# Ansible – Modules

## Agenda

- Explore the following modules
  - Group
  - User
  - Stat
  - Copy
  - Fetch
  - File
- Run Operations in Background

# Introduction

- Most common uses for Ansible’s ad-hoc commands is user and group management
- Ansible’s user and group modules make things pretty simple and standard across any Linux flavor
- First, add an admin group on the app servers for the server administrators
  - ansible app -s -m group -a "name=admin state=present“
- The group module is pretty simple
- You can remove a group by setting state=absent, set a group id with gid=[gid]

## Add User to the admin Group

- add the user sada to the app servers with the group
- give him a home folder in /home/sada
  - ansible app -s -m user -a "name=sada group=admin createhome=yes“

- If you want to automatically create an SSH key for the new user
  - additional parameter generate_ssh_key=yes
  - You can also set the UID of the user by passing in uid=[uid],
  - set the user’s shell with shell=[shell]

- Delete an account
  - ansible app -s -m user -a "name=sada state=absent remove=yes"

## Manage files and directories

- Another common use for ad-hoc commands is remote file management
- Ansible makes it easy to copy files from your host to remote servers, create directories, manage file and directory permissions and ownership, and delete files or directories
- Get information about a file using stat module
  - ansible multi -m stat -a "path=/bin/bash"

## Copy a file to the servers

- ansible multi -m copy -a "src=/etc/hosts dest=/tmp/hosts“
- The src can be a file or a directory. If you include a trailing slash, only the contents of the directory will be copied into the dest.
- If you omit the trailing slash, the contents and the directory itself will be copied into the dest
- Note: copy module is perfect for single-file copies, and works very well with small directories
- to copy hundreds of files, especially in very deeply-nested directory structures
  – unarchive module
  – synchronize module.

## Retrieve a file from the servers

- fetch module works almost exactly the same as the copy module, except in reverse
- $ ansible multi -s -m fetch -a "src=/etc/hosts dest=/tmp“
- files will be copied down to the local dest in a directory structure that matches the host from which you copied them
- Fetch will, by default, put the /etc/hosts file from each server into a folder in the destination with the name of the host (IP addresses in this case)

# Create directories and files

- use the file module to create files and directories (like touch), manage permissions and ownership on files and directories
  - $ ansible multi -m file -a "dest=/tmp/test mode=644 state=directory“
- create a symlink
  - $ ansible multi -m file -a "src=/src/symlink dest=/dest/symlink \ owner=root group=root state=link"
- Delete directories and files
  - $ ansible multi -m file -a "dest=/tmp/test state=absent"

# Run operations in the background

- Some operations take quite a while (minutes or even hours).
- For example, when you run
  - yum update or
  - apt-get update && apt-get dist-upgrade,

  it could be a few minutes before all the packages on your servers are updated
- In these situations, you can tell Ansible to run the commands asynchronously, and poll the servers to see when the commands finish
- -B : the maximum amount of time (in seconds) to let the job run.
- -P : the amount of time (in seconds) to wait between polling the servers for an updated job status
  - $ ansible multi -s -B 3600 -a "yum -y update“
  - $ ansible multi -s -m async_status -a "jid=123"

# Fire-and-forget tasks

- You may also need to run occasional long-running maintenance scripts, or other tasks that take many minutes or hours to complete, and you’d rather not babysit the task
- you can set the -B value as high as you want (be generous, so your task will complete before Ansible kills it!),
- and set -P to ‘0’,
- so Ansible fires off the command then forgets about it
  - ansible multi -B 3600 -P 0 -a "/path/to/fire-and-forget-script.sh“

- Note: Running the command this way doesn’t allow status updates via async_status and a jid, but you can still inspect the file ~/.ansible_async/ on the remote server.
- Tip: For tasks you don’t track remotely, it’s usually a good idea to log the progress of the task somewhere, and also send some sort of alert on failure

# Check log files

- When debugging application errors, or diagnosing outages or other problems, you need to check server log files
- Operations that continuously monitor a file, like tail -f, won't work via Ansible, because
- Ansible only displays output after the operation is complete, and you won't be able to send the Control-C command to stop following the file
- It's not a good idea to run a command that returns a huge amount of data via stdout via Ansible
- If you're going to cat a file larger than a few KB, you should probably log into the server(s) individually
- If you redirect and filter output from a command run via Ansible, you need to use the shell module instead of Ansible's default command module (add -m shell to your commands)
- $ ansible multi -s -a "tail /var/log/messages"
- $ ansible multi -s -m shell -a "tail /var/log/messages | \ grep ansible-command | wc -l"