

# Software Testing Help

## Ansible Roles, Integration with Jenkins in DevOps, and EC2 Modules

Last Updated: August 31, 2018

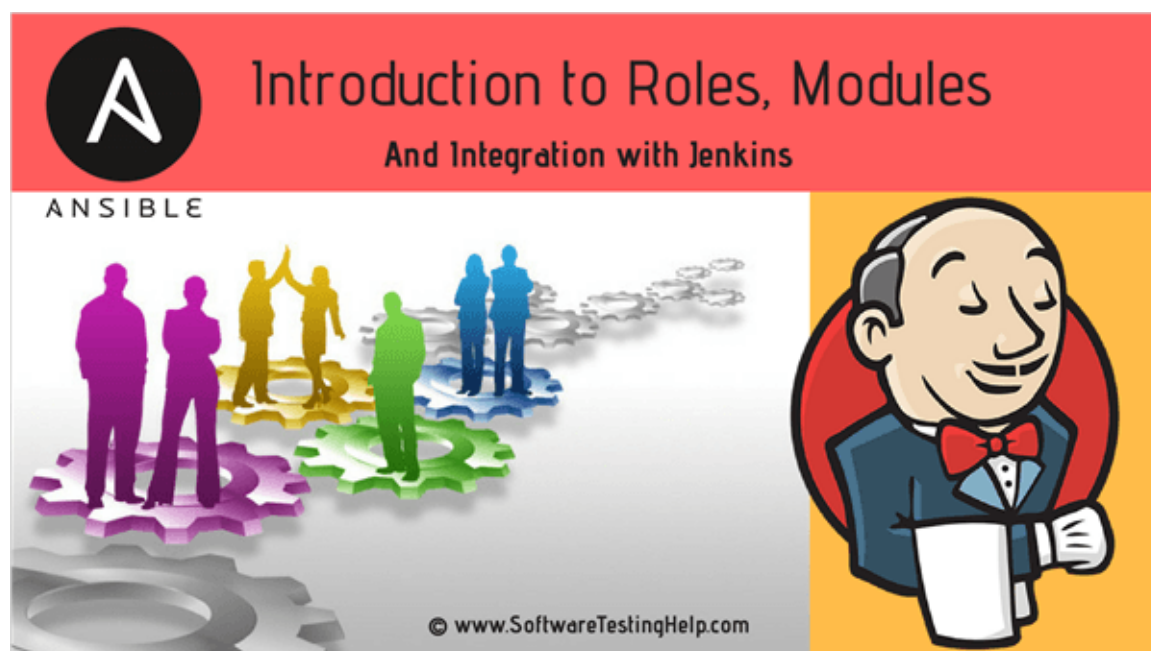
**An in-depth look at Ansible Roles, Integration with Jenkins, and Ansible S3 and EC2 modules:**

**In part 2 of the series on Ansible tutorials**, we learned how Ansible playbooks are used to execute multiple tasks and get all the target machines or servers to a particular desired state.

**Recommended Read => Exponential DevOps Training Series**

Ansible playbooks are primarily YAML files which contain multiple tasks in one single huge file which is not modular and reusable. But if you need to break up your entire configuration and be more modular and reusable then Ansible roles will help a lot.

***In this Ansible Tutorial**, we will look at Ansible roles, Integrate Ansible with Jenkins for continuous delivery from a DevOps point of view and most importantly look at Ansible S3 and EC2 modules for managing AWS EC2 instances (Create and Terminate EC2 instances).*



# Ansible Roles

With Ansible roles you can group your variables, tasks, handlers etc., which increase reusability and most certainly reduce syntax errors. It helps to de-clutter the whole code.

Ansible roles are similar to modules in Puppet and cookbooks in Chef.

In order to create roles, you use the **ansible-galaxy** command which has all the templates to create it.

## Example Scenario

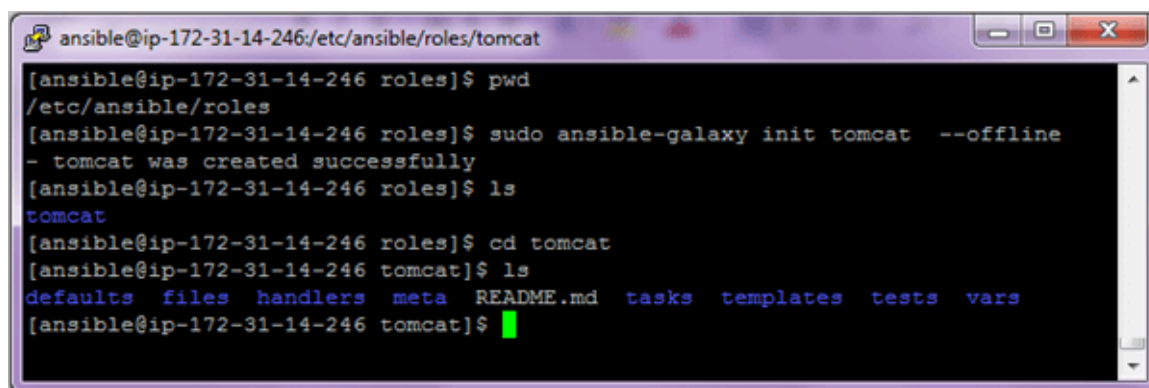
I have been a DevOps specialist for most of my life and have been working only in CI and CD.

**So for Example in Continuous Delivery where I am deploying a new build of my J2EE application (WAR file) to tomcat my steps would be as follows:**

- Stop the application
- Uninstall the application
- Deploy the new build of an application
- Start the application

So I would be creating a role with at least 4 tasks and one main file calling it. This way I am making my code more modular and reusable. So let's call this role as **tomcat** and create it.

```
$ cd /etc/ansible/roles
$ sudo ansible-galaxy init tomcat --offline
```

A terminal window titled 'ansible@ip-172-31-14-246:/etc/ansible/roles/tomcat' showing the following commands and output:

```
[ansible@ip-172-31-14-246 roles]$ pwd
/etc/ansible/roles
[ansible@ip-172-31-14-246 roles]$ sudo ansible-galaxy init tomcat --offline
- tomcat was created successfully
[ansible@ip-172-31-14-246 roles]$ ls
tomcat
[ansible@ip-172-31-14-246 roles]$ cd tomcat
[ansible@ip-172-31-14-246 tomcat]$ ls
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
[ansible@ip-172-31-14-246 tomcat]$
```

Once the role is created you can see the directory structure which it has created.

**The main components we will use in this section include:**

- **tasks/main.yml** – This is the starting point for tasks created for the role. You can use the main.yml file to point to the other task files.
- **vars** – This is to define any variables used.
- **meta** – This is to define information about yourself or the author.

Edit the **tasks/main.yml** file and add the below code. As per the example scenario mentioned above, we are defining 4 different tasks. In most cases, the deploy application will also start the application so the last one of the starting application may not be required.

```
---
# tasks file for tomcat
- import_tasks: stop_app.yml
- import_tasks: uninstall_app.yml
- import_tasks: deploy_app.yml
- import_tasks: start_app.yml
```

## Step 2: Create all the 4 files as per the scenario

In the below tasks **action: ec2\_facts** is used to get facts from remote EC2 instances and call them in plays/tasks

### tasks/stop\_app.yml file

```
---
- name: Gather EC2 instance metadata
  action: ec2_facts

- name: Stop application on {{ansible_hostname}}
  command: wget "http://{{tomcat_user}}:{{tomcat_pwd}}@{{ansible_ec2_public_ipv4}}:8080/m
```

### tasks/uninstall\_app.yml

```
---
- name: Gather EC2 instance metadata
  action: ec2_facts

- name: Undeploy application on {{ansible_hostname}}
  command: wget "http://{{tomcat_user}}:{{tomcat_pwd}}@{{ansible_ec2_public_ipv4}}:8080/m
```

### tasks/deploy\_app.yml

```
---
- name: Deploy the new WAR file to target servers

  copy: src=/var/lib/jenkins/workspace/Demo-Maven-Project/target/HelloWorld-Maven.war de
```

webapps location of tomcat.

### task/start\_app.yml

```
---
- name: Gather EC2 instance metadata
  action: ec2_facts

- name: Start application on {{ansible_hostname}}
  command: wget "http://{{tomcat_user}}:{{tomcat_pwd}}@{{ansible_ec2_public_ipv4}}:8080/m
```

### Step 3: Define Variables

Edit the **vars/main.yml** file and add the code as shown below.

```
---
# vars file for tomcat

tomcat_user: tomcat
tomcat_pwd: tomcat
```

### Step 4: Define information in the meta folder

Edit the meta/main.yml file and add your information like author, description, and company.

```
galaxy_info:

  author: V Niranjana

  description: Devops specialist

  company: <Company Name>
```

### Step 5: Create a main site.yml file

Lastly, create the main site.yml file to call the role created which in turn will help to deploy the application to the servers or a list of hosts as per the inventory file. Create the file as

**/etc/ansible/site.yml**

```
---
- hosts: webserver
  become: true
  roles:
    - apache
```

## Step 6: Run playbook file site.yml

\$ ansible-playbook site.yml

```
ansible@ip-172-31-18-176:/etc/ansible
[ansible@ip-172-31-18-176 ansible]$ pwd
/etc/ansible
[ansible@ip-172-31-18-176 ansible]$ ls
ansible.cfg  hosts  roles  site.yml
[ansible@ip-172-31-18-176 ansible]$ ansible-playbook site.yml

PLAY [webserver] *****

TASK [Gathering Facts] *****
ok: [172.31.27.39]

TASK [tomcat : Gather EC2 instance metadata] *****
[DEPRECATION WARNING]: The 'ec2_facts' module is being renamed '
by setting deprecation_warnings=False in ansible.cfg.
ok: [172.31.27.39]

TASK [tomcat : Stop application on ip-172-31-27-39] *****
[WARNING]: Consider using the get_url or uri module rather than
warn=False to this command task or set command_warnings=False in
changed: [172.31.27.39]

TASK [tomcat : Gather EC2 instance metadata] *****
ok: [172.31.27.39]

TASK [tomcat : Undeploy application on ip-172-31-27-39] *****
changed: [172.31.27.39]

TASK [tomcat : Gather EC2 instance metadata] *****
ok: [172.31.27.39]

TASK [tomcat : Deploy application on ip-172-31-27-39] *****
changed: [172.31.27.39]

TASK [tomcat : Gather EC2 instance metadata] *****
ok: [172.31.27.39]

TASK [tomcat : Start application on ip-172-31-27-39] *****
changed: [172.31.27.39]

PLAY RECAP *****
172.31.27.39 : ok=9  changed=4  unreachable=0

[ansible@ip-172-31-18-176 ansible]$
```

Launch the Tomcat URL to see if the application has been deployed and started.

HTTP://<HostnameorIPAddressOfEc2Instance>:portno/manager

[ec2-13-232-68.com:8080/manager/html/](#)

**Manager**

[List Applications](#)
[HTML Manager Help](#)
[Manager Help](#)

**Applications**

| Path              | Display Name                      | Running | Sessions | Commands   |
|-------------------|-----------------------------------|---------|----------|--|
| /                 | Welcome to Tomcat                 | true    | 0        | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /HelloWorld-Maven | Archetype Created Web Application | true    | 0        | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /docs             | Tomcat Documentation              | true    | 0        | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /examples         | Servlet and JSP Examples          | true    | 0        | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /host-manager     | Tomcat Host Manager Application   | true    | 0        | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |
| /manager          | Tomcat Manager Application        | true    | 1        | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |

**Deploy**

Deploy directory or WAR file located on server

Context Path (required):   
 XML Configuration file URL:   
 WAR or Directory URL:

## Jenkins Integration with Ansible

In this section, we will see how Jenkins can be integrated with Ansible. The WAR file built using the build process will be used to deploy to Tomcat on the target machine using Ansible. We will be calling the Ansible role created in the previous section in Jenkins using the Ansible plugin.

So once the build is done the deployment of WAR file will be automatically triggered using Ansible.

I am keeping this simple and have not configured Sonar or Artifactory or Junit during the continuous integration activities which can also be done.

### Step 1: Create a Jenkins job and configure the SCM repo using the code in GitHub

Jenkins > Demo-Maven-Project >

GeneralSource Code ManagementBuild TriggersBuild EnvironmentPre StepsBuildPost StepsBuild Settings

Post-build Actions

Source Code Management

None

Git

Repositories

Repository URLhttps://github.com/vniranjan1972/HW-Maven-Repo

Credentials- none -Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')\*/master

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Subversion

Step 2: Configure the build



Jenkins > Demo-Maven-Project >

General Source Code Management Build Triggers **Build Environment** Pre Steps Build

Post-build Actions

☐ Poll SCM

### Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Provide Configuration files
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Create Delivery Pipeline version
- ☐ Enable Artifactory release management
- ☐ Resolve artifacts from Artifactory
- ☐ With Ant

### Pre Steps

Add pre-build step ▾

### Build

Root POM

Goals and options

### Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☐ Run re

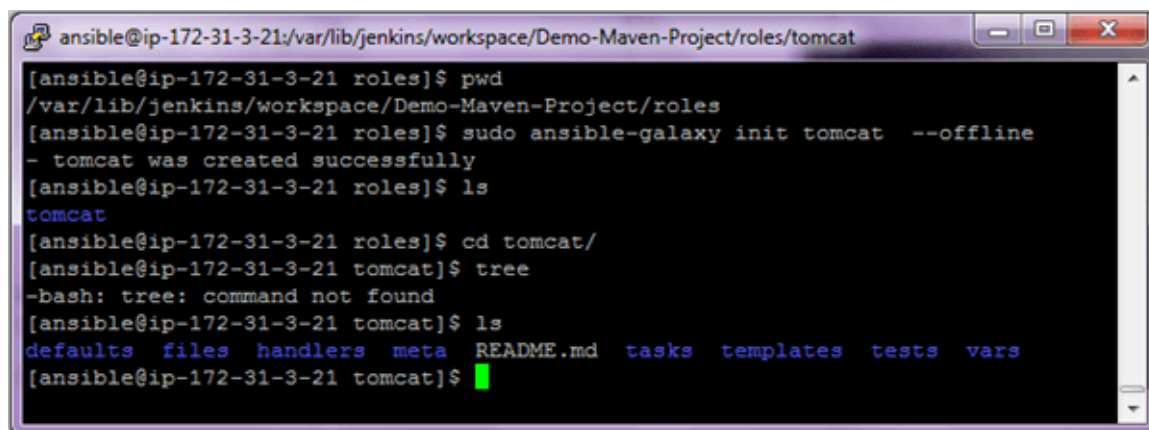
### Step 3: Create roles directory within the Jenkins workspace

```
ansible@ip-172-31-3-21:/var/lib/jenkins/workspace/Demo-Maven-Project/roles
[ansible@ip-172-31-3-21 roles]$ pwd
/var/lib/jenkins/workspace/Demo-Maven-Project/roles
[ansible@ip-172-31-3-21 roles]$
```

### Step 4: Create the tomcat role in the Jenkins workspace location using the command shown below

**\$ sudo ansible-galaxy init tomcat --offline**





```
ansible@ip-172-31-3-21:/var/lib/jenkins/workspace/Demo-Maven-Project/roles/tomcat
[ansible@ip-172-31-3-21 roles]$ pwd
/var/lib/jenkins/workspace/Demo-Maven-Project/roles
[ansible@ip-172-31-3-21 roles]$ sudo ansible-galaxy init tomcat --offline
- tomcat was created successfully
[ansible@ip-172-31-3-21 roles]$ ls
tomcat
[ansible@ip-172-31-3-21 roles]$ cd tomcat/
[ansible@ip-172-31-3-21 tomcat]$ tree
-bash: tree: command not found
[ansible@ip-172-31-3-21 tomcat]$ ls
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
[ansible@ip-172-31-3-21 tomcat]$
```

Follow the procedure as in the previous section to create all the files for **tasks, vars, meta and the main site.yml**.

The main **site.yml** file is created in **/var/lib/Jenkins/workspace/<Jenkins-Job-Name>** directory.

**Step 5: Configure the Jenkins post-build step to invoke the Ansible playbook and call the site.yml file. Save the job.**



Jenkins > Demo-Maven-Project >

General Source Code Management Build Triggers Build Environment Pre Steps Build **Post Steps**

Post-build Actions

### Post Steps

☒ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☐ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Invoke Ansible Playbook

Ansible installation Ansible 2.5.3

Playbook path site.yml

Inventory

- ☒ Do not specify Inventory
- ☐ File or host list
- ☐ Inline content

Host subset

Credentials - none - Add

Vault Credentials - none - Add

☐ become

☐ sudo

Add post-build step

### Build Settings

Email Notification

Save Apply

Post-build Actions

**Step 6: Trigger the build job and launch the Tomcat URL to verify if the application is deployed correctly.**

```

Jenkins > Demo-Maven-Project > #7
[JENKINS] Archiving /var/lib/jenkins/workspace/Demo-Maven-Project/pom.xml to HelloWorld-M
Maven-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/Demo-Maven-Project/target/HelloWorld-Maven
SNAPSHOT/HelloWorld-Maven-0.0.1-SNAPSHOT.war
channel stopped
[Demo-Maven-Project] $ /bin/ansible-playbook site.yml -f 5

PLAY [webserver] *****

TASK [Gathering Facts] *****
ok: [172.31.7.21]

TASK [tomcat : Gather EC2 instance metadata] *****
[DEPRECATION WARNING]: The 'ec2_facts' module is being renamed
'ec2_metadata_facts'. This feature will be removed in version 2.7. Deprecation
warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [172.31.7.21]

TASK [tomcat : Stop application on ip-172-31-7-21] *****
[WARNING]: Consider using the get_url or uri module rather than running wget.
If you need to use command because get_url or uri is insufficient you can add
warn=False to this command task or set command_warnings=False in ansible.cfg to
get rid of this message.
changed: [172.31.7.21]

TASK [tomcat : Gather EC2 instance metadata] *****
ok: [172.31.7.21]

TASK [tomcat : Undeploy application on ip-172-31-7-21] *****
changed: [172.31.7.21]

TASK [tomcat : Deploy the new WAR file to target servers] *****
changed: [172.31.7.21]

TASK [tomcat : Gather EC2 instance metadata] *****
ok: [172.31.7.21]

TASK [tomcat : Start application on ip-172-31-7-21] *****
changed: [172.31.7.21]

PLAY RECAP *****
172.31.7.21          : ok=8    changed=4    unreachable=0    failed=0

Finished: SUCCESS

```

## Manage AWS S3 Objects with Ansible

Ansible S3 module can be used to get or put a file to or from an S3 bucket. To use this module we will need to install and configure **boto** module of python which acts as an API(Application program interface) to access AWS. This has to be installed on the **Ansible control machine**.

### On Redhat Linux

```
$ sudo yum install -y python python-dev python-pip
```

### On Ubuntu



Once the above is done install boto

```
$ sudo pip install boto boto3
```

***If not able to install then you will need to enable EPEL repo. The procedure can be found in part 1 of the article series in the installing ansible section.***

Once the above is done we also have to provide AWS user credentials. You can export the AWS Access and Secret Key environment variables

```
export AWS_ACCESS_KEY_ID='AK123'  
export AWS_SECRET_ACCESS_KEY='abc123'
```

If in case even after setting the above environment variables you get an error as the credentials not found then you may also specify the same in the playbook.

Let's now look at some examples of how to use Ansible with S3 buckets and later on to create and terminate instances.

### **Example 1: Create an empty bucket with a folder**

```
---  
- hosts: localhost  
  become: true  
  tasks:  
    - name: Create an S3 bucket  
      s3: aws_access_key=<Access Key> aws_secret_key=<Secret Key> bucket=ansiblevnbucket
```

**Note: Here the mode is created for creating bucket and Permission can be public-read or public-read-write**

Run the playbook and view the bucket created with the development folder within it.



Amazon S3

Search for buckets

+ Create bucket Delete bucket Empty bucket

2 Buckets 1 Public

| Bucket name                              | Access       | Region                |
|--|--------------|-----------------------|
| ansiblevnbucket                          | Public       | Asia Pacific (Mumbai) |
| elasticbeanstalk-ap-south-1-236038045152 | Not public * | Asia Pacific (Mumbai) |

\* Objects might still be publicly accessible due to object ACLs. [Learn more](#)

Amazon S3 > ansiblevnbucket

Overview Properties Permissions Management

Upload + Create folder More

| Name        | Last modified |
|-------------|---------------|
| development | --            |

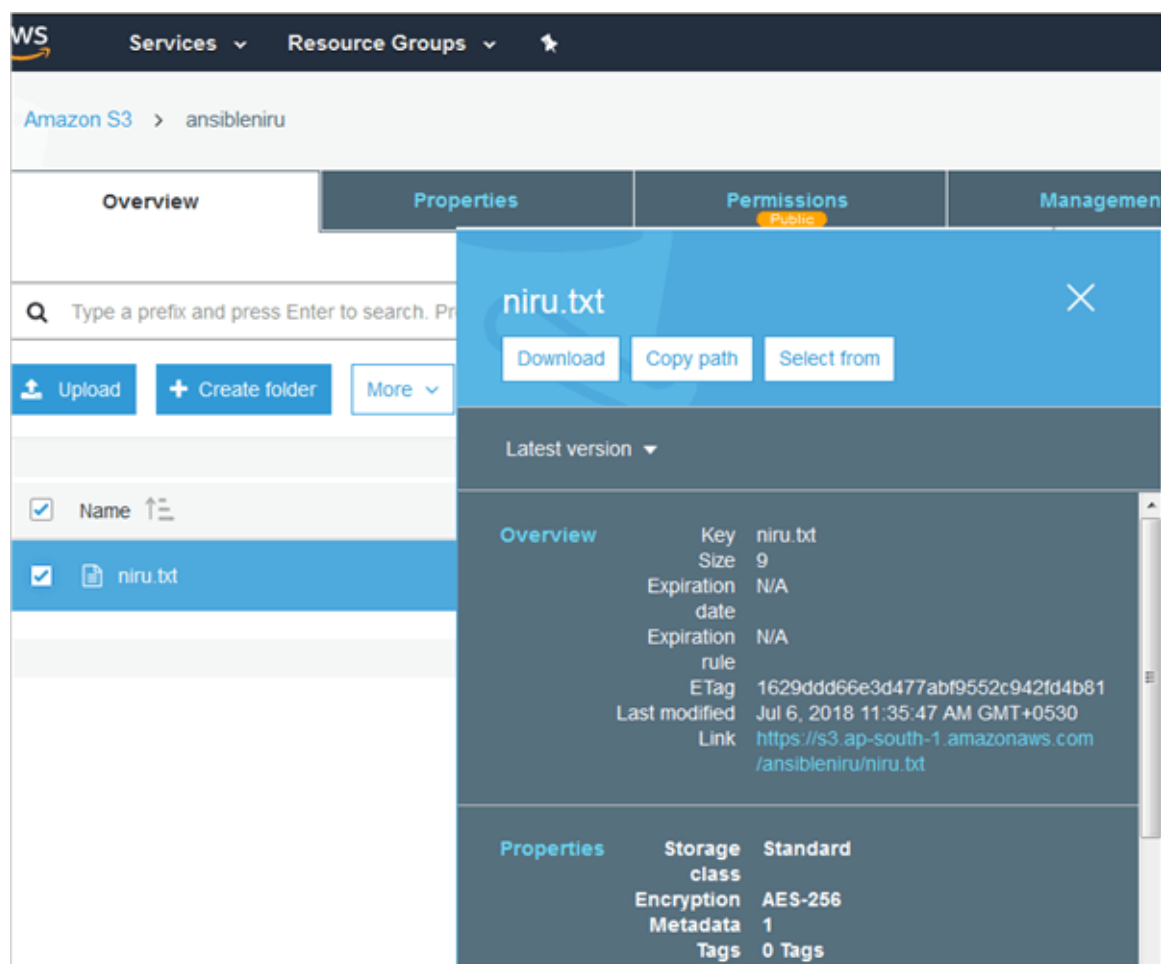
## Example 2: To copy (upload) a file to S3 bucket

```
---
- hosts: localhost
  become: true
  tasks:
    - name: Copy file to S3 bucket
      s3: aws_access_key=<AccessKey> aws_secret_key=<Secret Key> bucket=ansibleniru object=
```

Here the object is the file created within the bucket. It can be a file or folder. In this case, it is a file. The source is the file that is picked up from the local machine which is the Ansible control machine

**Note:** Here the mode is put for uploading object

Run the playbook and look at the S3 bucket.



### Example 3: Get (download) the file from S3 bucket

```
---
- hosts: localhost
  become: true
  tasks:
    - name: Download file from S3 bucket
      s3: aws_access_key=<AccessKey> aws_secret_key=<Secret Key> bucket=ansibleniru object=niru.txt
```

**Note:** Here the mode is get for download object

### Example 4: Delete an object or file from S3 bucket

```
---
- hosts: localhost
  become: true
  tasks:
    - name: Delete an S3 bucket
```

**Note:** Here the mode is delobj for delete object

### **Example 5: Delete a bucket and all contents**

```
---
- hosts: localhost
  become: true
  tasks:
    - name: Delete an S3 bucket
      s3: aws_access_key=<AccessKey> aws_secret_key=<Secret Key> bucket=ansiblevnbucket
```

**Note:** Here the mode is delete for the delete bucket

## Provision an AWS EC2 instance using Ansible

Lastly, I will leave you with one of the most important features of Ansible which is to create or spin up an AWS EC2 instance and also how to terminate the instance. Of course, do not forget to install **boto** which is a pre-requisite and also ensure to export the user “AWS\_ACCESS\_KEY\_ID” and “AWS\_SECRET\_ACCESS\_KEY”.

In case the export does not work ensure to add the same in the code as shown below.

**The code below will show you how to create an EC2 instance along with creating a security group and key pair.**

- Create a security group
- Create key pair and the PEM file
- Create EC2 instance
- Save the EC2 instance IP address to the ansible inventory file

I am assuming that the users doing this exercise are well versed with the AWS EC2 concepts.

Add the below code to a file and run the same to verify the EC2 instance creation in the AWS console. **As the code is big, it is split into 2 pages but ensure you save all to a single yml file.**

```
---
- hosts: localhost
  become: true
  gather_facts: False

vars:
  region: ap-south-1
  instance_type: t2.micro
  ami: ami-5b673c34 # RedHat Linux 7.5
  hosts_file: /etc/ansible/hosts
```

```

- name: Create security group
ec2_group:
  aws_access_key: <access key>
  aws_secret_key: <Secret key>
  name: "vniranjan"
  description: "V Niranjan Security Group"
  region: "{{ region }}"
  rules:
    - proto: tcp
      from_port: 22
      to_port: 22
      cidr_ip: 0.0.0.0/0

- name: Create an EC2 key
ec2_key:
  aws_access_key: <access key>
  aws_secret_key: <Secret key>
  name: "vniranjan"
  region: "{{ region }}"
  register: ec2_key

- name: Save private key (PEM file)
copy: content="{{ec2_key.key.private_key}}" dest=/home/ansible/vniranjan.pem mode=0600
when: ec2_key.changed

- name: Create an ec2 instance
ec2:
  aws_access_key: <access key>
  aws_secret_key: <secret key>
  key_name: vniranjan
  group: vniranjan # security group name
  instance_type: "{{ instance_type }}"
  image: "{{ ami }}"
  wait: true
  region: "{{ region }}"
  count: 1 # default
  count_tag:
    Name: Demo
  instance_tags:
    Name: Demo
  register: ec2

- name: Save IP to inventory file
lineinfile:
  dest: "{{ hosts_file }}"
  insertafter: '\[webservers\]'
  line: "{{ item.private_ip }}"
  with_items: "{{ ec2.instances }}"

```

## Run the playbook



```

ansible@ip-172-31-0-138:~
[ansible@ip-172-31-0-138 ~]$ ansible-playbook createec2ins.yml

PLAY [localhost] *****

TASK [Create security group] *****
changed: [localhost]

TASK [Create an EC2 key] *****
changed: [localhost]

TASK [Save private key (PEM file)] *****
changed: [localhost]

TASK [Create an ec2 instance] *****
changed: [localhost]

TASK [Save IP to inventory file] *****
changed: [localhost] => (item={u'kernel': None, u'root_device_type': u'ebs', u'p
.30.207', u'private_ip': u'172.31.12.15', u'id': u'i-0a8998c269bc8d9d9', u'ebs_o
me': u'/dev/sda1', u'ramdisk': None, u'block_device_mapping': {u'/dev/sda1': {u'
7bc'}}), u'key_name': u'vniranjan', u'image_id': u'ami-5b673c34', u'tenancy': u'd
207.ap-south-1.compute.amazonaws.com', u'state_code': 16, u'tags': {u'Name': u'D
6-30-207.ap-south-1.compute.amazonaws.com', u'region': u'ap-south-1', u'launch_t
_64', u'hypervisor': u'xen'})

PLAY RECAP *****
localhost                : ok=5    changed=5    unreachable=0    failed=0

[ansible@ip-172-31-0-138 ~]$ █

```

/etc/ansible/hosts inventory file updated with private IP

```

ansible@ip-172-31-0-138:~
# - A hostname/ip can be a member of mu

# Ex 1: Ungrouped hosts, specify before a

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging t

[webserver]
172.31.12.15
172.31.19.117
172.1.1.2
172.1.1.1
172.31.0.224

[database]

## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

```

## Login to the instance

`ssh -i "vniranjan.pem" ec2-user@ec2-13-126-30-207.ap-south-1.compute.amazonaws.com`

**(Note: Click on the below image for an enlarged view)**

```

ec2-user@ip-172-31-12-15:~
[ansible@ip-172-31-0-138 ~]$ ssh -i "vniranjan.pem" ec2-user@ec2-13-126-30-207.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-126-30-207.ap-south-1.compute.amazonaws.com (172.31.12.15)' can't be established.
ECDSA key fingerprint is SHA256:6yb5BylyX+1t4bkySvuOPUsTkudnwzk5seQ19U+YaBw.
ECDSA key fingerprint is MD5:e8:10:6c:dc:d2:72:6d:ad:36:5a:ea:a0:6c:3a:9f:ea.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-13-126-30-207.ap-south-1.compute.amazonaws.com,172.31.12.15' (ECDSA) to the list of known hosts.
[ec2-user@ip-172-31-12-15 ~]$

```

## Security Group created

EC2 Dashboard  
Events  
Tags  
Reports  
Limits  
INSTANCES  
Instances  
Launch Templates  
Spot Requests  
Reserved Instances  
Dedicated Hosts  
IMAGES  
AMIs  
Bundle Tasks  
ELASTIC BLOCK STORE  
Volumes  
Snapshots  
NETWORK & SECURITY  
**Security Groups**

Create Security Group Actions

Filter by tags and attributes or search by keyword

|                                     | Name | Group ID    | Group Name | VPC ID       |
|-------------------------------------|------|-------------|------------|--------------|
| <input type="checkbox"/>            |      | sg-7b9b6311 | vnir@6584  | vpc-843c1aed |
| <input type="checkbox"/>            |      | sg-a959f9c1 | default    | vpc-843c1aed |
| <input checked="" type="checkbox"/> |      | sg-b24db5d8 | vniranjan  | vpc-843c1aed |

Security Group: sg-b24db5d8

Description Inbound Outbound Tags

Edit

| Type | Protocol | Port Range |
|------|----------|------------|
| SSH  | TCP      | 22         |

## Key Pair created

EC2 Dashboard  
Events  
Tags  
Reports  
Limits  
NETWORK & SECURITY  
Security Groups  
Elastic IPs  
Placement Groups  
**Key Pairs**  
Network Interfaces  
IMAGES  
AMIs  
Bundle Tasks

Create Key Pair Import Key Pair Delete

Filter by attributes or search by keyword

|                                     | Key pair name | Fingerprint   |
|-------------------------------------|---------------|---|
| <input type="checkbox"/>            | ansiblelearn  | 47:bb:e5:36:63:2e:be:44:db:2e:23:5c:fa:7c:ed:84:36:cf:75:62 |
| <input checked="" type="checkbox"/> | vniranjan     | b8:04:15:1d:1f:df:39:fe:07:94:f0:36:5b:40:8a:94:b0:9c:eb:d1 |

Key Pair: vniranjan

Key pair name vniranjan  
Fingerprint b8:04:15:1d:1f:df:39:fe:07:94:f0:36:5b:40:8a:94:b0:9c:eb:d1

## Terminating EC2 Instances

In this section let's know more about terminating EC2 instances.

**In the following screen you can see that there are 2 instances running and the steps for terminating would be in the following order:**



- Remove security group
- Remove key pair

Resources ▾ Resource Groups ▾ ⭐

Launch Instance ▾ Connect Actions ▾

Filter by tags and attributes or search by keyword

| <input type="checkbox"/>            | Name ▾ | Instance ID ▴        | Instance Type ▾ | Availability Zone ▾ | Instance State ▾ | Status Checks  |
|-------------------------------------|--------|----------------------|-----------------|---------------------|------------------|----------------|
| <input type="checkbox"/>            | master | i-02aef93900ef2b8... | t2.micro        | ap-south-1a         | ● running        | ✓ 2/2 checks.. |
| <input checked="" type="checkbox"/> | Demo   | i-05945003313d2...   | t2.micro        | ap-south-1a         | ● running        | ✓ 2/2 checks.. |
| <input type="checkbox"/>            | Demo   | i-0bbfb7a0a67fb1...  | t2.micro        | ap-south-1a         | ● terminated     |                |
| <input checked="" type="checkbox"/> | Demo   | i-0ce5ce5820bddf...  | t2.micro        | ap-south-1a         | ● running        | ✓ 2/2 checks.. |
| <input type="checkbox"/>            | target | i-0ffe8e20d69a1b5... | t2.micro        | ap-south-1a         | ● running        | ✓ 2/2 checks.. |

### Playbook to terminate both the EC2 instances

```

---
- hosts: localhost
  gather_facts: false
  connection: local
  vars:
    instance_ids:
      - 'i-05945003313d20603' # Replace these with your EC2 instance id's
      - 'i-0ce5ce5820bddf610'

    region: ap-south-1
    keypair_name: vniranjan
    securitygroup_name: vniranjan
  tasks:
    - name: Terminate EC2 instance
      ec2:
        aws_access_key: <access key>
        aws_secret_key: <Secret key>
        instance_ids: '{{instance_ids}}'
        region: '{{region}}'
        state: absent
        wait: true

    - name: Remove EC2 Key
      ec2_key:
        aws_access_key: <access key>
        aws_secret_key: <Secret key>
        name: '{{keypair_name}}'
        state: absent
        region: '{{region}}'

    - name: Remove Security Group
      ec2_group:
        aws_access_key: <access key>
        aws_secret_key: <Secret key>
        name: '{{securitygroup_name}}'
        state: absent
        region: '{{region}}'

```

## Summary

Today, most of the IT organizations are looking at some kind of differentiators to win business and showcase the same to their clients. I would say automation is definitely one of the key differentiators.

With tools like Ansible, I am of the opinion that you should be able to automate most of the repetitive manual tasks.

Thus what we have learned from this **3-part Ansible Tutorial series** showcases Ansible as a very popular and powerful configuration management tool which helps in different areas of automation ranging from task automation, application deployment, and cloud provisioning. Thereby, we are primarily talking about IT orchestration.

***Hope you enjoyed the range of Ansible tutorials and I'm sure that you would have gained immense knowledge on the concept by now.***

*Next, we will learn how to Integrate Jenkins with Selenium which is also a part of our DevOps training series.*

**PREV Tutorial | NEXT Tutorial**

Join Over 200,000+ Testers

Get premium ebooks and testing tips.

SUBSCRIBE NOW!

### Top FREE Training Tutorials

[QA Training](#)

[Selenium Tutorials](#)

[QTP Tutorials](#)

[ISTQB Study Guide](#)

[ISTQB Premium Study Guide](#)