```

To run reboot for all your company servers in the group, 'abc', in 12 parallel forks:

```
$ ansible abc -a "/sbin/reboot" -f 12
```

By default, Ansible will run the above ad-hoc commands from the current user account. If you want to change then pass the username in ad-hoc command as follows:

```
$ ansible abc -a "/sbin/reboot" -f 12 -u username
```

2. File Transfer

You can use ad-hoc commands for doing SCP (secure copy protocol) which means lots of files in parallel on multiple machines or servers.

Transferring file on many machines or servers

```
$ ansible abc -m copy -a "src = /etc/yum.conf dest = /tmp/yum.conf"
```

Creating new directory

```
$ ansible abc -m file -
a "dest = /path/user1/new mode = 888 owner = user1 group = user1 state = directory"
```

Deleting all directory and files

```
$ ansible abc -m file -a "dest = /path/user1/new state = absent"
```

3. Managing Packages

Ad-hoc commands are available for apt and yum module. Here are the following ad-hoc commands using yum.

Below command checks, if the yum package is installed or not, but not update it.

```
$ ansible abc -m yum -a "name = demo-tomcat-1 state = present"
```

Below command checks the package is not installed.

```
$ ansible abc -m yum -a "name = demo-tomcat-1 state = absent"
```

And below command checks the latest version of package is installed.

```
$ ansible abc -m yum -a "name = demo-tomcat-1 state = latest"
```

4. anaging Users and Groups

You can manage, create, and remove a user account on your managed nodes with ad-hoc commands.

```
$ ansible all -m user -a "name=foo password=<crypted password here>"

$ ansible all -m user -a "name=foo state=absent"
```

5. Managing Services

Ensure a service is started on all the webservers.

```
$ ansible webservers -m service -a "name=httpd state=started"
```

Alternatively, restart a service on all webservers:

```
$ ansible webservers -m service -a "name=httpd state=restarted"
```

Ensure a service is stopped:

```
$ ansible webservers -m service -a "name=httpd state=stopped"
```

6. Gathering Facts

Fact represents the discovered variables about a system. You can use the facts to implement conditional execution of tasks, and also used to get ad-hoc information about your systems. To see all the facts:

```
$ ansible all -m setup
```