**Ansible**<https://www.youtube.com/watch?v=WJ8-hG7n0Xc&list=PLDhScTEBdP8wSi9eCMgL35iq44oqkRSXI>  
<https://www.youtube.com/watch?v=ajd4SvAMuy0&list=PLDhScTEBdP8yUtKxaaf7DwbGPmnrwnMgS>  
<https://www.youtube.com/watch?v=3cRhyISYxtE&list=PLDhScTEBdP8zRBt7R9bZ-tRER16ipYhSf>  
<https://www.youtube.com/watch?v=CIWIiiTsiDU&list=PLDhScTEBdP8zj2fpe_H3_R-mD9_-O3uXt>  
<https://www.youtube.com/watch?v=DpJ_v4C5uQY&list=PLDhScTEBdP8w28re6ij5w0BqWjDgYLhs1>  
<https://www.youtube.com/watch?v=X-MGKRk7ZgQ&list=PLDhScTEBdP8xMDkeDaZX5YLTabVAwq-xi>

**Configuration:**  configuration of servers.  
Each server contains:  
 --------------STATE--------------  
Software’s  
files, C | E | D   
directories, C | E | D   
packages, I | U | Update  
services, E | D | S | S | Restart  
policies, A | E | D  
users and groups C | E | D  
Process: who should have what policies, rights on files and directories  
Script  
Network  
Hardware: Ansible can’t manage this but it can manage hypervisor which in turn manages hardware.

**Tool:** Save cost (what? Managing a servers), time (what? Deploying apps at servers) and improve quality and efficiency (consistent change done all servers).

**Ansible:** Ansible is a powerful automation tool available in the market which manages the configuration of servers and helps us in saving cost, time and improves quality and efficiency.  
Server (ACS) – Client (ARS)  
Agentless =>   
ACS ARS  
 python (ssh & winrm)  
DSL of python  
Supports 2 protocol – SSH & WINRM

Roles in ansible:  
Developer: who writes program in ansible  
Admin: who apply programs in servers

how can we become programmer?  
Install ansible => .yaml (playbook) => roles => Admin

Ansible vs puppet  
===================================================  
Agent less With Agent  
SSH WINRM TCP  
Roles Modules  
Playbook Puppet program ( == Manifest)  
Python Ruby

Architecture of ansible:  
 Ansible control server – 64 bit linux  
 Ansible – A tool  
 Modules  
 inventory – a file [dir/list] which has list of servers  
 Config – Ansible config => /etc/ansible/ansible.cfg

Ansible release:  
Ansible => command line tool (free + Open-source tool)  
Ansible tower => GUI + command line tool (Free for 10 servers - paid - Enterprise)  
Ansible AWX => GUI + command line tool (Free for N servers - Free - No support + most recent).

Ansible can manage:  
Hardware: Ansible can manage hyperwisor which in turn manages the hardware.  
Software: files, directories, packages, services.  
People: users and groups.  
Process: policy.

**Why Ansible??**Deployment in servers with multiple configuration at onetime.  
Easy to learn (Compared to python, bash), test, debug, share and extend.  
Platform independent (bash script won’t support all servers i.e., windows, linux).  
Idempotent: Detect actual state and desired state and based on that it will do the changes.  
Consistent change server management.  
Save time because of parallel deployment across remote servers.

How ansible works? and architecture?  
Install Ansible command tool.  
Ansible configuration file (we can customize ansible default behaviour).  
Ansible playbook (Role).  
Ansible inventory.

**Note:** ACS must be linux and ARS can be anything.  
ACS Linux -> ARS Linux (SSH conection)  
ACS Linux -> ARS windows (Winrm connection)

**Components of ansible:**  
Ansible => moment we install ansible we get some utilities like modules (Unit of code that would execute in ARS E.g: copy, file for file operations, service, package. Each module requires parameters -> required and optional, 1000’s of built-in module) , plugins (Code which empowers various functionality of ansible), ansible, ansible-playbook, ansible-vault, ansible-doc, xxxx…  
Ansible configuration file => Default path: /etc/ansible/ansible.cfg  
Ansible playbook (Role) => YAML file, script that would execute in ARS.  
Ansible Inventory => it’s a file contains collection of IP address of ARS.

**Link to all modules available in ansible:** https://docs.ansible.com/ansible/2.9/modules/list\_of\_all\_modules.html

**Ansible adhoc commands:  
which one task we want to perform in sever?**

**Execute one task in one machine (local host)**--------------pseudocode------------  
Setup webserver

--------package----------------module-------------parameters------------  
Step-1: Install http server => yum module => name & installed  
Step-2: Copy index.html into /var/www/html => copy module => src & dest  
Step-3: Start a http server => service module => name & state

ansible localhost -m yum -a “name=httpd state=installed”  
ansible localhost -m copy -a “src=index.html dest=/var/www/html/index.html”  
ansible localhost -m service -a “name=httpd state=started”

**How authorization works in SSH?**

How to login to linux m/c?  
username password  
-u -k  
username key  
-u --key-file

**How to gain more privilege in remote machine??**

sudo without password => -b  
sudo with password => -b -K  
sudo with diff userId & password => -b –become-user -k  
sudo with diff userId & key => -b –become-user --key-file

**what is inventory?**List of IP address of ARS,  
12.3.4.102 => not a list  
12.3.4.103, => yes it is list  
12.3.4.103, 13.5.6.701 => yes it is list  
used at command line : 12.3.4.102, 12.3.4.103, 13.5.6.701  
file which is in INI format:   
12.3.4.102,   
12.3.4.103,   
13.5.6.701  
script which returns list of IP’s.  
Directory  
  
Can be grouped  
Default – all (all ip’s) & ungroup (IP address part of no group)

command:  
ansible all -i 12.3.4.102, 12.3.4.103, 13.5.6.701 -m yum -a “name=httpd state=installed” -u ec2-user --key-file =….pem -b (Do one task in one remote m/c)  
or  
ansible all -i inventory -m yum -a “name=httpd state=installed” -u ec2-user --key-file =….pem -b (Do one task in multiple remote m/c using inventory)  
or  
ansible web -i inventory -m yum -a “name=httpd state=installed” -u ec2-user --key-file =….pem -b (Do one task in multiple remote m/c using group in inventory)

ansible all -i 12.3.4.102, 12.3.4.103, 13.5.6.701 -m copy -a “src=index.html dest=/var/www/html/index.html” -u ec2-user --key-file =….pem -b  
  
ansible all -i 12.3.4.102, 12.3.4.103, 13.5.6.701 -m service -a “name=httpd state=started” -u ec2-user --key-file =….pem -b

| **Part** | **Meaning** |
| --- | --- |
| ansible all | Targets the group all, i.e., all hosts listed in the inventory. |
| -i "12.3.4.102,12.3.4.103,13.5.6.701," | Inline inventory. The trailing comma is required for a single-line inventory list. It tells Ansible not to look for a file. |
| -m service | Use the service module, which manages services on Linux (e.g., start, stop, restart). |
| -a "name=httpd state=started" | Arguments for the module: you want to start the httpd service. |
| -u ec2-user | SSH user to connect as (e.g., ec2-user for Amazon Linux). |
| --key-file /path/to/key.pem | Path to the SSH private key for authentication. No = sign should be used here. |
| -b | Run with sudo (become root) — needed to manage system services like httpd. |

**/etc/ansible/ansible.cfg**search for **host\_key\_checking** and this should be uncommented so that whenever ACS managing a new ARS it will not ask for permission yes/no again.

**To list all ip’s in inventory file/host patterns:  
Setup:**Login as root user >> cd /etc/ansible >> vim hosts => add managing nodes ip’s.  
**Commands:**  
ansible all --list-hosts => lists all ip’s.  
ansible webserver --list-hosts => list grouped ip’s.  
ansible webserver[1] --list-hosts => list indexed ip in the group webserver.  
ansible webserver[0:1] --list-hosts => list range of indexed ip in the group webserver.

How can I run multiple modules on multiple remote servers?  
===> playbook  
 .yaml

YAML Syntax:  
All members of a list are lines beginning at the same indentation level starting with a “- ”(a dash and a space)  
E.g:   
--- # Beginning of yaml  
fruits:  
 - Apple  
 - Orange  
 - Strawberry  
 - Mango  
… # End of yaml

A dictionary is represented in a simple key: value from (the colon must be followed by a space)  
E.g:   
martin:  
 name: Abhi  
 job: Software Engineer  
 skill: Elite

=======================================

What is playbook?  
Is a YAML file  
Contains collection of plays

What is play?  
//Play-1  
Hosts: all or <custom group>  
Tasks: List of modules with its param  
 - Module & Param  
 - Module & Param  
 - Module & Param  
 - Module & Param  
//Play-2  
Hosts: all or <custom group>  
Tasks: List of modules with its param  
 - Module & Param  
 - Module & Param  
 - Module & Param  
 - Module & Param  
//Play-3  
Hosts: all or <custom group>  
Tasks: List of modules with its param  
 - Module & Param  
 - Module & Param  
 - Module & Param  
 - Module & Param  
Contains many task  
contains Host group specifications  
Behaviour parameters + Built-in variables

What is task?  
Module in YAML format.

E.g: web.yaml  
---  
- name: Update web servers  
 hosts: web  
 sudo: yes

tasks:   
 - name: Ensure that apache is installed  
 yum: name=httpd state=present  
 - name: Copying index.html  
 copy:  
 src: /opt/index.html  
 dest: /var/www/html/index.html  
 - name: Start apache services  
 service: name=httpd enabled =yes state=started

- hosts: db  
 remote\_user: root

tasks:   
 - name: Install MySQL   
 yum: name=mysql-server state=present  
 - name: Start MySQL services  
 service: name=mysqld state=started  
…   
Command to validate yaml syntax: ansible-playbook --syntax-check web.yaml  
ansible-playbook --syntax-check web.yaml -i inventory => to remove warning msg in o/p.

Command to dry run: ansible-playbook -c web.yaml -i inventory -u user\_name -k -b

**-u user\_name (--user)**

* **Purpose**: Specifies the remote username to connect as
* **Example**: -u ubuntu, -u root, -u myuser
* **Default**: Uses your current local username if not specified
* **Usage**: This is the user account Ansible will use to SSH into the remote hosts

**-k (--ask-pass)**

* **Purpose**: Prompts you to enter the SSH password interactively
* **When to use**: When you don't have SSH key-based authentication set up
* **Security**: You'll be prompted to type the password (it won't be visible on screen)
* **Alternative**: Without -k, Ansible assumes you're using SSH keys for authentication

**-b (--become)**

* **Purpose**: Enables privilege escalation (like using sudo)
* **Default method**: Uses sudo to become root
* **Usage**: Allows tasks that require elevated privileges to run
* **Related flags**:
  + --become-user to specify which user to become (default is root)
  + --ask-become-pass or -K to prompt for the sudo password

Command:  
ansible-playbook -i inventory web.yaml -u ec2-user -k -b

Command to run in local machine:  
ansible-playbook localhost web.yaml

================================================================

Where to declare variables in ansible??  
22 places  
-------------------  
Playbook  
 - Embed variables in playbook  
 - Embed external variables files  
 - Embed using include vars into task  
 - Use variables using User prompt in playbook  
 - Use variables using output of task using register

Inventory:  
- Behavioural parameters: meaning controlling the behaviour of the ARS.  
Google search ansible Behavioural parameters   
<https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html>  
List of ansible inventory Behavioural parameters  
<https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html#connecting-to-hosts-behavioral-inventory-parameters>

Consider an example:  
-----------------------------------  
inventory file

[web] this group we have to login as UN & passwd & sudo without passwd  
1.2.3.4 ansible\_ssh\_user=user1, ansible\_ssh\_pass=1234 & ansible\_become=yes  
1.2.3.5 ansible\_ssh\_user=user2, ansible\_ssh\_pass=5676 & ansible\_become=yes  
1.2.3.6 ansible\_ssh\_user=user3, ansible\_ssh\_pass=9732 & ansible\_become=yes  
1.2.3.7

[web:vars]  
ansible\_ssh\_user=user3,   
ansible\_ssh\_pass=9732,  
ansible\_become=yes

[app] this group we have to login as UN & key & sudo with passwd  
2.2.3.4 ansible\_user=user1, **ansible\_ssh\_private\_key\_file**=node.pem & ansible\_become=yes **ansible\_become\_password**=passwd123   
3.2.3.5 ansible\_user=user2, **ansible\_ssh\_private\_key\_file**=node.pem & ansible\_become=yes **ansible\_become\_password**=passwd456   
4.2.3.6  
5.2.3.7

[app:vars]  
ansible\_user=user1,   
**ansible\_ssh\_private\_key\_file**=node.pem,  
ansible\_become=yes,   
**ansible\_become\_password**=passwd123

[log] this group we have to login as UN & key & sudo with diff user & passwd  
6.2.3.4 ansible\_user=user1, **ansible\_ssh\_private\_key\_file**=node.pem & ansible\_become=yes & **ansible\_become\_user**=user2 & **ansible\_become\_password**=passwd123  
7.2.3.5  
8.2.3.6  
9.2.3.7

[db] this group we have to login as UN & passwd & sudo with diff user & paswd  
6.2.3.1 ansible\_ssh\_user=user1, ansible\_ssh\_pass=1234 & ansible\_become=yes **& ansible\_become\_user**=user2 & **ansible\_become\_password**=passwd123  
7.2.3.2  
8.2.3.3  
9.2.3.4

Real time example:  
[web]  
52.146.89.226

[web:vars]  
ansible\_ssh\_user=ec2-user,   
ansible\_ssh\_pass=9732,  
ansible\_become=yes  
myname\_inv= Abhi in inventory

Command to execute playbook now: ansible-playbook var.yaml -i inventory

- vars in inventory file  
- Variable using host var: variables used for a specific host  
- Variable using group var

Tools (ansible, ansible-playbook)

Example:   
---  
- name: Update web servers  
 hosts: web  
 vars:  
 myname: “Abhishek”  
 age: “26”  
 packagename: “httpd”  
 servicename: “httpd”  
 vars\_files:  
 - “vars.yaml”  
 vars\_prompt:  
 - name: “version”  
 prompt: “Which version do you want to install?”  
 private: no

tasks:   
 - name: Ensure that apache is installed  
 ansible.builtin.yum:   
 name: “{{ packagename }} “  
 state=present  
 - name: Copying index.html  
 copy:  
 src: /opt/index.html  
 dest: /var/www/html/index.html  
 - name: Start apache services  
 ansible.builtin.service:   
 name: “{{ servicename }}”  
 state=started  
 - name: Print return information from previous task  
 ansible.builtin.debug:  
 var: myname  
 - name: include default step variable  
 include\_vars: tasks\_var.yaml  
 - name: Print return information from previous task  
 ansible.builtin.debug:  
 var: “My name is {{ myname }} and my age is {{ age }}”  
 - name: Print return information from previous task  
 ansible.builtin.debug:  
 var: version  
 - name: Ansible register variable basic example  
 shell: “find \*.txt”  
 args:   
 chdir: “/home/ec2-user/specify any folder exists in this path”  
 register: find\_output  
 - debug:  
 var: find\_output # prints lines based on no. of files with .txt in this case  
 or  
 var: find\_output.stdout\_lines[0] # prints first line based on no. of files with .txt in this case  
 - debug:  
 var: myname\_inv

vi vars.yml  
 myname: “Abhishek R from external var file”  
 age: “26”  
 packagename: “httpd”  
 servicename: “httpd”

Command: ansible-playbook var.yaml -i inventory -u ec2-user -k -b -vvvv  
-vvvv=> detailed logs will be printed in console

vi tasks\_var.yaml  
 myname: “Abhishek Rangaswamy from task var file”  
 age: “26”  
 packagename: “httpd”  
 servicename: “httpd”

mkdir group\_vars  
mkdir host\_vars  
cd group\_vars  
touch all web app db  
vim all  
myname\_inv: “Abhi is group var all”  
Note: if we declare myname\_inv in both all and web, variable precesence web >> all.

cd ..  
cd host\_vars  
vim 52.146.89.226  
myname\_inv: “Abhi is host var 52.146.89.226”  
Note: ip should be same as host ip mentioned in inventory file.  
Note: if we declare myname\_inv in both group\_var (all, web), host\_var, inventory variable precesence host\_var >> inventory >> group\_var (web) >> group\_var (all).

**Command line variable:** ansible-playbook var.yaml -i inventory -e myname\_inv=abhishekCommandline  
or  
create a file last.yaml  
vim last.yaml  
myname\_inv: abhishekBhai   
ansible-playbook var.yaml -i inventory -e @last.yml

Note: Variable precedence  
- Embed using include vars into task >> - Embed external variables files >> - Embed variables in playbook  
- Embed external variables files >> User prompt  
- inventory vars << Embed using include vars into task  
- inventory vars << - Embed external variables files   
- inventory vars << - Embed variables in playbook

---------------------------------- Fact variable -----------------------------------------

- fact of remote host  
- System/Env details of the host

Manually -  
 Module setup  
Whenever u run playbook – fact variable gather automatically meaning by default gather facts task will be executed first.

-----------------------------------------------------------------------  
Conditioning and looping with playbook  
-----------------------------------------------------------------------

When os = ubuntu  
 run apt module  
When os = RHEL  
 run yum module  
When os = windows  
 run ps module

Example:  
---  
- name: Update web servers  
 hosts: web  
 vars:  
 myname: “Abhishek Rangaswamy”

tasks:   
 - name: Ensure that apache is installed  
 ansible.builtin.yum:   
 name: httpd  
 state: latest  
 When: ansible\_os\_family == “RedHat”  
 or  
 When: (ansible\_os\_family == “RedHat” and ansible\_system == “Linux”)  
 - name: Copying index.html  
 copy:  
 src: /opt/index.html  
 dest: /var/www/html/index.html  
Note: grep linux with command ansible localhost -m setup | grep -i linux  
 - name: Start apache services  
 ansible.builtin.service:   
 name: httpd  
 state: started  
 When:   
 - ansible\_os\_family == “RedHat”  
 - ansible\_distribution\_major\_version == “7”  
 - ansible\_system == “Linux”  
 - name: Ansible when My name is Abhishek Rangaswamy  
 debug:  
 msg: “My name is Abhishek Rangaswamy”  
 When: myname == “Abhishek Rangaswamy”  
 - name: Ansible print when ubuntu  
 debug:  
 msg: “I am ubuntu”  
 When: ansible\_os\_family == “Debian”  
 - name: Ansible print when My name is not equals to Abhishek Rangaswamy  
 debug:  
 msg: “My name is not equals to Abhishek Rangaswamy”  
 When: myname != “Abhishek Rangaswamy”

Looping:  
Example:  
---  
- name: Update web servers  
 hosts: web  
 vars:  
 listvar1:  
 - ‘a’  
 - ‘b’  
 - ‘c’

tasks:   
 - name: Ensure that apache is installed  
 ansible.builtin.yum:   
 name: {{ item }}  
 state: latest  
 with\_items:  
 - git  
 - httpd  
 - ntp  
 - name: Copying index.html  
 copy:  
 src: /opt/index.html  
 dest: /var/www/html/index.html  
 ansible.builtin.service:   
 name: httpd  
 state: started  
 - command: echo {{ item }}  
 loop: [0, 2, 4, 6, 8, 10]  
 - command: echo {{ item }}  
 loop: [0, 2, 4, 6, 8, 10]  
 when: item > 5  
 - name: Remove users chuck and craig from the system  
 user:  
 name: “{{ item }}”  
 state: absent  
 remove: yes  
 with\_items:  
 - chuck  
 - craig  
 - shell: echo “nested test a={{ item[0] }} b={{ item[1] }} c={{ item[2] }}”  
 with\_nested:  
 - [‘red’, ‘blue’, ‘green’]  
 - [1, 2, 3]  
 - [‘up’, ‘down’, ‘strange’]  
 - shell: echo “nested test a={{ item[0] }} b={{ item[1] }}”  
 with\_nested:  
 - listvar1  
 - [1, 2, 3]

-----------------------------------------------------------------------

Declaring vars:  
playbook  
inventory  
command line  
And we use above vars in playbook tasks  
-----------------------------------------------------------------

Can we use these vars in some of the files used in playbook??  
Interpolate  
----------------------  
template  
- file must be .j2 ext  
- you must use a module called “template”  
Example:  
---  
- name: Update web servers  
 hosts: web  
 vars:  
 myname: “Abhishek R”   
 httpport: 8090

tasks:   
 - name: Ensure that apache is installed  
 ansible.builtin.yum:   
 name: {{ item }}  
 state: latest  
 with\_items:  
 - git  
 - httpd  
 - ntp  
 - name: Apacher server started  
 ansible.builtin.service:   
 name: httpd  
 state: started  
 - name: Copying index.html  
 copy:  
 src: /opt/index.html  
 dest: /var/www/html/index.html

- name: Copying index.html  
 template:  
 src: index.html.j2  
 dest: /var/www/html/index-template.html  
 - name: Apacher server stopped  
 ansible.builtin.service:   
 name: httpd  
 state: stopped  
 - name: Template for httpd.conf  
 template:  
 src: httpd.conf.j2  
 dest: /etc/httpd/conf/httpd.conf   
 - name: Apacher server restarted  
 ansible.builtin.service:   
 name: httpd  
 state: restarted  
Note: When we change apache static html content -> httpd restart not required.  
 When we change apache conf -> httpd restart is not required.

project:  
- http should run on 90  
- No restart should be done on static  
- Restart must be done at conf changes

Handlers: A section in playbook similar to tasks section  
It contains multiple tasks  
The task inside handlers would execute when you call from play tasks.  
But one condition - only when calling play tasks is changed == True

Example:  
---  
- name: Update web servers  
 hosts: web  
 vars:  
 myname: “Abhishek R”   
 httpport: 8090

tasks:   
 - name: Ensure that apache is installed  
 ansible.builtin.yum:   
 name: {{ item }}  
 state: latest  
 with\_items:  
 - git  
 - httpd  
 - ntp  
 - name: Apacher server started  
 ansible.builtin.service:   
 name: httpd  
 state: started  
 - name: Copying index.html  
 copy:  
 src: /opt/index.html  
 dest: /var/www/html/index.html

- name: Copying index.html  
 template:  
 src: index.html.j2  
 dest: /var/www/html/index-template.html  
 - name: Template for httpd.conf  
 template:  
 src: httpd.conf.j2  
 dest: /etc/httpd/conf/httpd.conf   
 notify:   
 - Apacher server restarted

Handlers Task  
 handlers:  
 - name: Apacher server restarted  
 ansible.builtin.service:   
 name: httpd  
 state: restarted

==============================================

We majorly use below components in project  
multiple playbooks  
multiple vars  
Templates  
Files  
Handlers

How to give right structure?  
like easy to share -> run -> debug -> documents  
Best practices:  
Scenario-1:  
--------------- main.yaml -------------------  
- hosts: all  
 tasks:  
 - debug:  
 msg: task1

- include: stuff.yaml

------------------ stuff.yaml ---------------------

---  
- name: http service state  
 service: name=httpd state=started enabled=yes

Scenario-2:  
- hosts: all  
 tasks:  
 - debug:  
 msg: task1

- include: stuff.yaml

------------------ stuff.yaml ---------------------

- hosts: all  
 tasks:  
 - debug:  
 msg: task1  
 - name: http service state  
 service: name=httpd state=started enabled=yes  
  
================================================================

Managing  
multiple playbooks  
multiple vars  
Templates  
Files  
Handlers

Solution: Roles  
It is a directory structure to manage :  
multiple playbooks - tasks  
multiple vars  
Templates  
Files  
Handlers

How to create a role??  
ansible-galaxy  
Note: ansible-galaxy -h => displays options  
ansible-galaxy role -h => displays options

Commands  
Examples:  
ansible-galaxy role init web  
===> role web created

================================================================

Inventory file:  
Groups of groups:  
[web]  
4.4.4.4  
5.5.5.5

[db]  
4.4.4.5  
5.5.5.6

[common]  
4.4.4.1  
5.5.5.2

[gog:children]  
web  
db

===================================================

Standard directory structure  
host\_vars [dir]  
4.5.5.6 [File]  
 age: fdsfds  
 name: fdsfds  
5.6.7.8 [File]  
 age: fdsfds  
 name: fdsfds  
4.5.5.2 [File]  
5.6.7.0 [File]  
group\_vars [dir]  
web [File]  
 age: fdsfds  
 name: fdsfds  
db [File]  
all [File]

=======================================================

Multi – environment directory structure  
- production  
 - group\_vars  
 - all  
 - db  
 - host\_vars  
 - web1  
 - inventory\_prod

- test  
 - group\_vars  
 - all  
 - db  
 - host\_vars  
 - web1  
 - inventory\_test

======================================================

**Variables in ansible:**  
**Custom variable before run**  
 Inventory file  
 ==================  
filename – inventory  
-----------------------------  
[web]  
5.5.5.5 age = ageininventory\_behaviour  
4.4.5.5

[web:vars]  
age=ageininventory  
name=nameininventory  
compname=compnameininventory  
 playbook  
 - in built with playbook  
 age=abhi\_playbook\_in  
 myname  
 companyname  
In playbook,  
declare a variable  
Use it – {{ age }} or {{ myname }} or {{ companyname }}  
How & where?? = module = debug module (google search for ansible module debug)  
====================================================================  
 - call a external var file  
 age=abhi\_playbook\_in  
 myname  
 companyname  
In playbook,  
declare a variable  
Use it – {{ age }} or {{ myname }} or {{ companyname }}  
How & where?? = module = debug module (google search for ansible module debug)  
E.g:  
vars\_files:  
 - “vars/external\_vars.yml”  
 - [“vars/{{ factor\_operatingsystem }}.yml”, “vars/defaults.yml”]  
 ansible.cfg  
 command prompt – Adhoc commands  
**Constant variable** – Pre exist  
 Fact module  
command: ansible 127.0.0.1 -m setup => gets complete info of host machine.  
**Note:** Default module for adhoc command is – command  
command: ansible 127.0.0.1 -m setup | grep os => gets specific info of host machine.  
**During runtime** Modules  
 vars\_prompt = function  
 register  
 --------------------  
Example:  
- hosts: all  
 tasks:  
 - name: Ansible register variable basic example  
 shell: “find \*.txt”  
 args:  
 chdir: /opt  
 register: find\_output  
  
 - debug:  
 var: find\_output

**Types of Variables in Ansible:   
Example (INI inventory):**[web]

web01 ansible\_host=192.168.1.10 app\_port=8080

**Usage in playbook:**  
- name: Start app on host-defined port

hosts: web

tasks:

- name: Print port

debug:

msg: "App running on port {{ app\_port }}"

**2. Playbook-Level Variables**- name: Example with vars

hosts: all

vars:

app\_name: "myapp"

tasks:

- name: Print app name

debug:

msg: "Deploying {{ app\_name }}"

3. **Host and Group Variables (host\_vars / group\_vars)**inventory/

├── group\_vars/

│ └── webservers.yml

├── host\_vars/

│ └── web01.yml

**group\_vars/webservers.yml**  
region: us-west  
**host\_vars/web01.yml**  
app\_port: 9090

4. **Facts**Collected automatically or defined using the set\_fact module.  
- name: Set custom fact

hosts: all

tasks:

- set\_fact:

release: "v1.2"

- debug:

msg: "Release version is {{ release }}"

5. **Role Variables**

Defined inside a role in:

defaults/main.yml → lowest priority

vars/main.yml → higher priority  
Example:  
# roles/myrole/defaults/main.yml

app\_env: "production"

**6. Extra Vars (Command Line)**

Highest priority. Useful for overrides.  
Example:

ansible-playbook deploy.yml -e "env=staging version=2.1"  
In playbook:  
msg: "Deploying version {{ version }} to {{ env }} environment"

**7. Prompting for Variables**

Useful for interactive playbooks.  
- name: Prompt for input

hosts: localhost

vars\_prompt:

- name: "db\_password"

prompt: "Enter database password"

private: yes

tasks:

- debug:

msg: "Password received"

8. **Task vars: Variables defined within a single task, applying only to that task.**  
- name: Restart application with overridden name

hosts: appservers

tasks:

- name: Restart app service with a custom name

vars:

service\_name: myapp-custom

service:

name: "{{ service\_name }}"

state: restarted  
  
 service\_name is only valid inside that specific task.

 It's useful when the variable doesn't need to be reused elsewhere.

9. **block vars:** Variables defined inside a block, applying to **all tasks inside the block only**.  
- name: Deploy configuration based on OS

hosts: all

tasks:

- name: Deploy on Debian

block:

- name: Install apt package

apt:

name: "{{ package\_name }}"

state: present

- name: Copy config file

copy:

src: "{{ config\_file }}"

dest: "/etc/{{ package\_name }}/conf.d/default.conf"

vars:

package\_name: nginx

config\_file: debian.conf

when: ansible\_os\_family == "Debian"

- name: Deploy on RedHat

block:

- name: Install yum package

yum:

name: "{{ package\_name }}"

state: present

- name: Copy config file

copy:

src: "{{ config\_file }}"

dest: "/etc/{{ package\_name }}/conf.d/default.conf"

vars:

package\_name: nginx

config\_file: redhat.conf

when: ansible\_os\_family == "RedHat"

**Explanation:**

* The vars: inside each block are scoped to that block only.
* You avoid polluting global variable space.
* It's **cleaner and easier to maintain** than repeating vars: in each task inside the block.

**Real-Time Example: Dynamic Web Deployment**  
- name: Deploy Nginx based on environment

hosts: webservers

vars:

nginx\_package: "{{ 'nginx' if ansible\_os\_family == 'Debian' else 'nginx-full' }}"

app\_port: 8080

tasks:

- name: Install nginx

apt:

name: "{{ nginx\_package }}"

state: present

when: ansible\_os\_family == "Debian"

- name: Ensure app is running

service:

name: nginx

state: started

- name: Show deployed info

debug:

msg: "Nginx is deployed on {{ inventory\_hostname }} using port {{ app\_port }}"

**Variable Precedence (High to Low):**

| **Priority** | **Source** |
| --- | --- |
| 1 | Extra vars (-e) |
| 2 | Task vars |
| 3 | Block vars |
| 4 | Role vars |
| 5 | Playbook vars |
| 6 | Inventory variables |
| 7 | Facts |
| 8 | Role defaults |

**Scenario: Web Server Deployment (Apache/Nginx)**

Your organization manages multiple environments (dev, staging, prod). You're deploying a web server and want the following:

* The **default server** is nginx.
* For production, the server should be apache2.
* For one specific task, you want to temporarily override it to httpd.
* If the deployment is run with -e, that value should take the highest priority.

**Variable Sources & Precedence (1=Highest):**

| **Priority** | **Source** | **Where Defined** |
| --- | --- | --- |
| 1 | Extra Vars (-e) | -e "web\_server=custom\_server" |
| 2 | Task Vars | Defined inside a task |
| 3 | Block Vars | Defined inside a block |
| 4 | Role Vars | roles/webserver/vars/main.yml |
| 5 | Playbook Vars | vars: block in the playbook |
| 6 | Inventory Vars | host\_vars/prod.yml or group\_vars/prod.yml |
| 7 | Facts | Collected by Ansible or set with set\_fact |
| 8 | Role Defaults | roles/webserver/defaults/main.yml |

**Directory Structure:**  
ansible-web-deploy/

├── inventory/

│ ├── hosts.ini

│ └── group\_vars/

│ └── prod.yml

├── roles/

│ └── webserver/

│ ├── defaults/

│ │ └── main.yml

│ ├── vars/

│ │ └── main.yml

│ ├── tasks/

│ │ └── main.yml

├── playbook.yml  
  
**inventory/hosts.ini**[prod]

prod-server ansible\_host=192.168.1.100 ansible\_user=ec2-user

**inventory/group\_vars/prod.yml (6 - Inventory Variables)**web\_server: apache2

**roles/webserver/defaults/main.yml (8 - Role Defaults)**web\_server: nginx

**roles/webserver/vars/main.yml (4 - Role Vars)**web\_server: nginx-role

**roles/webserver/tasks/main.yml**  
- name: Print from role vars

debug:

msg: "Role-level variable says: {{ web\_server }}"

- name: Install web server based on variable

block:

- name: Install web server (block vars)

package:

name: "{{ web\_server }}"

state: present

- name: Start service (override in task vars)

service:

name: "{{ web\_server }}"

state: started

vars:

web\_server: httpd # 2 - Task Vars

vars:

web\_server: nginx-from-block # 3 - Block Vars

**playbook.yml (5 - Playbook Vars + 7 - Facts + 1 - Extra Vars)**  
- name: Deploy web server using variable precedence

hosts: prod

become: yes

vars:

web\_server: nginx-playbook # 5 - Playbook Vars

pre\_tasks:

- name: Set a fact dynamically (7 - Facts)

set\_fact:

web\_server: fact-nginx

roles:

- webserver

tasks:

- name: Final result - which web server is used

debug:

msg: "FINAL: Web server is {{ web\_server }}"

**Execution Example with -e:**  
ansible-playbook -i inventory/hosts.ini playbook.yml -e "web\_server=httpd" -u ec2-user  
**Expected Output Behavior:**

| **Variable Level** | **Value Used** | **Notes** |
| --- | --- | --- |
| Extra Vars | overridden-nginx | CLI override, highest priority |
| Task Vars | httpd | Used in service task only |
| Block Vars | nginx-from-block | Used inside install task |
| Role Vars | nginx-role | Role-level |
| Playbook Vars | nginx-playbook | Defined in playbook |
| Inventory Vars | apache2 | From group\_vars/prod.yml |
| Facts | fact-nginx | Dynamically set |
| Role Defaults | nginx | Fallback if nothing else |

**Real-Time Use Logic:**

Imagine you're doing a deployment like this:

* **Default across all roles** → nginx
* **Production override from inventory** → apache2
* **Playbook-specific override** → nginx-playbook
* **Temporary testing block override** → block-nginx
* **Urgent fix for one task** → httpd
* **Final override via Jenkins or CLI** → -e web\_server=overridden-nginx

Only the highest-priority source gets used at any time.

============================================================

How to do config management in multiple hosts + multiple tasks  
-------------------------------------------------------------------------------------  
inventory  
[web]  
35.154.171.217 intevent\_var=helium\_123  
13.232.223.163

Playbook  
---  
- hosts: webservers  
 sudo: yes  
 vars:  
 packname: httpd  
  
 tasks:  
 - name: Ensure that apache is installed  
 yum: name={{ pacjname }} state=present  
 when: ansible\_os\_family == “Redhat”  
  
 - name: Start apache services  
 service: name=httpd enabled=yes state=present  
  
 - debug:  
 var: intevent\_var  
 var: packname

Command: ansible-playbook all -i inventory

================================================================

What is Role in ansible??  
Roles is a way to package the script (playbook) + All dependent file + dependencies on another roles + manage all playbook.

where can we find roles free? => ansible-galaxy (15k+ roles)

Directory structure:  
roles  
 1. builders  
 2. Server-common  
 a) defaults  
 b) files  
 c) handlers  
 d) meta => name of the role and dependencies for the role  
 e) tasks  
 f) templates  
 g) vars  
 3. webservers

How can I create my own role?  
ansible-galaxy init abhishek.role1 --offline  
===================================================================

Ansible is a powerful automation tool used for:  
Configuration management like OS installation/updates, package installation/updates, system dependencies, application dependencies...  
Provisioning infrastructure  
Application deployment in the CI-CD pipeline phase  
Network Automation  
Orchestration

It’s written in python and uses YAML (in the form of “Playbooks”) to describe automation jobs.

**Why Ansible??**

Agentless: No need to install software/agents on the target machines. Uses ssh. Simpler than puppet/chef.  
Simple YAML Syntax: Uses human-readable **YAML playbooks** - easy to write and understand.  
Idempotent: Re-running the same playbook gives consistent results (doesn't redo already-complete tasks).  
Push-based: Central server pushes configurations to the target nodes. Easier to control.  
Cross-platform: Works with Linux, macOS, Windows, cloud services (AWS, Azure, GCP), etc.  
Extensible: Integrates well with Docker, Kubernetes, cloud providers, Jenkins, Vault, and more.  
Great for DevOps: Automates repetitive tasks like server setup, app deployment, and config updates.  
  
**Real Use Cases:**Install packages on multiple servers  
Configure Nginx/Apache or databases  
Deploy apps to cloud servers  
Rotate secrets via Vault  
Setup Kubernetes clusters  
Automate CI/CD pipelines with Jenkins

**Example Ansible Playbook (to install Apache on a remote server:**- name: Install Apache on web servers

hosts: webservers

become: yes

tasks:

- name: Install Apache

apt:

name: apache2

state: present

- name: Start Apache

service:

name: apache2

state: started

enabled: yes

**How to install ansible in amazonLinux machine??**  
  
# 1. Enable the EPEL repository

sudo amazon-linux-extras enable epel

# 2. Install the EPEL release package

sudo yum install -y epel-release

# 3. Install Ansible

sudo yum install -y ansible

# 4. Verify installation

ansible --version

**How to setup passwordless authentication from control node EC2 instance to manage nodes EC2 instances??**

Note: Login as root user in control node and

Step-1: ssh-keygen

Step-2: cat ~/.ssh/id\_rsa.pub >> copy the key  
Step-3: Login to manage node ec2 instance  
Step-4: Make sure user is ec2-user  
Step-5: cd .ssh/  
Step-6: vim authorized\_keys >> paste here and save.  
Step-7: ssh-copy-id -i /root/.ssh/id\_rsa.pub ec2-user@<public ip of manage node> // Not required  
Step-8: ssh [ec2-user@13.233.98.229](mailto:ec2-user@13.233.98.229)

Ansible Adhoc commands:

**Syntax:** anisble -i <path of inventory file> -m <module name> <all/specific manage node/grouped nodes>

1. ansible -i inventory.ini -m ping all
2. ansible -i inventory.ini -m ping db/app => specify grouped nodes
3. ansible -i inventory.ini -m shell -a “ls /etc” all  
   -a => attribute

What is inventory file?  
In **Ansible**, the **inventory file** is a key configuration file that lists the **hosts (servers)** Ansible will manage and organize them into **groups**.  
Default location: /etc/ansible/hosts  
**Note:** we can have inventory file anywhere in the server.  
E.g: inventory.ini => static inventory file  
[webservers]  
web1.example.com  
web2.example.com

[dbservers]  
db1.example.com ansible\_user=ubuntu ansible\_port=22

What is YAML??  
YAML is human readable data serialization language and template language where data will be in the specific format. E.g: YAML, JSON etc.,

YAML Syntax:  
**Strings, Numbers and Booleans:**string: Hello, World!  
number: 42  
boolean: true

**List**fruits:  
 - Apple  
 - Orange  
 - Banana

**Dictionary**person:  
 name: John Doe  
 age: 30  
 city: New York

**List of dictionaries**YAML allows nesting of lists and dictionaries to represent more complex data.  
family:

parents:

- name: Jane

age: 50

- name: John

age: 52

children:

- name: Jimmy

age: 22

- name: Jenny

age: 20

**Playbook**A **Playbook** is a YAML file that defines a series of actions to be executed on managed nodes. It contains one or more "plays" that map groups of hosts to roles.

**Inventory file:**  
It is a file which contains collection of IP addresses of ansible remote servers (ARS).

**Module:**  
Unit of code which would execute in ARS. (might require mandatory parameters or optional parameters)  
E.g.: copy, file, service, package  
1000’s of built-in core modules are available in ansible.

**plugins:**code which empowers various functionality of ansible.

**Ansible Adhoc commands:**Suppose if I want to execute one task in local maction or in one remote machine or in multiple remote machines using inventory file.

Ansible builtin – methods: <https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html>

**first-playbook.yaml**

--- # YAML document start

- hosts: all # Apply this playbook to all hosts in the inventory

become: true # Run tasks with sudo/root privileges

tasks: # Start of the list of tasks to execute

- name: Install Apache HTTPD # Description of the task

ansible.builtin.yum: # Use the built-in yum module to manage packages (for RHEL-based systems)

name: httpd # Name of the package to install (httpd = Apache on RHEL)

state: present # Ensure the package is installed

update\_cache: yes # Update the package metadata cache before installing

- name: Copy file with owner and permission # Description of the task

ansible.builtin.copy: # Use the built-in copy module to copy files to remote hosts

src: index.html # Source file on control node (local system running Ansible)

dest: /var/www/html/index.html # Destination path on remote host

owner: root # Set file owner to root

group: root # Set file group to root

mode: '0644' # Set file permissions to read/write for owner, read for group and others

**What is a Role in Ansible?**

A role is a set of tasks, handlers, variables, templates, files, and defaults structured in a standard directory layout. It allows you to separate your configuration logic into reusable components.

**Basic Structure of a Role:**

myrole/

├── defaults/ # Default variables

│ └── main.yml

├── files/ # Static files to be copied (e.g., .conf, .html)

│ └── somefile.conf

├── handlers/ # Handlers (e.g., restart a service)

│ └── main.yml

├── meta/ # Metadata about the role (dependencies, author, etc.)

│ └── main.yml

├── tasks/ # Main list of tasks

│ └── main.yml

├── templates/ # Jinja2 templates (e.g., .j2 files)

│ └── config.j2

├── tests/ # Role testing inventory/playbooks

│ └── test.yml

└── vars/ # Role-specific variables

└── main.yml

**When Should You Use Roles?**

* When your playbook starts getting large
* When you need to reuse logic across multiple playbooks or projects
* When you work in teams (roles promote modular design)
* When you want to share code using **Ansible Galaxy**

**Benefits of Using Roles**

* **Reusability**: Use the same role in multiple playbooks
* **Modularity**: Clean separation of logic
* **Maintainability**: Easier to debug and update
* **Scalability**: Manage large projects easily

Example:

cd /etc/ansible/roles

yum install tree -y

ansible-galaxy init apache --offline

project/

├── inventory

├── playbook.yml

└── roles/

└── webserver/

├── defaults/

│ └── main.yml

├── files/

│ └── apache.conf

├── handlers/

│ └── main.yml

├── meta/

│ └── main.yml

├── tasks/

│ └── main.yml

├── templates/

│ └── index.html.j2

├── tests/

│ └── test.yml

└── vars/

└── main.yml

**inventory.ini file:**

[web]

IP address of target nodes/servers   
or

[web]

localhost ansible\_connection=local (To test in control node)

or

[web]

webserver ansible\_host=192.168.1.10

E.g: 192.168.56.10 ansible\_user=ec2-user ansible\_ssh\_private\_key\_file=~/.ssh/my-key.pem

**playbook.yml**

---

- hosts: web

become: true

roles:

- webserver

**roles/webserver/defaults/main.yml:**

---

apache\_port: 80

**roles/webserver/files/apache.conf:**

# Dummy Apache configuration

Listen 0.0.0.0:80

ServerName localhost

**roles/webserver/handlers/main.yml:**

---

- name: restart apache

service:

name: httpd

state: restarted

**roles/webserver/meta/main.yml:**

---

galaxy\_info:

author: Abhishek Rangaswamy

description: Simple Apache role

license: MIT

min\_ansible\_version: 2.9

platforms:

- name: EL

versions:

- 1

**roles/webserver/tasks/main.yml:**

---

- name: Install Apache

yum:

name: httpd

state: present

- name: Copy apache config file

copy:

src: apache.conf

dest: /etc/httpd/conf/httpd.conf

backup: yes

notify: restart apache

- name: Deploy website

template:

src: index.html.j2

dest: /var/www/html/index.html

notify: restart apache

- name: Ensure Apache is running and enabled

service:

name: httpd

state: started

enabled: true

**roles/webserver/templates/index.html.j2:**  
<!DOCTYPE html>

<html>

<head>

<title>My Ansible Site</title>

</head>

<body>

<h1>Welcome to {{ ansible\_hostname }}!</h1>

<p>Apache is running on port {{ apache\_port }}.</p>

</body>

</html>

**roles/webserver/tests/test.yml:**  
---

- hosts: web

become: true

roles:

- webserver

**roles/webserver/vars/main.yml:**---

httpd\_package: httpd

From the project/ directory:

**Run command:** ansible-playbook -i inventory playbook.yml