**Ansible** works by connecting to the nodes over SSH and pushing out small programs, called "**Ansible** Modules", to them. **Ansible** then executes and removes them when finished. Because these modules are simple Python scripts, and**Ansible** is agent-less, the target hosts only require an SSH connection and Python installed.

vi /etc/ansible/host🡪Inventory file this contains the agent info. agent IP/host name

ssh-keygen –t rsa🡪generate rsa public key

vi ~/.ssh/authorized\_keys🡪paste the public key in the agent

ssh-copy-id -i root@[IP of local host]🡪to copy the key to the host machine.

vi /etc/host🡪agent IP, hostname & controllerID, host name

ansible all –m ping🡪to ping all the agents

ansible all --list-hosts🡪to list all the hosts in the Inventory file

ansible all –m copy –a ’src=[source of the file] dest=[destination]🡪to copy a file

ansible all –m file –a ‘path=[destination of the file/dir to be created] state=directory/touch mode=777 owner=root’🡪to create a file or directory

ansible all –m file –a ‘path=[destination of file/dir] state=absent’

ansible all -m apt -a 'name=apache2 state=present'🡪to install packages

ansible all –m service –a ‘name=apache2 state=started’🡪to start service

ansible all –m setup🡪 to get all details like env variables , memory configurations etc

ansible [group name] –b –m user –a “name=[user name]”

ansible [group name] –b –m file –a “path=home/[username]/.ssh state=directory owner=[user name] group=[user name] mode=0755”

files, copy, setup, yum, apt, service, systemd, command, shell, raw, user, group, package, cron, template, uri, lineinfile, script

***GATHERING FACTS*🡪**When Ansible starts executing a play, the first thing it does is collect information about the server it is connecting to. The aim of this task is to gather useful metadata about the machine in the form of variable which are then used as a part of the task that follow in the playbook. The data include IP address , architecture of the system , host name etc.

***PLAY RECAP***🡪give us the necessary details of the tasks that are run on the remote machines and their status.

***List-tasks***🡪While running Ansible, there are cases where we need to check the tasks that we defined🡪**ansible-playbook apache.yml --list-tasks**

***Start-at-task***🡪There are cases sometimes where we need to start a particular task on the remote machine or re-run a task🡪**ansible-playbook apache.yml --start-at-task="restart apache"**

***Step***🡪Ansible also provides us with an option to ask for permission before running a task on the remote machine using the –step🡪**ansible-playbook apache.yml --step**

***Tags*** are another feature in Ansible that is used to run only a subset of tasks/roles.

**ansible-playbook sample-playbook.yml --tags sample**

**ansible-doc [module]🡪** **shows documentation about modules**

**ansible-galaxy init apache --offline🡪 create a role**

**default, file, handlers, meta, README.md, tasks, templates, test, vars**

**ansible-vault encrypt <file>🡪to encrypt secret files**

**ansible-vault edit <file>🡪to edit the secret file**

**ansible-vault view <file>🡪to view the content of the secret file**

**ansible-vault decrypt <file>🡪to decrypt the encrypted file**

**ansible-vault encrypt\_string ‘Hi Milan’ –n meaning🡪encrypt a string and can put it into a yaml file**

**ansible-vault encrypt\_string ‘Hi Milan’ –n meaning --vault-id dev@prompt 🡪this will assign the label dev to the piece of encrypted data**

**ansible-vault encrypt --vault-id prod@vault <file>🡪**

**ansible-playbook vault.yaml --vault-id prod@vault 🡪**

**ansible-playbook –v vault.yaml --vault-id prod@vault 🡪**

**tasks:**

**- name:**

**apt:**

**when: ansible\_distribution == 'Debian' or ansible\_distribution == 'Ubuntu'**

**- name: clone a private Git repository**

**git: repo=ssh://git@github.com/someone/example-repo.git**

**key\_file=/root/.ssh/id\_rsa.github**

**dest=/opt/example**

**- name:Download war to server**

**synchronize: scr={{ warLocalPath }}/{{ warName }} dest={{ warRemotePath }}/{{ warName }}**

**- name: Download Tomcat**

**get\_url: url=http://a.mbbsindia.com/tomcat/tomcat-8/v8.0.32/bin/apache-tomcat-{{ version }}.tar.gz dest=/tmp mode=0755**

**- name: Unzip WAR file**

**unarchive: src={{ warRemotePath }}/{{ warName }} dest=/var/lib/tomcat7/webapps/ROOT/ copy=no mode=0755 owner=tomcat7 group=tomcat7**

**- name: Wait for Tomcat to Start on Port 8084**

**wait\_for: host=172.16.202.96 port={{ http\_port }}**

**- name: scheduling**

**cron: {name: , weekday: ,minute: ,hour: ,user: ,job: }**

**- name: run a sh script**

**Script : file.sh**

**- name:create a new ec2 instance**

**ec2: {key\_name: ,group\_id: ,image: ,instance\_type: ,wait:true, region: ,count: ,count\_tag: ,instance\_tags: }**

**register: ec2**

**- name: Update packages on server**

**apt: update\_cache=yes upgrade=yes**

**- name: rebooting a server**

**reboot:**

**- name: execute a command**

**shell: echo"xyz"**

**- name:create a vagrantfile**

**command: vagrant init {{ box }} creates=Vagrantfile**

**- name: bring up a vagrant server**

**command: vagrant up**

**- name: add the vagrant hosts to the inventory**

**add\_host: >**

**name=vagrant**

**ansible\_ssh\_host=127.0.0.1**

**ansible\_ssh\_port=2222**

**ansible\_ssh\_user=vagrant**

**ansible\_ssh\_private\_key\_file=/Users/lorinhochstein/.vagrant.d/**

**insecure\_private\_key**

**- name: update nginx**

**apt: name=nginx update\_cache=yes**

**- name: Create Multiple users**

**shell: useradd {{ item }}**

**with\_items:**

**- hai**

**- bai**

**- name: Installing Packages**

**yum: name={{item.name}} state={{item.value}}**

**with\_items:**

**- {name: 'httpd', value: 'present'}**

**---------------------------------------------------------------**

**tasks:**

**- yum:**

**name: "{{ item }}"**

**state: installed**

**loop:**

**- httpd**

**- memcached**

**tags:**

**- packages**

**-------------------------------------------------------------------**

**- name: Checking for httpd**

**Package:**

**name: httpd**

**state: present**

**check\_mode: true**

**register: httpd\_check**

**- name: installing httpd and git if httpd not installed**

**yum:**

**name:**

**- httpd**

**- git**

**state: present**

**when: httpd\_check.changed**

**- name: install only git as httpd is already available**

**yum:**

**name: git**

**state: present**

**when: not httpd\_check.changed**

**------------------------------------------------------------------**

**- name: Pull Ubuntu image  
    docker\_image:  
      name: ubuntu**

**- name: Build Nginx image  
    docker\_image:  
      path: [Path of dockerfile]  
      name: my-nginx  
      tag: 0.1**

**- name: Create another Nginx container  
     docker\_container:  
       name: my-nginx  
       image: my-nginx:0.1  
       ports:  
         - "18880:80"  
       env:  
         KEY: value  
       command: sleep infinity**

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

cat install\_apache.yml

- set\_fact: package\_name=httpd

- name: install Apache

  yum: name=httpd state=present

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

---

- hosts: dev

  tasks:

**- include: install\_apache.yml**

    - name: check Apache Service

      service: name={{ package\_name }} state=restarted

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

cat test1.yml

---

- hosts: dev

**vars\_files:**

**- var1.yml**

  tasks:

   - name: install httpd Package

     yum: name=httpd state=present

   - name: restart Apache

     service: name="{{ package\_name }}" state=restarted

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

Now the var1.yml file contains the variables that we need to define as

 [root@vx111a 2test]# cat var1.yml

---

package\_name: "httpd"

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

cat apache.yml

---

- hosts: dev

**vars:**

**- package\_name: "httpd"**

  tasks:

   - name: install httpd Package

     yum: name=httpd state=present

   - name: restart Apache

     service: name="{{ package\_name }}" state=restarted

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

---

- hosts: tomcatServer

vars:

- warName: helloworld.war

- warRemotePath: /path/to/put/war

tasks:

- name: Download WAR to server

synchronize: src={{ warLocalPath }}/{{ warName }} dest={{ warRemotePath }}/{{ warName }}

- name: Unzip WAR file

unarchive: src={{ warRemotePath }}/{{ warName }} dest=/var/lib/tomcat7/webapps/ROOT/ copy=no mode=0755 owner=tomcat7 group=tomcat7

notify:

- Restart tomcat7

- name: Delete remote war file

file: path={{ warRemotePath }}/{{ warName }} state=absent

handlers:

- name: Restart tomcat7

service: name=tomcat7 state=restarted

---

- name: create a new ec2 instance

hosts: all

connection: local

gather\_facts: false

vars:

region: us-east-1

instance\_type: t2.micro

ami: ami-035be7bafff33b6b6

keypair: aws

sg: sg-09fdd74cdf67332ae

tasks:

- name: creating an ec2 instance

ec2:

key\_name: "{{ keypair }}"

group\_id: "{{ sg }}"

image: "{{ ami }}"

instance\_type: "{{ instance\_type }}"

wait: true

region: "{{ region }}"

count: 1

count\_tag:

Name: AnsibleDemo

instance\_tags:

Name: AnsibleEC2

register: ec2

---

- hosts: webservers

vars:

http\_port: 80

max\_clients: 200

remote\_user: root

tasks:

- name: ensure apache is at the latest version

yum:

name: httpd

state: latest

- name: write the apache config file

template:

src: /srv/httpd.j2

dest: /etc/httpd.conf

notify:

- restart apache

- name: ensure apache is running

service:

name: httpd

state: started

handlers:

- name: restart apache

service:

name: httpd

state: restarted

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

Playbooks can contain multiple plays. You may have a playbook that targets first the web servers, and then the database servers.

---

- hosts: webservers

remote\_user: root

tasks:

- name: ensure apache is at the latest version

yum:

name: httpd

state: latest

- name: write the apache config file

template:

src: /srv/httpd.j2

dest: /etc/httpd.conf

- hosts: databases

remote\_user: root

tasks:

- name: ensure postgresql is at the latest version

yum:

name: postgresql

state: latest

- name: ensure that postgresql is started

service:

name: postgresql

state: started

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

---

- hosts: isrestart

become: true

become\_user: root

tasks:

- name: Rebooting the cloud server/bare metal box

reboot:

.

├── defaults

│   └── main.yml

├── files

│   ├── apache2.conf

│   └── index.html

├── handlers

│   └── main.yml

├── meta

│   └── main.yml

├── README.md

├── tasks

│   ├── configure.yaml

│   ├── install.yaml

│   ├── main.yml

│   └── service.yaml

├── templates

├── tests

│   ├── inventory

│   └── test.yml

└── vars

└── main.yml

root@controller:/etc/ansible/roles/apache# cat tasks/main.yml

---

# tasks file for apache

- include: install.yaml

- include: configure.yaml

- include: service.yaml

root@controller:/etc/ansible/roles/apache# cat tasks/install.yaml

---

- name: install apache

apt:

name: apache2

state: latest

root@controller:/etc/ansible/roles/apache# cat tasks/configure.yaml

---

- name: apache2.conf file

copy: src=apache2.conf dest=/etc/apache2/apache2.conf

notify:

restart apache service

- name: send index.html

copy: src=index.html dest=/var/www/html/index.html

root@controller:/etc/ansible/roles/apache# cat tasks/service.yaml

---

- name: start service

service:

name: apache2

state: started

root@controller:/etc/ansible/roles/apache# cat handlers/main.yml

---

# handlers file for apache

- name: restart apache service

service:

name: apache2

state: started

root@controller:/etc/ansible/roles/apache# cat files/index.html

<html><body><h1>welcome to apache</h1></body></html>

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

.

├── defaults

│   └── main.yml

├── files

├── handlers

│   └── main.yml

├── meta

│   └── main.yml

├── README.md

├── tasks

│   └── main.yml

├── templates

├── tests

│   ├── inventory

│   └── test.yml

└── vars

└── main.yml

8 directories, 8 files

root@controller:/etc/ansible/roles/jenkins# cat tasks/main.yml

---

# tasks file for jenkins

- name: install wget

yum:

name: wget

state: present

- name: install openjdk

yum:

name: java-1.8.0-openjdk

state: present

- name: download jenkins.repo

get\_url:

url: http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo

dest: /etc/yum.repos.d/jenkins.repo

- name: import jenkins key

rpm\_key:

state: present

key: https://jenkins-ci.org/redhat/jenkins-ci.org.key

- name: install jenkins

yum:

name: jenkins

state: present

- name: start jenkins

systemd:

name: jenkins

state: started

- name: enable jenkins

systemd:

name: jenkins

enabled: true

- name: sleep for 30 seconds and continue with play

wait\_for: timeout=30

delegate\_to: localhost

- name: init password jenkin

shell: cat /var/lib/jenkins/secrets/initialAdminPassword

changed\_when: false

register: result

- name: print init password jenkins

debug:

var: result.stdout

node('Slave')

{

stage('SCM Checkout'){

git 'https://github.com/javahometech/my-app'

}

stage('Compile-Package'){

def mvnHome = tool name: 'maven-3', type: 'maven'

sh "${mvnHome}/bin/mvn package"

}

stage('Deploy to Tomcat'){

sshagent(['tomcat-dev']) {

sh 'scp -o StrictHostKeyChecking=no target/\*.war ec2-user@172.31.29.242:/opt/tomcat8/webapps/'

}

}

stage('Build Docker Imager'){

sh 'docker build -t kammana/myweb:0.0.1 .'

}

|  |
| --- |
| stage('Push to Docker Hub'){ |
|  |  |
|  | withCredentials([string(credentialsId: 'github-pwd', variable: 'dockerHubPwd')]) { |
|  | sh "docker login -u kammana -p ${dockerHubPwd}" |
|  | } |
|  | sh 'docker push kammana/myweb:0.0.1' |
|  | } |

stage('Deploy to Dev Environment'){

def dockerRun = 'docker run -d -p 8080:8080 --name myweb kammana/myweb:0.0.1 '

sshagent(['docker-dev']) {

sh "ssh -o StrictHostKeyChecking=no ec2-user@172.31.17.196 ${dockerRun}"

}

stage('Email Notification'){

mail bcc: '', body: '''Hi Welcome to jenkins email alerts

Thanks

Hari''', cc: '', from: '', replyTo: '', subject: 'Jenkins Job', to: 'hari.kammana@gmail.com'

}

}

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

|  |
| --- |
| node{ |
|  | parallel firstBranch: { |
|  | stage('one'){ |
|  | git url: 'http://github.com/javahometech/my-app', branch: 'master123' |
|  | } |
|  | }, secondBranch: { |
|  | stage('Two'){ |
|  | git url: 'http://github.com/javahometech/my-app', branch: 'feature-1' |
|  | } |
|  | }, |
|  | failFast: true |
|  | } |

**Install the Active directory using Jenkins** ->  Manage Jenkins -> Manage plugins->Active directory plugin -> Manage Jenkins -> Configure Global Security->Access Control – Security Realm, choose Active directory. Once you select the Active directory , click on the “Add domain” button

Build Triggers-->Build after other projects are built

Build Triggers-->Build periodically-->schedule job

Build Triggers-->GitHub hook trigger for GITScm polling

Build Triggers-->poll scm

Post-build Actions-->Deploy artifacts to Artifactory

Post-build Actions-->Deploy war to container(deploy to container)

Post-build actions🡪publish Junit test result report

Manage plugin🡪TestNG Result plugin

Post-build actions🡪Publish TestNG result🡪test results

Build Settings-->E-mail Notification

Configure system-->E-mail Notification,Extended E-mail Notification, GitHub(webhooks), Artifactory

Configue Global Security-->Matrix-based security(for giving access to users)

Manage users(create users)

Git source code issue--> someone screwed up the code-->contact to the devloper to resolve

stated the build then-->compiler issue-->inform the developer

server is down or unreachable--> we contact network team

issue while taking backup-->server 100% disk full-->clean up the server

while deploying disk space-->increase the space

smoke testing-->login issue--> we need to see the code and then resolve the issue

issue while triggering the email-->check the credentials

ways to connect the slaves?

ssh

java web start

execution of command on the master

let jenkins control this windows slave as a windows service

to configure multiple environment deployment(like dev,QA,prod)-->we need to check"this project is premeterized"

to delete old builds-->we need to check "discard old builds" in job configuration

types of job--> feestyle,pipeline, maven project, external job, multi-configuration project, milti-branch

configure scaling policy-->use scaling policy to adjust the capacity of the group

Endpoint and host to connect a RDS.

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

rpm -ql Jenkins 🡪Here are the major locations where jenkins files will be installed

In order to remove the cache we need remove the following,

rm -rf /var/cache/jenkins/\*

rm -rf  /var/lib/jenkins/\*

rm -rf /var/log/jenkins/\*

Once these locations are cleaned we can now perform a new installation with no previous cache

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

Database migration service🡪Replication Instance🡪Database endpoints, here need to provide source database connection details(source endpoint) and target database connection details(Target endpoint)🡪run test🡪task, here we review the provided details and modify the details accordingly🡪create

Ova file🡪s3 bucket🡪create a role vmimport🡪attach IAM policy to the role🡪import the file, this will create an image 🡪create an instance from the image

Aws iam create-role --role-name vmimport --as sume-role-policy-document file://trust-policy.json

Aws ec2 import-image --description “ “ --disk-containers file://containers.json

Application Load Balancer instead of a Classic Load Balancer has the following benefits:

* Support for routing requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
* Support for redirecting requests from one URL to another.
* Support for returning a custom HTTP response.
* Support for registering targets by IP address, including targets outside the VPC for the load balancer.