# Ansible

## Architecture

* Control node – node on which ansible is installed and from where playbooks are triggered (ansible+ inventory)
* Managed node – node on which playbooks are executed and part of inventory
* Playbook – list of plays
* Play – list of tasks (modules)
* Task- task that need to be run on the targeted nodes

## Installation and setup

Install Ansible as a regular user

• pip3 install ansible ansible-doc

Check Ansible version and run ping module without Playbook to check Ansible status

• # ansible --version

• # ansible localhost –m ping

Ansible config files

• /etc/ansible → Default directory

• /etc/ansible/ansible.cfg

• /etc/ansible/hosts

• /etc/ansible/roles

## pre requisites

Python version 2.7 and above for linux

Windows – powershell 5.1 and .net 4.0

## Modules - idempotency

An operation is idempotent if the result of performing it once is exactly same as the result of performing it repeatedly without any intervening actions

Shell, command, raw – non idempotent

Script is also non idempotent but the script module assume the unlying script to maintain idempotency

## ansible.cfg

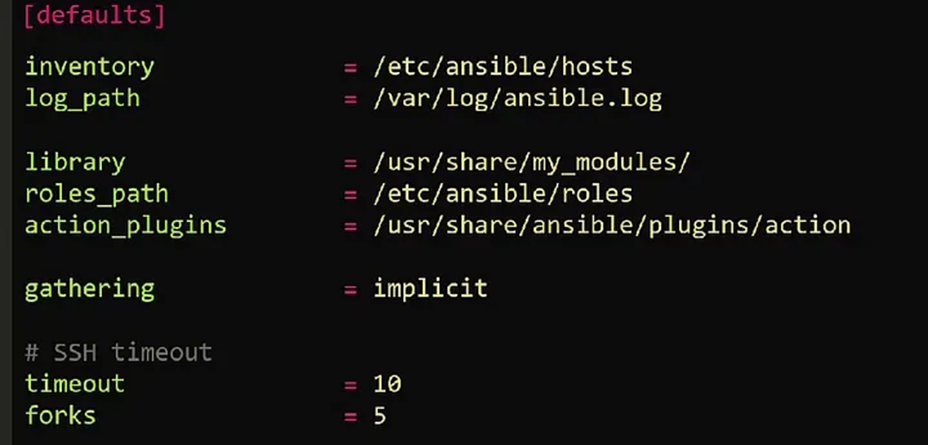
Certain settings in Ansible are adjustable with a configuration file (ansible.cfg).

You can generate an Ansible configuration file, ansible.cfg, that lists all default settings as follows:

**$** ansible-config init --disabled > ansible.cfg

Include available plugins to create a more complete Ansible configuration as follows:

**$** ansible-config init --disabled -t all > ansible.cfg



## hosts.ini / inventory.ini

All remote clients are considered inventory in Ansible

• Ansible keeps its inventory information in host file located: /etc/ansible/hosts

• The hosts file is created during Ansible installation

• Inventory host file can either be static or dynamic (using additional plug-ins)

• Listing host file # ansible-inventory --list OR # ansible all --list-hosts

## Linux/windows setup

Populate the hosts file with IP or FQDN for our clients:

[labclients] = For grouping

10.253.1.18

10.253.1.20

• Generate SSH Keys on the control node and copy over to clients for password less SSH connections

# ssh-keygen

# Leave everything default and enter

# ssh-copy-id 10.253.1.18

# ssh-copy-id 10.253.1.20

• Now SSH into the clients to test # ssh 10.253.1.18

• Run Ansible add-hoc to ping remote nodes (make sure hosts file has remote clients IPs)

# ansible all -m ping # ansible –a “uptime” all (To run a command on the remote clients

## Playbook

Start a playbook at a specific task

# anisble-playbook yamlfile.yml --start-at-task ‘Task name’

# anisble-playbook http.yml --start-at-task ‘Intall telnet’

## Roles

Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure.

An Ansible role has a defined directory structure with seven main standard directories.

By default, Ansible will look in most role directories for a main.yml file for relevant content (also main.yaml and main):

* tasks/main.yml - A list of tasks that the role provides to the play for execution.
* handlers/main.yml - handlers that are imported into the parent play for use by the role or other roles and tasks in the play.
* defaults/main.yml - very low precedence values for variables provided by the role (see [Using Variables](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#playbooks-variables) for more information). A role’s own defaults will take priority over other role’s defaults, but any/all other variable sources will override this.
* vars/main.yml - high precedence variables provided by the role to the play (see [Using Variables](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#playbooks-variables) for more information).
* files/stuff.txt - one or more files that are available for the role and it’s children.
* templates/something.j2 - templates to use in the role or child roles.
* meta/main.yml - metadata for the role, including role dependencies and optional Galaxy metadata such as platforms supported. This is required for uploading into galaxy as a standalone role, but not for using the role in your play.

You can use roles in the following ways:

* at the play level with the roles option: This is the classic way of using roles in a play.
* at the tasks level with include\_role: You can reuse roles dynamically anywhere in the tasks section of a play using include\_role.
* at the tasks level with import\_role: You can reuse roles statically anywhere in the tasks section of a play using import\_role.
* as a dependency of another role (see the dependencies keyword in meta/main.yml in this same page).

## Vault

• Ansible can automate tasks for teams such as:

• Hardware

• Operating systems

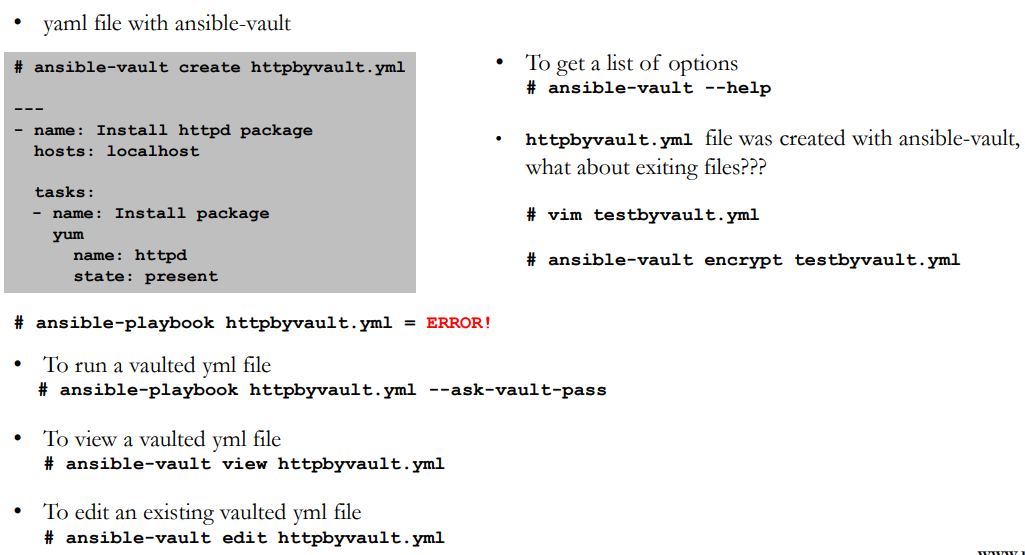
• Virtualization

• Database or Storage

• Applications/Software etc.

• Oftentimes you have to share Ansible code with these groups over the network and anything you share over network has a risk to end up in wrong hands

• It is best practice to use Ansible vault feature which will password protect your code





## Vars - Groups vars, host vars, inventory vars, extra vars, role vars

Lower to highest

* 1. command line values
  2. role defaults
  3. inventory file group\_vars script
  4. inventory group\_vars/all
  5. playbook group\_vars/all
  6. inventory group\_vars/\*
  7. playbook group\_vars/\*
  8. inventory file host\_vars script
  9. inventory host\_vars/\*
  10. playbook host\_vars/\*
  11. host facts /cached facts
  12. play vars
  13. play vars\_prompt
  14. play vars\_file
  15. role vars
  16. block vars
  17. task vars
  18. include\_vars
  19. set\_facts/registered vars
  20. include role vars
  21. include task vars
  22. extra vars

## ansible adhoc commands

* Ping localhost

# ansible localhost –m ping

* Creating a file on all remote clients

# ansible all –m file –a “path=/home/iafzal/adhoc1 state=touch mode=700”

* Deleting a file on all remote clients

# ansible all –m file –a “path=/home/iafzal/adhoc1 state=absent”

* Copying a file to remote clients

# ansible all –m copy –a “src=/tmp/adhoc2 dest=/home/iafzal/adhoc2”

* Installing package (telnet and httpd-manual)

# ansible all –m yum –a “name=telnet state=present”

# ansible all –m yum –a “name=httpd-manual state=present”.

* Starting httpd package service

# ansible all –m service –a “name=httpd state=started”

* Start httpd and enable at boot time

# ansible all –m service –a “name=httpd state=started enabled=yes”

* Checking httpd service status on remote client

# ansible all –m shell -a “systemctl status httpd”

* Remove httpd package

# ansible all –m yum –a “name=httpd state=absent” OR

# ansible all –m shell -a “yum remove httpd”.

* Creating a user on remote clients

# ansible all –m user –a “name=jsmith home=/home/jsmith shell=/bin/bash state=present”

* To add a user to a different group

# ansible all –m user –a “name=jsmith group=iafzal”

* Deleting a user on remote clients

# ansible all –m user –a “name=jsmith home=/home/jsmith shell=/bin/bash state=absent” OR

# ansible all –m shell –a “userdel jsmith”

* Getting system information from remote clients

# ansible all –m setup

* You can run commands on the remote host without a shell module e.g. reboot client1

# ansible client1 –a “/sbin/reboot”

## Collections

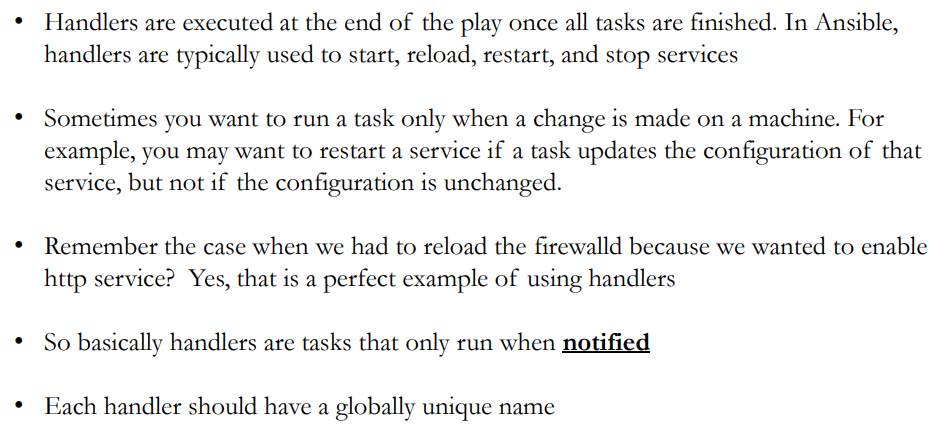
Collections are a distribution format for Ansible content that can include playbooks, roles, modules, and plugins. You can install and use collections through a distribution server, such as Ansible Galaxy.

If your playbook uses both the collections keyword and one or more roles, the roles do not inherit the collections set by the playbook. This is one of the reasons we recommend you always use FQCN.

Eg. Ansible.builtin, ansible.community

https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections

## handlers



A screenshot of a computer

Description automatically generated

## Conditions

A screenshot of a computer

Description automatically generated

A close-up of a computer screen

Description automatically generated

## Loops

A white text with black text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

## Tags

If you have a large playbook, it may be useful to run only specific parts of it instead of running the entire playbook. You can do this with Ansible tags. Using tags to execute or skip selected tasks is a two-step process:

1. Add tags to your tasks, either individually or with tag inheritance from a block, play, role, or import.
2. Select or skip tags when you run your playbook.

Ansible reserves several tag names for special behavior: always, never, tagged, untagged and all. Both always and never are mostly for use in tagging the tasks themselves, the other three are used when selecting which tags to run or skip.

* If you assign the always tag to a task or play, Ansible will always run that task or play, unless you specifically skip it (--skip-tags always) or another tag defined on that task.
* If you assign the never tag to a task or play, Ansible skips that task or play unless you specifically request it (--tags never) or another tag defined for that task.

ansible-playbook example.yml --tags "tag1,tag3" --skip-tags "tag4"

## jinja templates

{% %} – for expressions

{{ }} – outputting a variable value

{# #} – for comments

## set\_facts, set\_stats

* These variables will be available to subsequent plays during an ansible-playbook run via the host they were set on. **-** **name:** Creating list and dictionary variables using 'shorthand' YAML

**ansible.builtin.set\_fact:**

**two\_dict:** **{**'something'**:** **here2,** 'other'**:** **somewhere}**

**two\_list:** **[1,2,3]**

* This module allows setting/accumulating stats on the current ansible run, either per host or for all hosts in the run.

**-** **name:** Aggregating packages\_installed stat per host

**ansible.builtin.set\_stats:**

**data:**

**packages\_installed:** 31

**per\_host:** yes

## set\_fact vs Register

set\_fact is specific to a host and is valid until the entire list of plays are executed, register is specific to a play and same for all hosts so it gets destroyed when play ends

## Magic variables

Variables related to remote systems are called facts. With facts, you can use the behavior or state of one system as a configuration on other systems. For example, you can use the IP address of one system as a configuration value on another system. Variables related to Ansible are called magic variables.

You can access information about Ansible operations, including the Python version being used, the hosts and groups in inventory, and the directories for playbooks and roles, using “magic” variables.

The most commonly used magic variables are hostvars, groups, group\_names, and inventory\_hostname

{% **for** **host** **in** **groups[**'app\_servers'**]** %}

# something that applies to all app servers.

{{ hostvars[‘I’][‘html\_content’] }}

{% **endfor** %}

{% **for** **host** **in** **groups[**'app\_servers'**]** %}

{{ **hostvars[host][**'ansible\_facts'**][**'eth0'**][**'ipv4'**][**'address'**]** }}

{% **endfor** %}

## serial/parallel, job slicing/forking

By default, Ansible runs in parallel against all the hosts in the [pattern](https://docs.ansible.com/ansible/latest/inventory_guide/intro_patterns.html#intro-patterns) you set in the hosts: field of each play. If you want to manage only a few machines at a time, for example during a rolling update, you can define how many hosts Ansible should manage at a single time using the **serial** keyword:

**---**

**-** **name:** test play

**hosts:** webservers

**serial:** 3

**gather\_facts:** False

**tasks:**

**-** **name:** first task

**command:** hostname

**-** **name:** second task

**command:** hostname

The default behaviour described above is the [linear strategy](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/linear_strategy.html#linear-strategy). Ansible offers other strategies, including the [debug strategy](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/debug_strategy.html#debug-strategy) (see also [Debugging tasks](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_debugger.html#playbook-debugger)) and the [free strategy](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/free_strategy.html#free-strategy), which allows each host to run until the end of the play as fast as it can:

strategy **=** free/ debug/ linear

By default, Ansible runs each task on all hosts affected by a play before starting the next task on any host, using 5 forks. If you want to change this default behavior, you can use a different strategy plugin, change the number of forks, or apply one of several keywords like serial.

If you have the processing power available and want to use more forks, you can set the number in ansible.cfg:

**[defaults]**

forks **=** 30

## Ansible runner

## Ansible molecule

## Ansible tower

## Project

## template

## inventory

## credential types

## credentials

## tower api

## Ansible-lint

## Yaml-lint

## GitLab ci