5.Write IaC using terraform to create EC2 machine on aws or azure or google cloud. (Compulsory to use Input and output variable files)

- Name: Om Deore
- PRN: 22110318
- Div: TY(A)
- RollNo: 321022
- github: All Assignments

What is YAML

YAML stands for "yet another markup language"
YAML is a human-readable data serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted. YAML targets many of the same communications applications as Extensible Markup Language but has a minimal syntax that intentionally differs from Standard Generalized Markup Language. It uses Python-style indentation to indicate nesting and does not require quotes around most string values. Custom data types are allowed, but YAML natively encodes scalars, lists, and associative arrays. These data types are based on the Perl programming language, though all commonly used high-level programming languages share very similar concepts.

eg. YAML is used for configuration of "Alacritty terminal"
the config looks something like this where there are variables, then nested variables who hold some values like `padding, opacity, font`

```
import:
 - ~/.config/alacritty/new.yml

env:
  TERM: alacritty

window:
  padding:
    x: 5
    y: 5

  opacity: 0.7
```

```
  startup_mode: Maximized
  dynamic_title: true

font:
  normal:
    family: HackNerdFont
    style: Regular

  bold:
    family: HackNerdFont
    style: Bold

  italic:
    family: HackNerdFont
    style: Italic

  bold_italic:
    family: HackNerdFont
    style: Bold Italic

  # Point size
  size: 10

  glyph_offset:
    x: 0
    y: 0

draw_bold_text_with_bright_colors: true
```

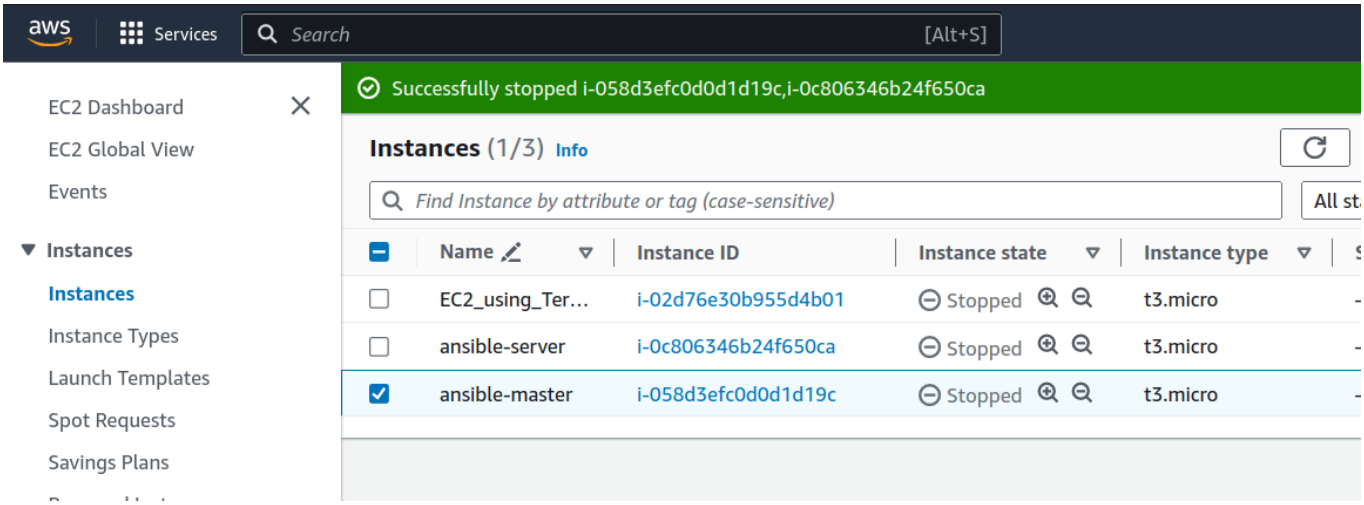
What is Ansible

Ansible is an open source, command-line IT automation software application written in Python. It can configure systems, deploy software, and orchestrate advanced workflows to support application deployment, system updates, and more.
Ansible's main strengths are simplicity and ease of use. It also has a strong focus on security and reliability, featuring minimal moving parts. It uses OpenSSH for transport (with other transports and pull modes as alternatives), and uses a human-readable language that is designed for getting started quickly without a lot of training.
Ansible is a suite of software tools that enables infrastructure as code. It is open-source and the suite includes software provisioning, configuration management, and application deployment functionality. Originally written by Michael DeHaan in 2012, and acquired by Red Hat in 2015, Ansible is designed to configure both Unix-like systems and Microsoft Windows. Ansible is agentless, relying on temporary remote connections via SSH or Windows Remote Management

which allows PowerShell execution. The Ansible control node runs on most Unix-like systems that are able to run Python, including Windows with Windows Subsystem for Linux installed. System configuration is defined in part by using its own declarative language

Commands to install Ansible

To use Ansible you need at least 2 remote servers or 1 remote servers and one local machine On which we will be installing Ansible, we will call that machine "Ansible-master". It can be an remote server or your local machine. And one with which we will be interacting is called as ansible server.



On Ansible-master run following commands

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Inventory

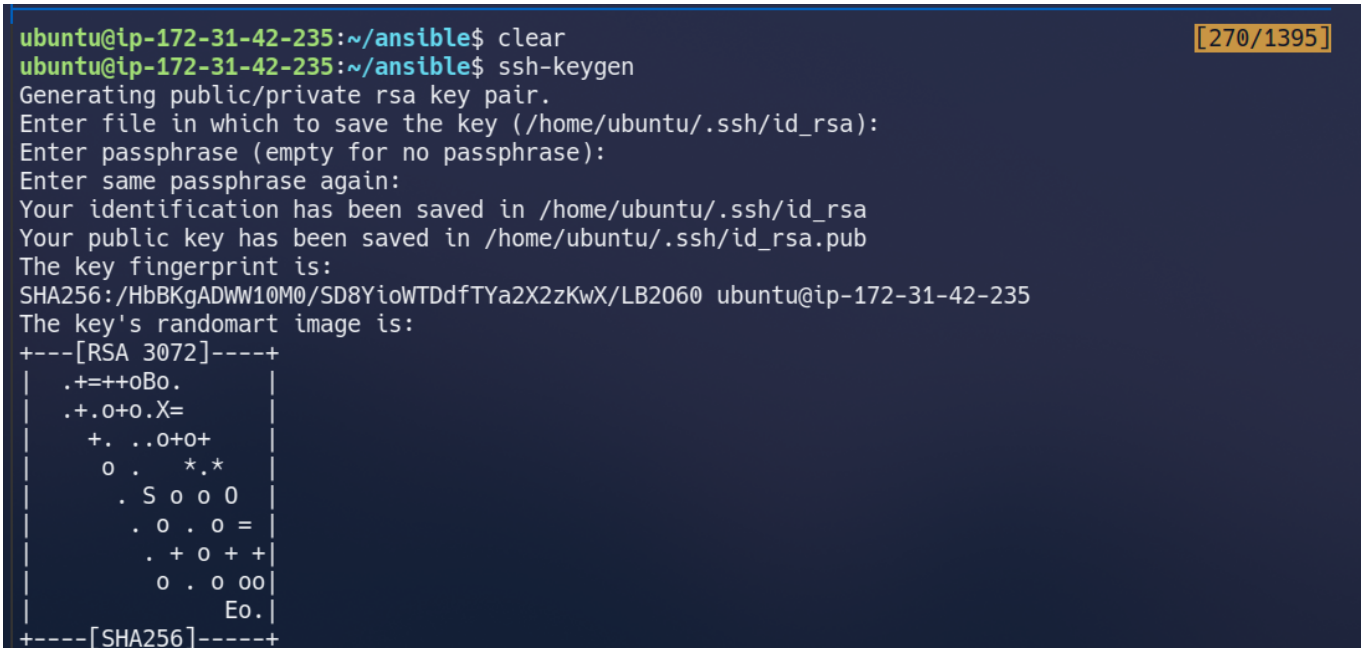
Before we start, we need to tell Ansible installed on our `master` machine the address of `server`
We can do this by creating `inventory.cfg` file

We have to put public ip address of our `ansible server` machine in this file

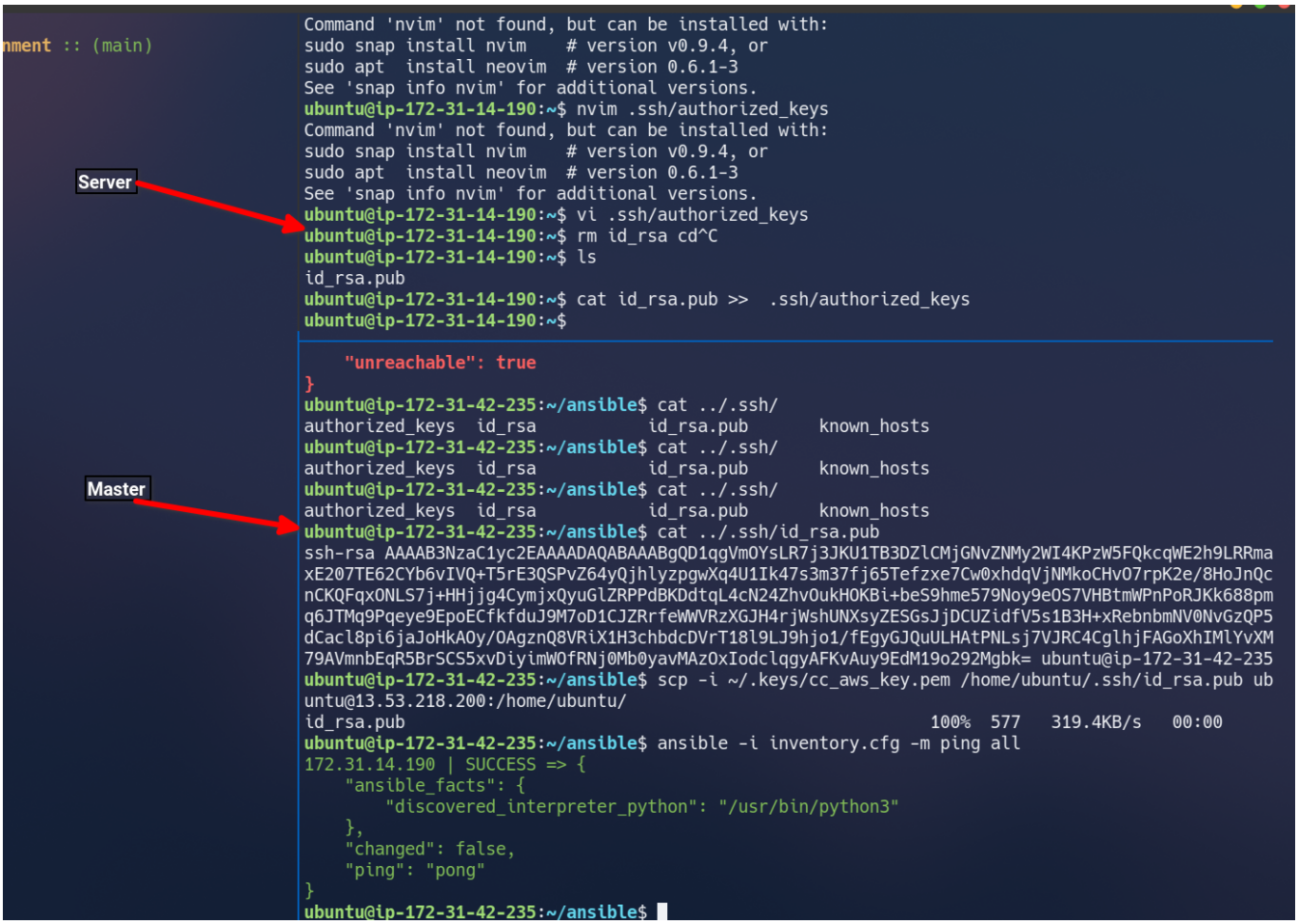
```
$ touch inventory.cfg
$ nvim inventory.cfg
$ cat inventory.cfg
13.53.218.200
```

Authorized Keys

To access our server from master via Ansible we need to add master to server's authorised keys
For this run `ssh-keygen` on master. It will create a public and private key. We have to paste this public key fingerprint in server's authorised keys file



Then to get fingerprint in server machine, we can use clipboard or we can scp this file `id_rsa.pub` file on server and cat it into `authorised_keys` file.
To check we have done everything correct we ping all machines in `inventory.cfg`



ad-hoc command

Ad-hoc commands are one of the simplest ways of using Ansible. These are used when you want to issue some commands on a server or bunch of servers. The ad-hoc commands are not stored for future use, but it represents a fast way to interact with the desired servers.

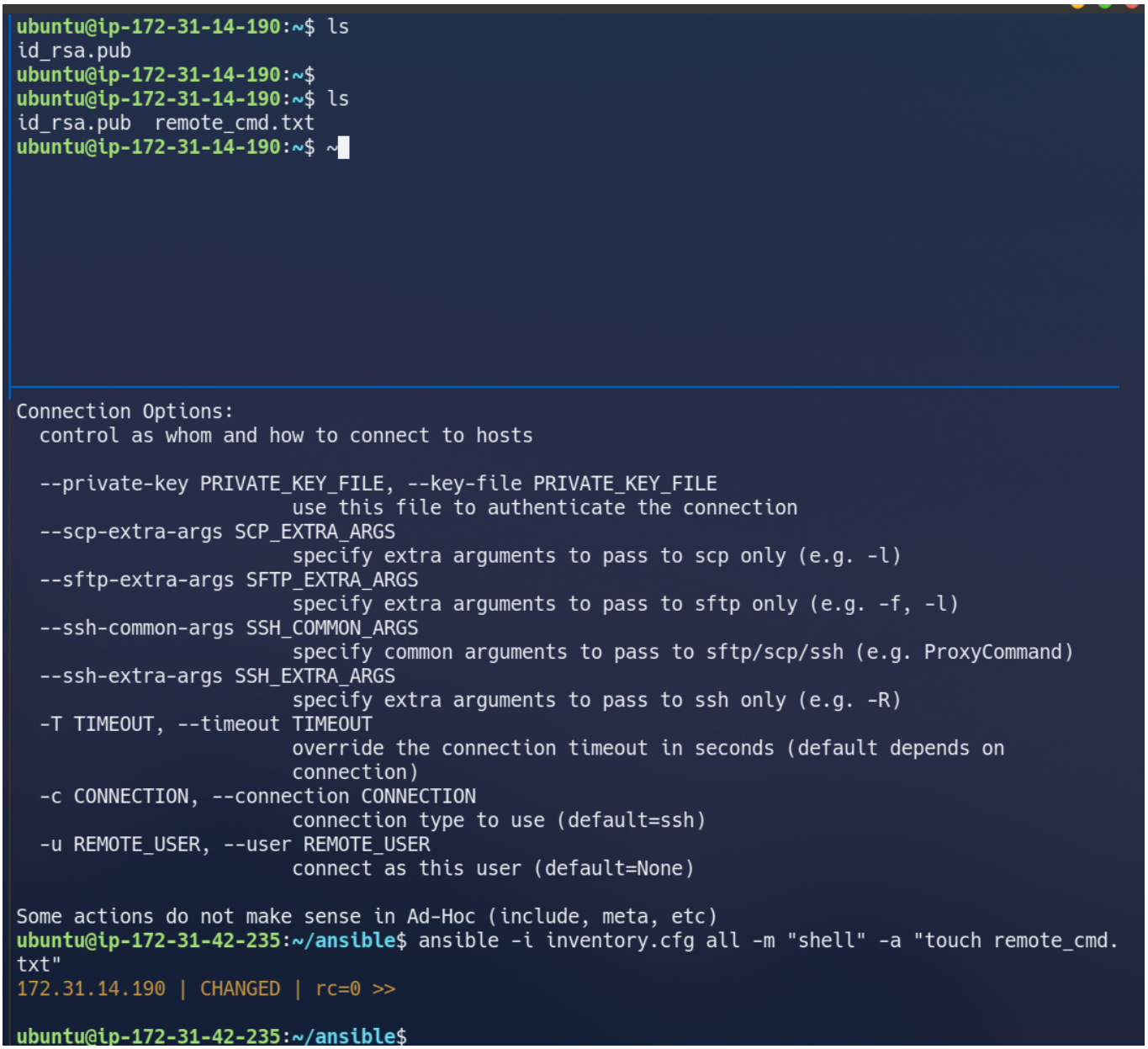
```
$ ansible <hosts> -m <module> -a <arguments>
```

Hosts: It can be an entry in the inventory file. For specifying all hosts in the inventory, use all or "*".

module_name: It is an optional parameter. There are hundreds of modules available in the Ansible, such as shell, yum, apt, file, and copy. By default, it is the command.

Arguments: We should pass values that are required by the module. It can change according to the module used.

on `master` i execute `touch remote_cmd.txt` and on `server` we can see file being created



Ansible Playbooks

Ansible Playbooks offer a repeatable, reusable, simple configuration management and multi-machine deployment system, one that is well suited to deploying complex applications. If you need to execute a task with Ansible more than once, write a playbook and put it under source control. Then you can use the playbook to push out new configuration or confirm the configuration of remote systems.
In our playbook file

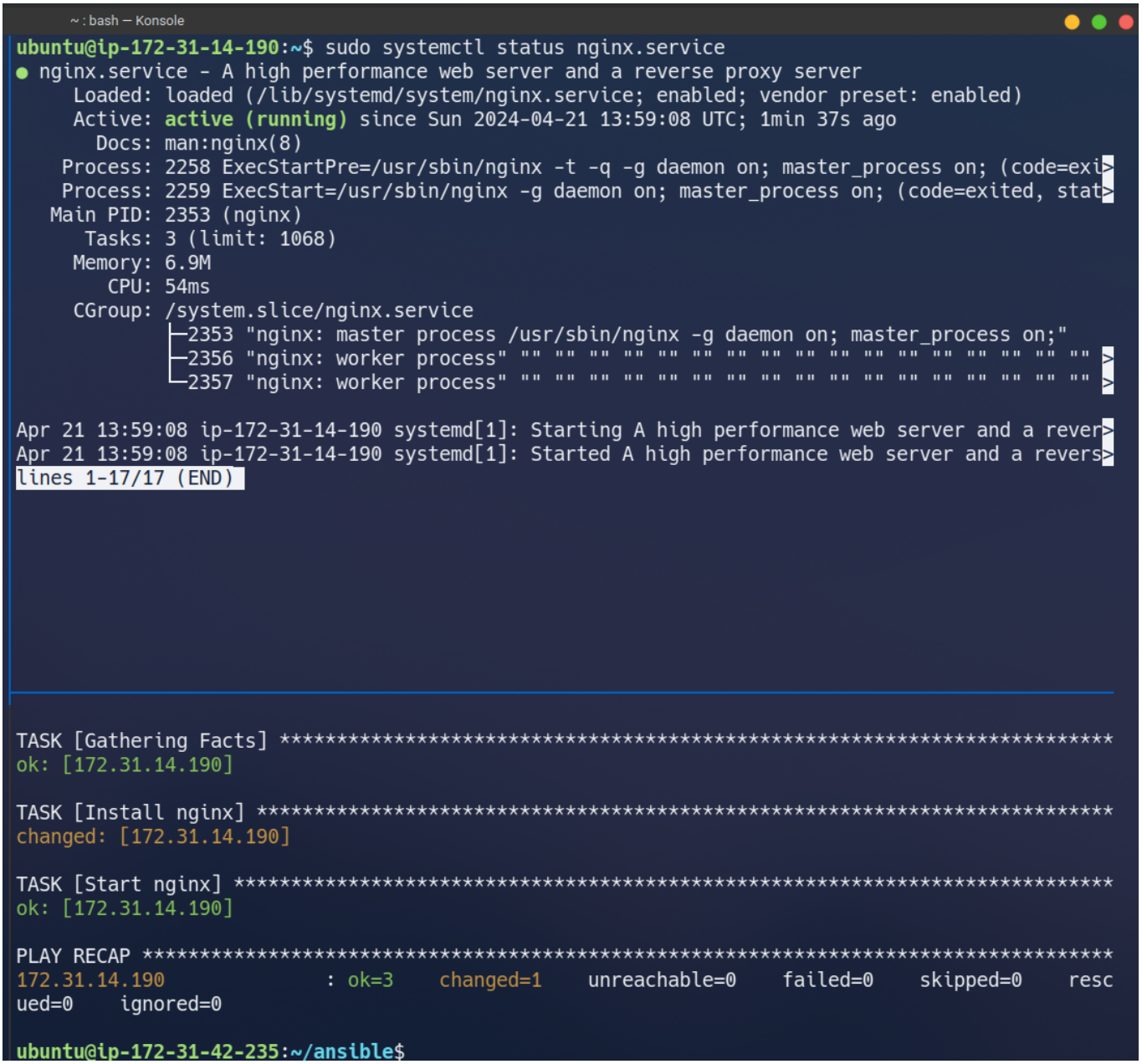
```
- name: Install and Restart nginx
  hosts: all
  become: true
  tasks:
    - name: Install nginx
      apt: name=nginx  state=latest

    - name: Start nginx
      service:
        name: nginx
        state: started

~
~
~
~
~
~
~
~
~
~
~
~
~
~
first_playbook.yml                                                  11,22         All
"first_playbook.yml" 12L, 218C written
```

In above file we declare two tasks one being installing nginx and second is starting the service.

Run following command to execute these tasks on our `server`
Here we can see all tasks being done and their outputs

```
ubuntu@ip-172-31-42-235:~/ansible$ ansible -i inventory.cfg all -m "shell" -a "touch remote_cmd.
txt"
172.31.14.190 | CHANGED | rc=0 >>

ubuntu@ip-172-31-42-235:~/ansible$ nvim first_playbook.yml
ubuntu@ip-172-31-42-235:~/ansible$ ansible
ansible             ansible-connection  ansible-galaxy      ansible-pull
ansible-community   ansible-console     ansible-inventory   ansible-test
ansible-config      ansible-doc         ansible-playbook    ansible-vault
ubuntu@ip-172-31-42-235:~/ansible$ ansible-playbook -i inventory.cfg first_playbook.yml

PLAY [Install and Restart nginx] ***********************************************************

TASK [Gathering Facts] *********************************************************************
ok: [172.31.14.190]

TASK [Install nginx] ***********************************************************************
changed: [172.31.14.190]

TASK [Start nginx] *************************************************************************
ok: [172.31.14.190]

PLAY RECAP *********************************************************************************
172.31.14.190              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0

ubuntu@ip-172-31-42-235:~/ansible$
```

Now let's see what changed at `server`



From here we used Ansible to install and start nginx service.