Running Ad Hoc Commands

Performing Ad Hoc Commands with Ansible

An *ad hoc command* is a way to execute a single Ansible task quickly, one that you don't need to save to run again later. They're simple, one-line operations that can be run without writing a playbook.
They're useful for quick tests and changes. For example, you can use an ad hoc command to make sure a certain line exists in the **/etc/hosts** file on a group of servers. You could use another to efficiently restart a service on many different machines or ensure that a particular software package is up-to-date. You could also use it to run an arbitrary command on one or more hosts to run a program or collect information. Ad hoc commands are a very useful tool to quickly perform simple tasks with Ansible. They do have their limits, and in general you'll want to use Ansible Playbooks to realize the full power of Ansible. In many situations, however, ad hoc commands are exactly the tool you need to do something simple quickly.

Create inventory file: -

#cd /home/ansible/playbook

#cat inventory

```
[ansible@robo playbook]$ cat inventory
[local]
robo

[dev]
robo2

[everyone:children]
robo
robo2
```

Running Ad Hoc Commands

#ansible all -m ping

```
[ansible@robo playbook]$ ansible all -m ping
SUDO password:
robo | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
robo2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Performing Tasks with Modules in Ad Hoc Commands

Modules are the tools that ad hoc commands use to accomplish tasks. Ansible provides hundreds of modules which do different things. You can usually find a tested, special-purpose module that does what you need as part of the standard installation. The **ansible-doc -l** command lists all the modules that are installed on the system. You can then use **ansible-doc** to view the documentation of modules by name and find information about what arguments the modules take as options. For example, the following command displays the documentation for the **ping** module, which has no options:

#ansible-doc ping

For example, the following ad hoc command uses the **user** module to make sure that the **newbie** user exists and has UID 4000 on **robo2**:

#ansible -m user -a 'name=newbie uid=4000 state=present' robo2 -b

```
robo2 | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 4000,
    "home": "/home/newbie",
    "name": "newbie",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 4000
}
```

#ansible robo2 -a 'id newbie'

```
robo2 | CHANGED | rc=0 >>
uid=4000(newbie) gid=4000(newbie) groups=4000(newbie)
```

ansible dev -m command -a /usr/bin/hostname

```
robo2 | CHANGED | rc=0 >>
robo2
```

ansible local -m shell -a set

```
robo | CHANGED | rc=0 >>
BASH=/bin/sh
BASHOPTS=cmdhist:extquote:force_fignore:hostcomplete:interactive_comments:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_EXECUTION_STRING=set
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="2" [2]="46" [3]="2" [4]="release" [5]="x86_64-redhat-linux-gnu")
BASH_VERSION='4.2.46(2)-release'
DIRSTACK=()
EUID=1000
GROUPS=()
HOME=/home/ansible
HOSTNAME=robo
HOSTTYPE=x86_64
```

...output omitted...

Important

In most circumstances, it is a recommended practice that you avoid the **command**, **shell**, and **raw** "run command" modules. Most other modules are idempotent and can perform change tracking automatically. They can test the state of systems and do nothing if those systems are already in the correct state. By contrast, it's much more complicated to use the "run command" modules in a way which will be idempotent. Depending on them might make it harder for you to be confident that re-running an ad hoc command or playbook won't cause an unexpected failure.

There are times when the "run command" modules are valuable tools and a good solution to a problem. If you do need to use them, it's probably best to try to use **command** first, resorting to **shell** or **raw** only if you need their special features.

Ansible Command-line Options

Setting	Command-line option
inventory	-i
remote_user	-u
become	--become, -b
become_method	--become-method
become_user	--become-user
become_ask_pass	--ask-become-pass, -K

References

ansible(1) man page
Patterns: Ansible Documentation
http://docs.ansible.com/ansible/intro_patterns.html
Introduction to Ad-Hoc Commands: Ansible Documentation
http://docs.ansible.com/ansible/intro_adhoc.html
Module Index: Ansible Documentation
http://docs.ansible.com/ansible/modules_by_category
command - Executes a command on a remote node: Ansible Documentation
http://docs.ansible.com/ansible/command_module.html
shell - Execute commands in nodes: Ansible Documentation
http://docs.ansible.com/ansible/shell_module.html

Guided Exercise: Running Ad Hoc Commands

Create a dep-adhoc directory
#cd /home/ansible/playbook/dep-adhoc
ansible everyone -m ping
ansible local -m command -a 'id'

```
[ansible@robo playbook]$ ansible local -m command -a 'id'
SUDO password:
robo | CHANGED | rc=0 >>
uid=1000(ansible) gid=1000(ansible) groups=1000(ansible),10(wheel)
```

#ansible dev -m command -a 'cat /etc/motd' -u ansible

```
[ansible@robo playbook]$ ansible dev -m command -a 'cat /etc/motd' -u ansible
SUDO password:
robo2 | CHANGED | rc=0 >>
This is the system robo2.
Today's date is: 2019-04-18.
Only use this system with permission.
You can ask deepan.redhat@gmail.com for access.
```

ansible dev -m copy -a 'content="Managed by ansible" dest=/etc/motd' -b

```
[ansible@robo playbook]$ ansible dev -m copy -a 'content="Managed by ansible" dest=/etc/motd' -b
SUDO password:
robo2 | CHANGED => {
    "changed": true,
    "checksum": "182e35a0ebf73d6e38ef5d41aa4c757447d987f4",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "a54d99460d23116c3591009d85cb6cf4",
    "mode": "0644",
    "owner": "root",
    "size": 18,
    "src": "/home/ansible/.ansible/tmp/ansible-tmp-1555979183.56-175034359536370/source",
    "state": "file",
    "uid": 0
}
```

Guided Exercise: Writing and Running Playbooks

In your directory, use a text editor to create a playbook named **site.yml**. This playbook contains one play, which should target members of the **dev** host group. The playbook should use tasks to ensure that the following conditions are met on the managed hosts:
1. The *httpd* package is present, using the **yum** module.
2. The local **files/index.html** file is copied to **/var/www/html/index.html** on each managed host, using the **copy** module.
3. The **httpd** service is started and enabled, using the **service** module.
You can use the **ansible-doc** command to help you understand the directives needed for each of the modules.

After the playbook is written, verify its syntax and then use **ansible-playbook** to run the playbook to implement the configuration.

Steps
To make all playbook exercises easier, if you use the Vim text editor you may want to use it to edit your **~/.vimrc** file (create it if necessary), to ensure it contains the following line:
Only the space character can be used for indentation; tab characters are not allowed. If you use the Vim text editor, you can apply some settings which might make it easier to edit your playbooks. For example, by adding the following line to your **$HOME/.vimrc** file, when **vim** detects that you're editing a YAML file, it will perform a two space indentation when the **Tab** key is pressed, will autoindent subsequent lines, and will expand tabs into spaces.

#echo "autocmd FileType yaml setlocal ai ts=2 sw=2 et" >>/home/ansible/.vimrc

Create a basic-playbook directory and index.html file.
#cd /home/ansible/playbook/basic-playbook
#cat index.html

```
[ansible@robo basic-playbook]$ cat index.html
This is anible node
```

cat site.yml

```
---
- name: Install and start the web apache httpd
  hosts: dev
  become: yes
  tasks:
    - name: install the pkg
      yum: name=httpd state=present
    - name: copy index.html file
      copy: src=index.html dest=/var/www/html/index.html
    - name: start the service and enable it
      service: name=httpd state=started enabled=true
```

ansible-playbook --syntax-check site.yml
ansible-playbook site.yml

```
[ansible@robo basic-playbook]$ ansible-playbook site.yml
SUDO password:

PLAY [Install and start the web apache httpd] *********************************

TASK [Gathering Facts] ********************************************************
ok: [robo2]

TASK [install the pkg] ********************************************************
changed: [robo2]

TASK [copy index.html file] ***************************************************
changed: [robo2]

TASK [start the service and enable it] ****************************************
changed: [robo2]

PLAY RECAP ********************************************************************
robo2                      : ok=4    changed=3    unreachable=0    failed=0
```

curl http://robo2

```
[ansible@robo basic-playbook]$ curl http://robo2
This is anible node
```

Note:- Please clean up once playbook executed successfully.

Guided Exercise: Implementing Multiple Plays

In this directory, create a playbook named **intranet.yml** which contains two plays. The first play requires privilege escalation and must perform the following tasks in the specified order:
1. Use the **yum** module to ensure that the latest versions of the *httpd* and *firewalld* packages are installed.
2. Ensure the **firewalld** service is enabled and started.
3. Ensure that **firewalld** is configured to allow connections to the **httpd** service.
4. Ensure that the **httpd** service is enabled and started.
5. Ensure that the managed host's **/var/www/html/index.html** file consists of the content **"Welcome to the robo2 intranet!"**.
The second play does not require privilege escalation and should run a single task using the **uri** module to confirm that the URL **http://robo2** returns an HTTP status code of 200.

Create a multiple-playbook directory
#cd /home/ansible/playbook/multiple-playbook

```
[ansible@robo multiple-playbook]$ cat index.html
Welcome to the robo2 intranet!
```

#cat intranet.yml

```
---
- name: Enable the intranet service
  hosts: dev
  become: yes
  tasks:
    - name: Install httpd and firewalld pkg
      yum:
        name:
          - httpd
          - firewalld
        state: latest
    - name: Firewalld enabled and running
      service: name=firewalld enabled=true state=started

    - name: Firewall permits the http service
      firewalld:
        service:  http
        permanent:  true
        state:  enabled
        immediate:  yes

    - name: httpd enabled and running
      service: name=httpd enabled=true state=started

    - name: test html page is installed
      copy: content="Welcome to the robo2 intranet!\n" dest=/var/www/html/index.html

- name: Test the intranet web service from ansible controlnode
  hosts: local
  become: yes
  tasks:
    - name: connect to intranet web server
      uri:
        url: http://robo2
        status_code: 200
```

ansible-playbook --syntax-check intranet.yml
ansible-playbook intranet.yml

```
[ansible@robo multiple-playbook]$ ansible-playbook intranet.yml
SUDO password:

PLAY [Enable the intranet service] *********************************************

TASK [Gathering Facts] *********************************************************
ok: [robo2]

TASK [Install httpd and firewalld pkg] *****************************************
changed: [robo2]

TASK [Firewalld enabled and running] *******************************************
changed: [robo2]

TASK [Firewall permits the http service] ***************************************
ok: [robo2]

TASK [httpd enabled and running] ***********************************************
changed: [robo2]

TASK [test html page is installed] *********************************************
changed: [robo2]

PLAY [Test the intranet web service from ansible controlnode] ******************

TASK [Gathering Facts] *********************************************************
ok: [robo]

TASK [connect to intranet web server] ******************************************
ok: [robo]

PLAY RECAP *********************************************************************
robo                       : ok=2    changed=0    unreachable=0    failed=0
robo2                      : ok=6    changed=4    unreachable=0    failed=0
```

curl http://robo2

```
[ansible@robo multiple-playbook]$ curl http://robo2
Welcome to the robo2 intranet!
[ansible@robo multiple-playbook]$
```

Note: - Please clean up once playbook executed successfully.