ansible notes:

Inventory:

[**Assigning a variable to many machines: group variables**](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#id12)

If all hosts in a group share a variable value, you can apply that variable to an entire group at once.

**In INI:**

[atlanta]

host1

host2

[atlanta:vars]

ntp\_server=ntp.atlanta.example.com

proxy=proxy.atlanta.example.com

**In YAML:**

atlanta**:**

hosts**:**

host1**:**

host2**:**

vars**:**

ntp\_server**:** ntp.atlanta.example.com

proxy**:** proxy.atlanta.example.com

### [**Inheriting variable values: group variables for groups of groups**](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#id13)**:**

You can make groups of groups using the :children suffix in INI or the children: entry in YAML. You can apply variables to these groups of groups using :vars or vars::

**In INI:**

[atlanta]

host1

host2

[raleigh]

host2

host3

[southeast:children]

atlanta

raleigh

[southeast:vars]

some\_server=foo.southeast.example.com

halon\_system\_timeout=30

self\_destruct\_countdown=60

escape\_pods=2

[usa:children]

southeast

northeast

southwest

northwest

Child groups have a couple of properties to note:

* Any host that is member of a child group is automatically a member of the parent group.
* A child group’s variables will have higher precedence (override) a parent group’s variables.
* Groups can have multiple parents and children, but not circular relationships.
* Hosts can also be in multiple groups, but there will only be one instance of a host, merging the data from the multiple groups.

You can also add group\_vars/ and host\_vars/ directories to your playbook directory. The ansible-playbook command looks for these directories in the current working directory by default.

 Other Ansible commands (for example, ansible, ansible-console, etc.) will only look for group\_vars/ and host\_vars/ in the inventory directory.

If you load inventory files from both the playbook directory and the inventory directory, variables in the playbook directory will override variables set in the inventory directory.

## [Connecting to hosts: behavioral inventory parameters](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#id17)

# **Tags:**

If you have a large playbook, it may become useful to be able to run only a specific part of it rather than running *everything* in the playbook. Ansible supports a “tags:” attribute for this reason.

- name: Adding a user

user:

name: test

createhome: true

home: /home/test

shell: /bin/bash

generate\_ssh\_key: yes

ssh\_key\_bits: 4096

tags:

- user

Tags can be applied to many structures in Ansible, but its simplest use is with individual tasks. When you execute a playbook, you can filter tasks based on tags in two ways:

* On the command line, with the --tags or --skip-tags options
* In Ansible configuration settings, with the TAGS\_RUN and TAGS\_SKIP options

ansible-playbook example.yml --tags "user "

+++++++++

**Password changing using ansible:**

Within Ansible, we will leverage the ‘user’ module to quickly change the password for a user account on all our servers. Ansible doesn’t allow you to pass a cleartext password through its playbooks so you have to use a hashed password. [mkpasswd --method=SHA-512]

- name: create new users

user:

name: '{{ item.username }}'

password: '{{ item.password }}'

state: present

shell: /bin/bash

groups: admin

update\_password: always

with\_items:

- "{{ users }}"

Define group vars.

Under group vars/main.yml

Users:

* Username: test

Password: hashvalue generated using mkpasswd command

+++++++++

**Ansible roles:**

Roles are ways of automatically loading certain vars\_files, tasks, and handlers based on a known file structure. Grouping content by roles also allows easy sharing of roles with other users.

Roles expect files to be in certain directory names. Roles must include at least one of these directories, however it is perfectly fine to exclude any which are not being used. When in use, each directory must contain a main.yml file, which contains the relevant content:

* tasks - contains the main list of tasks to be executed by the role.
* handlers - contains handlers, which may be used by this role or even anywhere outside this role.
* defaults - default variables for the role (see [Using Variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#playbooks-variables) for more information).
* vars - other variables for the role (see [Using Variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#playbooks-variables) for more information).
* files - contains files which can be deployed via this role.
* templates - contains templates which can be deployed via this role.
* meta - defines some meta data for this role. See below for more details.

**Ansible Vault:**

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control. To enable this feature, a command line tool - [ansible-vault](https://docs.ansible.com/ansible/latest/cli/ansible-vault.html#ansible-vault) - is used to edit files, and a command line flag ([**--ask-vault-pass**](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-ask-vault-pass), [**--vault-password-file**](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-vault-password-file) or [**--vault-id**](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-vault-id)) is used. Alternately, you may specify the location of a password file or command Ansible to always prompt for the password in your ansible.cfg file. These options require no command line flag usage.

To create a new encrypted data file, run the following command:

ansible-vault create foo.yml

First you will be prompted for a password. After providing a password, the tool will launch whatever editor you have defined with $EDITOR, and defaults to vi. Once you are done with the editor session, the file will be saved as encrypted data.

[**Editing Encrypted Files**](https://docs.ansible.com/ansible/latest/user_guide/vault.html#id12)**:**

To edit an encrypted file in place, use the [ansible-vault edit](https://docs.ansible.com/ansible/latest/cli/ansible-vault.html#ansible-vault-edit) command.

ansible-vault edit foo.yml

To edit a file encrypted with the ‘vault2’ password file and assigned the ‘pass2’ vault ID:

ansible-vault edit --vault-id pass2@vault2 foo.yml

ansible changed configs:

ansible-config dump --only-changed

✘ hanitha@DECIBEL-0211  ~/Desktop/Git\_repo/ansible   jenkins\_test  ansible-config dump |grep VAULT

DEFAULT\_ASK\_VAULT\_PASS(default) = False

DEFAULT\_VAULT\_ENCRYPT\_IDENTITY(default) = None

DEFAULT\_VAULT\_IDENTITY(default) = default

DEFAULT\_VAULT\_IDENTITY\_LIST(default) = []

DEFAULT\_VAULT\_ID\_MATCH(default) = False

DEFAULT\_VAULT\_PASSWORD\_FILE(/Users/hanitha/Desktop/Git\_repo/ansible/ansible.cfg) = /Users/hanitha/Desktop/Git\_repo/ansible/vault-env

#grep -i vault /Users/hanitha/Desktop/Git\_repo/ansible/ansible.cfg

# Vault

vault\_password\_file = ./vault-env