What is Ansible?

* Ansible is an open-source automation tool used for configuration management, application deployment, and task automation.
* It uses a simple syntax, called YAML, to define tasks.
* Ansible uses an agentless architecture, meaning it does not require any software or agent to be installed on the target machine.
* Instead, it connects to remote machines using SSH or WinRM protocols and executes tasks using modules.
* Ansible enables users to automate complex tasks with ease, such as deploying applications, configuring servers, and managing network devices.

Why Ansible?

* **Easy to learn and use**: Ansible uses a simple syntax and doesn't require any special coding skills. It is easy to understand and use even for non-programmers.
* **Agentless architecture**: Ansible uses an agentless architecture, which means there is no need to install any agent on the target machine. This reduces the complexity of the setup process and makes it easy to manage large numbers of machines.
* **Idempotent**: Ansible ensures that the desired state of the system is maintained, irrespective of the number of times the playbook is run. This makes it idempotent, reducing the risk of errors caused by repeated execution.
* **Supports multiple platforms**: Ansible is capable of managing a wide range of systems, including Linux, macOS, and Windows. It supports multiple cloud platforms like AWS, Azure, and Google Cloud.
* **Large and active community**: Ansible has a large and active community of users who contribute to its development and offer support. This means that users can easily find help and resources, making it easier to learn and use.

Ansible Vocabulary

* **Inventory**: A file or set of files that contain information about the hosts and groups of hosts that Ansible manages.
* **Playbook**: A YAML file that defines a set of tasks to be executed on one or more hosts.
* **Task**: A unit of work to be performed on a target host by Ansible.
* **Module**: A built-in or custom Python script that performs a specific task in Ansible, such as managing files, packages, or users.
* **Role**: A collection of tasks, files, templates, and other resources organized into a directory structure that can be reused across multiple playbooks.
* **Play**: A collection of tasks and associated metadata that are executed in a specific order.
* **Facts**: Variables that represent the state of a target host, such as the hostname, IP address, or installed packages.
* **Handler**: A task that is triggered by another task and performs an action if a condition is met, such as restarting a service.
* **Vault**: A feature that allows sensitive data such as passwords, keys, or certificates to be encrypted and stored securely in Ansible playbooks.
* **Ad-hoc command**: A single command that is run on one or more hosts using the ansible command-line tool, without the need for a playbook or inventory file.

Ansible Real time usage:

* **Server provisioning**: Ansible can be used to automate the provisioning of servers, including the installation of operating systems, applications, and configuration files.
* **Application deployment**: Ansible can be used to automate the deployment of applications, including the installation of dependencies, configuration files, and deployment scripts.
* **Configuration management**: Ansible can be used to automate the configuration of servers and applications, including the management of user accounts, network settings, and system services.
* **Continuous integration and continuous delivery (CI/CD):** Ansible can be used to automate the build, test, and deployment of applications as part of a CI/CD pipeline.
* **Cloud infrastructure management**: Ansible can be used to automate the provisioning and management of cloud infrastructure, including virtual machines, containers, and cloud services.
* **Security compliance**: Ansible can be used to automate security compliance checks and remediation, ensuring that servers and applications are configured to meet security standards.
* **Disaster recovery**: Ansible can be used to automate the deployment and configuration of backup servers and applications in the event of a disaster.

Ansible Playbooks:

* **Hosts**: The list of hosts or groups of hosts that the playbook will run on. This can be defined at the top of the playbook, or in the inventory file.
* **Variables**: The variables that are used in the playbook, such as package names, file paths, or user accounts. Variables can be defined at the top of the playbook, in the inventory file, or in external files.
* **Tasks**: The list of tasks that will be executed on the target hosts. Tasks can be simple shell commands, or complex sets of instructions using Ansible modules.
* **Modules**: The built-in or custom Python scripts that perform specific actions on the target hosts, such as managing files, users, or packages.
* **Handlers**: Special tasks that are only executed if triggered by another task. Handlers are often used to restart services or reload configuration files after a change has been made.
* **Roles**: Reusable collections of tasks, variables, and other resources that can be included in multiple playbooks. Roles are organized into a directory structure, and can be installed from Ansible Galaxy or created from scratch.
* **Conditionals**: Statements that allow tasks to be executed only if certain conditions are met, such as checking if a file exists or a package is installed.
* **Loops**: Statements that allow tasks to be repeated multiple times, such as installing multiple packages or creating multiple user accounts.
* **Templates**: Jinja2 templates that allow files to be dynamically generated based on variables and other data. Templates are often used to create configuration files for services or applications.

Ansible Modules:

Ansible modules are the building blocks of automation in Ansible. They are small pieces of code that Ansible uses to perform specific tasks, such as managing files, installing packages, configuring network devices, and deploying applications.

There are three types of modules in Ansible:

* **Core modules**: These are modules that are shipped with Ansible and are maintained by the Ansible community.
* **Extras modules**: These are modules that are contributed by the Ansible community and are not included in the core modules.
* **Custom modules**: These are modules that are created by users to perform specific tasks that are not covered by the core or extras modules.

Some of the commonly used modules are:

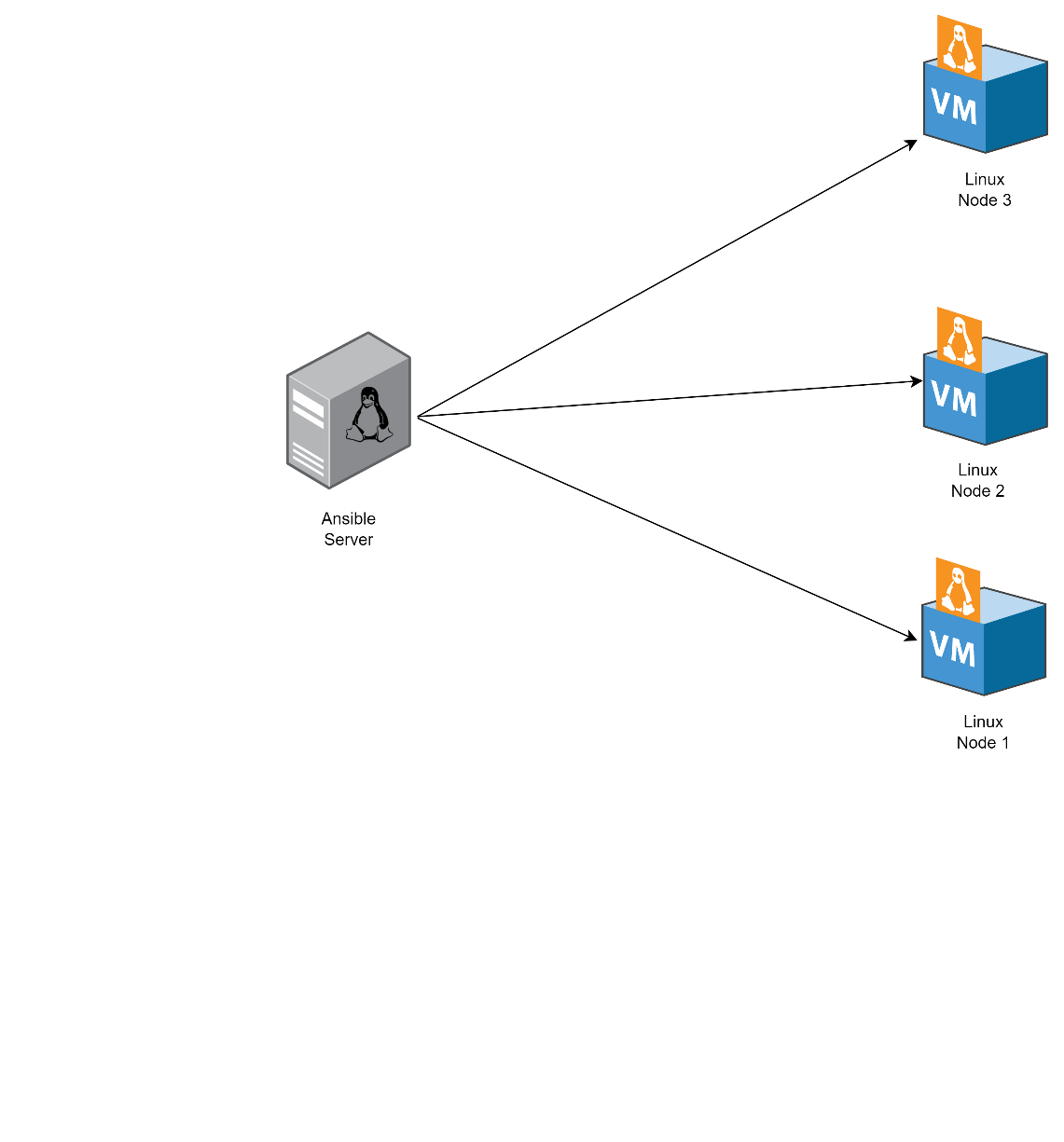
* **File module**:
  1. This module is used to manage files and directories on remote hosts.
  2. Examples: *creating files and directories, modifying file permissions, and copying files*.
* **Service module**:
  1. This module is used to manage system services on remote hosts.
  2. Examples: starting, stopping, enabling, disabling services, and restarting services.
* **Package module**:
  1. This module is used to manage packages on remote hosts.
  2. Examples: installing, removing packages, updating and listing installed packages.
* **Template module**:
  1. This module is used to generate files from templates.
  2. Examples: creating configuration files from templates, generating HTML files from templates, and creating scripts from templates.
* **User module**:
  1. This module is used to manage users and groups on remote hosts.
  2. Examples: creating and deleting users and groups, modifying user and group attributes, and managing user passwords.
* **Git module**:
  1. This module is used to manage Git repositories on remote hosts.
  2. Examples: cloning Git repositories, fetching and pulling changes, and checking out specific branches or tags.
* **Shell module**:
  1. This module is used to execute shell commands on remote hosts.
  2. Examples: running shell scripts, executing one-liner commands, and running system commands.
* **Docker module**:
  1. This module is used to manage Docker containers on remote hosts.
  2. Examples of its usage are creating and deleting containers, starting and stopping containers, and configuring container networks.
* **AWS module**:
  1. This module is used to manage AWS resources, such as EC2 instances, S3 buckets, and RDS instances.
  2. Examples: launching and terminating EC2 instances, managing S3 buckets and objects, and managing RDS instances.

Conditionals and Control Flow:

* Ansible provides a number of conditionals and control flow statements to enable complex playbooks and tasks to be created.
* These allow tasks to be executed conditionally or repeatedly, or for tasks to be skipped or failed based on the results of other tasks.

|  |
| --- |
| When conditional - This allows a task to be executed only when a certain condition is met.  Example:    In this example, the service task will only be executed when the changed\_files variable contains the string 'httpd.conf'. |
| Loop control flow - This allows a task to be executed repeatedly for each item in a list.  Example:    In this example, the yum task will be executed three times, once for each package in the list. |
| Until conditional: This allows a task to be repeated until a certain condition is met.  Example:    In this example, the wait\_for task will be repeated until the result variable contains the string 'succeeded'. |
| Failed\_when: This allows a task to be marked as failed based on the result of the task.  Example:    In this example, the shell task will be marked as failed if the string '/dev/sda1' is not found in the output of the command df -h /. |

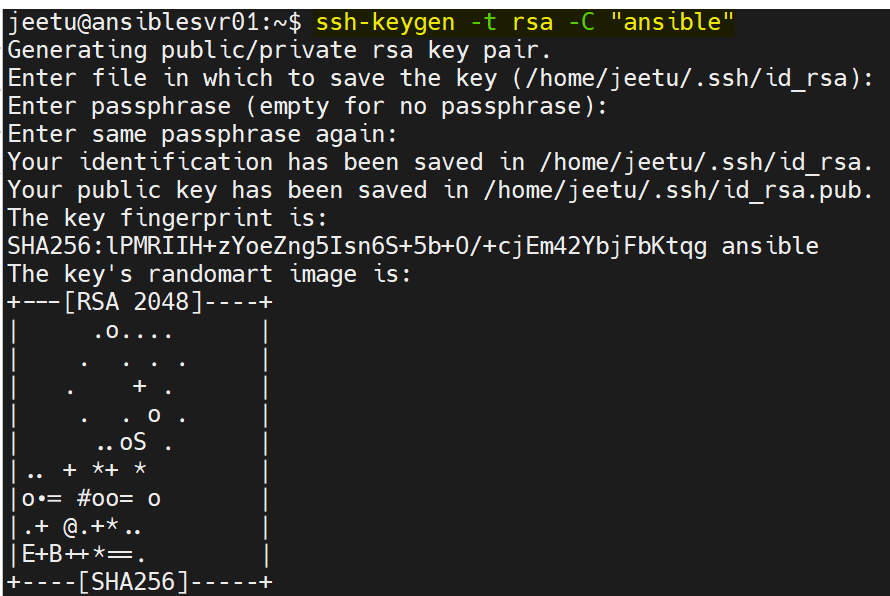
Lab setup for Ansible practical:



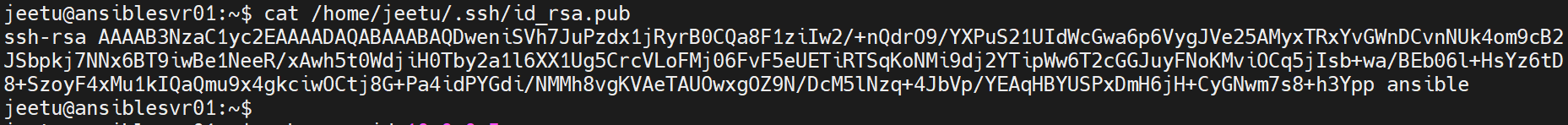
Steps for Ansible:

1. Create secure private-public keys on ansible server.
2. Copy these keys to all the node/client machines.
3. Install Ansible on ansible server & update all the packages.

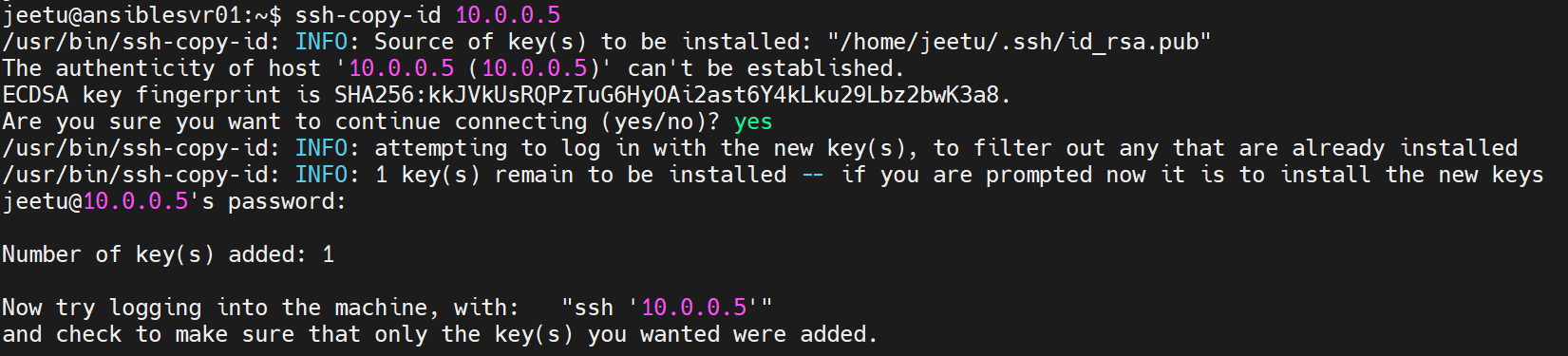
Creating SSH keys on ansible server:



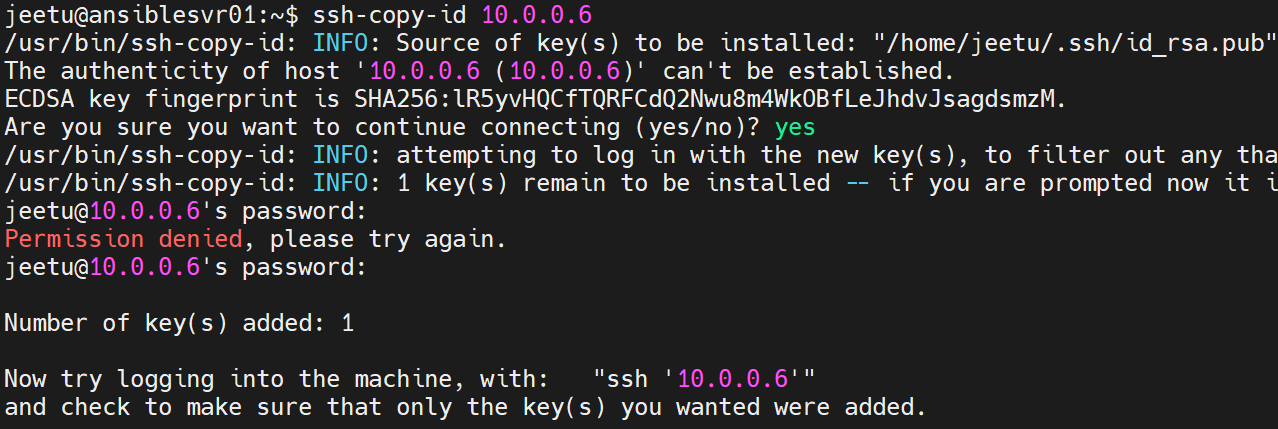
Verifying keys:



Adding keys to 1st ansible client machine:

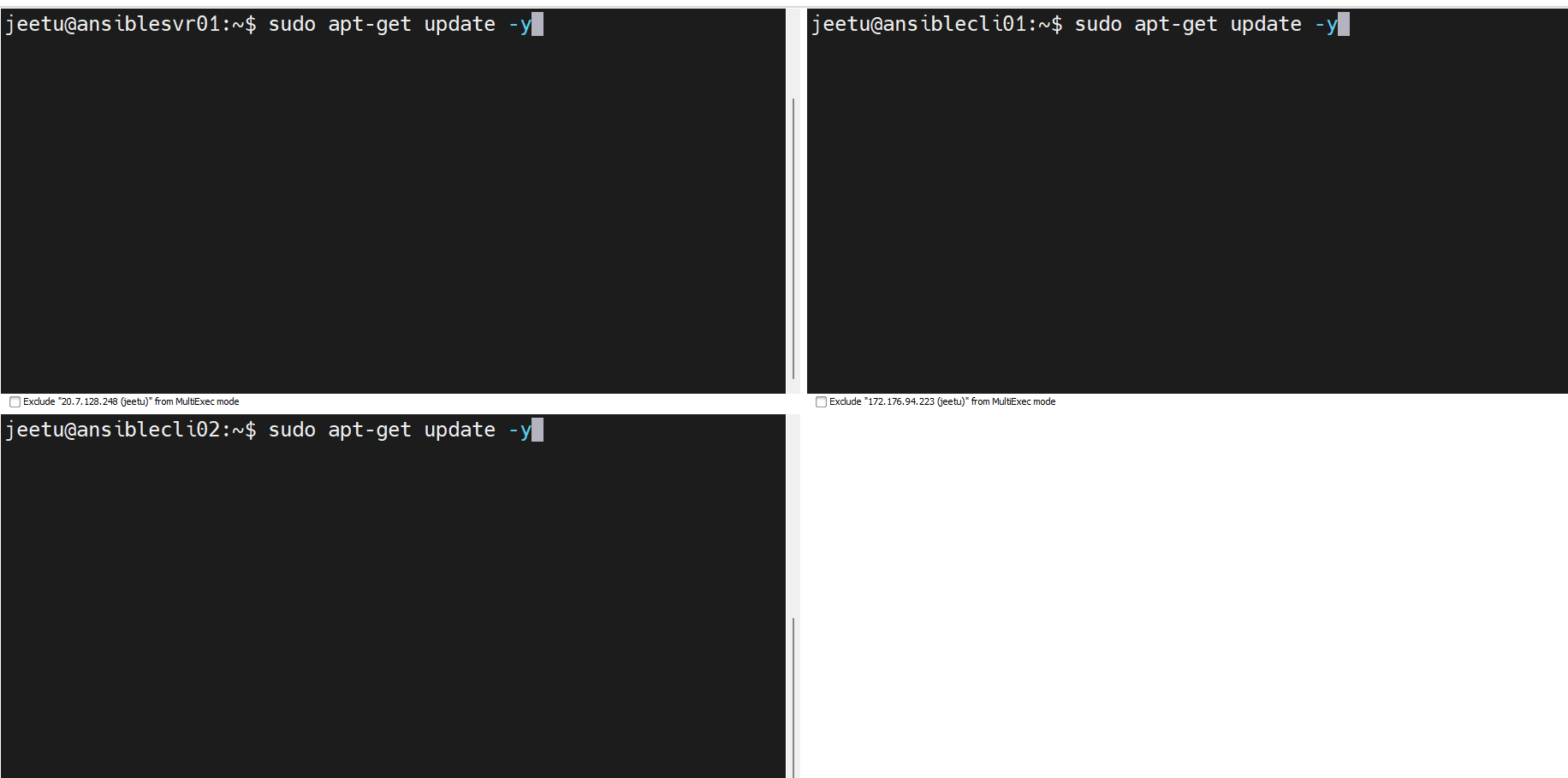


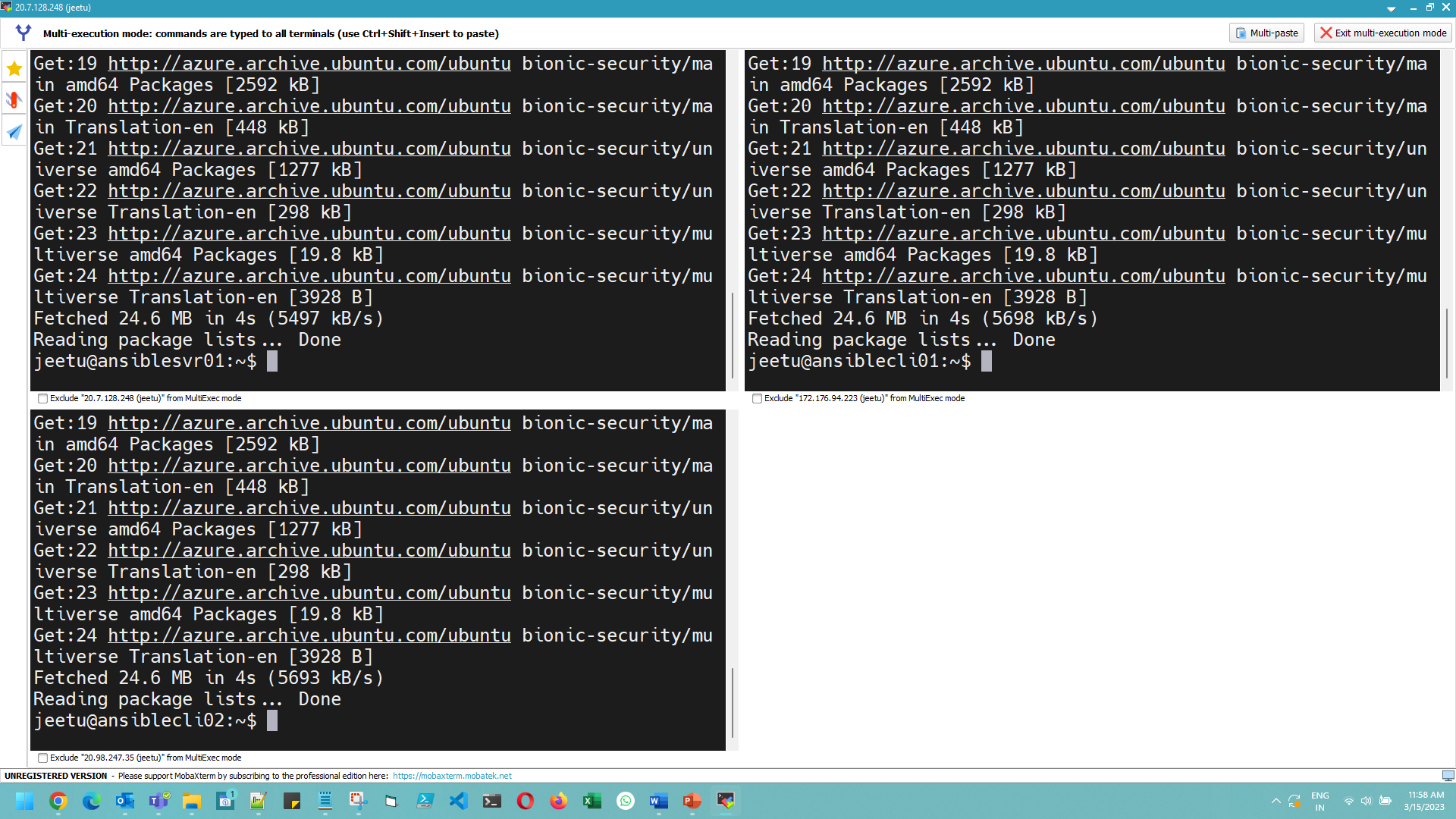
Adding SSH key to 2nd ansible client:



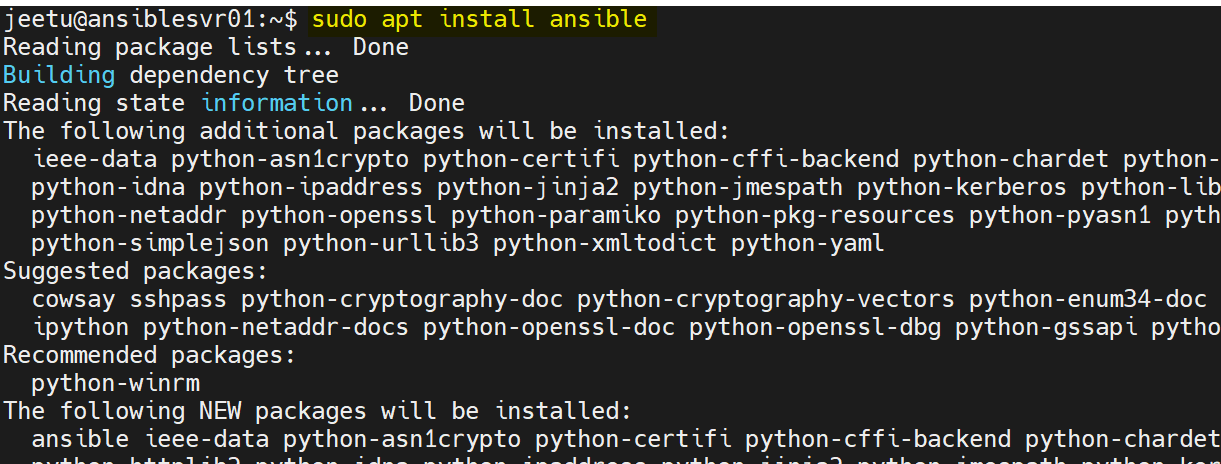
* After copying the keys, use SSH command to login to both clients one-by-one to verify if it’s working.

Updating all 3 Ubuntu VMs parallelly using [MobaxTerm](https://mobaxterm.mobatek.net/download.html) utility.





Installing ansible package on ansible-server01 using APT:



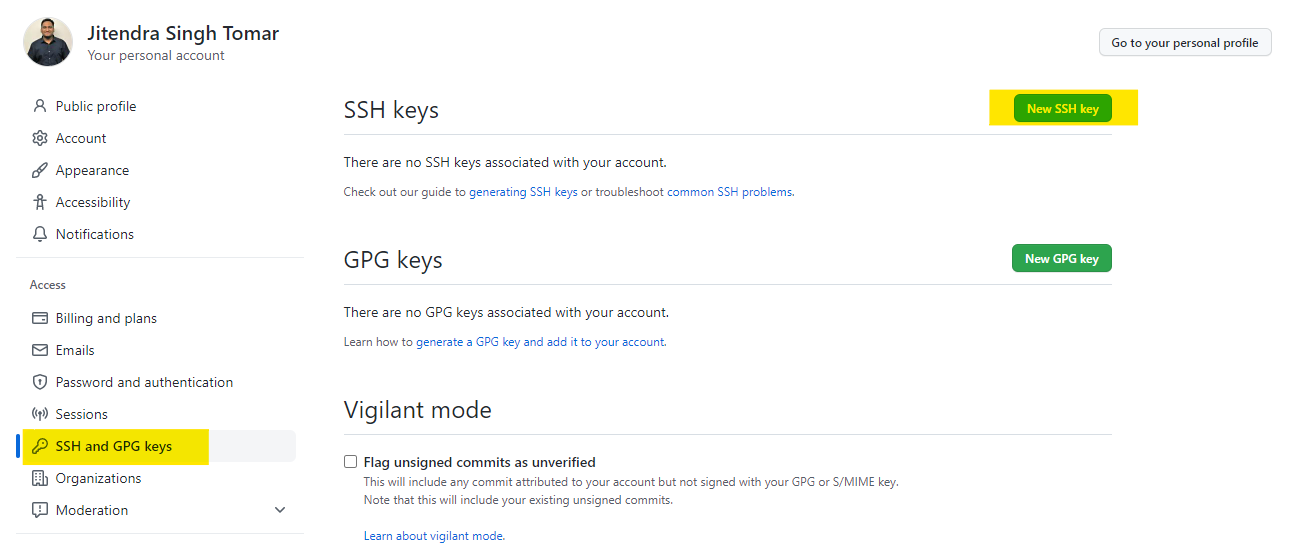
CMD: sudo apt install ansible

Verify:

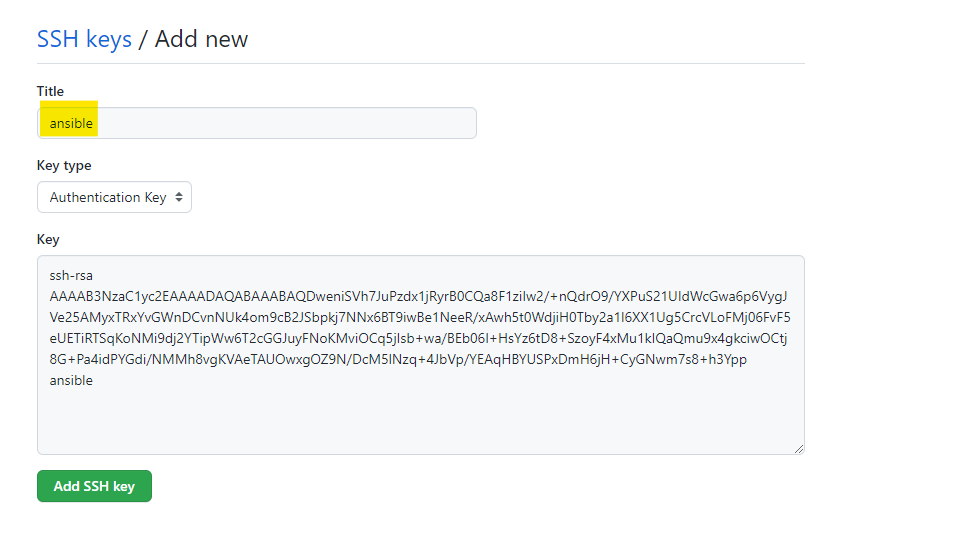


Now to save all the file for future, I am storing them in [GitHub](https://github.com/) repository. It’s not mandatory but very useful in DevOps. So, if you haven’t created a GitHub account yet, please go ahead & create it.

To connect ansible server with GitHub account, copy the public key (created earlier) & paste it in the GitHub account 🡪 Settings 🡪 SSH and GPG Keys.



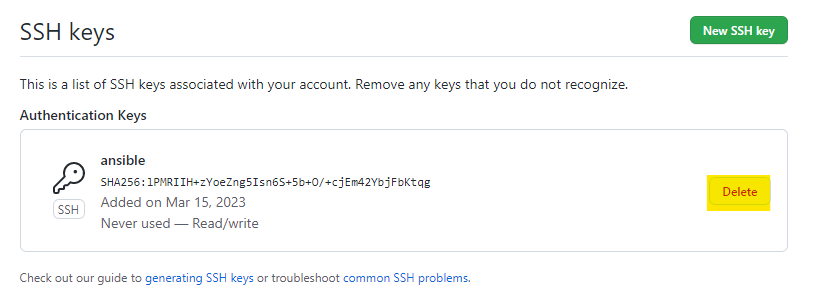
Fill any name in the title (I have used the same name as the key’s name):



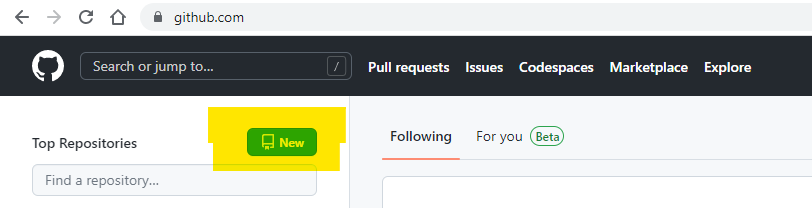
& click on “Add SSH key” & confirm the password for your GitHub account.

Don’t worry, I will be deleting the VMs used in this tutorial, which will make those keys un-usable/useless.

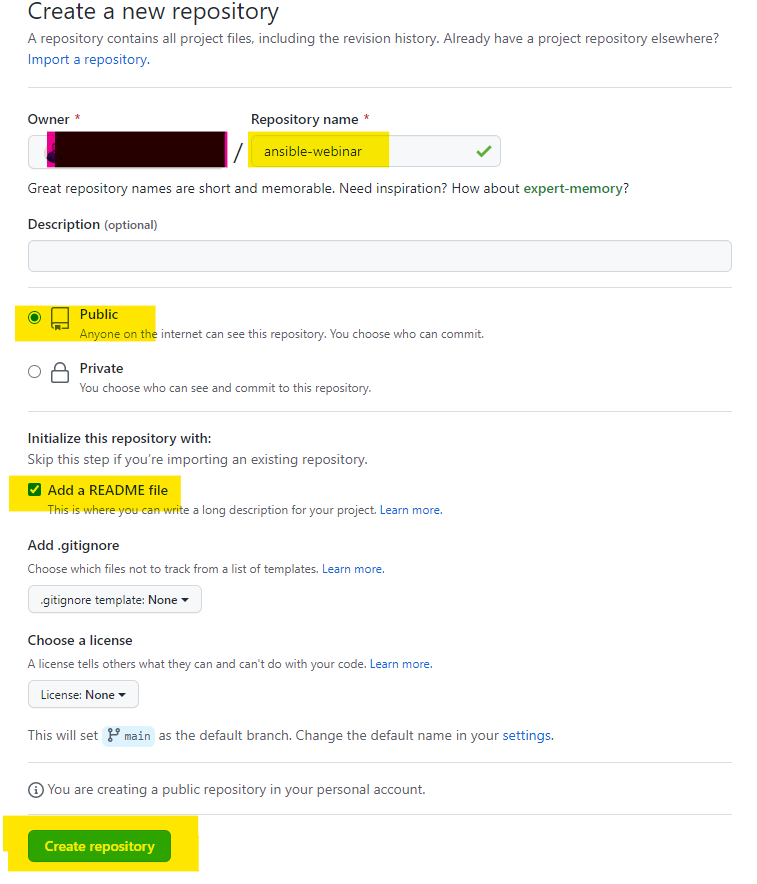
One the practical is over; you can delete the keys too.



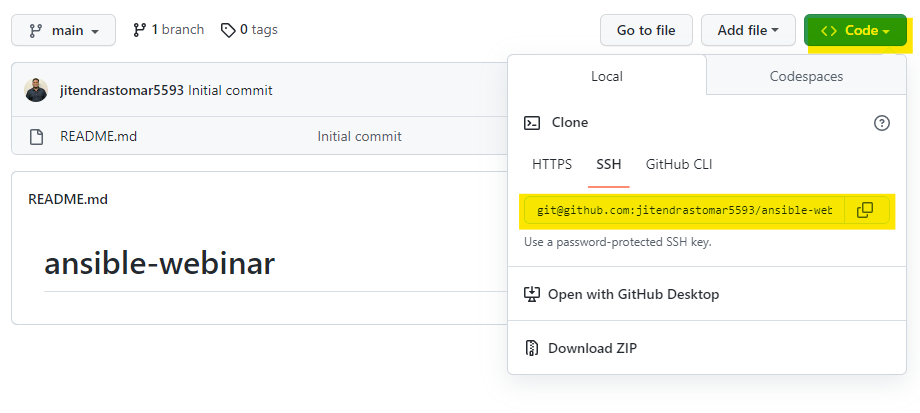
Now, create a new repository on GitHub:



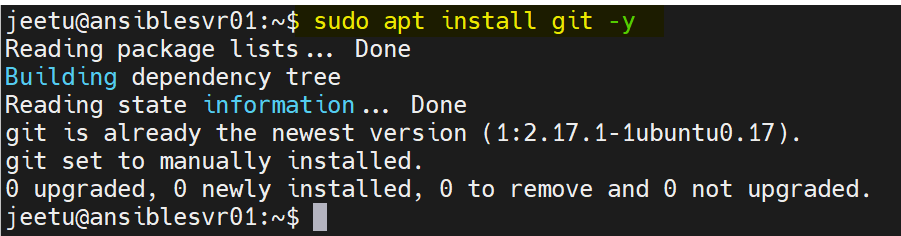
& fill basic details:



Inside repo, expand “code” & copy the SSH URL. This URL will be now used in the ansible server.

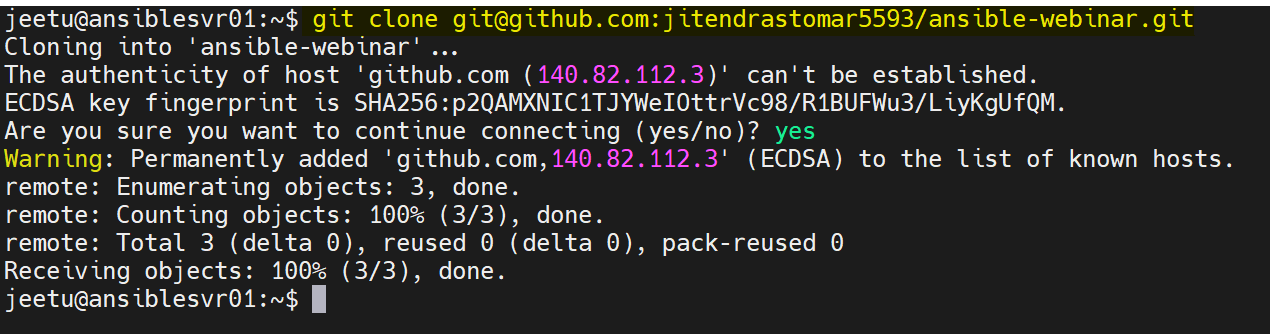


Now switch to ansible server & install “git” on it.

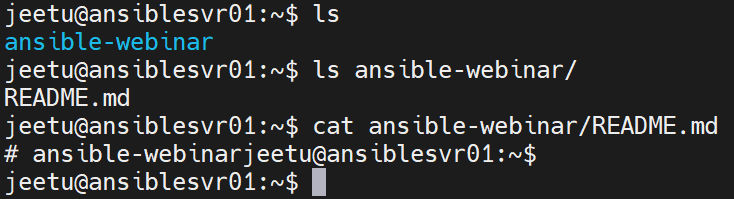


& execute below command

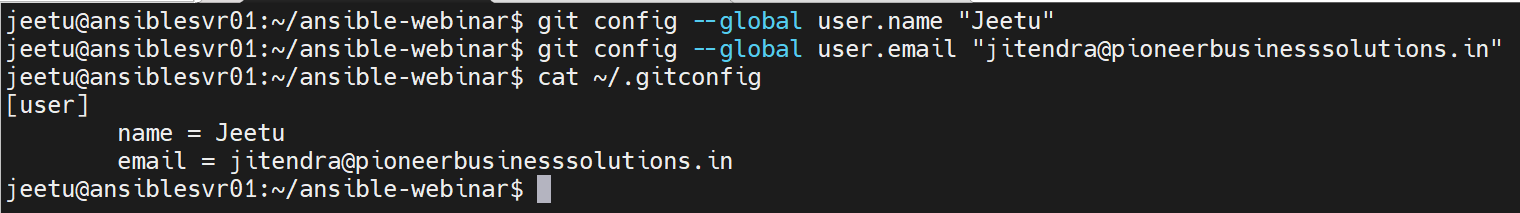
$git clone [git@github.com:jitendrastomar5593/ansible-webinar.git](mailto:git@github.com:jitendrastomar5593/ansible-webinar.git)



& verify:



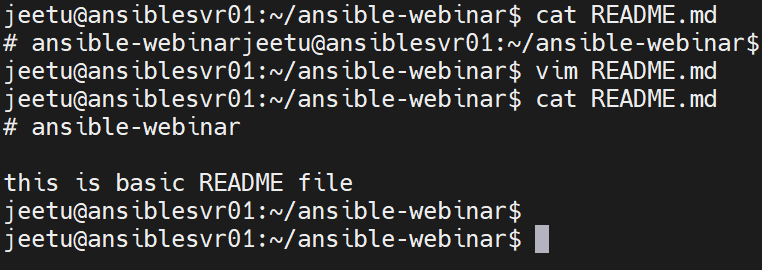
Now, creating configuration for git:



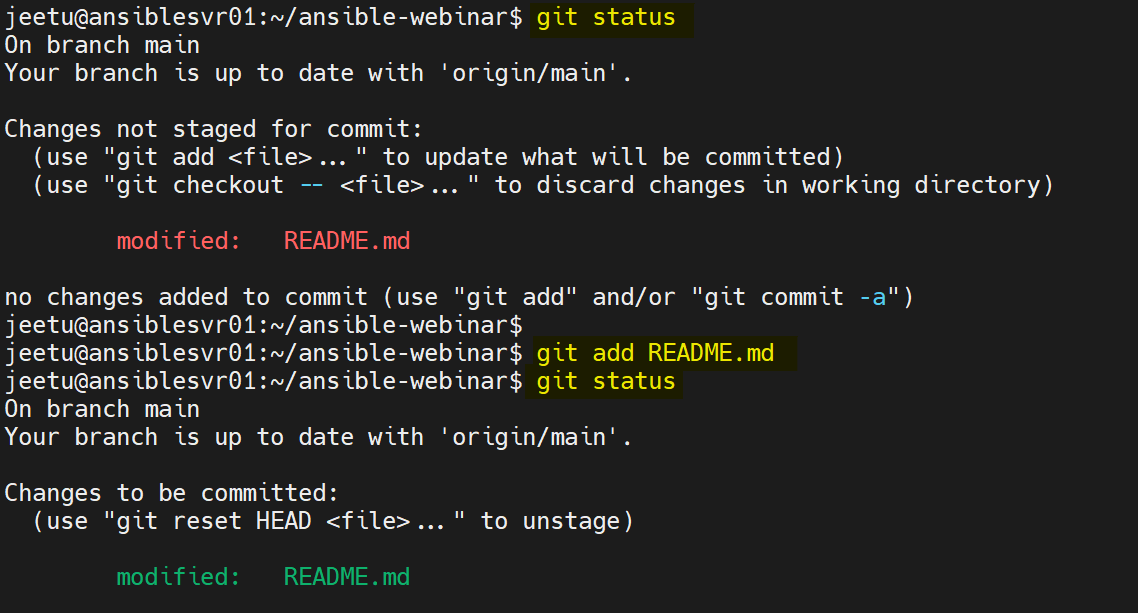
Commands:

* git config --global user.name "Jeetu"
* git config --global user.email "jitendra@pioneerbusinesssolutions.in"
* cat ~/.gitconfig

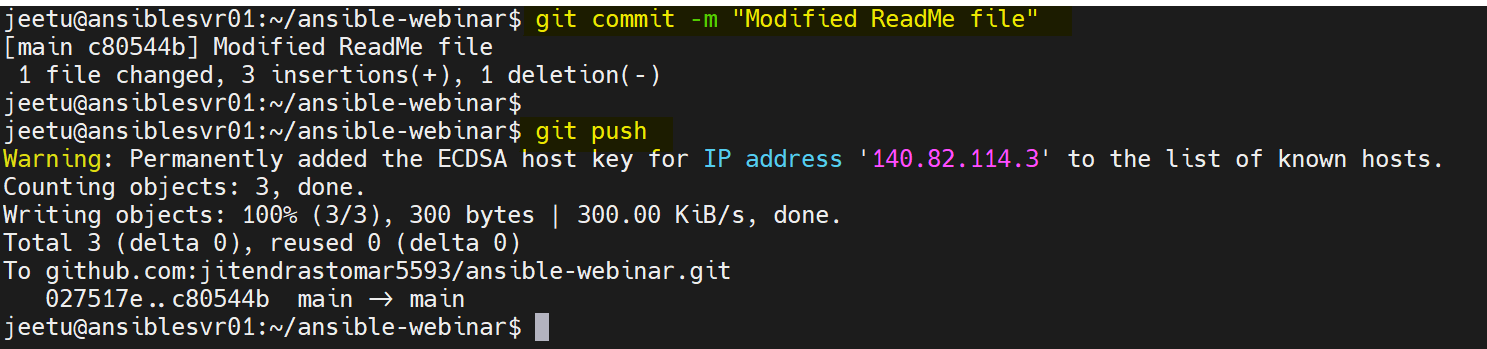
Modifying basic README file:



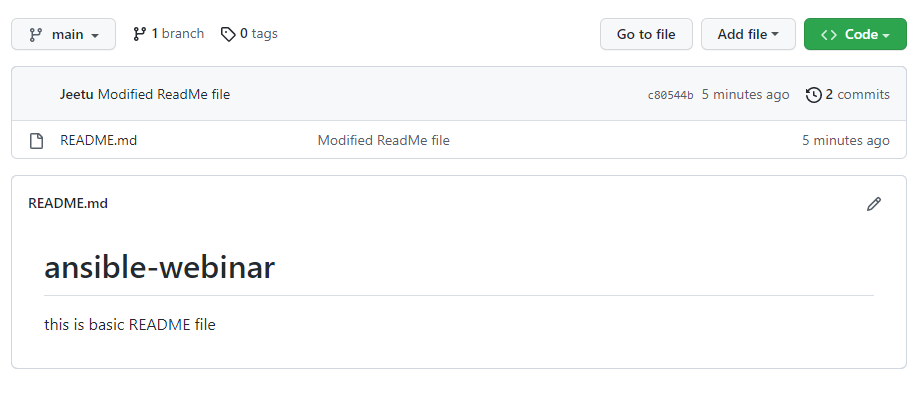
Preparing README file for the GitHub:



Committing the changes & then pushing the changes to the GitHub account:

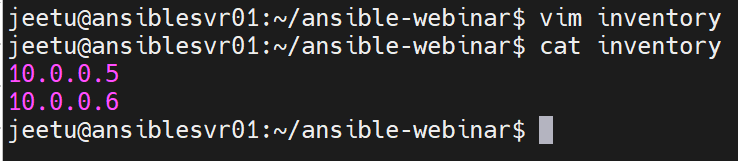


Verifying the changes on the browser:



Creating an inventory for all the hosts:

Command: #vim inventory



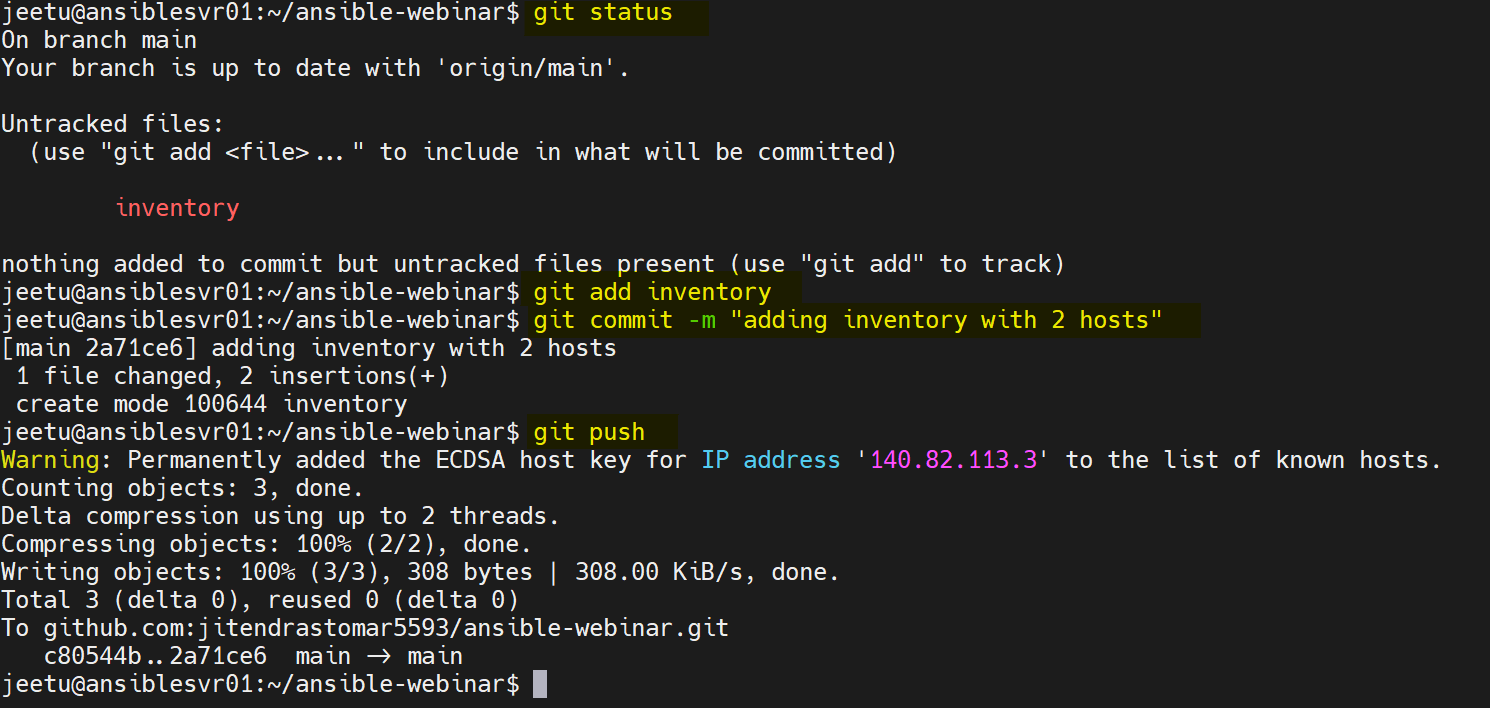
Adding inventory file in repository, committing the changes & then pushing the changes.

$ git status

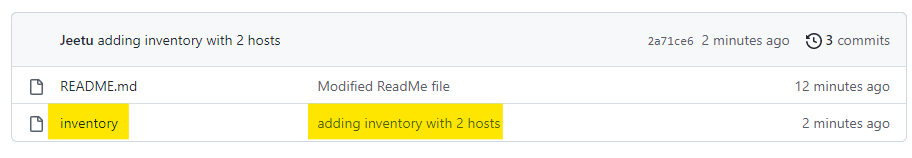
$ git add inventory

$ git commit -m "adding inventory with 2 hosts"

$ git push

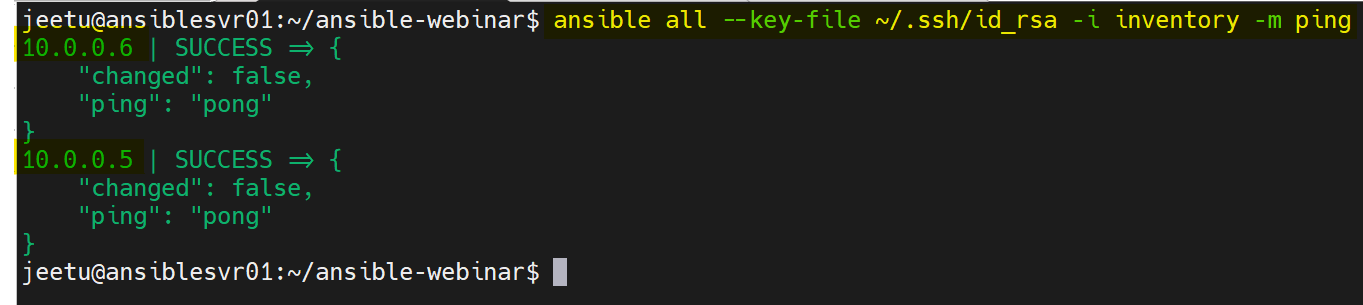


Verifying on the browser:



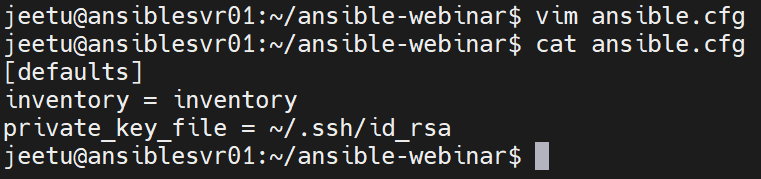
Ping the hosts to verify if its reachable using ansible module:

Command: ansible all --key-file ~/.ssh/id\_rsa -i inventory -m ping



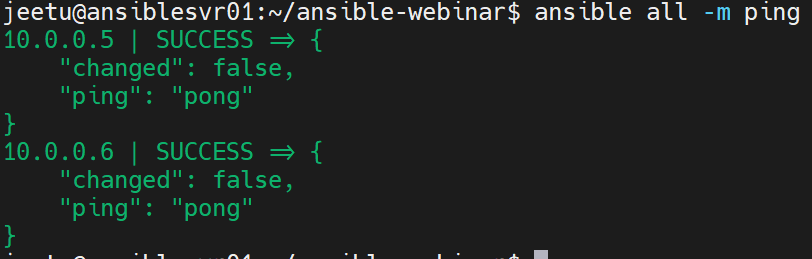
Now adding the default values to shorten the command length:

File: ansible.cfg



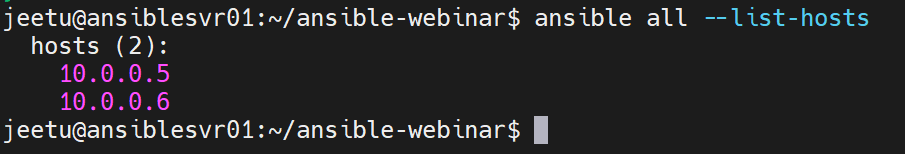
Ping all hosts after shortening:

Command: ansible all -m ping



Listing all registered hosts:

Command: ansible all --list-hosts



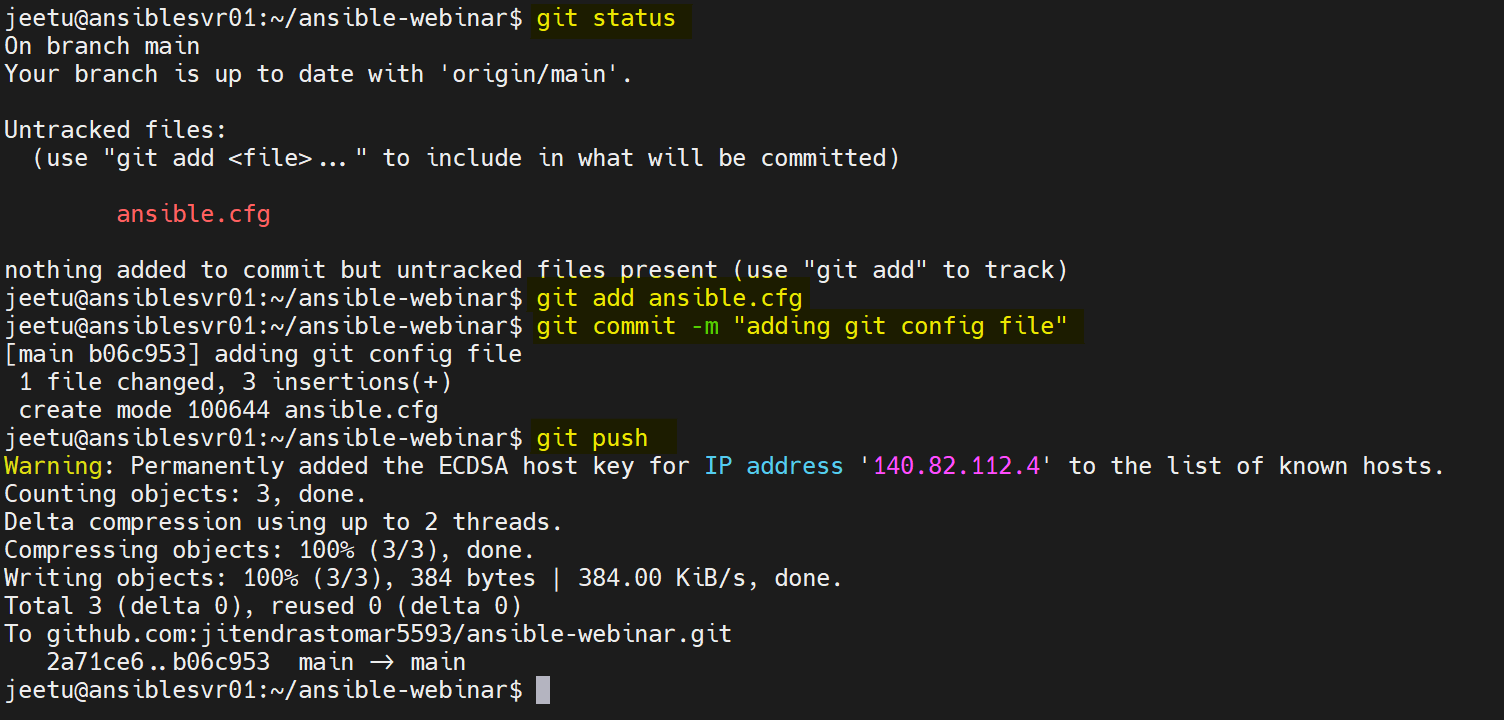
Adding newly created files or modified files to GitHub:

$ git status

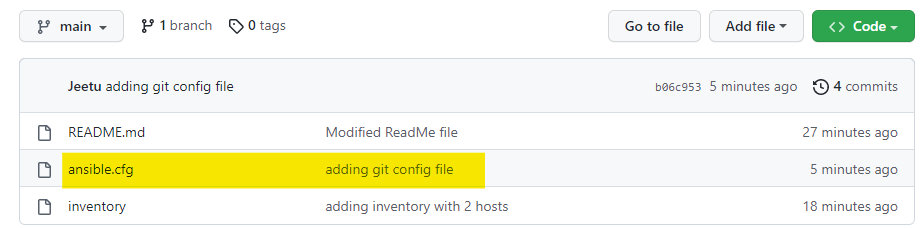
$ git add ansible.cfg

$ git commit -m "adding git config file"

$ git push

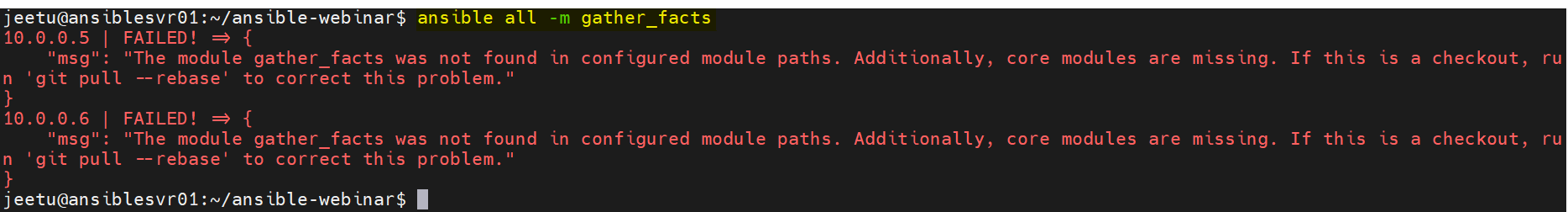


Verifying on browser:



Gathering facts about hosts (running SUDO commands, but fails)

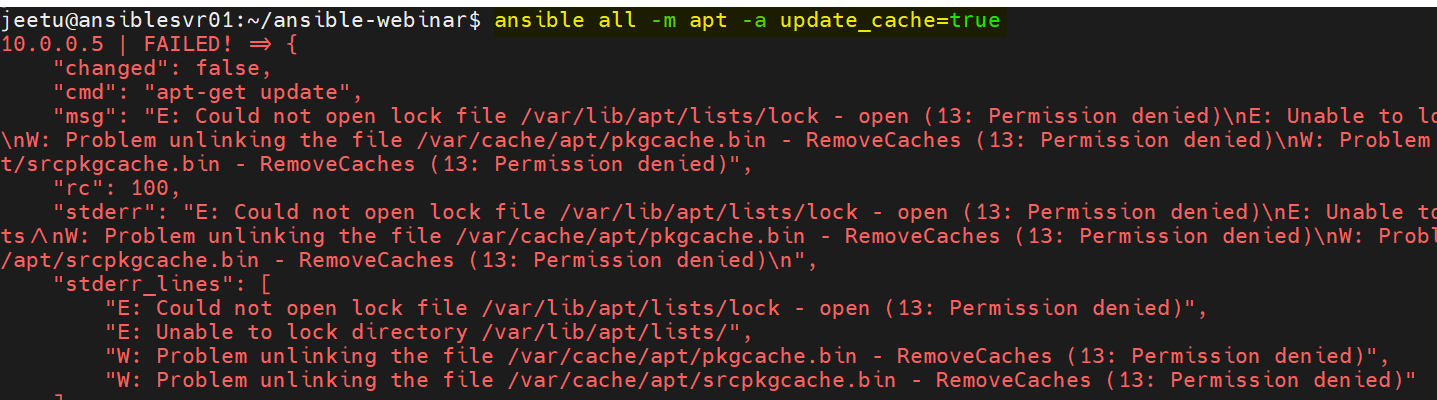
Command: $ ansible all -m gather\_facts



It fails because there is NO ROOT/SUDO access given to ansible.

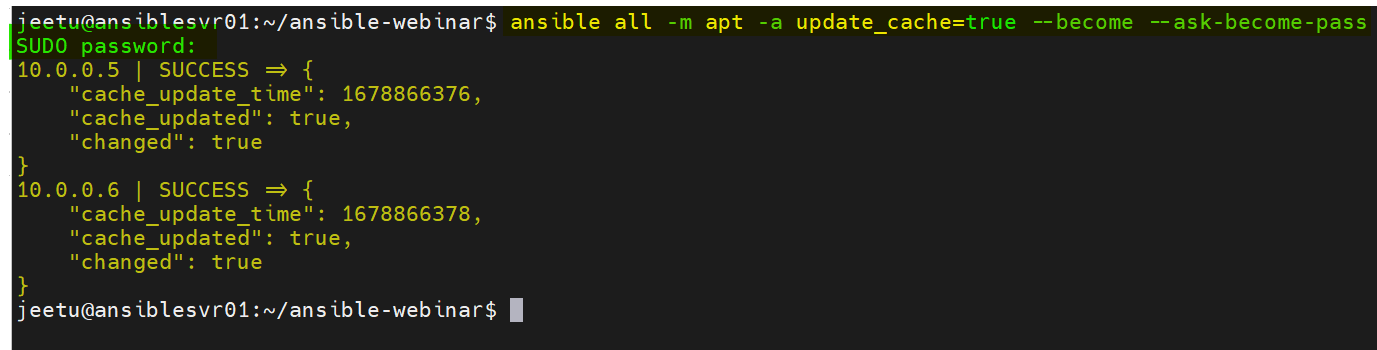
Running another package updating command:

Command: $ ansible all -m apt -a update\_cache=true



Now, allowing SUDO accessing while passing the ansible commands:

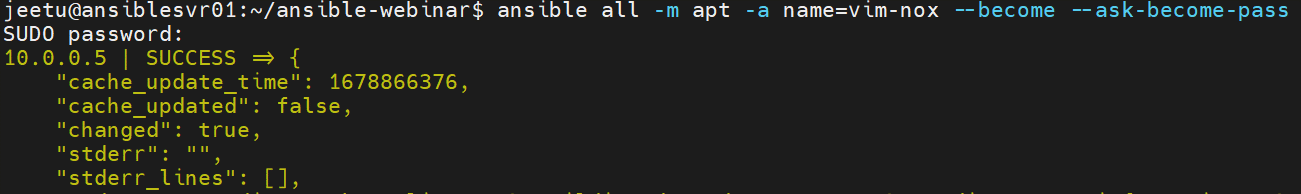
Command: $ ansible all -m apt -a update\_cache=true --become --ask-become-pass



Here, “SUDO password” is actually your SUDO user password and SUCCESS is the message for successfully updating the cache on both hosts.

Now, installing a new package (vim-nox) on both machines:

Command: $ ansible all -m apt -a name=vim-nox --become --ask-become-pass



Command to upgrade Ubuntu machines: $ ansible all -m apt -a "upgrade=dist" --become --ask-become-pass

Writing ansible playbook:

SCRIPT: - this YAML script installs Apache & PHP on the registered hosts (for UBUNTU only)

|  |
| --- |
| **Script:**  - hosts: all  become: true  tasks:  - name: install apache2 package  apt:  name: apache2  state: latest  - name: add php support for apache  apt:  name: libapache2-mod-php  state: latest    Running the script: $ ansible-playbook --ask-become-pass install\_apache.yml |

Script 2: for CentOS & Ubuntu both:

|  |
| --- |
| Script:  ---  - hosts: all  become: true  tasks:  - name: install apache2 package & libapache2-mod-php  apt:  name:  - apache2  - libapache2-mod-php  state: latest  update\_cache: yes  when: ansible\_distribution == "Ubuntu"  - name: install HTTPD & PHP packages on CentOS  dnf:  name:  - httpd  - php  state: latest  update\_cache: yes  when: ansible\_distribution == "CentOS" |

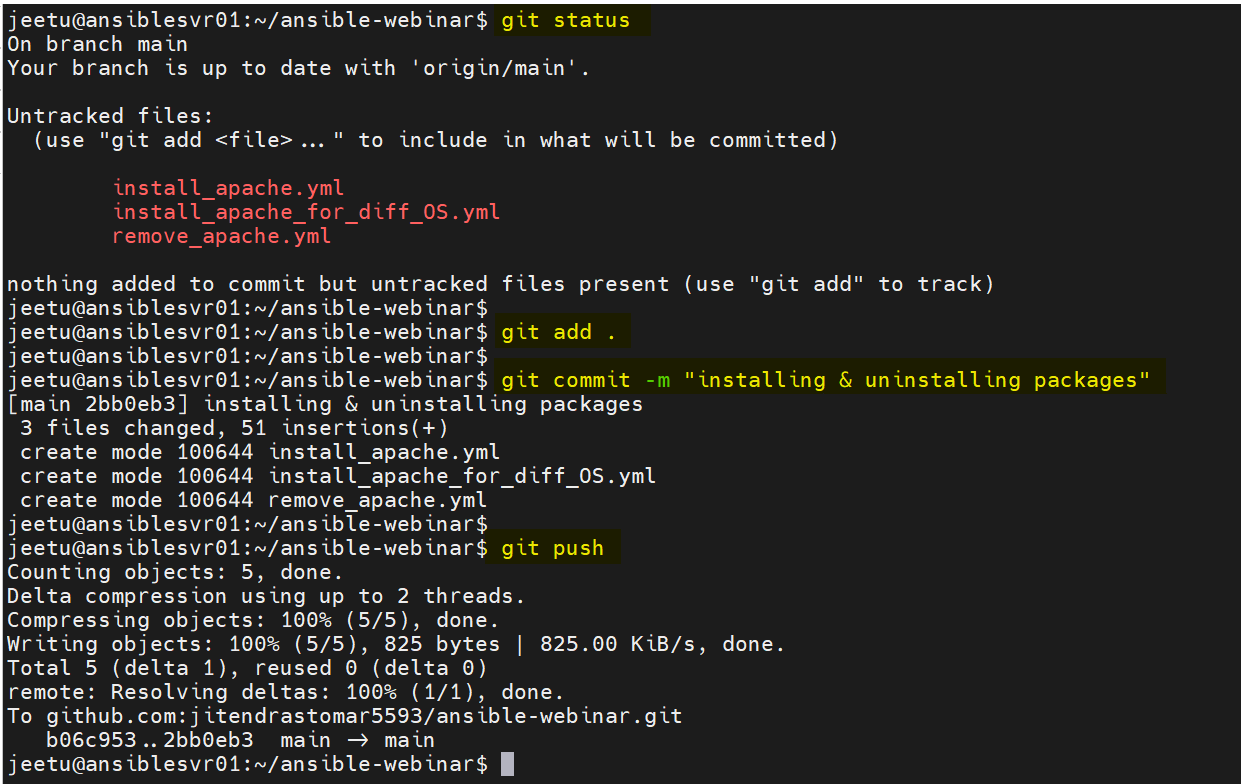
Script 3: removing Apache & PHP packages from the registered hosts:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Script:   |  | | --- | | --- | |  |  | |  | - hosts: all | |  | become: true | |  | tasks: | |  |  | |  | - name: remove apache2 package | |  | apt: | |  | name: apache2 | |  | state: absent | |  |  | |  | - name: remove php support for apache | |  | apt: | |  | name: libapache2-mod-php | |  | state: absent |     Command: $ ansible-playbook --ask-become-pass remove\_apache.yml |

Adding, committing & pushing changes to GIT.

Commands:

* $ git status
* $ git add .
* $ git commit -m "installing & uninstalling packages"
* $ git push

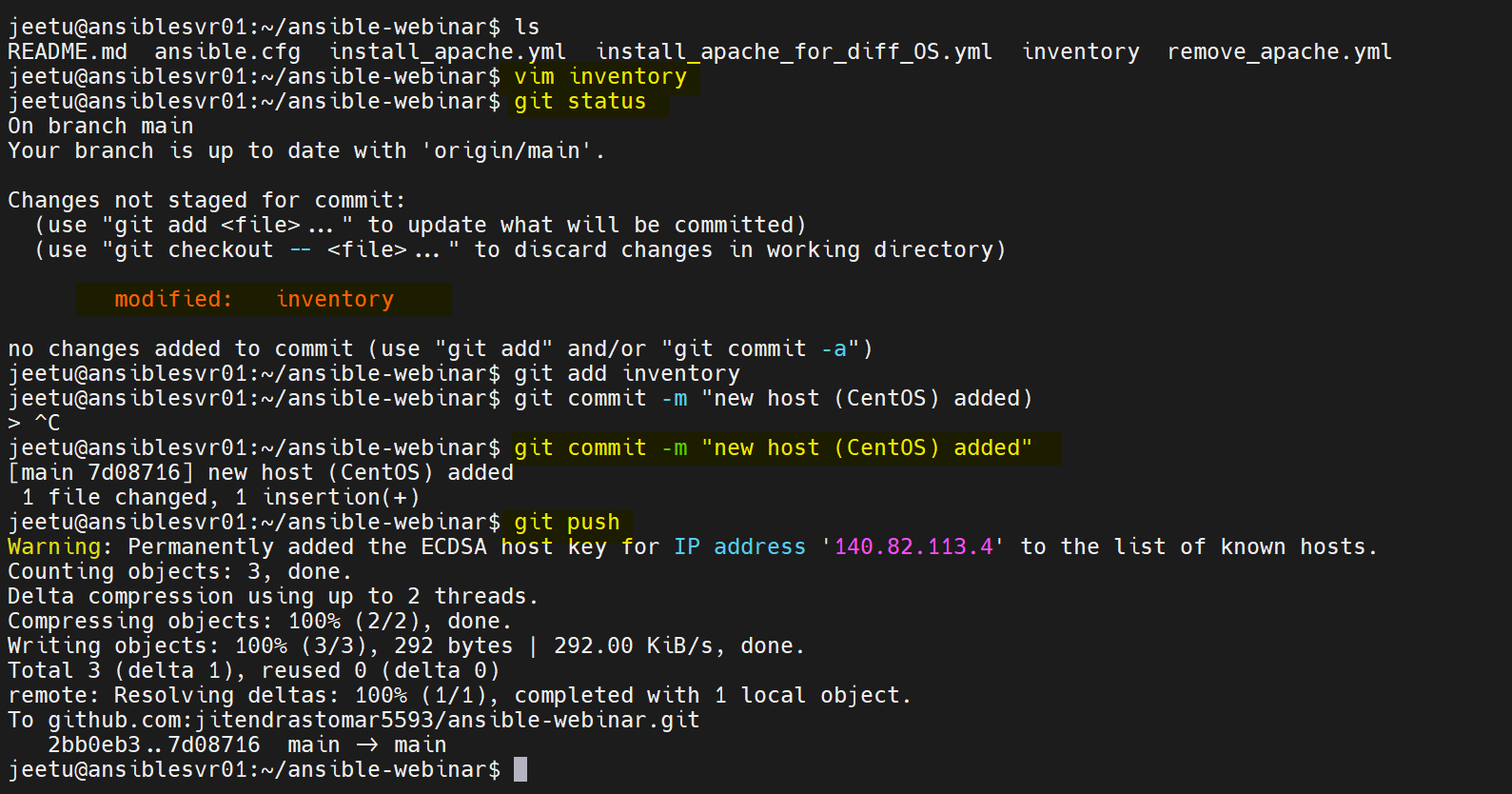


Using WHEN in playbooks:

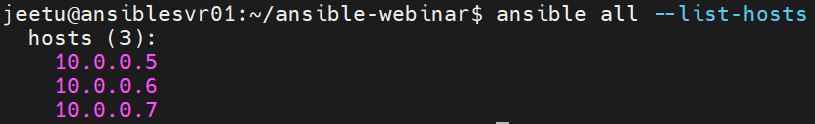
Before this adding a CentOS machine to the fleet:

Commands:

* git status
* git add inventory
* git commit -m "new host (CentOS) added"
* git push



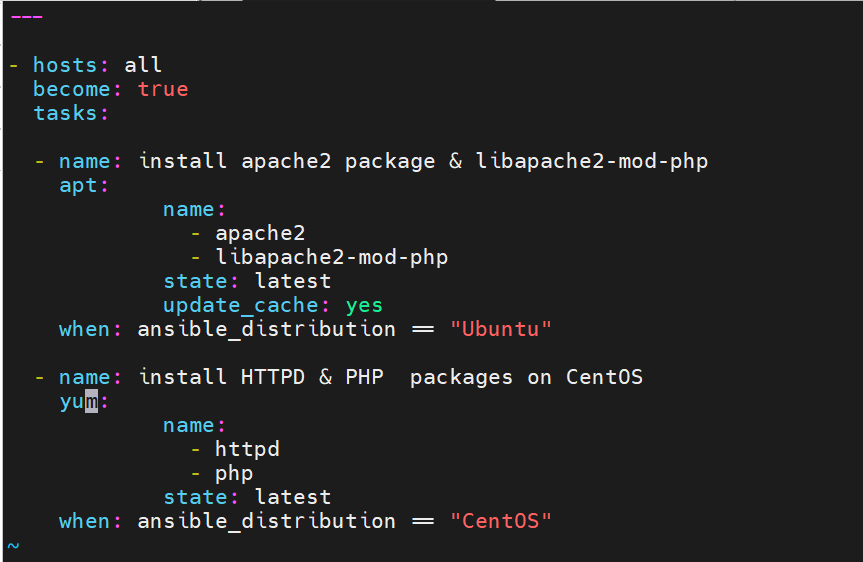
List of current hosts: $ansible all --list-hosts



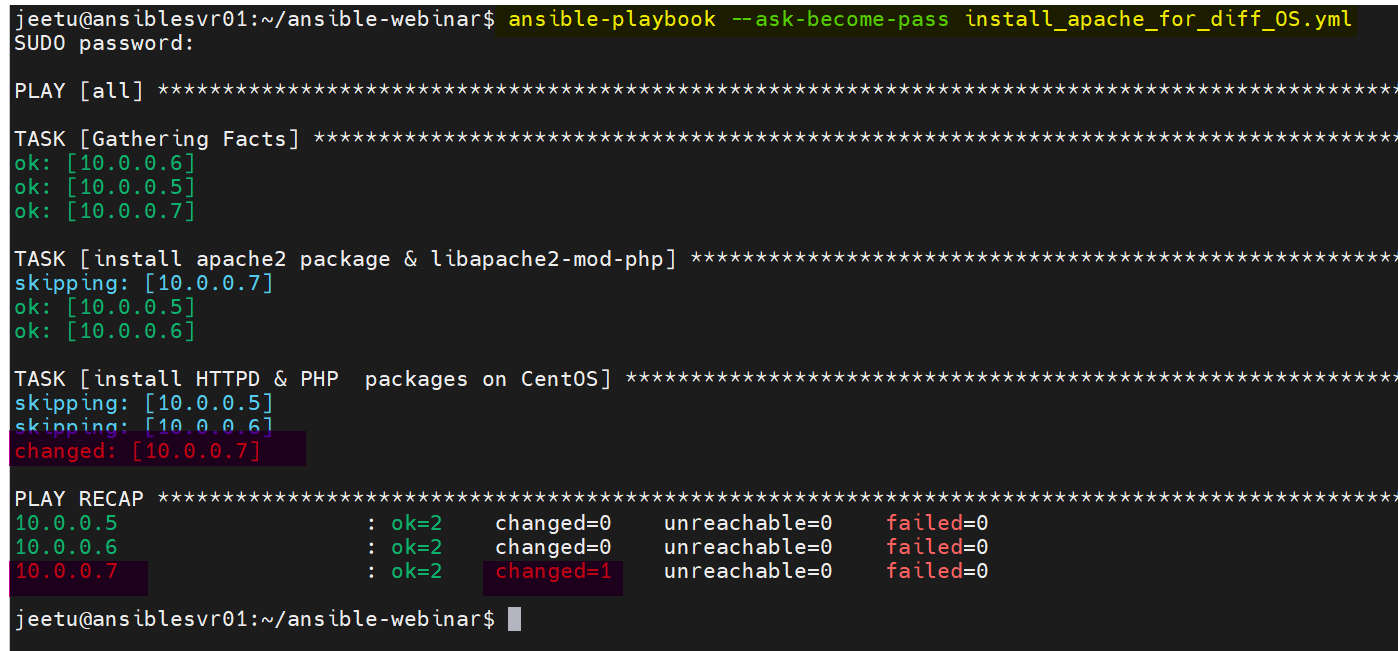
Running the package install for Ubuntu & CentOS 7:

Code with “WHEN” keyword:

|  |
| --- |
| - hosts: all  become: true  tasks:  - name: install apache2 package & libapache2-mod-php  apt:  name:  - apache2  - libapache2-mod-php  state: latest  update\_cache: yes  when: ansible\_distribution == "Ubuntu"  - name: install HTTPD & PHP packages on CentOS  yum:  name:  - httpd  - php  state: latest  when: ansible\_distribution == "CentOS" |

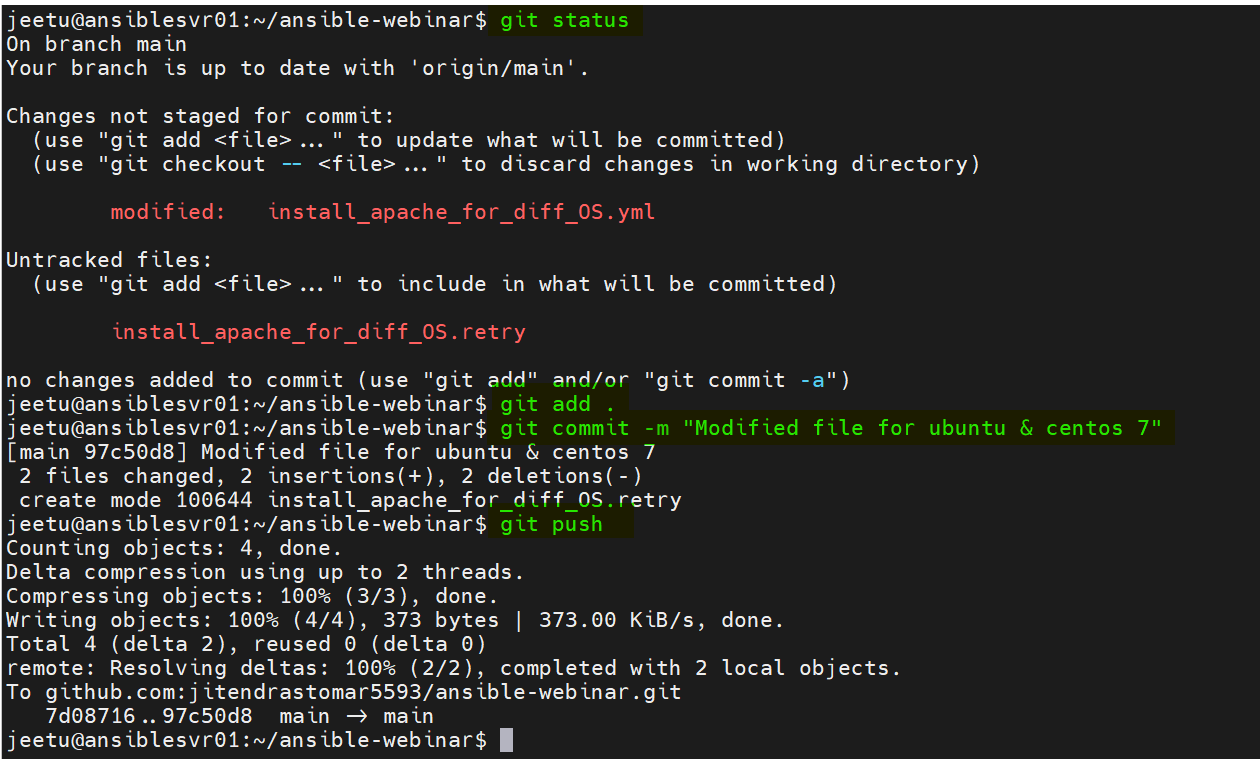


Command: $ ansible-playbook --ask-become-pass install\_apache\_for\_diff\_OS.yml



Adding, committing & pushing the changes:

* $ git status
* $ git add .
* $ git commit -m "Modified file for ubuntu & centos 7"
* $ git push

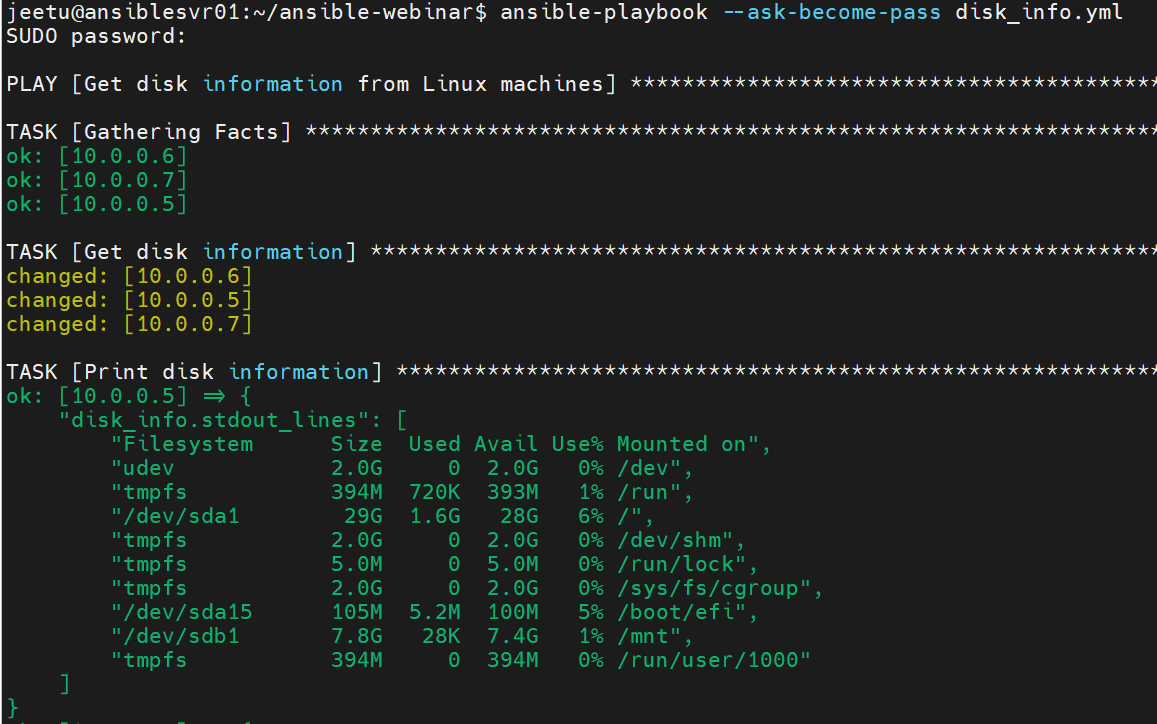


Script 4: Getting disk information from all the machines:

Code:

|  |
| --- |
| jeetu@ansiblesvr01:~/ansible-webinar$ vim disk\_info.yml  jeetu@ansiblesvr01:~/ansible-webinar$ cat disk\_info.yml  ---  - name: Get disk information from Linux machines  hosts: all  gather\_facts: yes  tasks:  - name: Get disk information  command: "df -h"  register: disk\_info  - name: Print disk information  debug:  var: disk\_info.stdout\_lines |

Output:



Getting disk information for complete “/”:

Code:

|  |
| --- |
| Command: $ ansible-playbook --ask-become-pass disk\_info.yml  Output: |

Script 5: To update/patch complete operating system (Ubuntu/CentOS)

|  |
| --- |
| Code:  jeetu@ansiblesvr01:~/ansible-webinar$ cat full\_update.yml  ---  - name: Update CentOS and Ubuntu  hosts: all  become: true  tasks:  - name: Update CentOS  yum:  name: '\*'  state: latest  when: ansible\_os\_family == 'RedHat'  - name: Update Ubuntu  apt:  upgrade: dist  when: ansible\_os\_family == 'Debian'    Running script: $ ansible-playbook --ask-become-pass full\_update.yml  Output:    Verify: |

Targeting Specific Nodes: (grouping of servers)

Making changes in inventory file

|  |
| --- |
| [Ubuntu\_servers]  10.0.0.5  10.0.0.6  [CentOS\_servers]  10.0.0.7 |

To install packages based on groups: change “ALL” to the group name, that needs to be changed.

|  |
| --- |
| jeetu@ansiblesvr01:~/ansible-webinar$ cat install\_apache.yml  - hosts: all  become: true  tasks:  - name: install apache2 package  apt:  name: apache2  state: latest  - name: add php support for apache  apt:  name: libapache2-mod-php  state: latest |

################# The END ################

Find all scripts & files: git@github.com:jitendrastomar5593/ansible-webinar.git