SSH Key Generation

Ansible uses SSH to communicate between the nodes.
To set up an SSH connection, follow the steps mentioned below:

- Setting Up SSH Command
- Generating SSH Key
- Copy the SSH Key on the Hosts
- Check the SSH Connection

```
#Setting Up SSH Command
$ sudo apt-get install openssh-server

#Generating SSH Key
$ ssh-keygen

#Copy the SSH Key on the Hosts
$ ssh-copy-id hostname

#Check the SSH Connection
$ ssh <nodeName>
```

Install Ansible

To install Ansible in Debian based Linux, you

Install Ansible

To install Ansible in Debian based Linux, you can follow the following steps:

- Add the Ansible repository to your system, before installing Ansible
- Then run the update command before installing, to update the existing packages
- After that install the Ansible package

```
#Add Ansible repository
$ sudo apt-add-repository ppa:ansible/ansible


#Run the update command
$ sudo apt-get update


#Install Ansible package
$ sudo apt-get install ansible


#Check Ansible Version
$ ansible –version
```

Inventory Files & Hosts Patterns

Set & Check Hosts Connection

Follow the below steps to set hosts and then check their connection.

```
#Set up hosts by editing the hosts' file in the Ansible directory
$ sudo nano /etc/ansible/hosts

#To check the connection to hosts
#Change your directory to /etc/Ansible
$ cd /etc/ansible

#Ansible’s ping module allows you to check whether Ansible is connecting to hosts
$  ansible –m ping <hosts>

#To check on servers individually
$ ansible -m ping server name

#To check a particular server group
$ ansible -m ping servergroupname
```

Ansible Host Patterns

Refer to the following table to know the Ansible Host Patterns, as described in playbooks.

Ansible Host Patterns	
all	All hosts in inventory
*	All hosts in inventory
ungrouped	All hosts in inventory not appearing within a group
10.0.0.*	All hosts with an IP starting 10.0.0.*
webservers	The group webservers
webservers:!moscow	Only hosts in webservers, not also in group moscow
webservers:&moscow	Only hosts in the group's webservers and moscow

Example Inventory File

Below is an example inventory file, which you can refer to understand the various parameters.

```
#Default location for host file
$ /etc/ansible/hosts


#To define location for inventory, in CLI
-i<path>


#example host file

ungrouped.example.com                                    #An ungrouped host

[webservers]                                             #a group called webservers
beta.example.com ansible_host = 10.0.0.5                 #ssh to 10.0.0.5
```

```
github.example.com ansible_ssh_user = abc                    #ssh as user a
bc

[clouds]
cloud.example.com fileuser = alice                           #fileuser is a host vari
able

[moscow]
beta.example.com                                             #host (DNS will resolve)
telecom.example.com                                          #host(DNS will resolve)

[dev1:children]                                              #dev1 is a group containing
webservers                                                   #all hosts in group webservers
clouds                                                       #all hosts in group clouds
```

Ad-Hoc Commands

Ad-Hoc commands are quick commands which are used to perform the actions, that won't be saved for later.
Some of the tasks that you can perform using Adhoc commands are as follows:

- Parallelism and Shell Commands
- File Transfer
- Managing Packages
- Deploying From Source Control
- Managing Services

I am going to explain all these tasks, with a basic example.

Parallelism & Shell Commands

In this section, I am going to tell you the commands, for parallelism and shell.

```
#To set up SSH agent
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa

#To use SSH with a password instead of keys, you can use --ask-pass (-K)
$ ansible europe -a "/sbin/reboot" -f 20

#To run /usr/bin/ansible from a user account, not the root
$ ansible europe -a "/usr/bin/foo" -u username

#To run commands through privilege escalation and not through user account
$ ansible europe -a "/usr/bin/foo" -u username --become [--ask-bec
ome-pass]

#If you are using password less method then use  --ask-become-pass (-K)
#to interactively get the  password to be  used

#You can become a user, other than root by using --become-user
$ ansible europe -a "/usr/bin/foo" -u username --become --become-
user otheruser [--ask-become-pass]
```

File Transfer

Ansible can perform secure transmissions of files to multiple machines in parallel.

```
#Transfer a file directly to many servers
$ ansible europe -m copy -a "src=/etc/hosts dest=/tmp/hosts"

#To change the ownership and permissions on files
$ ansible webservers -m file -a "dest=/srv/foo/a.txt mode=600"
```

```
$ ansible webservers -m file -a "dest=/srv/foo/b.txt mode=600 owne
r=example group=example"


#To create directories
$ ansible webservers -m file -a "dest=/path/to/c mode=755 owner=e
xample group=example state=directory"


#To delete directories (recursively) and delete files
$ ansible webservers -m file -a "dest=/path/to/c state=absent"
```

Manage Packages

This section consists of commands to manage packages.

```
#To ensure that a package is installed, but doesn’t get updated
$ ansible webservers -m apt -a "name=acme state=present"

#To ensure that a package is installed to a specific version
$ ansible webservers -m apt -a "name=acme-1.5 state=present"

#To ensure that a package at the latest version
$ ansible webservers -m apt -a "name=acme state=latest"

#To ensure that a package is not installed
$ ansible webservers -m apt -a "name=acme state=absent”
```

Manage Services

This section consists of commands to manage services.

```
#To ensure a service is started on all web servers
$ ansible webservers -m service -a "name=httpd state=started"

#To restart a service on all web servers
$ ansible webservers -m service -a "name=httpd state=restarted"

#To ensure a service is stopped
$ ansible webservers -m service -a "name=httpd state=stopped
```

Playbooks

Playbooks in Ansible are written in YAML format. It is a human-readable data serialization language that is commonly used for configuration files. It can also be used in many applications where data is being stored.

A playbook has various parameters that you need to mention, like Hosts & Users, Variables, Tasks, Handlers, Modules and Return Values.

Sample Playbook

This is the sample playbook to start the Apache httpd Server program.

```
#Every YAML file starts with ---
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    apt: 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 (and enable it at boot)
    service: name=httpd state=started enabled=yes
```

```
handlers:
  - name: restart apache
    service: name=httpd state=restarted
```

Writing a Playbook

Follow the below steps to write a run a playbook. For the ease of understanding, the commands are in a generalized format.

```
#SSH Key Generation
$ ssh key-gen

#Copy the  generated public SSH key on your hosts
$ ssh-copy-id -i root@<IP address of your host>

# List the IP addresses of your hosts/nodes in your inventory
$ vi /etc/ansible/hosts

#Ping to ensure a connection has been established
$ ansible -m ping <Name of the Host>


#You do not have to follow the above steps, if you already have host connected to th
e control machine.

#Create a Playbook
$ vi <name of your file>.yml


#To write the playbook refer to the snapshot here.


#Run the playbook
$ ansible-playbook <name of your file>.yml
```