Ansible

Notes:

* ansible-playbook first-playbook.yaml -i inventory --extra-vars "node1\_ip=10.221.242.53 username=root root\_password=Gyp.s8m" -v
* ansible -i inventory -m ping all --extra-vars "node1\_ip=10.221.242.53 username=root root\_password=Gyp.s8m" -v
* ansible -i inventory -m shell -a "ls /etc" all --extra-vars "node1\_ip=10.221.242.53 username=root root\_password=Gyp.s8m" -v
* role can be imported from ansible galaxy. It acts like docker hub.

ansible-galaxy role init test

* install roles from ansible galaxy

ansible-galaxy role install <role\_name>

#by running above command role will be available in ~/.ansible/role folder.

#import custom roles to ansible galaxy

#create a role and push the whole dircetory to github repo.

#import the role to galaxy using below command

#(api-token can be generated from ansible galaxy website - collections->api-token->load-token)

ansible-galaxy import <github\_username> <repo\_name> <api\_token>

* install new collections (collection\_name can be found on official documentation)

ansible-galaxy collection install <collection\_name>

* error handling in ansible

ansible playbook will fail even if a single task fails. If we want to avoid this then we can use

ignore\_error: true in our playbook.

Also, we can handle whether task should be performed or not by adding conditionals "failed\_when"

let's say we are doing cd or ls to a directory which does not exist then we can use failed\_when like -

- name: Fail task when the command error output prints FAILED

ansible.builtin.command: /usr/bin/example-command -x -y -z

register: command\_result

failed\_when: "'FAILED' in command\_result.stderr"

- name: Check if a file exists in temp and fail task if it does

ansible.builtin.command: ls /tmp/this\_should\_not\_be\_here

register: result

failed\_when:

- result.rc == 0

- '"No such" not in result.stdout'

* ansible-vault

It is like a bank locker. We can store sensitive info inside the vault and then access directly inside the playbook.

Steps to create and use vault ->

let's say we must hide aws credentials.

1) create a random password for vault: #openssl rand -base64 2048 > vault.pass

2) create a vault and pass the above password:

#ansible-vault create group\_vars/all/aws-creds.yaml --vault-password-file vault.pass

a vault (vi editor window) will open where we need to put your credentials in "key: value" pairs. Now save the file and use the same keys in playbook to access the values. If we run -> cat aws-creds.yaml then we can see encrypted data.

If we want to see decrypted data then run:

# ansible-vault decrypt aws-creds.yaml --vault-password-file vault.pass

note that we need password to decrypt.