Ansible

Contents

Ansible

https://docs.ansible.com/ansible/latest/community/index.html → **Ansible Doc. Page**
https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html → **Modules**
https://galaxy.ansible.com/ui/ → **open-source website for Ansible roles**

1) Ansible:

- Ansible is an open-source IT automation engine that can be used to automate many IT processes, including:
  - **Configuration management:** Automating storage, servers, and networking
  - **Application deployment:** Supporting application deployment and system updates
  - **Orchestration:** Making it easier to orchestrate complex workflows.
- Ansible is widely used in DevOps for continuous integration and continuous deployment (CI/CD) pipelines, cloud provisioning, and managing infrastructure as code (IaC). Its simplicity, flexibility, and extensive community support make it a popular choice for IT automation.
- Ansible understands YAML Language and indentation is very important in this language.

Key Features:

(a) Agentless architecture: Doesn't require agents on worker nodes, simplifying setup and maintenance.

(b) Build on Python: Build using Python and hence provides a lot of python functionality.

(c) Declarative language (YAML): Uses human-readable YAML format for playbooks, making it easy to write and understand.

(d) SSH-based: Primarily uses SSH protocol for secure connection between master and worker nodes.

(e) Push based architecture: Follows push-based architecture for sending configuration.

(f) Very easy and fast setup.

(g) Idempotency: Ensures tasks are executed consistently, regardless of how many times they are run.

(h) Modularity: Offers a vast library of modules for various tasks, from package management to configuration management.

Practical:

i) Create 3 Amazon linux instances, one for master and 2 for workers

ii) Connect to master server (ssh -i pemfilename.pem ec2-user@PublicIP)

sudo yum update -y	# Update latest patches and other updates
sudo yum install ansible -y	# installing ansible
ansible --version	# validating, even python will be installed with Ansible installed as Ansible is build using Python
sudo find / -iname "ansible"	# checking/finding where ansible is present
cd /etc/ansible **ls**	# folder where all configuration related to ansible are present
Host file is where all hosts server (worker nodes) will be present	# contains list of hosts (hosts inventory file/worker nodes details)
vim hosts 191.122.3.4 [app_server] 108.91.0.22 108.91.23.34 [db_server] 10.0.0.1 10.2.33.22	# can give worker nodes to groups like app_server and db_server so we can use for particular tasks like for apps and for database purpose. We should make entry here/ assign any worker nodes in this file (host file = registry book). If host file not present then create file as hosts in ansible folder (**/etc/ansible**)

2) *Modules:*

- Ansible modules are the building blocks of Ansible playbooks. They are reusable, standalone scripts that Ansible runs to perform specific tasks on managed hosts.
- Each module is designed to handle a particular function, such as installing software, copying files, or managing services.
  - **apt:** Install/remove packages using this module
  - **copy:** copy the files from local (master) to hosts (worker nodes) server.
  - **file:** create file, directory or link
  - **service:** start, stop and restart services

➔ *Password and Passwordless authentication between servers (Master-Worker):*

Setting up password authentication
Master-server: i) create user sudo adduser ansible ii) create password for user sudo passwd ansible (passwd: ansible123) iii) configure to make user password sudo vim /etc/ssh/sshd_config PasswordAuthentication yes PermitRootLogin yes (remove hash) sudo service sshd restart iv) add user to sudo group sudo vim /etc/sudoers Add after root, ansible ALL=(ALL) NOPASSWD: ALL save v) change hostname to identify sudo hostnamectl set-hostname Master-Node
Similarly, do for other 2 servers (worker servers) (i) sudo adduser ansible (ii) sudo passwd ansible (passwd: ansible123) (iii) sudo vim /etc/ssh/sshd_config PasswordAuthentication yes (remove hash) PermitRootLogin yes (remove hash) sudo service ssh restart (iv) add user to sudo group sudo vim /etc/sudoers Add after root, ansible ALL=(ALL) NOPASSWD: ALL save (v) Change hostname to identify sudo hostnamectl set-hostname worker 1 and 2 respectively (v) validation: su ansible

Passwordless authentication
Connect to Master-Node: (i) create key ssh-keygen (ii) copy ID of Master server to worker1 server ssh-copy-id ansible@worker1IP (iii) copy ID of Master server to worker2 server ssh-copy-id ansible@worker2IP Login to worker 1 and worker 2 from master server without password (Master can communicate with worker server without password as master node fingerprint has been added to worker node) ssh ansible@worker1PublicIP ssh ansible@worker2PublicIP

3) *Ansible ad hoc commands:*

Ansible ad hoc commands are used to execute a single task on one or more managed nodes without the need for a full playbook. They are ideal for quick, one-time actions.

(i) create host file in master node

sudo vim hosts

[app_server]

Add 2 worker publicIP

save

(ii) ping: checking server status of hosts for pong if server responds

(a) ansible all -m ping	#checking server status of all hosts -m = module
(b) ansible app_server -m ping	# checks app_server hosts only
(c) ansible app_server -m shell -a "uptime"	#Check current time, the system's uptime, the number of users, and the load average
(d) ansible app_server -m shell -a "df -h"	#check information about total space and available space on a file system

(e) ansible app_server -m shell -a "mkdir demo_ansible"	#Create folder in both workers
(f) ansible app_server -m yum -a "name=httpd state=present" -b	#install httpd, checks present or not, if not then will install httpd
(g) ansible app_server -m shell -a "service httpd status"	#check status of httpd
(h) ansible app_server -m shell -a "service httpd started"	#to start httpd
(i) validate: Go to AWS --> add inbound rule for http for worker node Copy paste serverPublicIP in browser	#check in browser
Pem file authentication	
(j) check ping for new server where no password configurations are done Add new server IP in hosts in Master server sudo vim /etc/ansible/hosts [new] --> new group New_serverPublicIP save ansible new -m ping --private-key test.pem -u ec2-user	#check ping of instance where no configuration is done (using pem file authentication)
Password authentication	
ansible app_server -m shell -a "rm -r demo_ansible" -k	#removes folder
Passwordless authentication	
ansible app_server -m shell -a "mkdir demo_ansible"	#creates folder

[***Note:*** user should be ansible
su ansible → to change user (ec2-user to ansible user)]

4) 30 Modules

SL No	Modules	Description
1	ping	use to check the server response
2	setup	gather facts about remote hosts
3	yum	install/remove packages for redhat or centos based linux os
4	apt	install/remove packages for Debian/ubuntu based linux os
5	package	install/remove packages for linux distribution
6	service	used for start/stop/reload/restart the services
7	pip	python packages
8	npm	node package manager install
9	file	used for create/delete/modify files/directory/link
10	command/shell	to run your linux commands
11	script	run local script in remote node
12	copy	used for copy the files/directory from local (Control machine) to remote
13	fetch	used to copy from remote to local
14	user/group	creation of linux user and group
15	template	used to run dynamic files
16	cron	set the cron exection
17	archive	used to compress
18	unarchive	used for uncompress
19	find	to find the files/directories
20	systemd	used to manage the system reload action
21	git	used to clone your repos to local
22	lineinfile	used to find/replace the lines in a file
23	blockinfile	complete block replace/modify
24	debug	used to print the message or variables values
25	register	used to register the state of task
26	import_task	used to import the tasks
27	include_var	used to include your varible files
28	syncronize	used to local to remote

29	wait_for	wait your taks exection
30	include_role	including the role for our exection
31	shell	Executes shell commands on a remote host
32	get_url	Downloads files from URLs to remote hosts
33	uri	Interacts with HTTP/HTTPS services and APIs

5) Playbook:

An Ansible playbook is a YAML file that defines a series of tasks to be executed on remote hosts. It is a core component of Ansible, used for automating the configuration, deployment, and manage infrastructure as code.

Key concepts:

i) **YAML Format:** Playbooks are written in a simple, human-readable YAML format.

ii) **Plays:** A play maps hosts to tasks, defining what tasks should be executed on which machines.

iii) **Tasks:** Individual actions that Ansible performs, like installing packages or copying files.

iv) **Modules:** Reusable units of code that perform specific tasks, such as apt, yum, service, etc.

v) **Handlers:** Tasks that are triggered by other tasks, often used to restart services when changes occur.

vi) **Variables:** Dynamic values used in playbooks for flexibility, making it easier to manage different environments.

vii) **Roles:** A way to organize playbooks into reusable components, including tasks, variables, files, and templates.

Practical:

Connect to Linux instance,
can use playbook.yml (or) playbook.yaml

vim test.yml

1st line/playbook in a file should start with 3 hyphens (---)
2nd line should be given hosts along with hyphen (- hosts:), where we want playbook to run

Playbook1: ***vim test.yml*** **---** **- hosts: all/group_name/localhost** **tasks:** **- name: testing folder creation** **file:** **path: /home/ec2-user/folder1** **state: directory** **mode: 0777** **Save** **Run playbook:** **Syntax:** **ansible-playbook <playbook.yml>** **ansible-playbook test.yml** For more detailed output, we can use verbose while running playbook: ansible-playbook -vvv test.yml -v: Basic output, showing task start and end times, and simple status messages. -vv: More detailed output, including module parameters and results. -vvv: Very verbose output, showing all Ansible internal workings. **Validate:** **ls -l folder1/**	# hosts should start with hyphen (-hosts) followed by where we want to run playbook. # we can either give all which will select all IP that's in host (or) can give group_name (or) can give localhost which is Master-Node where playbook will run. #task should come below host as indentation is important #under task we can give tasks so can give name of task (name = heading) #since, we need to create folder we need to use file module (file = module) #under file, steps should have 2 spaces as indentation #state is of 6 types (absent = delete directory recursively, directory = create directory, file = create file, hard = create hardlink, link = create softlink, touch = create empty file) #mode is used for giving permission to file/folder
Playbook2: ***vim test2.yml*** **---** **- hosts: all/group_name/localhost** **tasks:** **- name: testing folder creation** **file:** **path: /home/ec2-user/folder2** **state: directory** **mode: 0777**	# output should show changed, which indicates playbook has been executed and files are created

- name: testing file creation **file:** **path: /home/ec2-user/folder1/file1** **state: touch** **mode: 0666** **- name: testing file creation** **file:** **path: /home/ec2-user/folder2/file2** **state: touch** **mode: 0666** **Save and run** **ansible-playbook -v test2.yml** **Validate:** **ls folder1/** **ls -l folder1/** **ls -l folder2/**	
Playbook3: **vim test3.yml** **---** **- hosts: localhost** **vars:** **home: /home/ec2-user** **tasks:** **- name: debug variable** **debug: var=home** **- name: creating file** **file:** **path: "{{ home }}/test-ansible"** **state: touch** **register: task1** **- name: debugging task output** **debug: var=task1** **Save and run:** **ansible-playbook -v test2.yml**	# vars = variable Declaring variable home as vars: home: /home/ec2-user and using it in path as {{ home }} # register is module used to register task output of "creating file task" in variable (task1) (i.e, output of task will be stored in variable, task1) # to print output of variable (register) we need to use debug, debug: var=task1

Tomcat-installation
Playbook chain: ***Playbook --> place --> tasks --> modules***
Add below content to file called as tomcat.services which should be present in same path where installation file (tomcat-installation.yml) is present **vim tomcat.services** **[Unit]** **Description=Apache Tomcat Web Application Container** **After=network.target** **[Service]** **Type=forking** **Environment=JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk** **Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid** **Environment=CATALINA_HOME=/opt/tomcat** **Environment=CATALINA_BASE=/opt/tomcat** **Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'** **Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'** **ExecStart=/opt/tomcat/bin/startup.sh** **ExecStop=/opt/tomcat/bin/shutdown.sh** **User=tomcat** **Group=tomcat** **UMask=0007** **RestartSec=10** **Restart=always** **[Install]** **WantedBy=multi-user.target** [Note: tomcat.service and tomcat-installation.yml should be in same path]

Tomcat-installation in Linux	
vim tomcat-installation.yml **---** **- hosts: localhost** **become: yes** **become_method: sudo**	#test group #Become used for sudo access

tasks: **- name: Update and upgrade apt packages** **shell: "yum update -y"** **- name: install JDK 11** **yum:** **name: java-1.8.0** **state: present** **- name: add tomcat group** **group:** **name: tomcat** **- name: add tomcat user** **user:** **name: tomcat** **group: tomcat** **home: /user/share/tomcat** **createhome: no** **- name: create /opt/tomcat directory** **file:** **path: /opt/tomcat** **state: directory** **mode: 0755** **- name: download & unarchive** **unarchive:** **src:http://apache.cs.utah.edu/tomcat/tomcat-9/v9.0.93/bin/apache-tomcat-9.0.93.tar.gz** **dest: /opt/tomcat** **remote_src: yes** **extra_opts: [--strip-components=1]** **- name: Change ownership** **file:** **path: /opt/tomcat** **owner: tomcat** **group: tomcat** **mode: "u+rwx,g+rx,o=rx"** **recurse: yes**	#remote_user: ubuntu/ansible but this user should be in worker node #state: present/installed/latest If package present no changes will be made, if not then will be installed (for uninstall we can use absent as state) #--strip-components=1 will extract and copies content inside folder to /opt/tomcat instead of copying folder

```
    state: directory

 - name: Copy Tomcat service from local to remote
   copy:
     src: tomcat.service
     dest: /etc/systemd/system/
     mode: 0755

 - name: reload the daemon
   shell: "systemctl daemon-reload"

 - name: Start and enable Tomcat service
   systemd:
     name: tomcat
     state: started
     enabled: true
     daemon_reload: true
```

Save and run:
ansible-playbook -v tomcat-installation.yml

Validate in AWS:
Allow traffic for 8080 as tomcat runs in 8080 ports

Tomcat-installation in Ubuntu

```
---
- hosts: localhost
  vars:
    download_url:
https://dlcdn.apache.org/tomcat/tomcat-
9/v9.0.93/bin/apache-tomcat-9.0.93.tar.gz
  tasks:

  - name: Install OpenJDK
    become: yes
    apt:
     name: openjdk-17-jre-headless
     update_cache: yes
     state: present

  - name: Validate if Java is available
```

```
  shell: java -version
  register: java_version
  ignore_errors: yes
  changed_when: false

- name: Ensure Java is installed
  fail:
   msg: "Java is not installed!"
  when: java_version.rc != 0

- name: Create the group
  become: yes
  group:
   name: tomcat
   state: present

- name: Create the user
  become: yes
  user:
   name: tomcat
   state: present

- name: Create a directory /opt/tomcat9
  become: yes
  file:
   path: /opt/tomcat9
   state: directory
   mode: 0755
   owner: tomcat
   group: tomcat

- name: Download Tomcat using unarchive
  become: yes
  unarchive:
   src: "{{ download_url }}"
   dest: /opt/tomcat9
   mode: 0755
   remote_src: yes
   group: tomcat
   owner: tomcat
```

```
 - name: Move files to the /opt/tomcat9
directory
   become: yes
   shell: "mv /opt/tomcat9/apache*/*
/opt/tomcat9"

 - name: Remove unnecessary directory
   become: yes
   file:
    path: "/opt/tomcat9/apache-tomcat-9.0.93"
    state: absent

 - name: Create a service file
   become: yes
   copy:
    content: |-
     [Unit]
     Description=Tomcat Service
     Requires=network.target
     After=network.target

     [Service]
     Type=forking
     User=tomcat

Environment="CATALINA_PID=/opt/tomcat9/logs
/tomcat.pid"

Environment="CATALINA_BASE=/opt/tomcat9"

Environment="CATALINA_HOME=/opt/tomcat9"
     Environment="CATALINA_OPTS=-Xms512M -
Xmx1024M -server -XX:+UseParallelGC"
     ExecStart=/opt/tomcat9/bin/startup.sh
     ExecStop=/opt/tomcat9/bin/shutdown.sh
     Restart=on-abnormal
     [Install]
     WantedBy=multi-user.target
    dest: /etc/systemd/system/tomcat.service
    mode: 0644
```

```
 - name: Reload the SystemD to re-read
configurations
   become: yes
   systemd:
    daemon-reload: yes

 - name: Enable the tomcat service and start
   become: yes
   systemd:
    name: tomcat
    enabled: yes
    state: started

 - name: Connect to Tomcat server on port 8080
and check status 200 - Try 5 times
   tags: test
   uri:
    url: http://localhost:8080
   register: result
   until: "result.status == 200"
   retries: 5
   delay: 10
```

Save and run:
ansible-playbook -v tomcat-installation.yml
Validate in AWS:
Allow traffic for 8080 as tomcat runs in 8080 ports

[Note: Try to use Ansible modules, keep shell as last option as when we use shell, we may not get much details and will be difficult to solve errors but when we use Ansible modules then we will get more details which will be useful to solve errors if any]

When to use ansible?
If we have multiple servers and want same configuration to be done in all servers then we can go with servers and it's used for repetitive tasks.
If we have one or two servers then no need of ansible.

6) *Lineinfile module:*

The lineinfile module is used to manage lines in text files. It allows to add, modify, or remove specific lines in a file with precision.

This module is particularly useful for managing configuration files on your target systems.

#adding line after host line **vim test.yml** **---** **- hosts: localhost** **tasks:** **- name: adding line content** **lineinfile:** **path: /home/ec2-user/local** **insertafter: 127.0.0.1 local** **line: "text add"** **Save and run:** **ansible-playbook test.yml** **cat test.yml**	o/p 127.0.0.1 local text add
#adding line before host line **vim test2.yml** **---** **- hosts: localhost** **tasks:** **- name: adding line content** **lineinfile:** **path: /home/ec2-user/local** **insertbefore: 127.0.0.1 local** **line: "testing"** **Save and run:** **ansible-playbook test.yml** **cat test1.yml**	o/p testing 127.0.0.1 local text add

7) *Ansible Role:*

- An Ansible role is a reusable, self-contained unit of automation that is used to organize and manage tasks, variables, files, templates, and handlers in a structured way.
- Roles help to encapsulate and modularize the logic and configuration needed to manage a particular system or application component.
- This modular approach promotes reusability, maintainability, and consistency across different playbooks and environments.

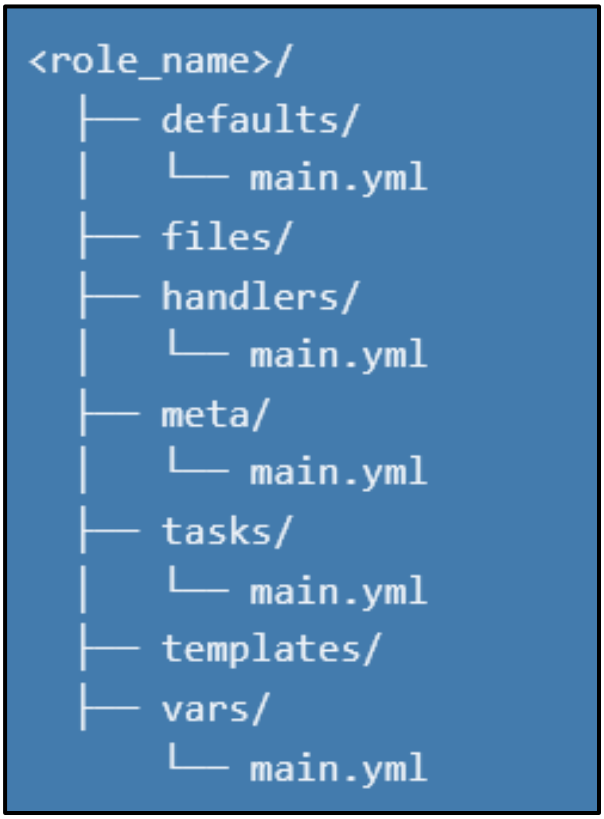
Create role:

ansible-galaxy init <role_name>

[Note: folder will be created inside which there will be many sub-folders/components such as defaults, files, handlers, meta, tasks, templates, tests, vars]

sudo yum install tree

tree



i) Components of Role:

- **defaults:** define your default variables eg) versions of applications
- **files:** placing static files
- **handlers:** service control
- **meta:** info of this role, dependencies have to be declared in meta
- **tasks:** consists of our playbooks like installation deployment
- **templates:** used for placing a dynamic file (j2 (jinja) format)
- **vars:** define variables

ii) Why Use Ansible Roles?

- **Modularity:** Roles allow you to break down complex playbooks into smaller, reusable components. Each role handles a specific part of the configuration or setup.
- **Reusability:** Once created, roles can be reused across different playbooks and projects. This saves time and effort in writing redundant code.
- **Maintainability**
  By organizing related tasks into roles, it becomes easier to manage and maintain the code. Changes can be made in one place and applied consistently wherever the role is used.
- **Readability:** Roles make playbooks cleaner and easier to read by abstracting away the details into logically named roles.
- **Collaboration:** Roles facilitate collaboration among team members by allowing them to work on different parts of the infrastructure independently.
- **Consistency:** Using roles ensures that the same setup and configuration procedures are applied uniformly across multiple environments, reducing the risk of configuration drift.

variables define:

vars: port=85 --> 2 (priority)
defaults: port=83 --> 4 (priority)
playbook: port=82 --> 3 (priority)
run time: port=84 --> 1 (priority)

ansible-playbook -v test.yml --extra-vars variable_name=variable_value

{{ JAVA_HOME }} --
{{ ui_admin_username }} --
{{ ui_admin_pass }} --
{{ ui_manager_user }} --
{{ ui_manager_pass }} --
{{ tomcat_ver }} --

role/tasks/main.yml

- import_tasks: install.yml
- import_tasks: configure.yml
- import_tasks: service.yml

➔ *Create Ec2 Instance from Ansible Master*

Step1: switch to ansible user or can create in ec2-user

```
su ansible
```

Step2: create role

```
ansible-galaxy init awsrole
```

Step3: create user and access key

create IAM user and assign permission for: - ec2-fullaccess

Select user --> create access key

```
[aws_access_key: AKIA4MTWIW6EE6THYPG3
aws_secret_key: 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
region-us-east-1]
```

Step4: Install required packages and configure

```
sudo yum update
sudo yum install -y python3-boto python3-boto3
sudo yum install -y python3-pip
sudo pip3 install boto boto3
python3 -m pip show boto | grep Version
python3 -m pip show boto3 | grep Version

aws configure
aws_access_key: AKIA47CRZ4HOYQFC5M57
aws_secret_key: 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
region-us-east-1
```

Step 5: If want to install in worker nodes

```
cd /etc/ansible:
sudo vim hosts
    [group_name]
    Worker1_IP
    Worker2_IP
```

Step 6: create file to run role in home path

```
sudo vi aws.yml
  ---
  - hosts: localhost (or)  <group_name>
    gather_facts: no
    become: yes
    connection: local
    become_user: ec2-user
    roles:
      - awsrole
```

Step 7: ***add below code and make reqired chances for creation of EC2-Instance***

sudo vim awsrole/tasks/main.yml

```
---
- name: Launch EC2 instance
  amazon.aws.ec2_instance:
    key_name: devops   #Replace with your keypair
    instance_type: t2.micro
    image_id: ami-0ae8f15ae66fe8cda #urs ami desired AMI ID
    region: us-east-1 #urs region
    wait: yes
    count: 1
    tags:
      Name: MyEC2Instance
    vpc_subnet_id: subnet-03ad4b60d670ec516 # Replace with your subnet ID
    security_group: sg-088b51b2fdb35cc7b #Replace your security group ID
    network:
      assign_public_ip: yes
  register: ec2
- name: Debug EC2 instance information
  debug:
    var: ec2
- name: Wait for the instance's public IP to be available
  pause:
    seconds: 30
- name: Wait for SSH to come up
  wait_for:
    host: "{{ ec2.instances[0].network_interfaces[0].association.public_ip}}"
    port: 22
    delay: 60
    timeout: 320
    state: started
- name: Add the new instance to the local host group (optional)
  add_host:
    hostname: "{{ ec2.instances[0].network_interfaces[0].association.public_ip
}}"
    groupname: launched
```

Step 8: ***Run playbook in home path (/home/ansible/)***

ansible-playbook -vv aws.yml