Variables in Ad-hoc commands

- Ansible variables are dynamic values used within Ansible playbooks and roles to enable customization, flexibility, and reusability of configurations.

ansible all -m shell -e servers='[test01,test02]' -a 'echo {{ servers }}'

m → module

e → extra variables

a → module arguments

Manage users and groups

#To add a group on the node servers

ansible all -m group -a "name=admin state=present“

#To remove a group on the node servers

ansible all -m group -a "name=admin state=absent“

We can remove a group by setting **state=absent**, set a group id with **gid=[gid]**, and indicate that the group is a system group with **system=yes**

Cont.

To add users into group

ansible devservers -m user -a "name=neopolean group=admin createhome=yes“

- If you want to automatically create an SSH key for the new user (if one doesn’t already exist), you can run the same command with the 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]**, and the password with **password=[encrypted-password].**

#To get list of users

ansible devservers -m command -a "cut -d: -f1 /etc/passwd"

To delete an account

ansible all -m user -a "name=neopolean state=absent remove=yes"

Manage files and directories

Get information about a file

#If we need to check a file's permissions, MD5, or owner, use Ansible's stat module:

ansible all -m stat -a "path=/etc/hosts"

ansible multi -m stat -a 'path=/etc/environment'

#To extract only size

ansible all -m stat -a "path=/etc/hosts" | grep -i size

#To get mtime – modified time alone

ansible all -m stat -a "path=/etc/hosts" | grep -i mtime | cut -d ":" -f2 | sed 's/,//' | sed 's/\s//'

Cont.

Copy a file to the servers

- We probably use **scp and/or rsync** to copy files and directories to remote servers, and while Ansible has recently gained an rsync module, most file copy operations can be completed with Ansible's **copy** module:

ansible all -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.

Cont.

- The **copy** module is perfect for single-file copies, and works very well with small directories.
- When you want to copy hundreds of files, especially in very deeply-nested directory structures, you should consider either copying then expanding an archive of the files with Ansible's **unarchive** module, or using Ansible's **synchronize** module.

Retrieve a file from the servers

- The **fetch** module works almost exactly the same as the **copy** module, except in reverse.
- The major difference is that files will be copied down to the local dest in a directory structure that matches the host from which you copied them.

Cont.

- For example, use the following command to grab the hosts file from the servers:

  ansible all -m fetch -a "src=/etc/hosts dest=/tmp“

- Fetch will, by default, put the /etc/hosts file from each server into a folder in the destination with the name of the host

  ansible all -m fetch -a "src=/etc/hosts dest=/tmp/ flat=yes"

- You can add the parameter **flat=yes**, and set the dest to **dest=/tmp/** (add a trailing slash), to make Ansible fetch the files directly into the /tmp directory. However, filenames must be unique for this to work, so it’s not as useful when copying down files from multiple hosts. Only use **flat=yes** if you’re copying files from a single host.