Ansible Ad-Hoc Commands: A Detailed Guide

Ansible **ad-hoc commands** allow you to quickly execute single tasks or commands on one or more managed nodes without having to write an entire playbook. These commands are very useful for quick automation, testing, and managing hosts.

In this guide, we'll go over the **basic syntax** of ad-hoc commands, as well as provide detailed examples of common tasks and use cases.

---

1. Basic Syntax of Ansible Ad-Hoc Commands

The general syntax for running an ad-hoc command is:

bash
CopyEdit
```
ansible <group or host> -m <module> -a "<arguments>" [options]
```

- **`<group or host>`**: Specifies the host or group of hosts from your **inventory file** (e.g., `webservers`, `localhost`, `db_servers`).
- **`-m <module>`**: The Ansible module to use (e.g., `ping`, `command`, `yum`, `copy`).
- **`-a "<arguments>"`**: The arguments to pass to the module (in quotes).

Example:

bash
CopyEdit
```
ansible webservers -m ping
```

This command will ping all the hosts in the `webservers` group to check connectivity.

Common Options:

- `-i`: Specify an **inventory file**. By default, it looks for `/etc/ansible/hosts`.
- `-u`: Specify the SSH user (e.g., `ec2-user`, `root`).
- `--private-key`: Specify a custom **SSH private key** for authentication.
- `-v` or `-vv`: Enable verbose output to get more details about what's happening.

---

2. Commonly Used Ansible Ad-Hoc Commands

2.1 Ping Command

The **ping module** is used to check if Ansible can communicate with a host. It sends a "ping" to the target and expects a response. It doesn't actually use ICMP like the `ping` command, but uses Ansible's internal communication.

bash
CopyEdit
```
ansible <host or group> -m ping
```

Example:

bash
CopyEdit
```
ansible webservers -m ping
```

2.2 Running Shell Commands

The **command module** is used to run shell commands on the remote host.

bash
CopyEdit
```
ansible <host or group> -m command -a "<command>"
```

Example:

bash
CopyEdit

```
ansible webservers -m command -a "uptime"
```

This will run the `uptime` command on all the servers in the `webservers` group.

> **Note**: The `command` module **does not process** shell features like pipes (`|`), redirection (`>`) or shell-specific commands. If you need those features, use the `shell` module instead.

Example using `shell`:

bash
CopyEdit
```
ansible webservers -m shell -a "df -h | grep '/dev/xvda1'"
```

2.3 Copy Files from Local to Remote

You can use the **copy module** to copy a file from your local machine to a remote server.

bash
CopyEdit
```
ansible <host or group> -m copy -a "src=<local_file_path>
dest=<remote_file_path>"
```

Example:

bash
CopyEdit
```
ansible webservers -m copy -a "src=/path/to/local/file.txt
dest=/tmp/file.txt"
```

This will copy `file.txt` from your local machine to `/tmp/file.txt` on the `webservers` group hosts.

2.4 Install Packages

Ansible's **package management modules** are great for installing software on remote machines. You can use **`yum`**, **`apt`**, or other package managers depending on the target host's OS.

For **RedHat/CentOS** systems (using `yum`):

bash
CopyEdit
```
ansible <host or group> -m yum -a "name=<package_name> state=present"
```

Example:

bash
CopyEdit
```
ansible webservers -m yum -a "name=httpd state=present"
```

This installs the `httpd` package (Apache) on all hosts in the `webservers` group.

For **Debian/Ubuntu** systems (using `apt`):

bash
CopyEdit
```
ansible <host or group> -m apt -a "name=<package_name> state=present"
```

Example:

bash
CopyEdit
```
ansible webservers -m apt -a "name=nginx state=present"
```

This installs the `nginx` package on all hosts in the `webservers` group.

2.5 Start or Stop Services

You can use the **service module** to start, stop, or restart a service on a remote machine.

bash
CopyEdit
```
ansible <host or group> -m service -a "name=<service_name>
state=<state>"
```

Example:

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

This will start the `httpd` service (Apache) on all hosts in the `webservers` group.

To **stop** the service:

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

2.6 Managing Users

You can add or remove users using the **user module**.

Add a user:

bash
CopyEdit
```
ansible <host or group> -m user -a "name=<username> state=present"
```

Example:

bash
CopyEdit
```
ansible webservers -m user -a "name=testuser state=present"
```

This adds a user named `testuser` to all hosts in the `webservers` group.

To **remove** a user:

bash
CopyEdit
```
ansible webservers -m user -a "name=testuser state=absent"
```

2.7 Change File Permissions

Use the **file module** to modify file permissions, ownership, and other file properties.

bash
CopyEdit

```
ansible <host or group> -m file -a "path=<file_path>
mode=<permissions> owner=<owner> group=<group>"
```

Example:

bash
CopyEdit
```
ansible webservers -m file -a "path=/tmp/file.txt mode=0644 owner=root
group=root"
```

This sets the permissions of `/tmp/file.txt` to `0644` with owner `root` and group `root`.

2.8 Gathering System Facts

You can gather detailed system information from your managed nodes using the **setup module**. This is useful for dynamic inventory management, filtering, and debugging.

bash
CopyEdit
```
ansible <host or group> -m setup
```

Example:

bash
CopyEdit
```
ansible webservers -m setup
```

This will gather and display detailed system information about all hosts in the `webservers` group.

You can also filter the facts to retrieve specific information:

bash
CopyEdit
```
ansible webservers -m setup -a "filter=ansible_facts"
```

---

3. Using Extra Variables with Ad-Hoc Commands

You can pass **extra variables** to customize the behavior of the module you're using by appending the `-e` or `--extra-vars` option.

Example:

bash
CopyEdit
```
ansible webservers -m user -a "name=testuser state=present" -e
"ansible_user=ubuntu"
```

This passes `ansible_user=ubuntu` as an extra variable, which allows you to override the default user for that run.

---

4. Using Wildcards and Groups

You can also target a specific set of hosts or use wildcards.

4.1 Target All Hosts:

To run a command on all hosts defined in the inventory:

bash
CopyEdit
```
ansible all -m command -a "uptime"
```

4.2 Target Specific Hosts by Group:

If your inventory is organized into groups, you can target a specific group:

bash
CopyEdit
```
ansible webservers -m ping
```

4.3 Target Hosts by Wildcard:

You can use wildcard patterns to target multiple hosts:

bash
CopyEdit

```
ansible web* -m ping
```

This will target all hosts whose names start with `web`.

---

5. Useful Ad-Hoc Command Options

- **`-v`**, **`-vv`**, **`-vvv`**: Increase verbosity of the output. You can use this to debug the execution or get more information on what Ansible is doing behind the scenes.
- **`--ask-pass`**: If you need to supply an SSH password, use this option to prompt for the password.
- **`--private-key`**: If you need to specify a custom private key file for SSH authentication.
- **`--become`**: If you need to run commands with elevated privileges (similar to `sudo`).

Example:

bash
CopyEdit
```
ansible webservers -m command -a "mkdir /tmp/test" --become
```

---

6. Conclusion

Ansible **ad-hoc commands** are powerful tools for quick management tasks, making it easy to automate and troubleshoot. By using ad-hoc commands, you can perform one-off tasks without needing to write complete playbooks, which is especially useful for short-term automation or testing.

With the flexibility of Ansible's modules, you can automate tasks ranging from simple command execution to more complex system administration tasks like package management, user management, and service control.

Let me know if you'd like to dive deeper into any specific module or use case!