

Ansible is an open-source software provisioning, configuration management, and application-deployment tool. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration.

**Stable release:**2.8.1 / June 6, 2019

### Overview Of Ansible

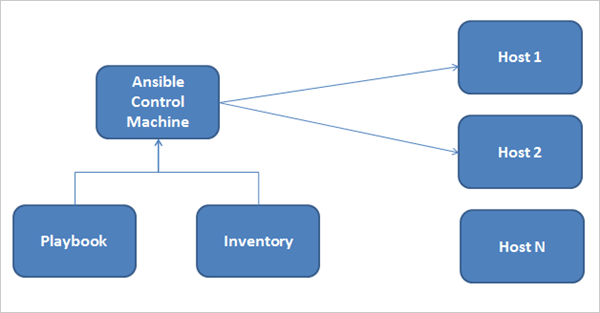
Most importantly Ansible does not use an agent to automate tasks on different machines.

Ansible ensures maintaining exact versions and up to date information to the software packages.

**For Example**,  if you want to install JDK 8 or Tomcat or any other software package in 10 or 20 different machines it is not actually feasible to go to all the machines and install them rather use Ansible to automate the installation or even software deployments using Playbooks and Inventory written in a very simple language.

**So Ansible is:**

* Free and Open Source
* Maintained by Redhat
* Essentially a server configuration
* Configuration Management



Inventory : is host files (contains node ip addres)

Playbook: yml/yaml scripting

* Installation and configuration process
* Inventory
* Ansible Modules
* Ad-hoc commands,
* Task automation using playbooks
* Ansible roles
* Ansible vault
* Ansible and AWS

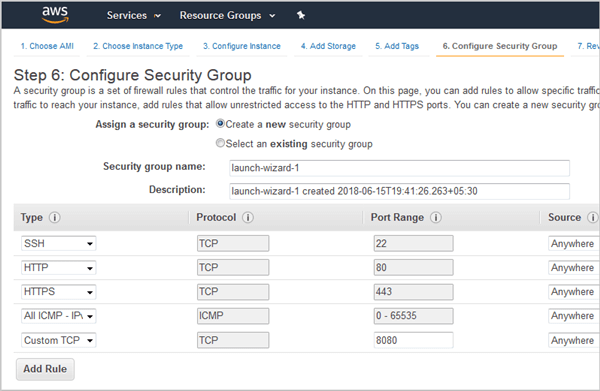
### Ansible Installation Process

Ansible can be installed and run from any machine.

Typically you will need a **Control machine** for installation which should be Linux. Windows machine does not support being a control machine. The control machine will manage the other remote machines. As mentioned earlier Ansible uses SSH to manage remote machines.

Throughout this tutorial, I will be using AWS EC2 instances to showcase the examples. I have used 2 instances (one control machine and other as a target for automating tasks) and Redhat Linux 7.5.

Whether on-premise or cloud instances you will need to open ports appropriately based on the tasks being automated. I have the following ports open as a part of the security group for the EC2 instances to demonstrate the examples mentioned in the tutorial.



In the above screen, I have mentioned opening port 8080 as I will be showing about automating software deployment automation using Tomcat which will be useful from a DevOps point of view, especially during the continuous delivery process.

#### Installation of Ansible using YUM

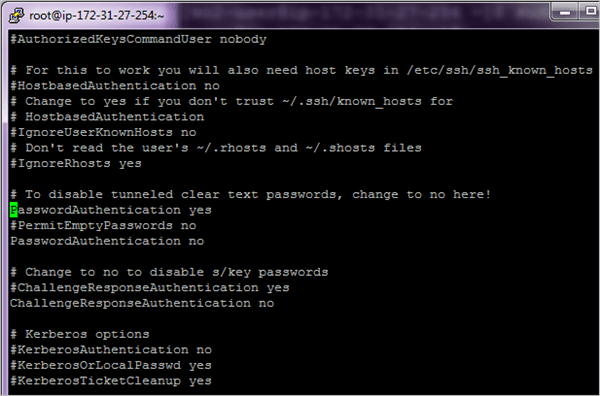
As mentioned before I will be using one control machine and a target machine. To start with installation, perform the steps as shown below in both the machines.

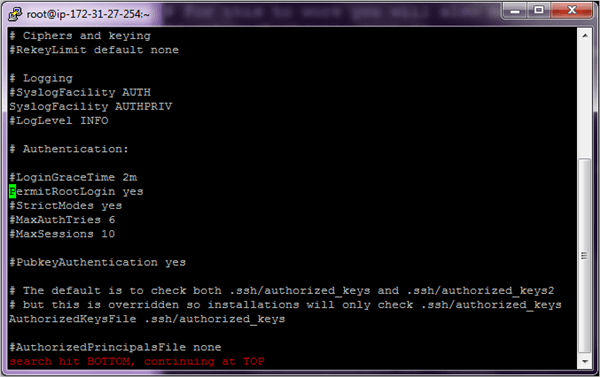
**a)** Create a common id on both the machines, for **Example**, **ansible**with SUDO privileges. This id will be used for communicating across all the machines involved for automation of tasks.

**# useradd ansible**

**# passwd ansible**

**b)** Edit the**/etc/ssh/sshd\_config**file on the **control machine** and uncomment out the lines for **PasswordAuthentication and PermitRootLogin**





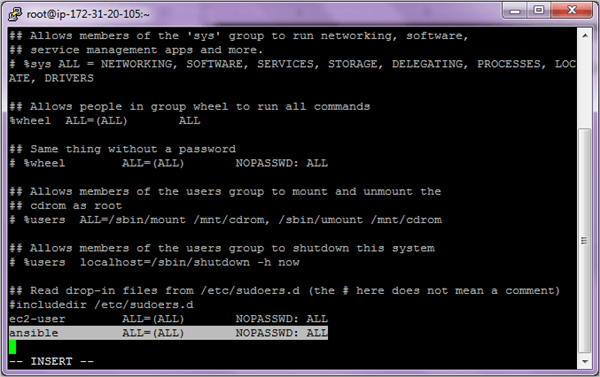
Perform the above steps on both the machines. Once completed, restart the **sshd** service on both the machines.

**# systemctl restart sshd**

**c)** For complete automation of tasks, we will need passwordless SSH authentication else the whole process will not be used if you have to key in the password every time.

So post the changes done above if we run the command ssh <target machine> and ssh <control machine> we will need to key in the password every time which is not the right procedure to execute Ansible tasks.

**d)** To enable passwordless authentication to perform the steps shown below. Firstly add the user **ansible**to the **/etc/sudoers** file on both the machines which will enable the user **ansible**to run any command which requires root privileges.



Save and exit the file after adding the user.

**e)** Going forward we will use the user **ansible**to perform all the steps. So switch to the user **ansible.**

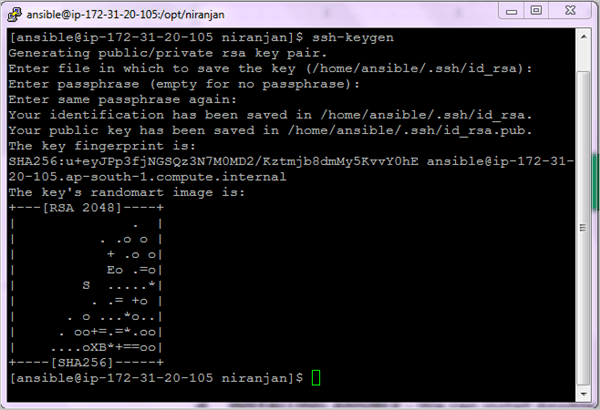
**Control Machine**

**su – ansible AND Target Machine su – Ansible**

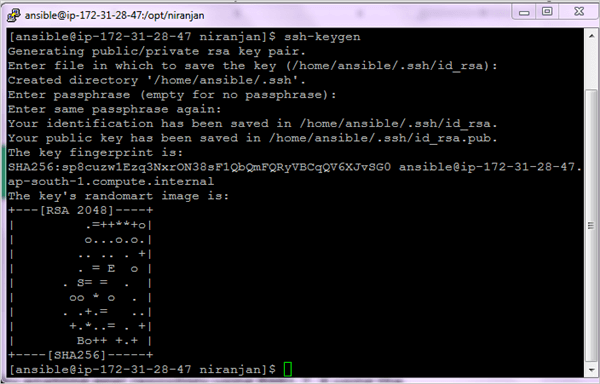
5.user ansible

**Control Machine ssh-keygen**

(/home/ansible/.ssh/id\_rsa):

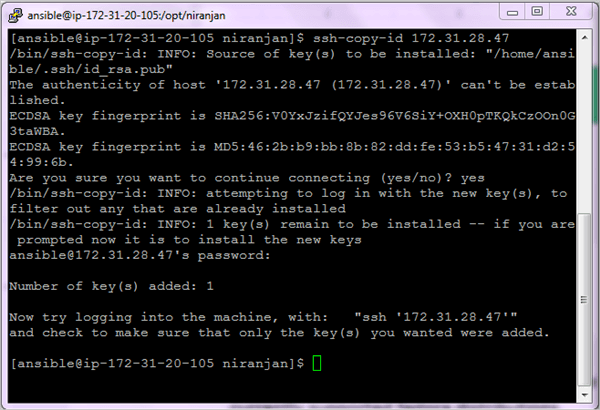


**Target Machine ssh-keygen**

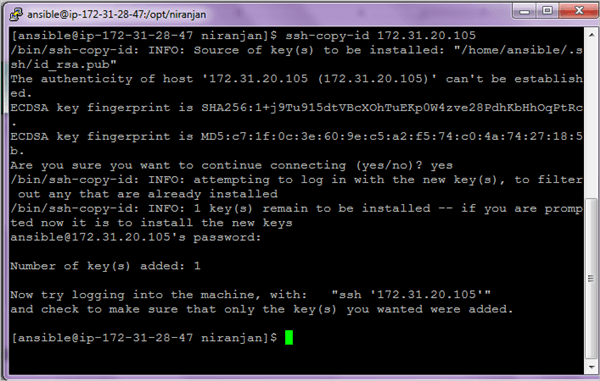


Copy the ssh key to the target machine and vice versa.

**Control Machine ssh-copy-id <IP-Address-Host-Machine>**

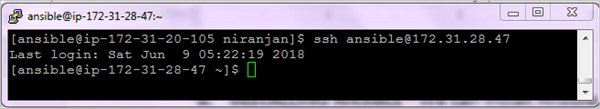


**Target Machine ssh-copy-id <IP-Address-Control-Machine>**

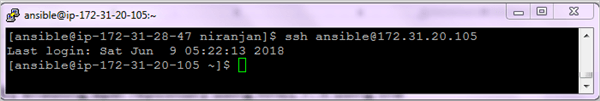


We are now able to log in without entering the password. After the check out of the ssh connectivity on both the machines and be logged in as ansible user.

**Control Machine: ssh ansible@<IP-Address-Host-Machine**



**Target Machine: ssh ansible@<IP-Address-Control-Machine>**



**f)** Install wget if not installed on both the machines.

**$ sudo yum install wget -y**

**g)** We can now install **ansible** **on the Control machine only**by enabling the EPEL repo from fedora which provides add-on software packages. Perform the following steps to install **ANSIBLE.**

**$ wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm**

**$ sudo rpm -ivh epel-release-latest-7.noarch.rpm**

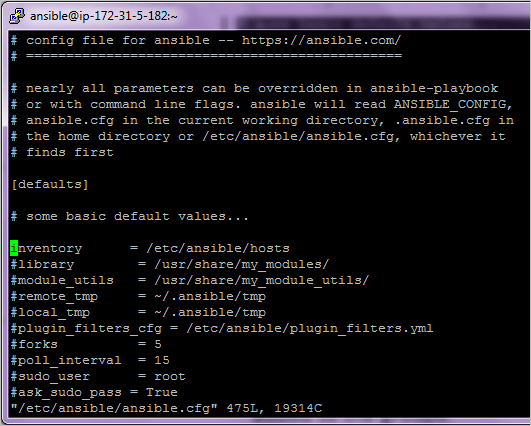
**$ sudo yum install ansible -y**

**$ ansible --version**

**The ansible version used is 2.5.3**

**h)** Edit the **ansible.cfg** file and enable the inventory file parameter on the Control machine.

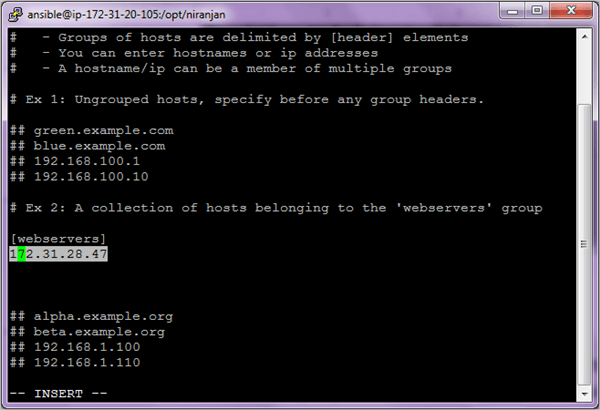
$ **sudo vi /etc/ansible/ansible.cfg**



**i)** Ansible uses the concept of Inventory to manage and track the target machines. By default, this file is located in **/etc/ansible/hosts** and can be changed as well. A host file consists of groups for better classification and multiple machines under the group. All the required machines can be added to those groups.

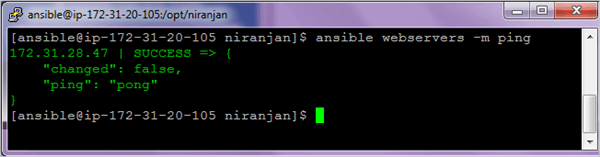
Every group is denoted by a square bracket and a group name within. A server can actually exist in multiple groups.

Edit the inventory file **/etc/ansible/hosts**and add all the servers which need to be managed.



**j)** To test the connectivity of the servers under the webserver's group run the **ansible ping** command as shown. Here **ping** is a module which performs a particular function to test whether the hosts can be connected as defined in the inventory file or not. We will see more about various modules and its examples in the next section.

**$ ansible webservers –m ping**



To list the hosts in the inventory file, you can run the below command

**$ ansible webservers --list-hosts**

**Ansible consist of 3 main components**

* Control Machine
* Inventory
* Playbook

The control machine manages the execution of the Playbook. It can be installed on your laptop or on any machine on the internet.

The Inventory file provides a complete list of all the target machines on which various modules are run by doing an ssh connection and install the necessary software’s.

The playbook consists of steps that the control mechanism will perform on the servers defined in the inventory file.

Very important to understand here is that Ansible interacts with all the servers defined in the inventory through the SSH protocol which is a secure method of remote login. Every operation is done and file transfer is encrypted.

So as you would have seen in the previous section Ansible does not use any kind of database for installation and is very easy to install, we will now proceed with the actual usage of Ansible starting with Modules which is the main building block.

### Ansible Modules

Modules are the main building blocks of Ansible and are basically reusable scripts that are used by Ansible playbooks. Ansible comes with a number of reusable modules. These include functionality for controlling services, software package installation, working with files and directories etc.

The syntax is as follows while running the ad-hoc commands which help in running single or simple tasks just once and which need not be run later. For **E.g.** just installing Tomcat on all servers.

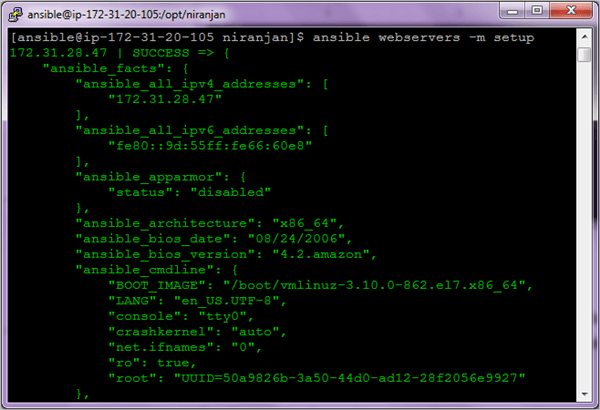
ansible hostORgroup -m module\_name -a "arguments" -u username --become

Let’s have a look at some of the most popular Ansible modules and their usage through the ad-hoc commands and later on in the playbook.

#### #1) Setup Module

To get information about the network or hardware or OS version or memory related information the setup module will help to gather the same about the target machines. On the control, the machine runs the below command.

**$ ansible webservers –m setup**



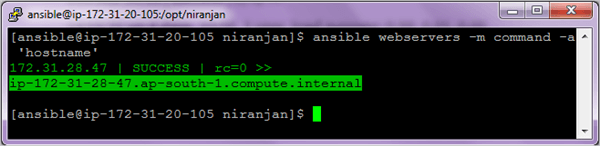
#### #2) Command Module

The command module simply executes a specific command on the target machine and gives the output.

**Some of the Examples are given below**

**$ ansible webservers –m command - an ‘uptime’**

**$ ansible webservers –m command –a ‘hostname’**

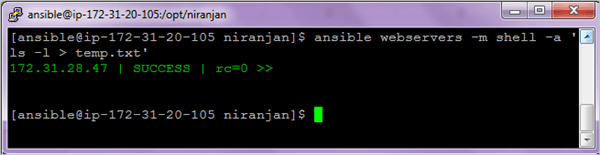


#### #3) Shell Module

To execute any command in the shell of your choice you can use the Shell module. The shell module commands are run in /bin/sh shell and you can make use of the operators like ‘>’ or ‘|’ (pipe symbol or even environment variables.

So primarily the difference between the Shell and Command module is that if you actually do not need to use the operators like the ones mentioned then you could use the command module.

**$ ansible webservers -m shell -a 'ls -l > temp.txt'**



On the machines under webservers group check for the file created and run the command to view the text file.

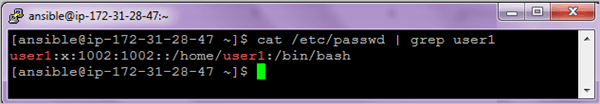
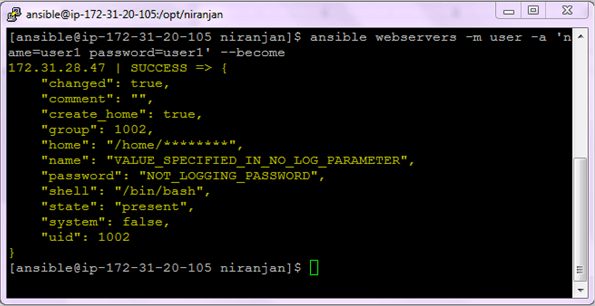
**$ ansible webservers –m command -a ‘cat temp.txt’**

#### #4) User Module

Using this module one can create or delete users.

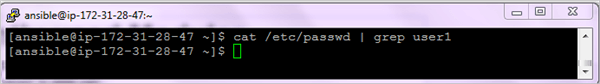
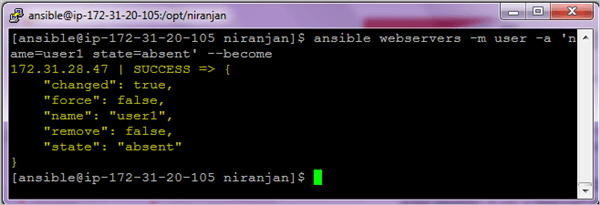
**To add user**

**$ ansible webservers -m user -a 'name=user1 password=user1' --become**



**To delete user**

**$ ansible webservers -m user -a 'name=user1 state=absent' --become**



**Options:**

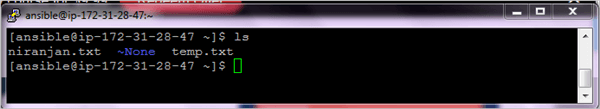
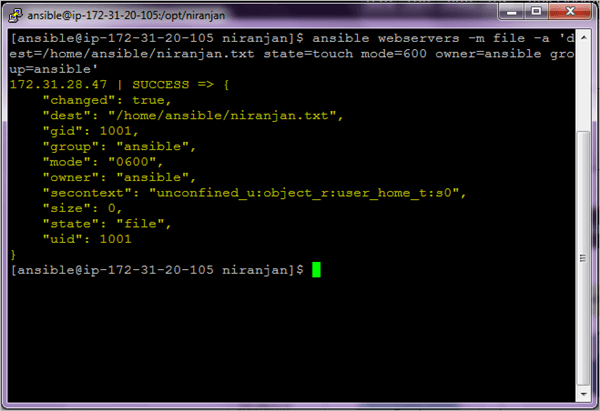
* **become** – Privilege to the superuser to run the command
* **state=absent** to delete the user

#### #5) File Module

This module is used to create files, directories, set, or change file permissions and ownership etc

**Example 1:** Create a file

**$ ansible webservers -m file -a 'dest=/home/ansible/niranjan.txt state=touch mode=600 owner=ansible group=ansible'**

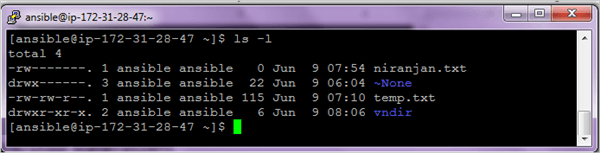
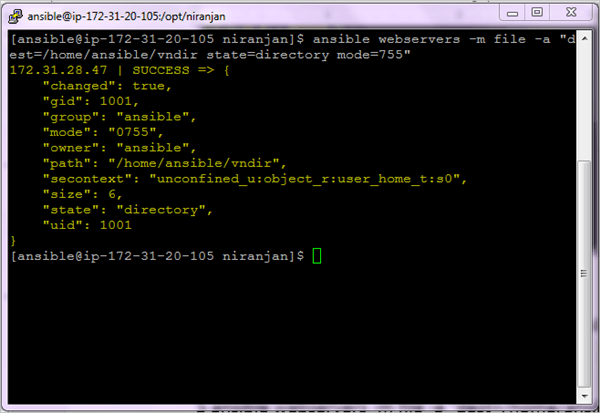


**Example 2:** Create a directory

To create a directory using the file module, you need to set two parameters.

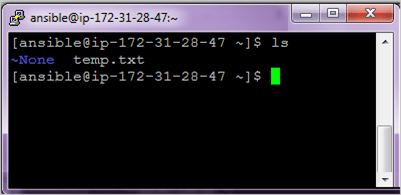
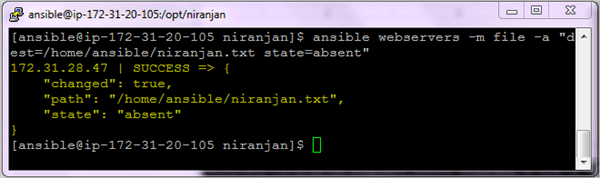
* Path(alias – name, dest) – This is the absolute path of the directory to be created.
* State – You should enter the value as ‘directory.’ By default, the value is ‘file’.

**$ ansible webservers -m file -a "dest=/home/ansible/vndir state=directory mode=755"**



**Example 3:**Delete a file

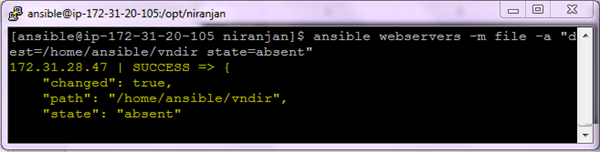
**$ ansible webservers -m file -a "dest=/home/ansible/niranjan.txt state=absent"**

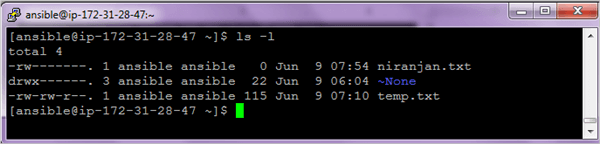


**Example 4:** Delete a directory

You can delete a directory by setting the state parameter value to **absent**. The directory and all its contents will be deleted.

**$ ansible webservers -m file -a "dest=/home/ansible/vndir state=absent"**

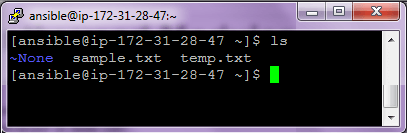
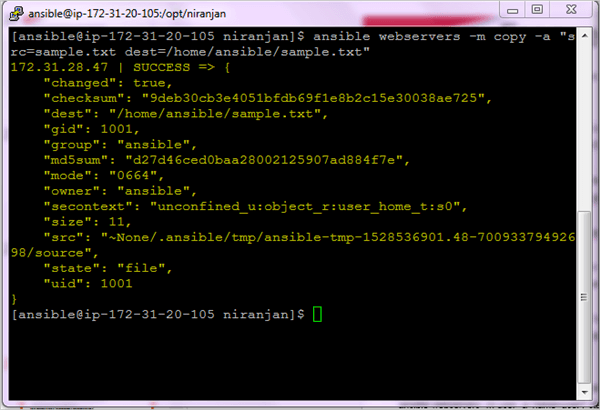




#### #6) Copy Module

It is used for copying files to multiple target machines.

**$ ansible webservers -m copy -a "src=sample.txt dest=/home/ansible/sample.txt"**

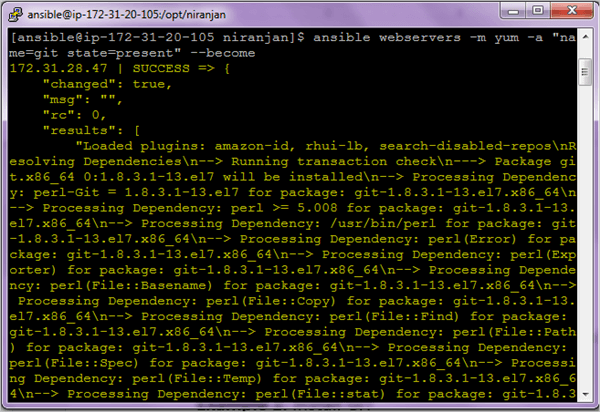


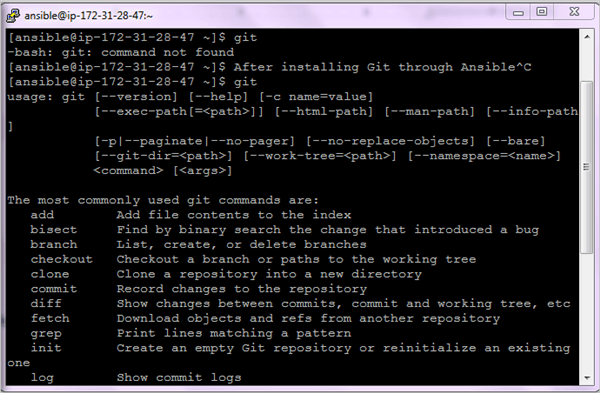
#### #7) Managing Software Packages

If you need to install software packages through ‘yum’ or ‘apt’ you can use the below commands.

**Example 1:** Install GIT

**$ ansible webservers –m yum -a “name=git state=present” --become**





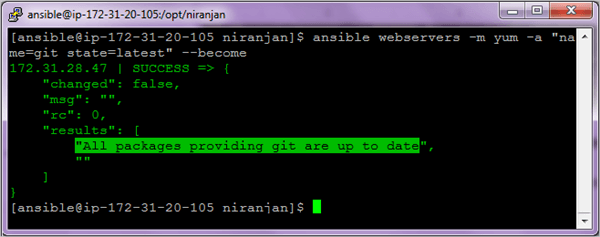
On the right-hand window, you can see if git is not installed it will give command not found and once installed it will show up the output.

In this command, **state=present** will check if the package is installed or not and if not installed it will install the latest version.

**Example 2:** Check if the package is installed & update it to the latest version.

**$ ansible webservers -m yum -a “name=git state=latest”**

In the above command, **state=latest**will update the package to the latest version only.



**Example 3:** Install Apache Webserver

**$ ansible webservers -m yum -a "name=httpd state=present" –become**

**Example 4:**Check if Maven is installed or not.

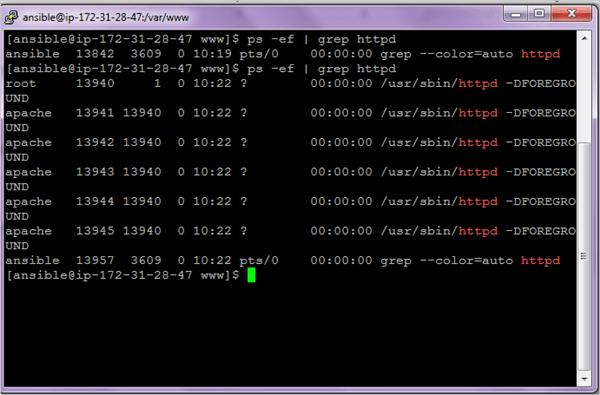
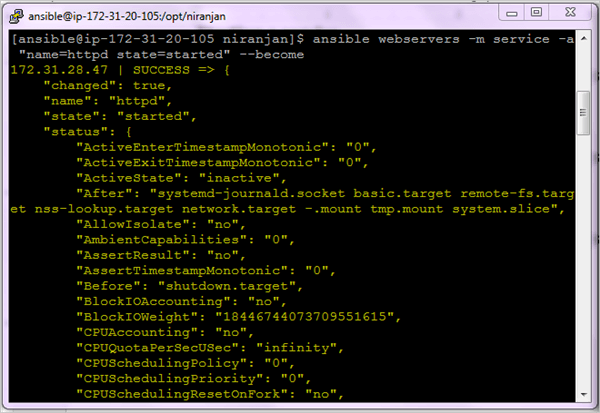
**$ ansible webservers -m yum -a "name=maven state=absent" –become**

#### #8) Managing Services Module

To manage services with ansible, we use a module **‘service’.**

**Starting a service**

**$ ansible webservers -m service -a “name=httpd state=started” --become**



**Stopping a service**

**$ ansible webservers -m service -a “name=httpd state=stopped” --become**

**Restarting a service**

**$ ansible webservers -m service -a “name=httpd state=restarted --become**

Installation Process

Mainly, there are two types of machines when we talk about deployment −

* **Control machine** − Machine from where we can manage other machines.
* **Remote machine** − Machines which are handled/controlled by control machine.

### Installation through Apt on Ubuntu Machine

For installing Ansible you have to configure PPA on your machine. For this, you have to run the following line of code −

$ sudo apt-get update

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible $ sudo apt-get update

$ sudo apt-get install ansible

## Understanding YAML

## Every YAML file optionally starts with “---” and ends with “...”.

In this section, we will learn the different ways in which the YAML data is represented.

### key-value pair

YAML uses simple key-value pair to represent the data. The dictionary is represented in key: value pair.

**Note** − There should be space between : and value.

### Example: A student record

--- #Optional YAML start syntax

james:

name: james john

rollNo: 34

div: B

sex: male

… #Optional YAML end syntax

### Abbreviation

You can also use abbreviation to represent dictionaries.

### Example

James: {name: james john, rollNo: 34, div: B, sex: male}

## Representing List

We can also represent List in YAML. Every element(member) of list should be written in a new line with same indentation starting with “- “ (- and space).

### Example

---

countries:

- America

- China

- Canada

- Iceland

…

### Abbreviation

You can also use abbreviation to represent lists.

### Example

Countries: [‘America’, ‘China’, ‘Canada’, ‘Iceland’]

### List inside Dictionaries

We can use list inside dictionaries, i.e., value of key is list.

### Example

---

james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

…

### List of Dictionaries

We can also make list of dictionaries.

### Example

---

- james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

- robert:

name: robert richardson

rollNo: 53

div: B

sex: male

likes:

- biology

- chemistry

…

YAML uses “|” to include newlines while showing multiple lines and “>” to suppress newlines while showing multiple lines. Due to this we can read and edit large lines. In both the cases intendentation will be ignored.

We can also represent **Boolean** (True/false) values in YAML. where **boolean**values can be case insensitive.

### Example

---

- james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

result:

maths: 87

chemistry: 45

biology: 56

physics: 70

english: 80

passed: TRUE

messageIncludeNewLines: |

Congratulation!!

You passed with 79%

messageExcludeNewLines: >

Congratulation!!

You passed with 79%

## Some common words related to Ansible.

**Service/Server** − A process on the machine that provides the service.

**Machine** − A physical server, vm(virtual machine) or a container.

**Target machine** − A machine we are about to configure with Ansible.

**Task** − An action(run this, delete that) etc managed by Ansible.

**Playbook** − The yml file where Ansible commands are written and yml is executed on a machine.

## Playbook Structure

Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a playbook.

The function of a play is to map a set of instructions defined against a particular host.

YAML is a strict typed language; so, extra care needs to be taken while writing the YAML files. There are different YAML editors but we will prefer to use a simple editor like notepad++. Just open notepad++ and copy and paste the below yaml and change the language to YAML (Language → YAML).

A YAML starts with --- (3 hyphens)

## Create a Playbook

Let us start by writing a sample YAML file. We will walk through each section written in a yaml file.

---

name: install and configure DB

hosts: testServer

become: yes

vars:

oracle\_db\_port\_value : 1521

tasks:

-name: Install the Oracle DB

yum: <code to install the DB>

-name: Ensure the installed service is enabled and running

service:

name: <your service name>

The above is a sample Playbook where we are trying to cover the basic syntax of a playbook. Save the above content in a file as **test.yml**. A YAML syntax needs to follow the correct indentation and one needs to be a little careful while writing the syntax.

## The Different YAML Tags

Let us now go through the different YAML tags. The different tags are described below −

### name

This tag specifies the name of the Ansible playbook. As in what this playbook will be doing. Any logical name can be given to the playbook.

### hosts

This tag specifies the lists of hosts or host group against which we want to run the task. The hosts field/tag is mandatory. It tells Ansible on which hosts to run the listed tasks. The tasks can be run on the same machine or on a remote machine. One can run the tasks on multiple machines and hence hosts tag can have a group of hosts’ entry as well.

### vars

Vars tag lets you define the variables which you can use in your playbook. Usage is similar to variables in any programming language.

### tasks

All playbooks should contain tasks or a list of tasks to be executed. Tasks are a list of actions one needs to perform. A tasks field contains the name of the task. This works as the help text for the user. It is not mandatory but proves useful in debugging the playbook. Each task internally links to a piece of code called a module. A module that should be executed, and arguments that are required for the module you want to execute.