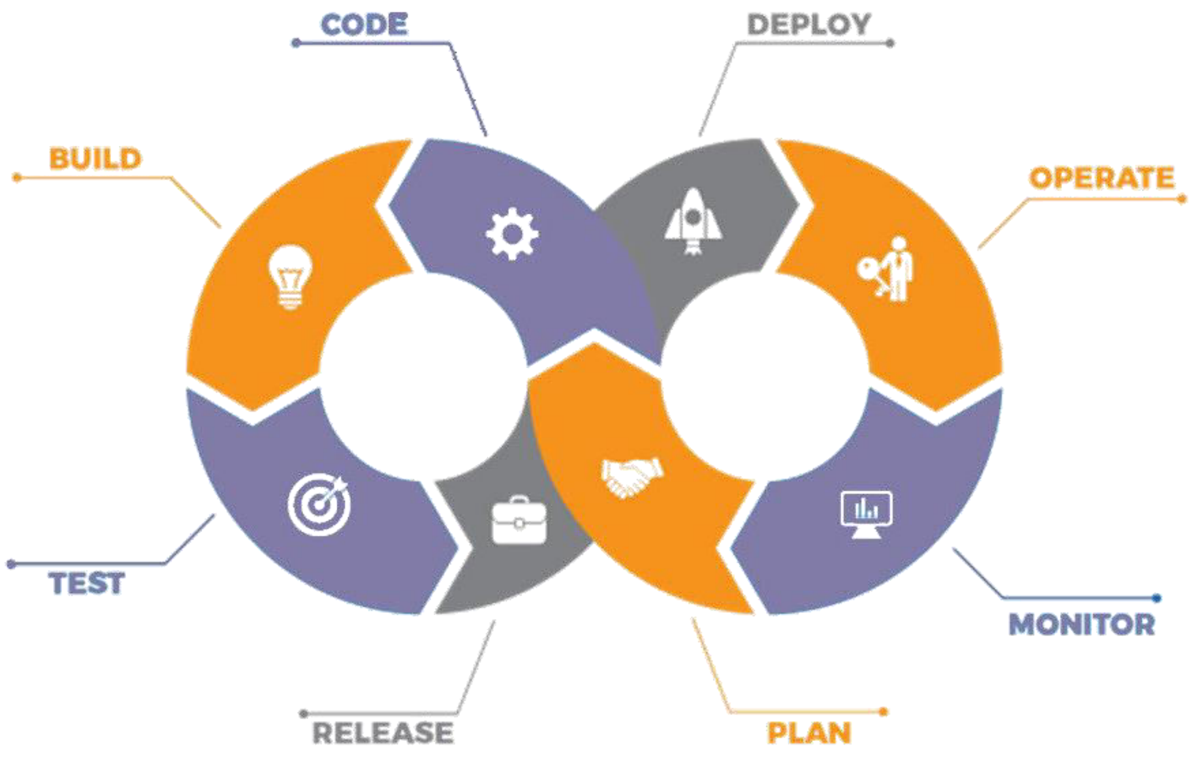
Introduction to Ansible



Agenda

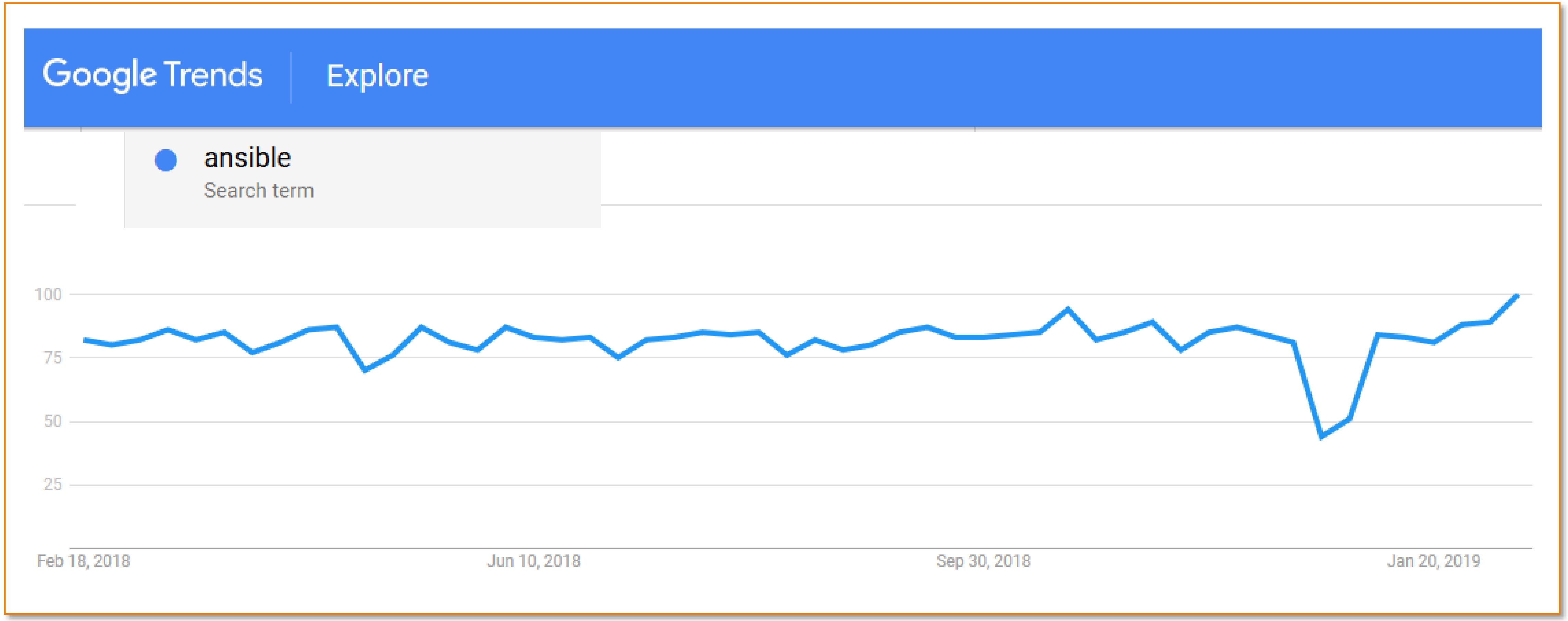
What is Ansible?

What is Ansible?

- Ansible is an open-source configuration management tool
- Used for configuration management
- Can solve wide range of automation challenges
- Written by Michael DeHaan
- Named after a fictional communication device, first used by Ursula K. LeGuin in her novel Rocannon's World in 1966
- In 2015 Red Hat acquired Ansible

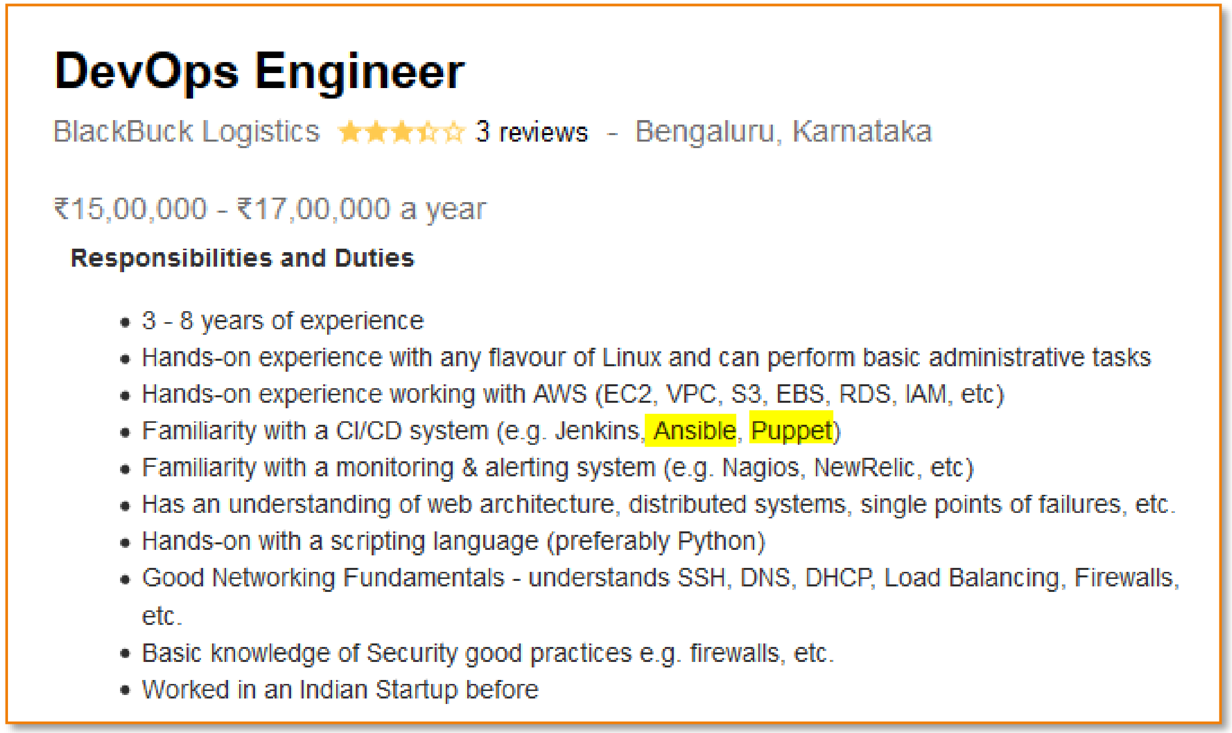
Why Ansible?

Why Ansible?



Google Trends Results for Ansible

Career Opportunities of Ansible



DevOps Engineer

BlackBuck Logistics ★★★☆☆ 3 reviews - Bengaluru, Karnataka

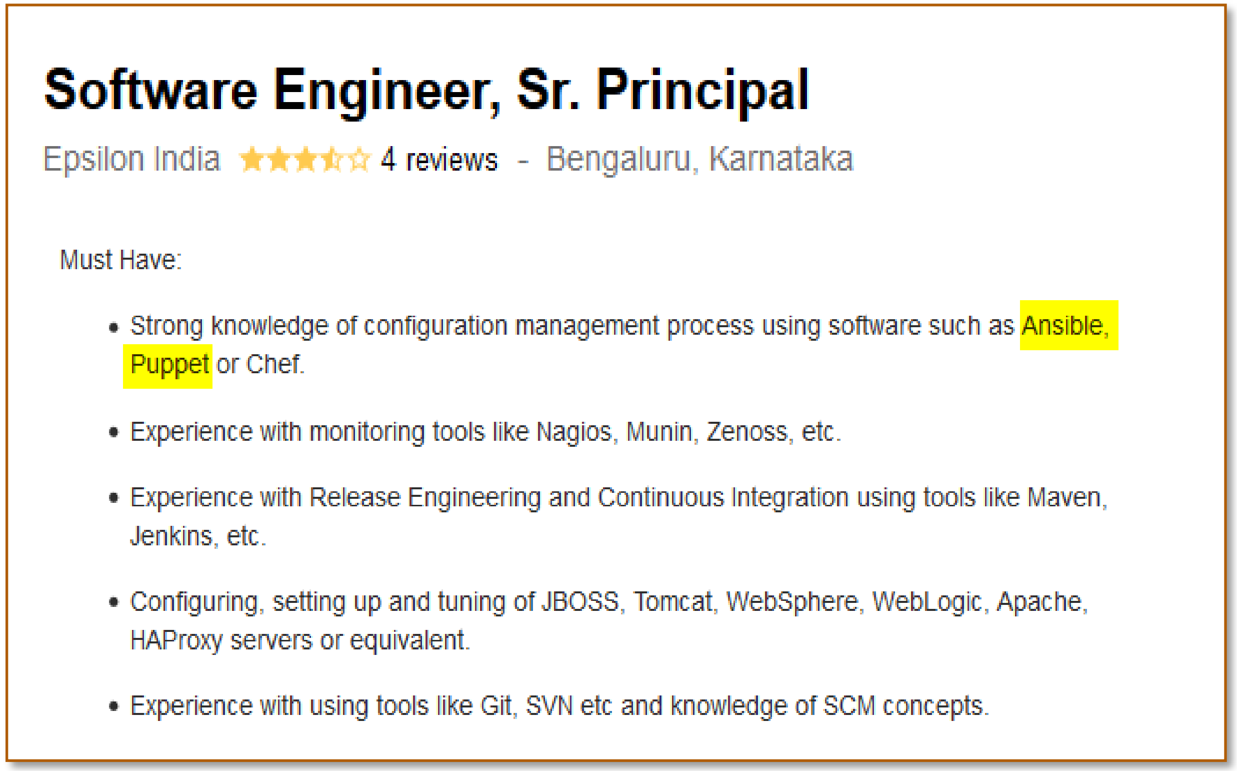
₹15,00,000 - ₹17,00,000 a year

Responsibilities and Duties

- 3 - 8 years of experience
- Hands-on experience with any flavour of Linux and can perform basic administrative tasks
- Hands-on experience working with AWS (EC2, VPC, S3, EBS, RDS, IAM, etc)
- Familiarity with a CI/CD system (e.g. Jenkins, Ansible, Puppet)
- Familiarity with a monitoring & alerting system (e.g. Nagios, NewRelic, etc)
- Has an understanding of web architecture, distributed systems, single points of failures, etc.
- Hands-on with a scripting language (preferably Python)
- Good Networking Fundamentals - understands SSH, DNS, DHCP, Load Balancing, Firewalls, etc.
- Basic knowledge of Security good practices e.g. firewalls, etc.
- Worked in an Indian Startup before

BLACKBUCK

Career Opportunities of Ansible



Software Engineer, Sr. Principal

Epsilon India ★★★★☆ 4 reviews - Bengaluru, Karnataka

Must Have:

- Strong knowledge of configuration management process using software such as Ansible, Puppet or Chef.
- Experience with monitoring tools like Nagios, Munin, Zenoss, etc.
- Experience with Release Engineering and Continuous Integration using tools like Maven, Jenkins, etc.
- Configuring, setting up and tuning of JBOSS, Tomcat, WebSphere, WebLogic, Apache, HAProxy servers or equivalent.
- Experience with using tools like Git, SVN etc and knowledge of SCM concepts.

EPSILON®

Advantage of Ansible

- ✓ Easy to learn
- ✓ Written in Python
- ✓ Easy installation and configuration steps
- ✓ No need to install ansible on slave
- ✓ Highly scalable

Popularity of Ansible

Apple

NASA

Intel

Percussion

Cisco

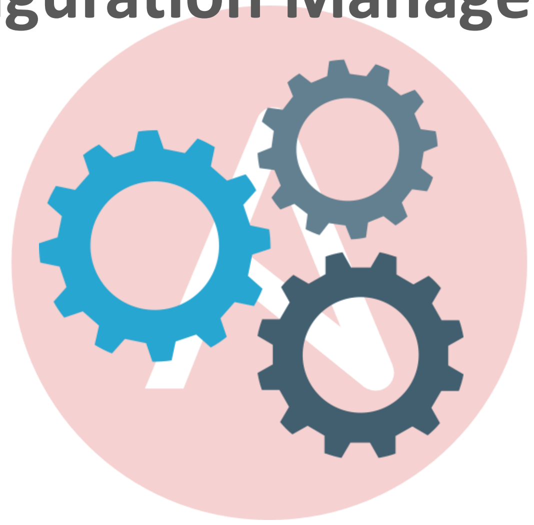
Twitter

How does Ansible work?

How does Ansible work?

With the help of **Ansible Playbooks**,
which are written in a very simple language, **YAML**

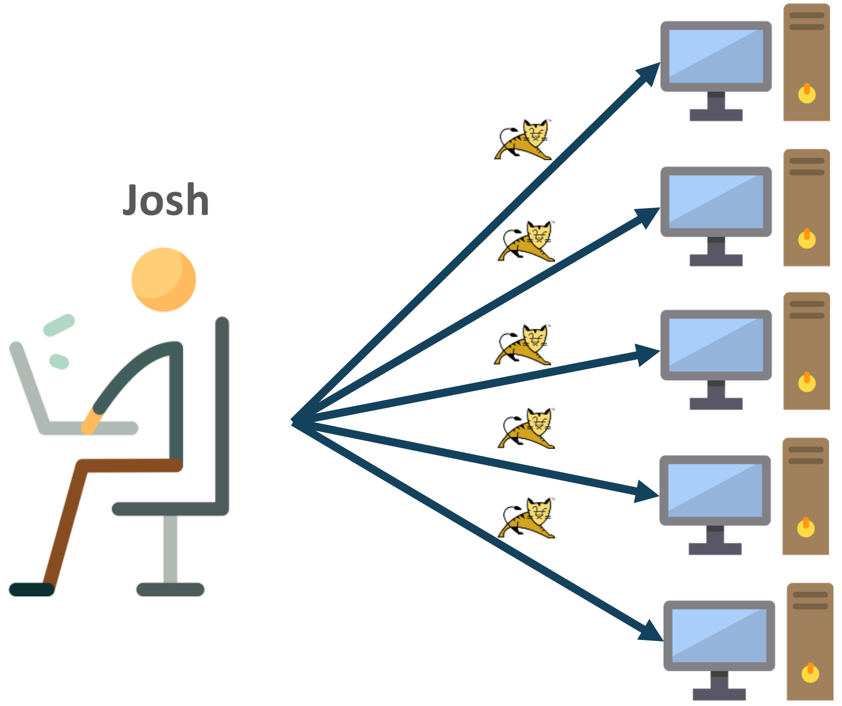
Configuration Management



Problem Statement

Say, Josh runs an enterprise, wants to install a new version of Apache Tomcat in all the systems

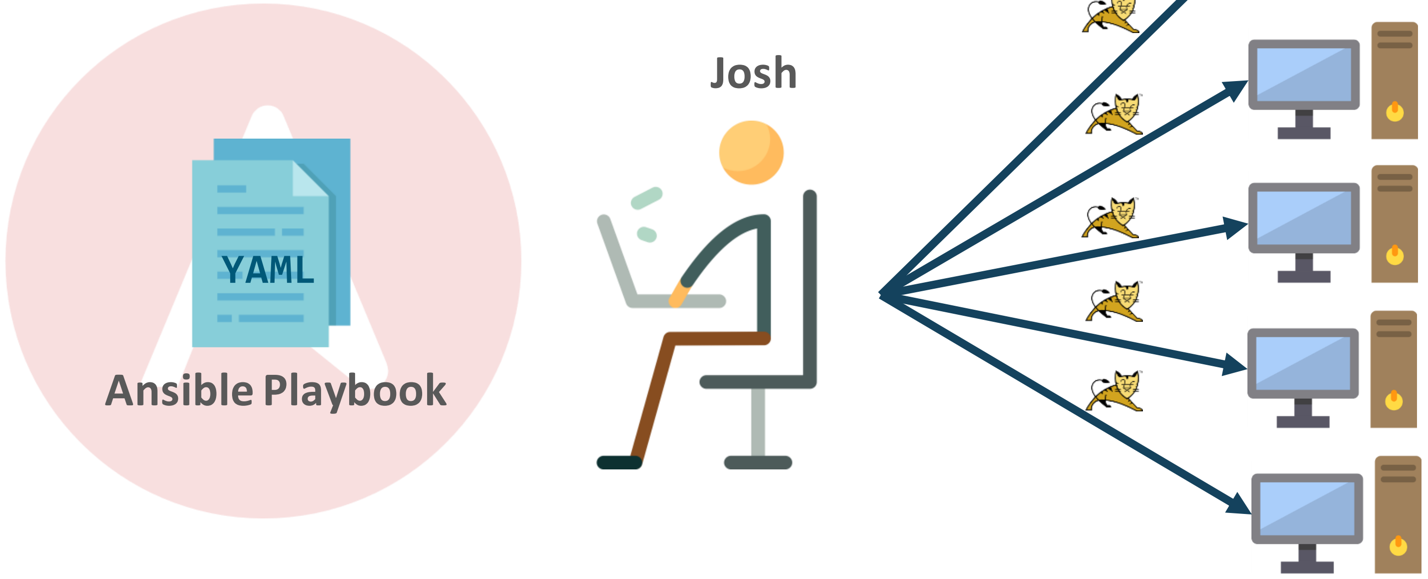
Configuration Management



Problem Statement-Solution with Ansible

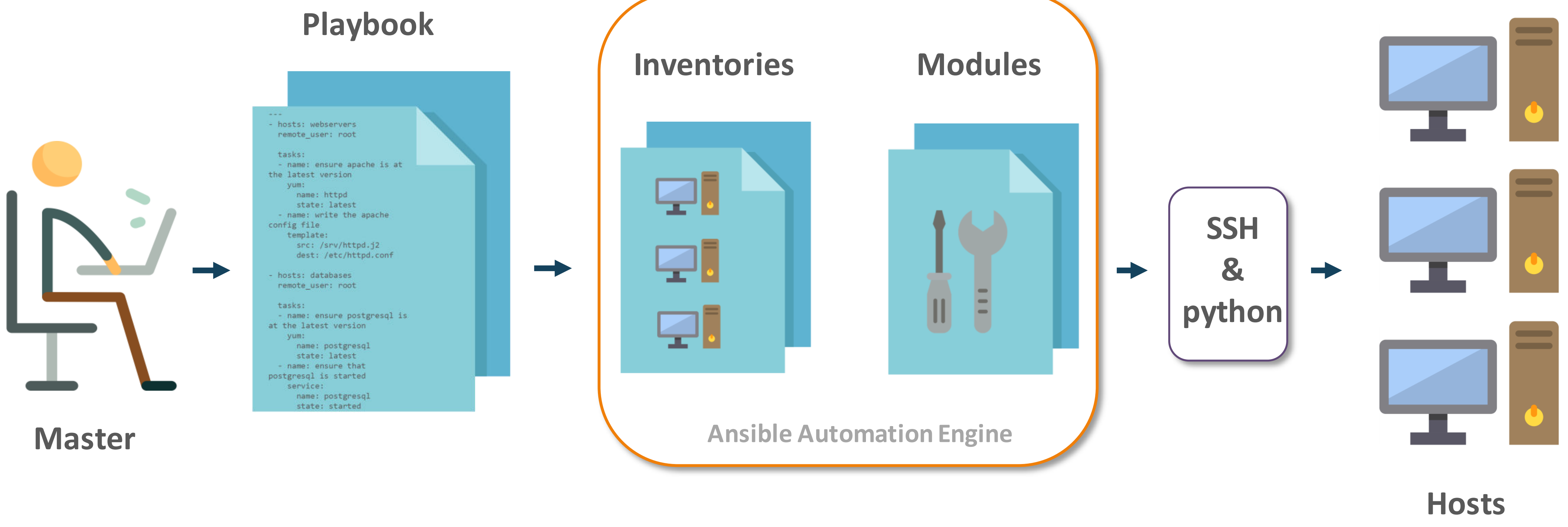
Instead of going to each system, manually updating, Josh can use Ansible to automate the installation using Ansible Playbooks

Configuration Management



Ansible Architecture

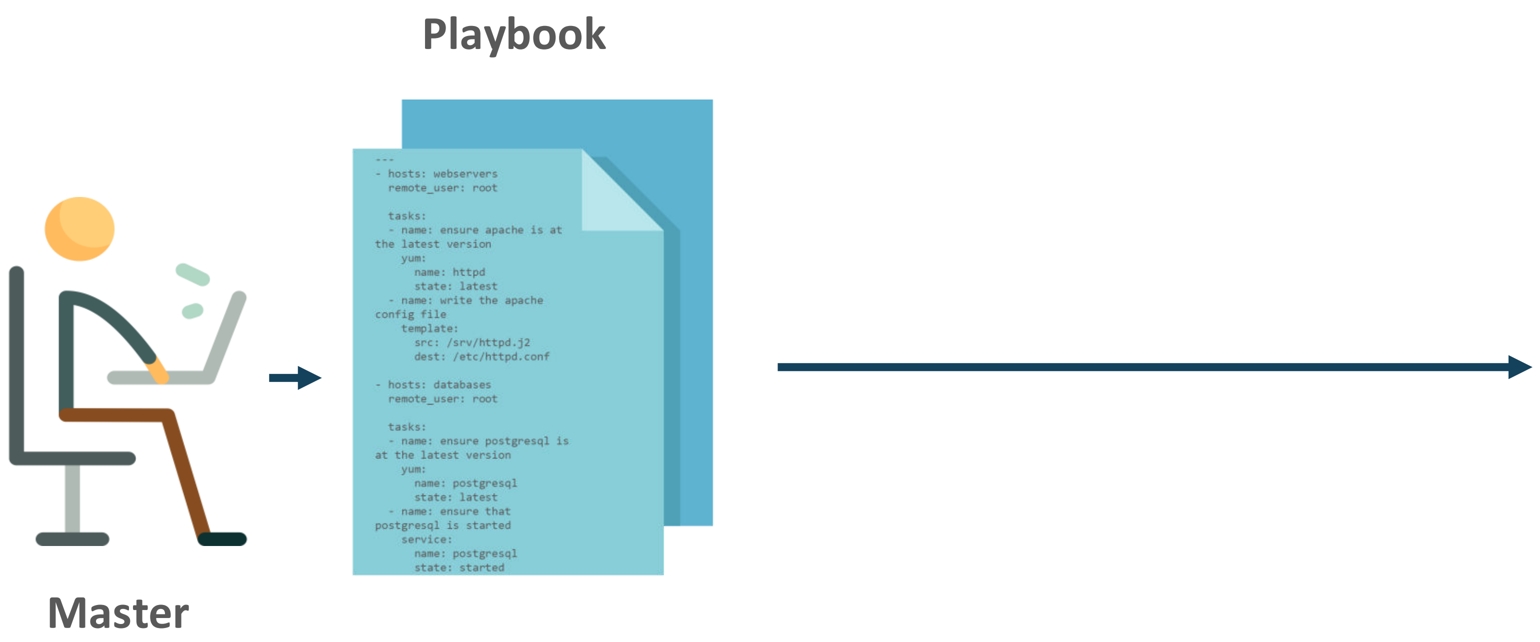
Ansible Architecture



```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

Basic Ansible Architecture

Ansible Architecture- Master

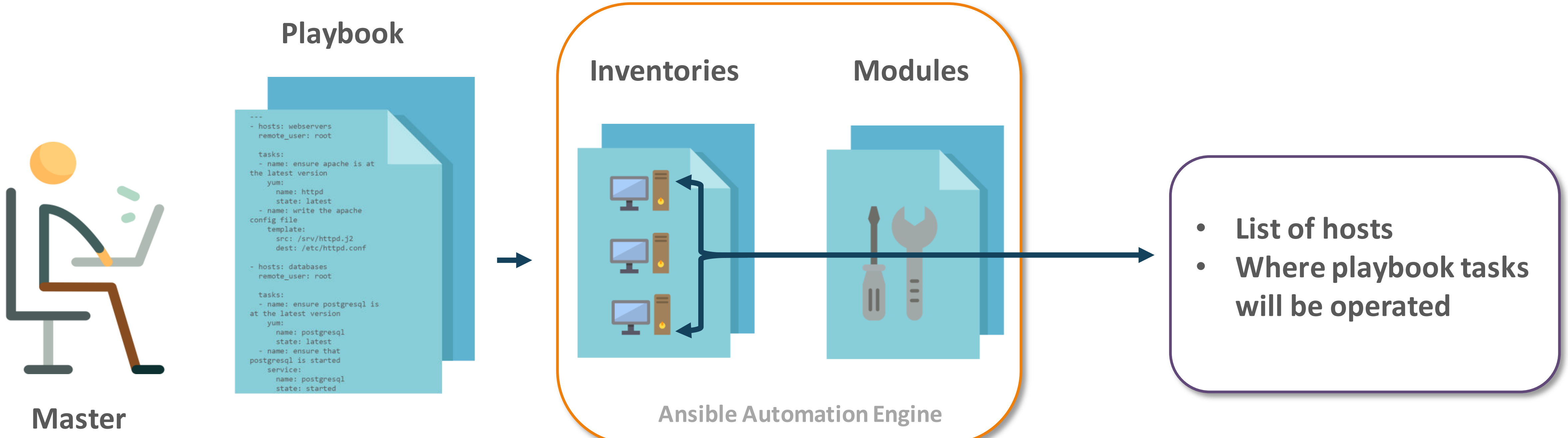


```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

- Describes the tasks to be executed
- Written in simple language
- Playbooks are like instruction manuals

Ansible Architecture- Inventories

Play

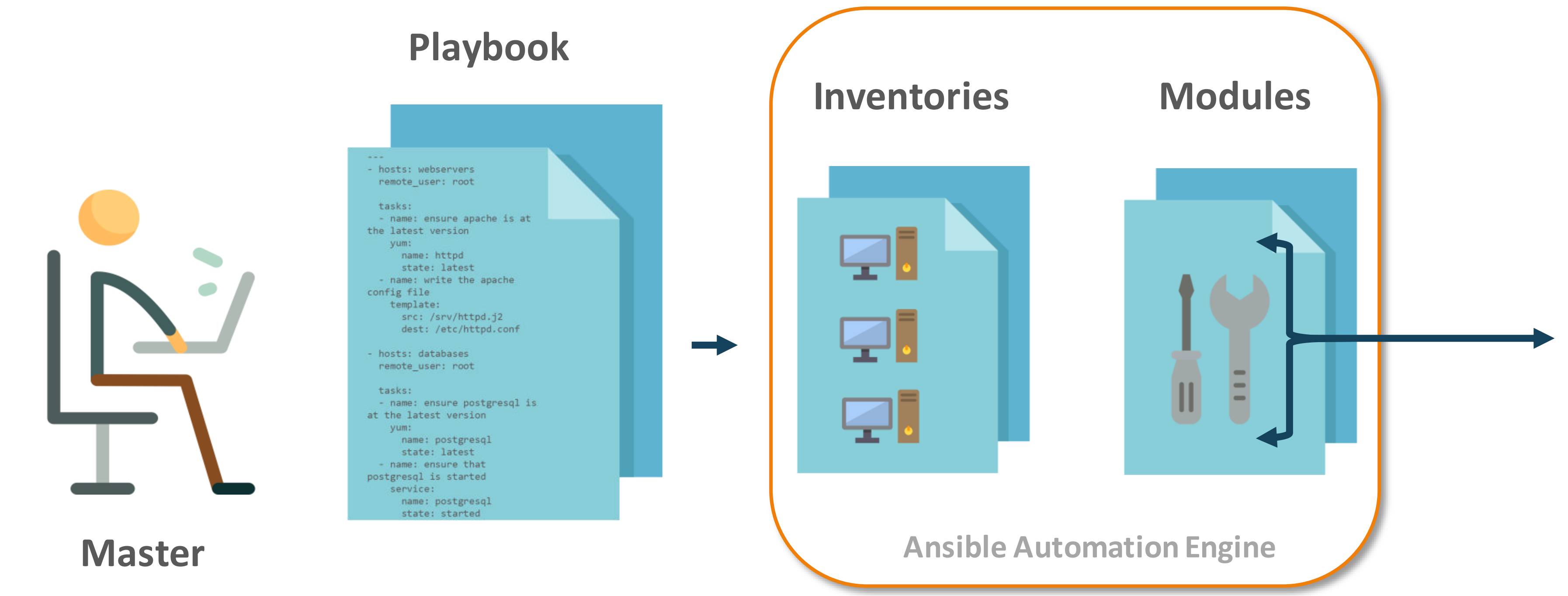


```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

- List of hosts
- Where playbook tasks will be operated

Ansible Architecture- Modules

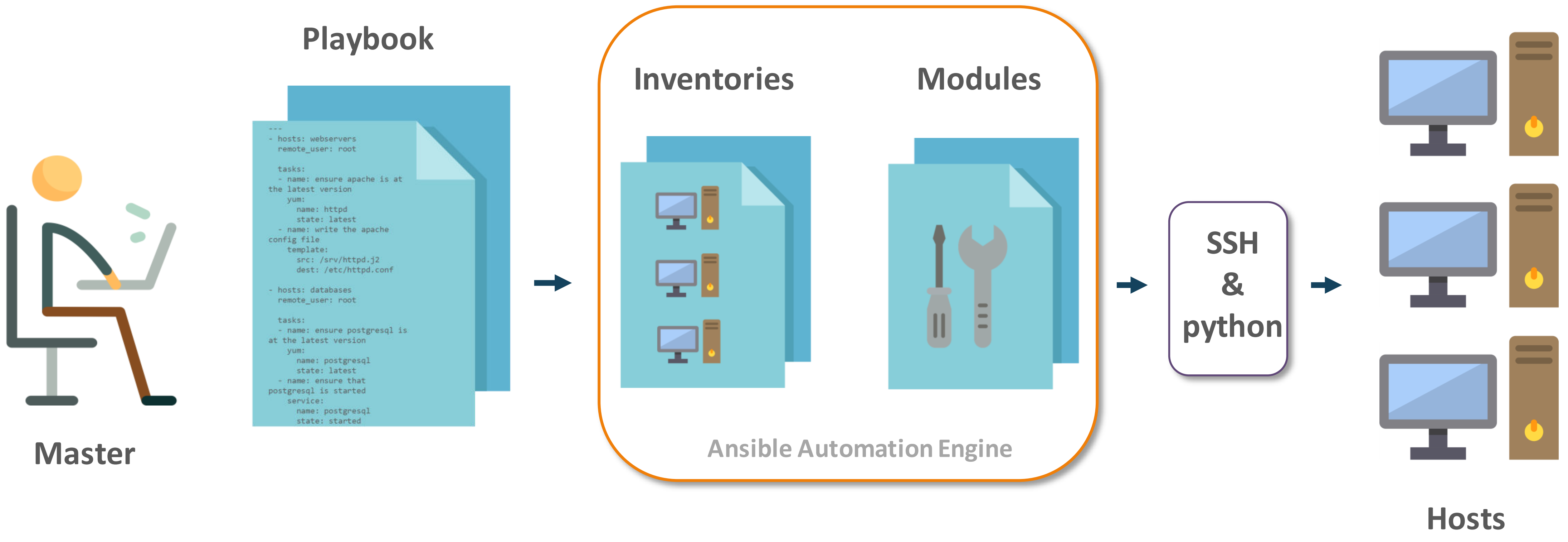
Play



```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

- Modules are like tools
- Can control system resources, like services, packages etc.
- 500+ core modules
- Also allows custom

Ansible Architecture- Hosts



```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

Case Study: Ansible being used in NASA

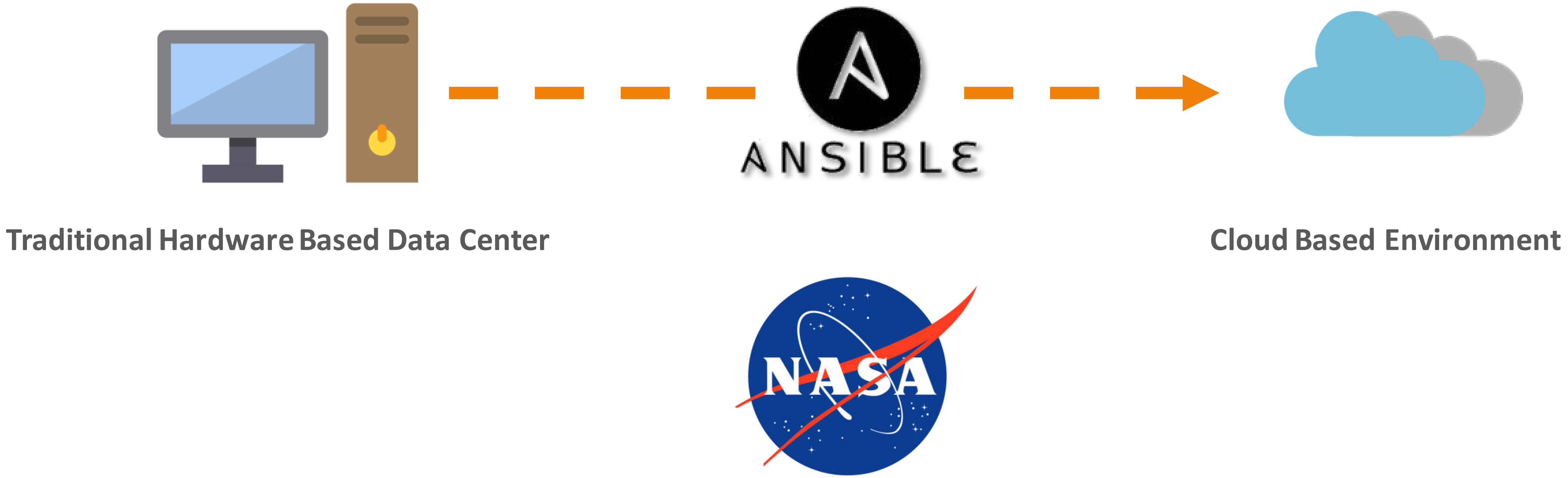
Case Study- Business Challenge

NASA needed to move roughly 65+ applications from a Traditional Hardware Based Data Center to Cloud Based Environment for better agility and cost saving

Traditional Hardware Based Data Center
Cloud Based Environment

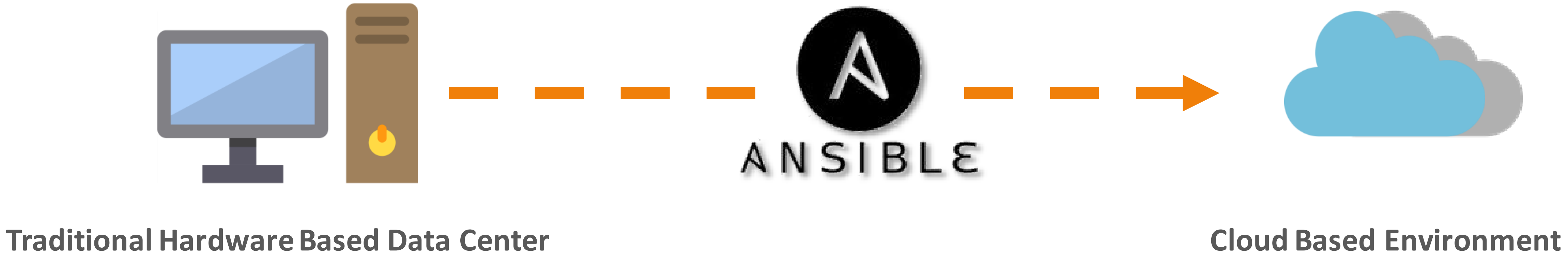
Case Study- Solution

NASA used Ansible to manage and schedule the cloud environment



Case Study- Results

- ✓ Could provide better operations and security to its clients
- ✓ Increased team efficiency
- ✓ Patching updates went from a multi-day process to 45 minutes



Installing Ansible on AWS

Installing Ansible on AWS

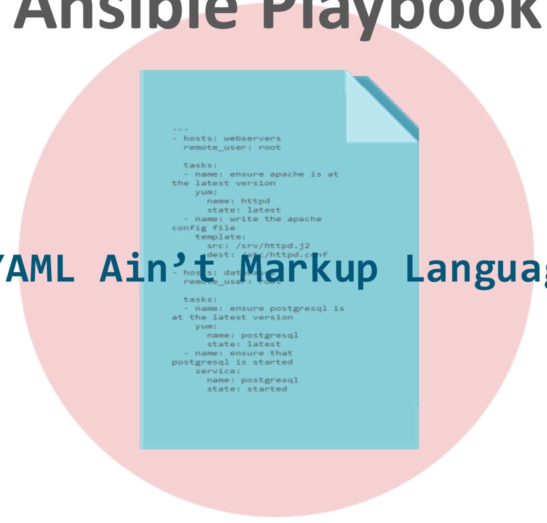
1. Install Ansible on Master
2. Configure SSH access to Ansible Host
3. Setting up Ansible Host and testing connection

Creating Ansible Playbooks

What is Ansible Playbook?

An organized unit of scripts
Defines work for a server configuration
Written in **YAML**

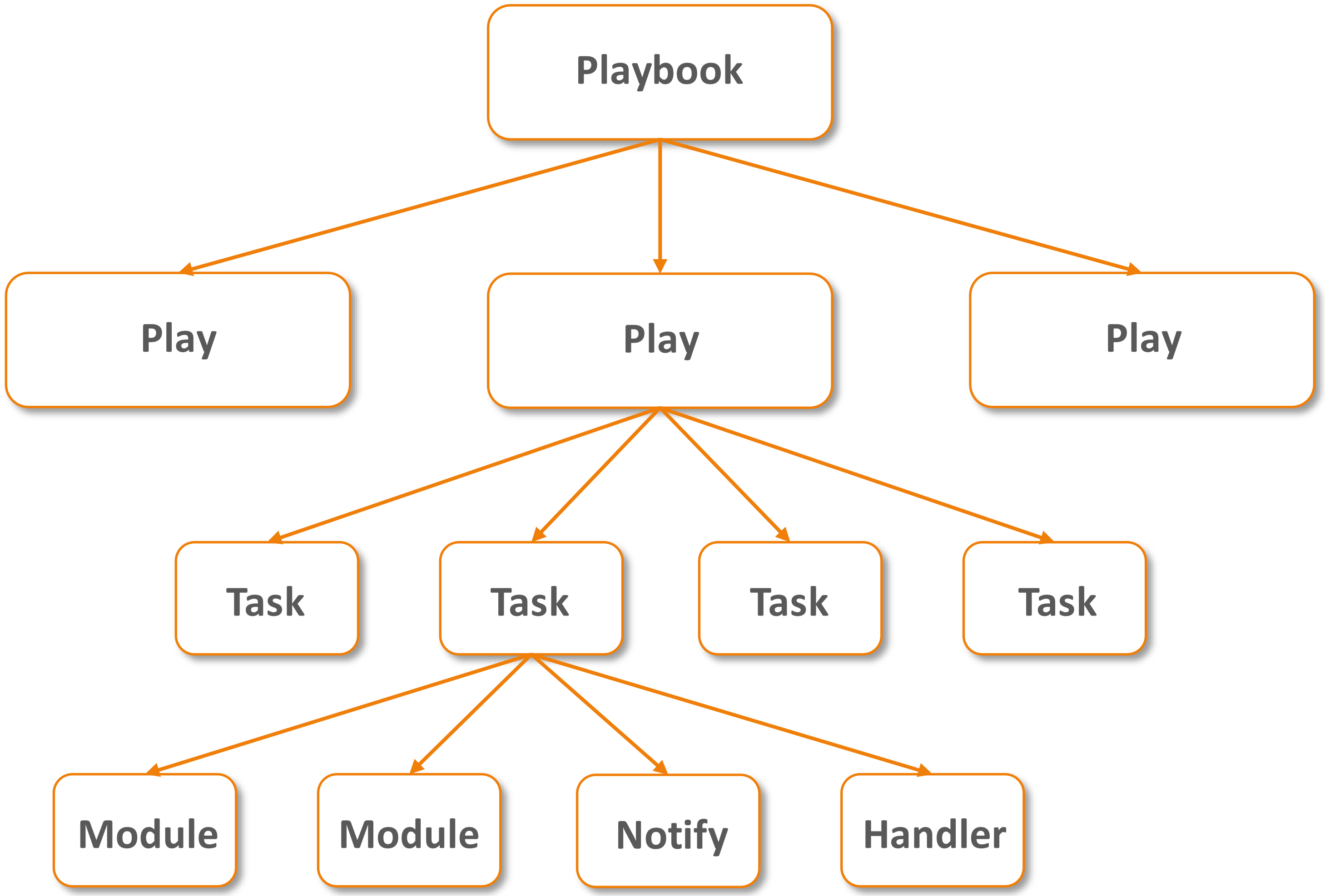
Ansible Playbook



```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

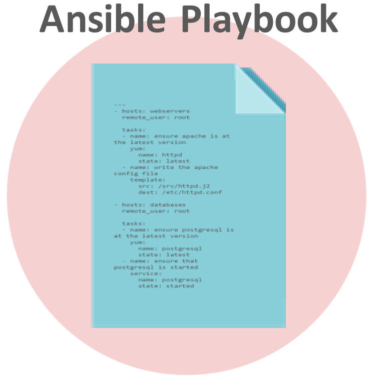
YAML Ain’t Markup Language

Ansible Playbook Structure



- **Playbook** have number of **plays**
- **Play** contains **tasks**
- **Tasks** calls core or custom **modules**
- **Handler** gets triggered from **notify** and executed at the end only once.

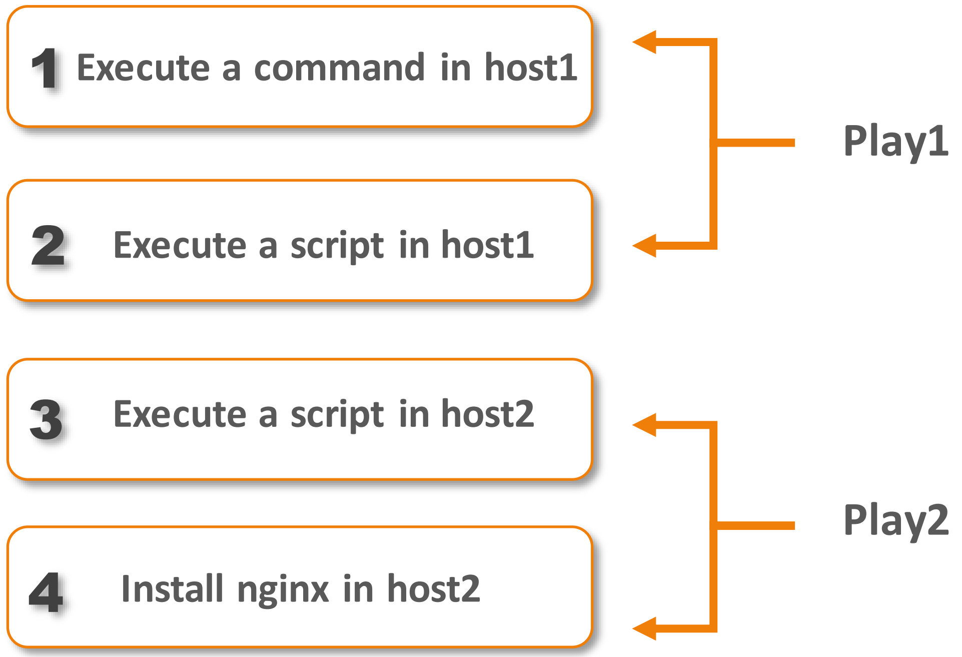
Ansible Playbook



```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at
the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache
config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is
at the latest version
    yum:
      name: postgresql
      state: latest
  - name: ensure that
postgresql is started
    service:
      name: postgresql
      state: started
```

Creating Ansible Playbook-Example

Say, we want to create a playbook with two plays with following tasks


1 Execute a command in host1
2 Execute a script in host1
3 Execute a script in host2
4 Install nginx in host2
Play1
Play2

Creating Ansible Playbook-Example

```
---

- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command ‘Date’
       command: date
     - name: Execute script on server
       script: test_script.sh


- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: Install nginx
       apt: name=nginx state=latest
```

Say we want to create a playbook with two plays with following tasks

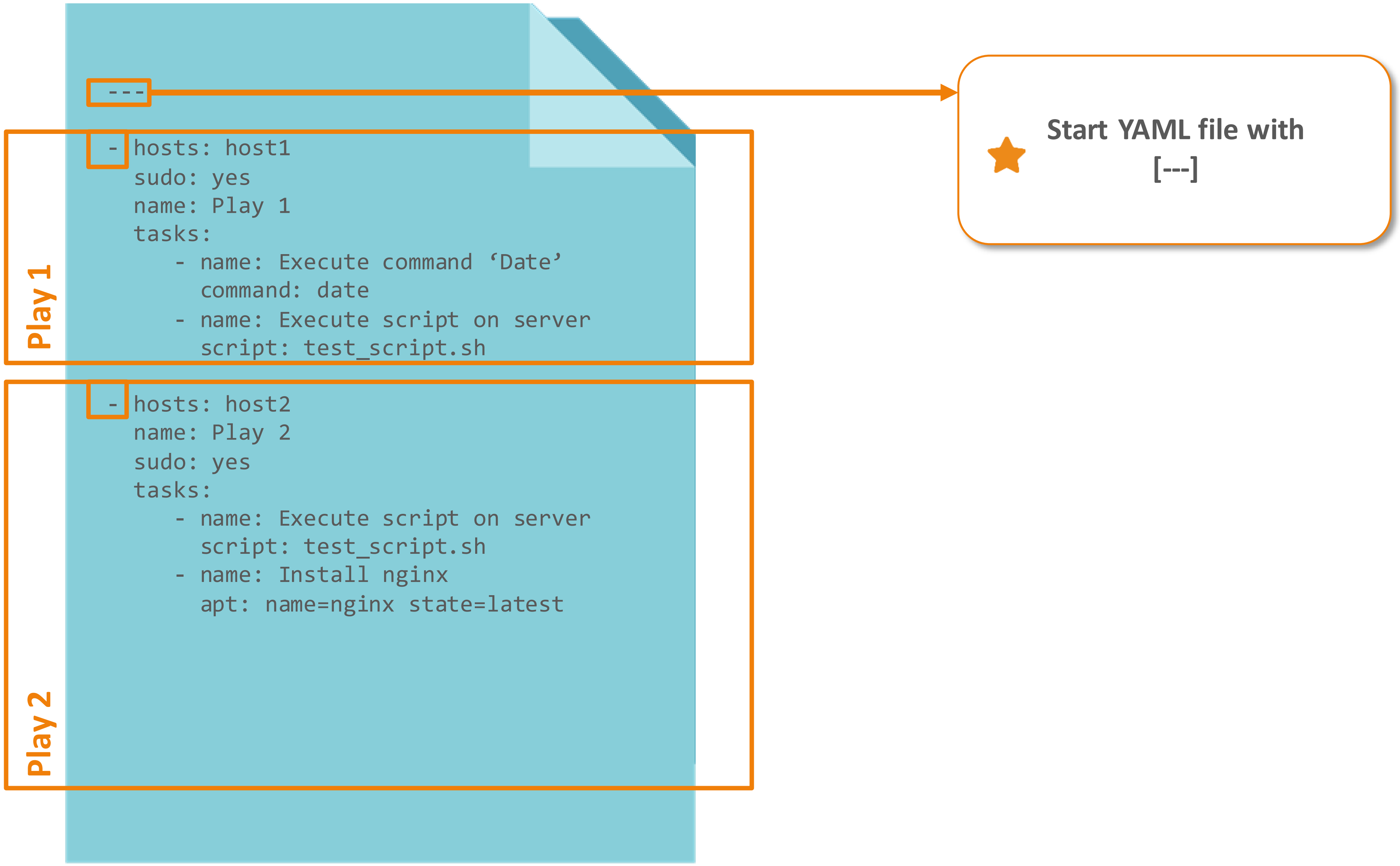
1 Execute a command in host1

2 Execute a script in host1

3 Execute a script in host2

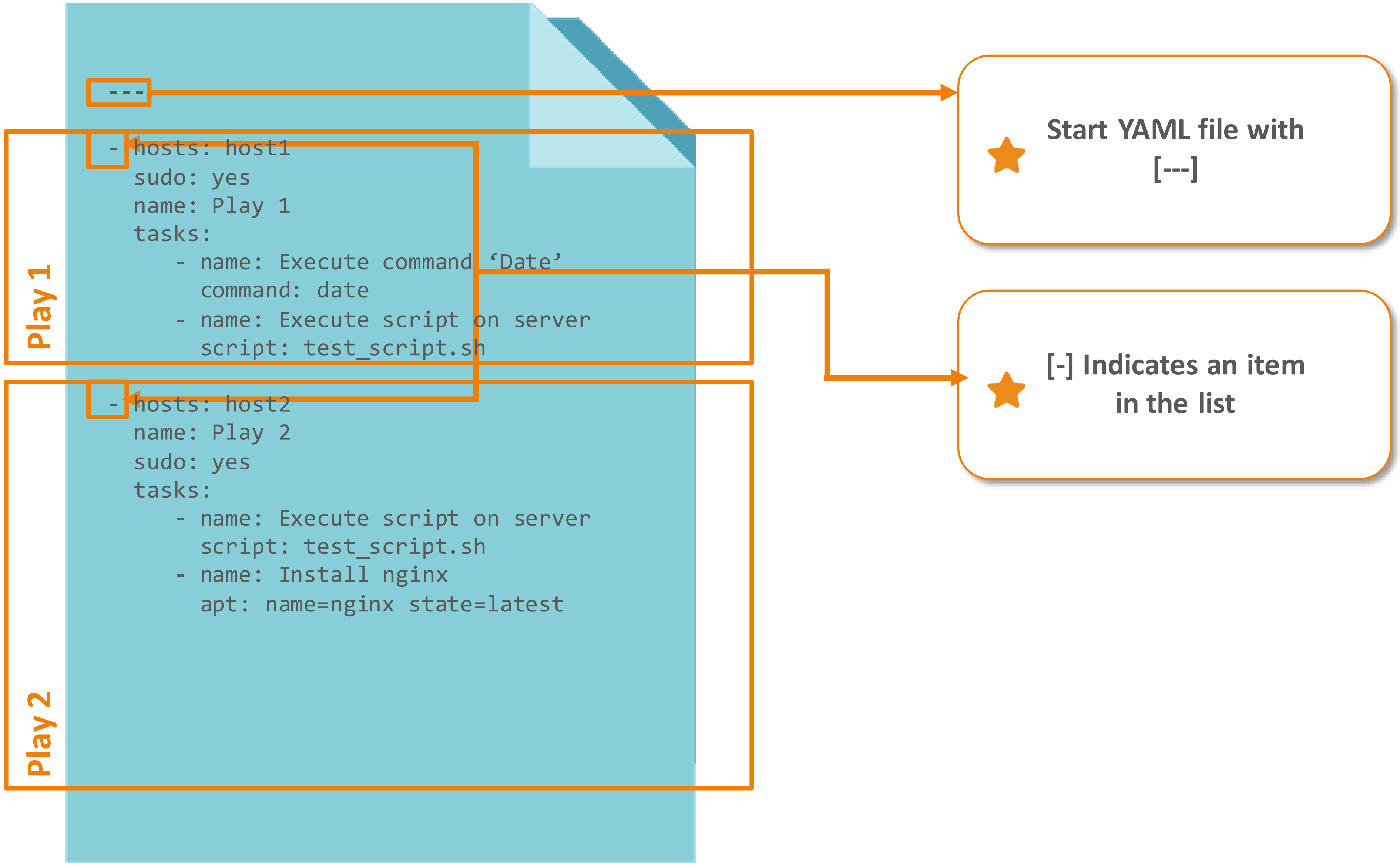
4 Install nginx in host2

Creating Ansible Playbook-Example



```
---
- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command ‘Date’
       command: date
     - name: Execute script on server
       script: test_script.sh
- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: Install nginx
       apt: name=nginx state=latest
```

Creating Ansible Playbook-Example

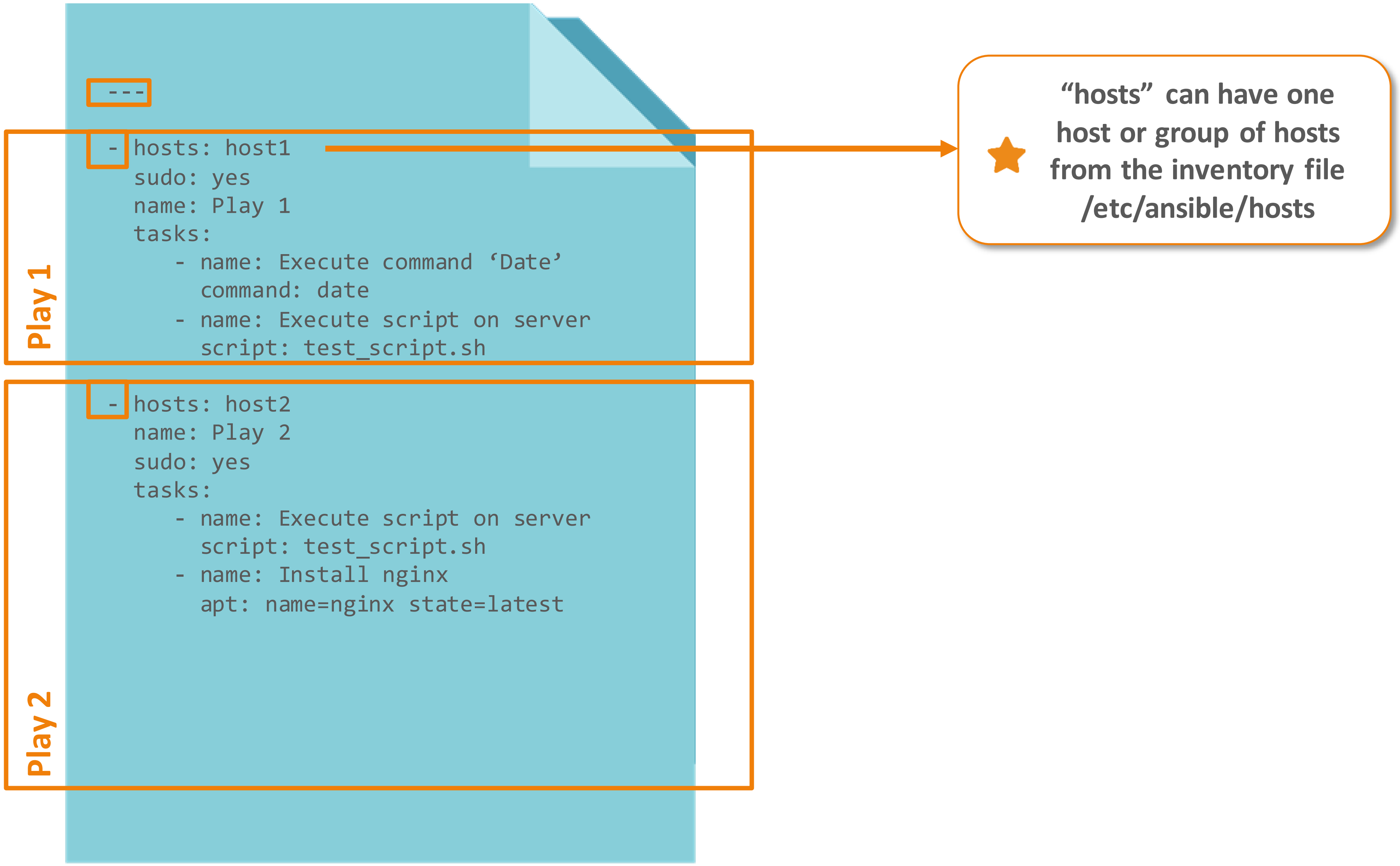


```
---
- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command ‘Date’
       command: date
     - name: Execute script on server
       script: test_script.sh
- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: Install nginx
       apt: name=nginx state=latest
```

Start YAML file with [---]

[-] Indicates an item in the list

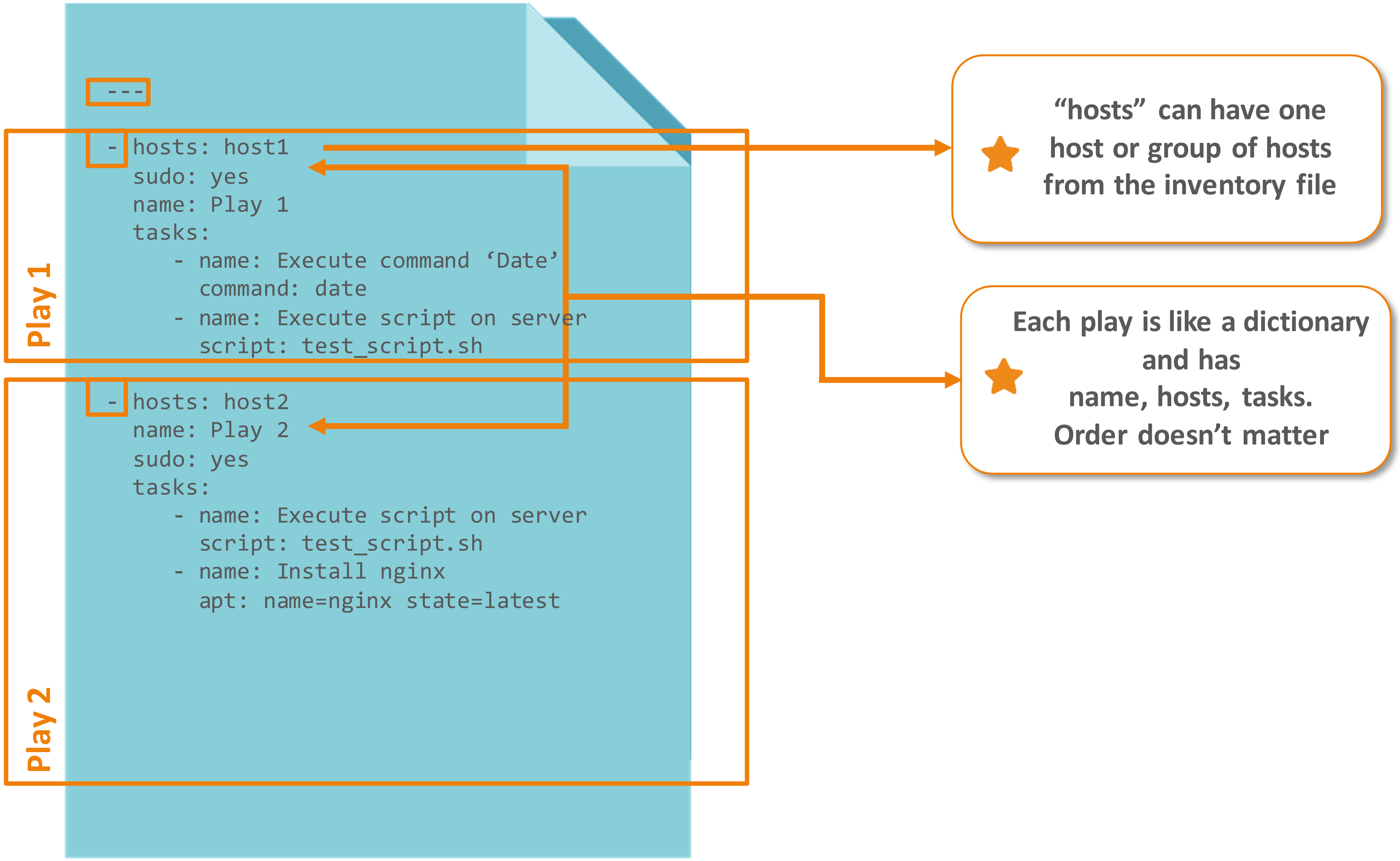
Creating Ansible Playbook-Example



```
---
- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command ‘Date’
       command: date
     - name: Execute script on server
       script: test_script.sh
- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: Install nginx
       apt: name=nginx state=latest
```

“hosts” can have one host or group of hosts from the inventory file /etc/ansible/hosts

Creating Ansible Playbook-Example

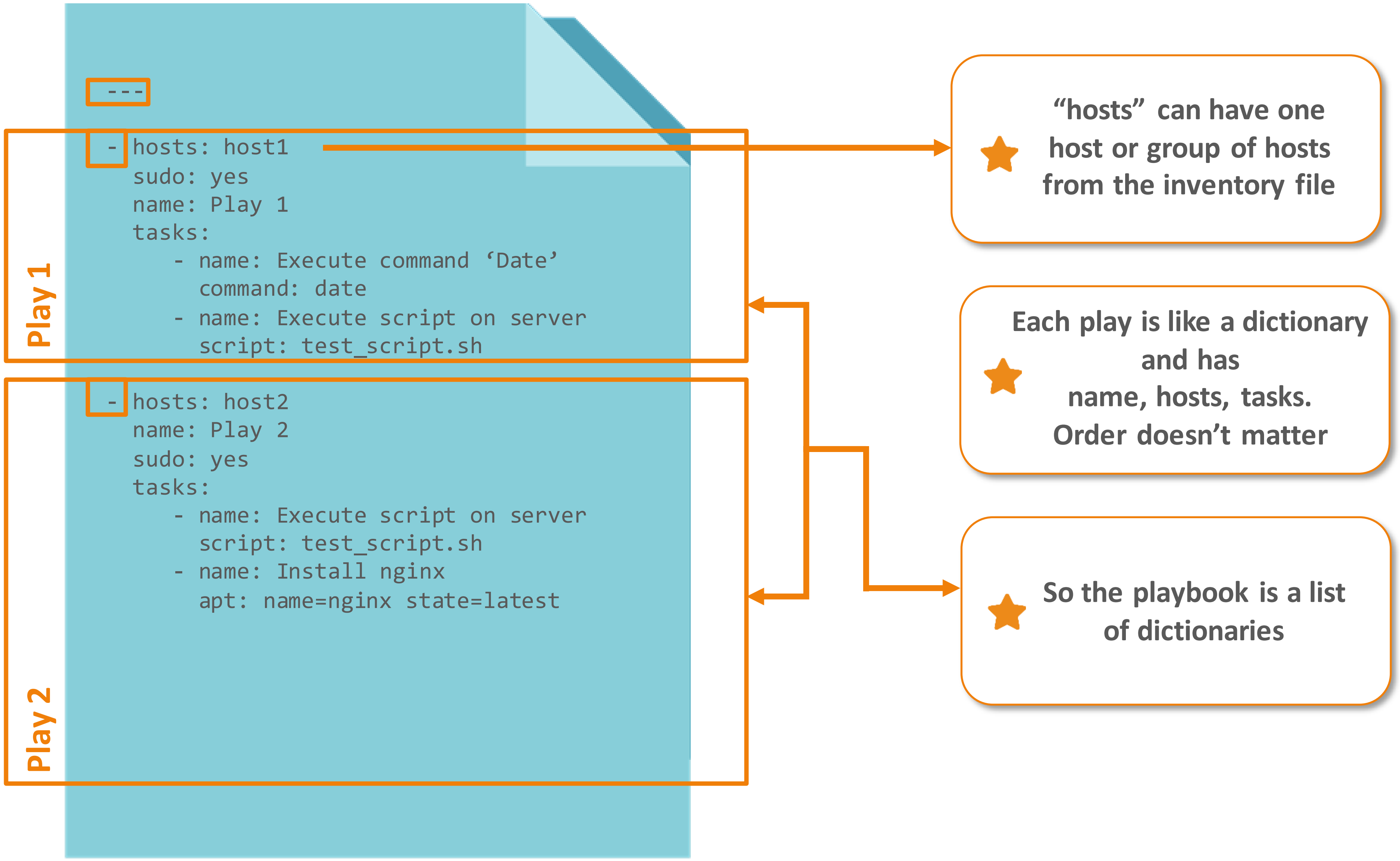


```
---
- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command ‘Date’
       command: date
     - name: Execute script on server
       script: test_script.sh
- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: Install nginx
       apt: name=nginx state=latest
```

★ “hosts” can have one host or group of hosts from the inventory file

★ Each play is like a dictionary and has name, hosts, tasks. Order doesn’t matter

Creating Ansible Playbook-Example



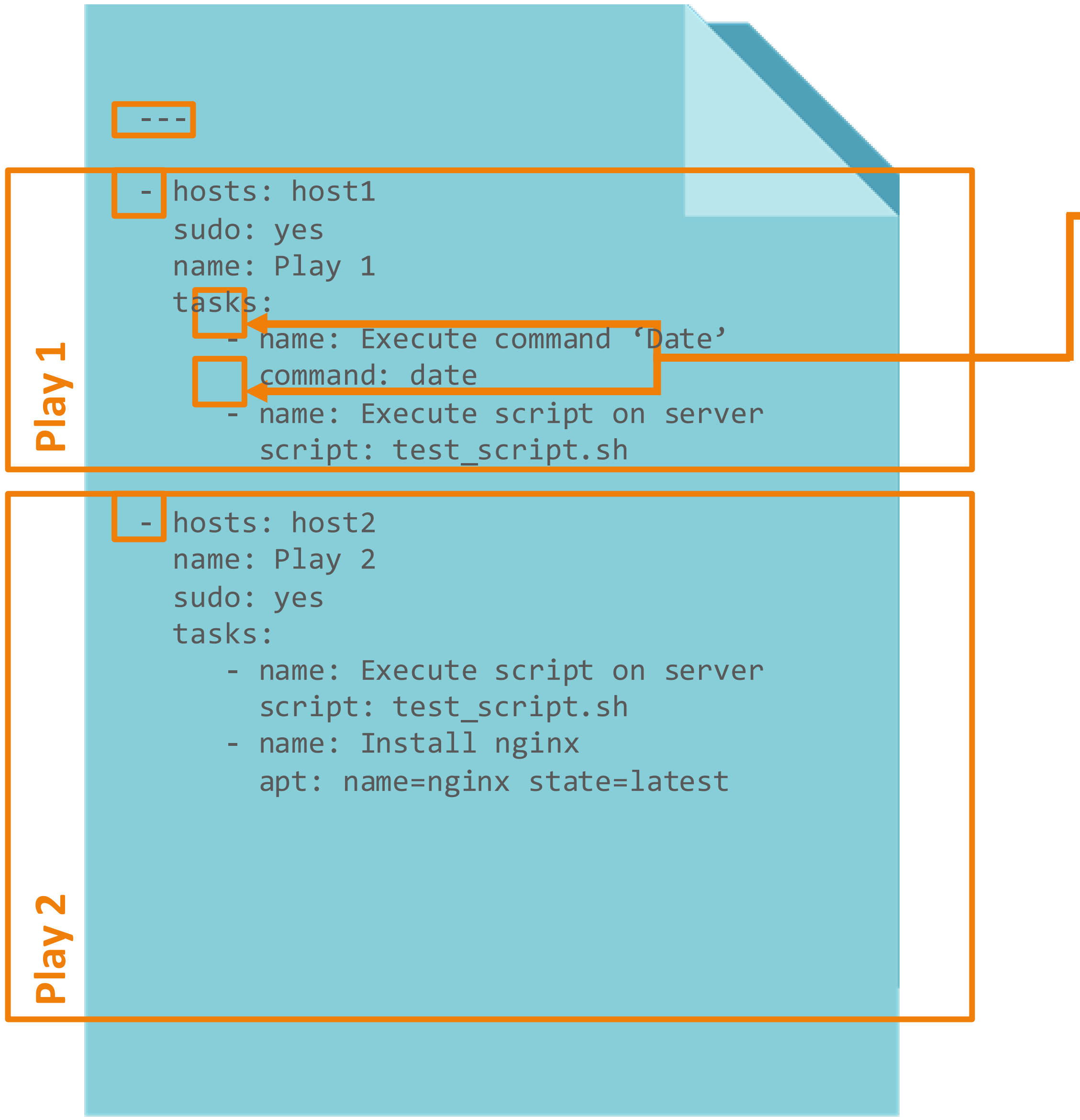
```
---
- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command ‘Date’
       command: date
     - name: Execute script on server
       script: test_script.sh
- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: Install nginx
       apt: name=nginx state=latest
```

“hosts” can have one host or group of hosts from the inventory file

Each play is like a dictionary and has name, hosts, tasks. Order doesn’t matter

So the playbook is a list of dictionaries

Creating Ansible Playbook-Example



```
---
- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
    - name: Execute command ‘Date’
      command: date
    - name: Execute script on server
      script: test_script.sh
- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
    - name: Execute script on server
      script: test_script.sh
    - name: Install nginx
      apt: name=nginx state=latest
```

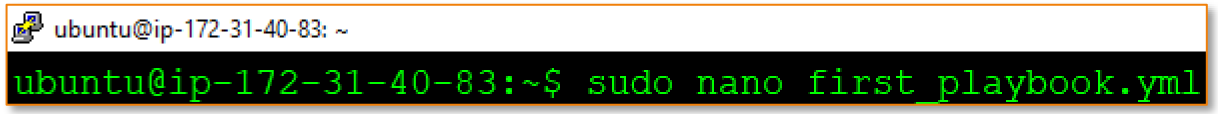
Similarly tasks are nothing but lists
Denoted by [-]

For tasks ordered collection.
Position of entry matters

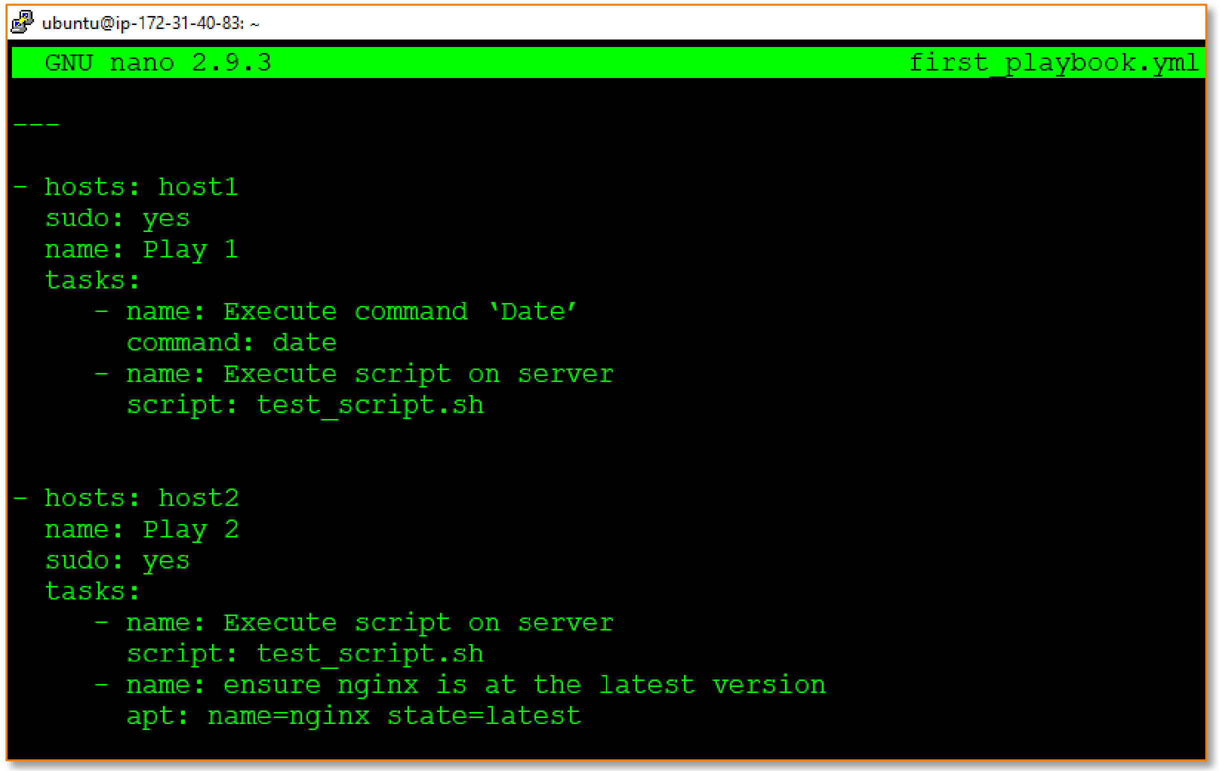
First entry gets performed first

Creating Ansible Playbook-Example

Create first_playbook.yml using
sudo nano <playbookname>



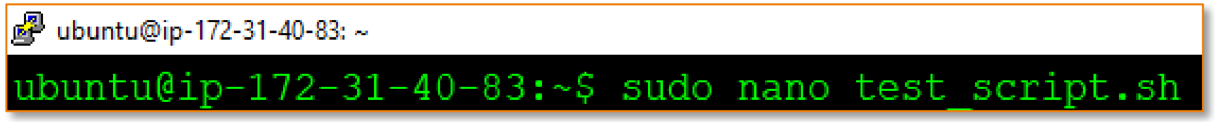
```
ubuntu@ip-172-31-40-83: ~
ubuntu@ip-172-31-40-83:~$ sudo nano first_playbook.yml
```



```
ubuntu@ip-172-31-40-83: ~
  GNU nano 2.9.3                                              first_playbook.yml

---

- hosts: host1
  sudo: yes
  name: Play 1
  tasks:
     - name: Execute command 'Date'
       command: date
     - name: Execute script on server
       script: test_script.sh


- hosts: host2
  name: Play 2
  sudo: yes
  tasks:
     - name: Execute script on server
       script: test_script.sh
     - name: ensure nginx is at the latest version
       apt: name=nginx state=latest
```

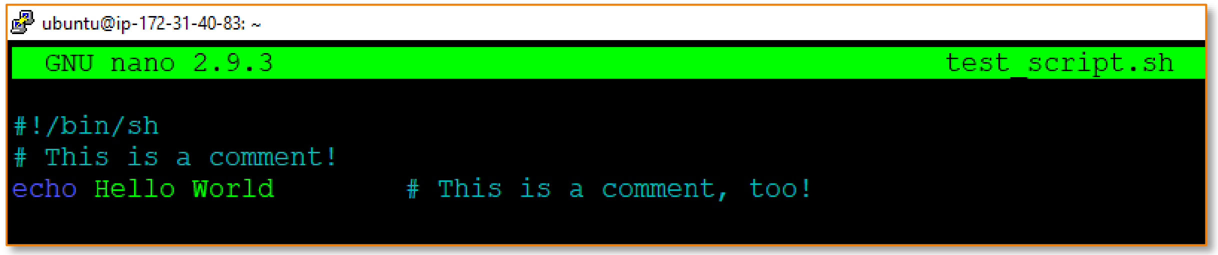
Creating Ansible Playbook-Example

Create test_script.sh using
sudo nano <file_name>



ubuntu@ip-172-31-40-83: ~

```
ubuntu@ip-172-31-40-83:~$ sudo nano test_script.sh
```

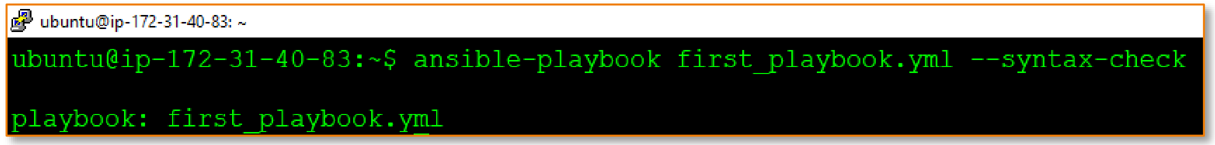


ubuntu@ip-172-31-40-83: ~

```
  GNU nano 2.9.3                                              test_script.sh

#!/bin/sh
# This is a comment!
echo Hello World        # This is a comment, too!
```

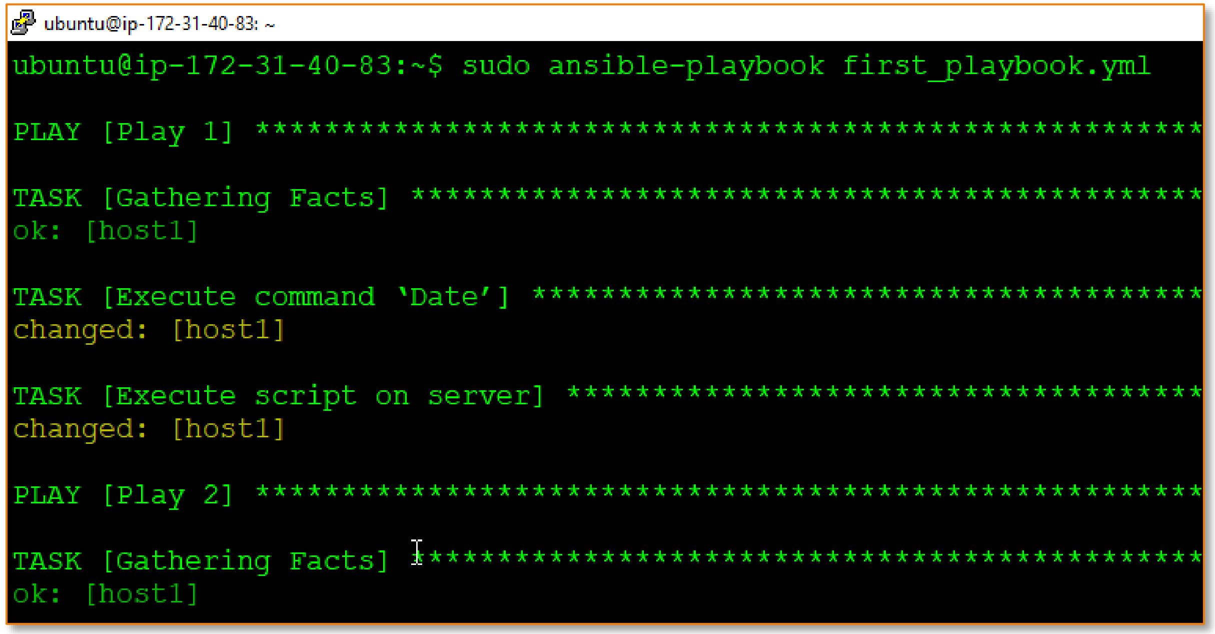
Creating Ansible Playbook-Example

Syntax-check and execute ansible playbook using
ansible-playbook <playbook> --syntax-check and
ansible-playbook <playbook>



ubuntu@ip-172-31-40-83: ~

```
ubuntu@ip-172-31-40-83:~$ ansible-playbook first_playbook.yml --syntax-check

playbook: first_playbook.yml
```



ubuntu@ip-172-31-40-83: ~

```
ubuntu@ip-172-31-40-83:~$ sudo ansible-playbook first_playbook.yml

PLAY [Play 1] ***************************************************************

TASK [Gathering Facts] ******************************************************
ok: [host1]

TASK [Execute command 'Date'] ***********************************************
changed: [host1]

TASK [Execute script on server] *********************************************
changed: [host1]

PLAY [Play 2] ***************************************************************

TASK [Gathering Facts] ******************************************************
ok: [host1]
```

Ansible Roles

What is Ansible Roles?

An ansible role is group of tasks, files, and handlers stored in a standardized file structure.
Roles are small functionalities which can be used independently used but only within playbook

Ansible Playbook

Ansible playbook organizes tasks

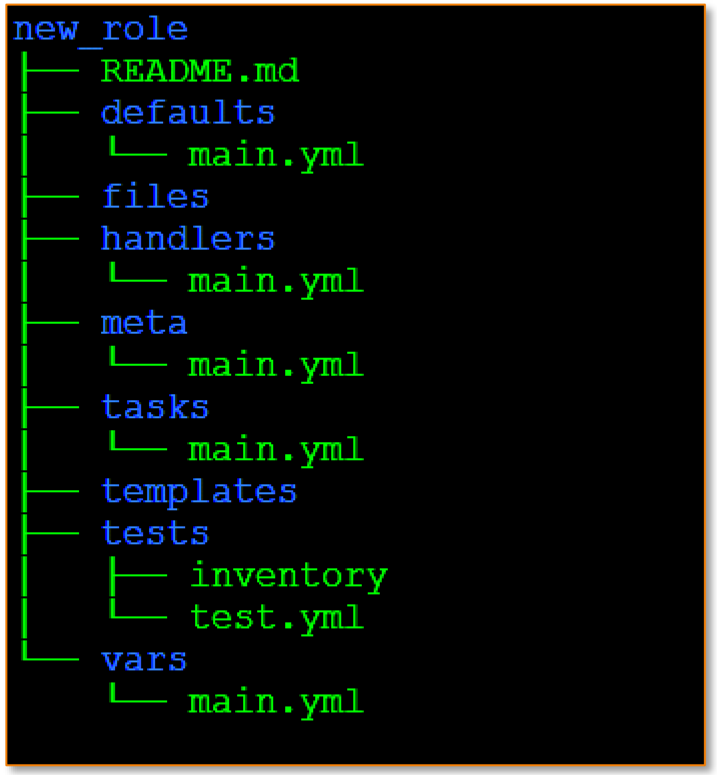
Ansible Roles

Ansible roles organizes playbooks

Why do we need Ansible Roles?

- Roles simplifies writing complex playbooks
- Roles allows you to reuse common configuration steps between different types of servers
- Roles are flexible and can be easily modified

Structure of Ansible Role



```
new_role
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Structure of an Ansible Role

Structure of an ansible role consists of below given components

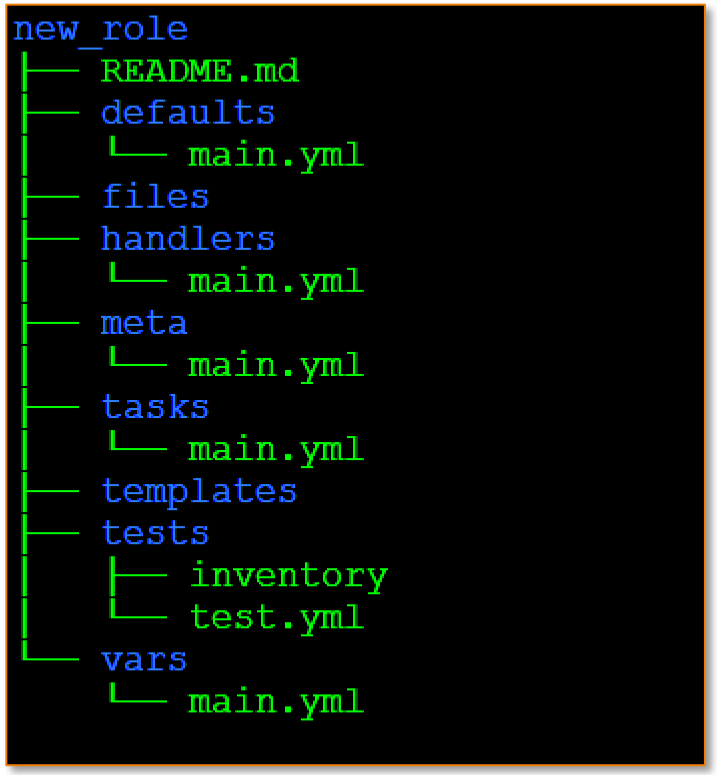
Defaults: Store data about the role, also store default variables.

Files: Store files that needs to be pushed to the remote machine.

Handlers: Tasks that get triggered from some actions.

Meta: Information about author, supported platforms and dependencies.

Structure of Ansible Role



```
new_role
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Structure of an Ansible Role

Structure of an ansible role consists of below given components

Tasks: Contains the main list of tasks to be executed by the role.

Templates: Contains templates which can be deployed via this role.

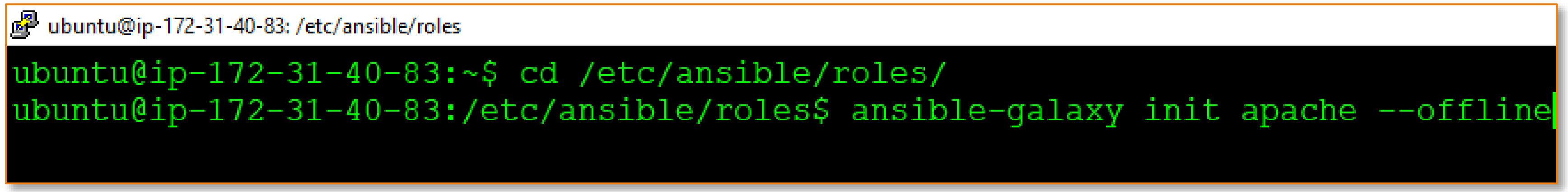
Handlers: Tasks that get triggered from some actions.

Vars: Stores variables with higher priority than default variables. Difficult to override.

Creating an Ansible Role

1 Use the *ansible-galaxy init <role name> --offline* command to create one Ansible role

★ Remember that Ansible roles should be written inside */etc/ansible/roles/*

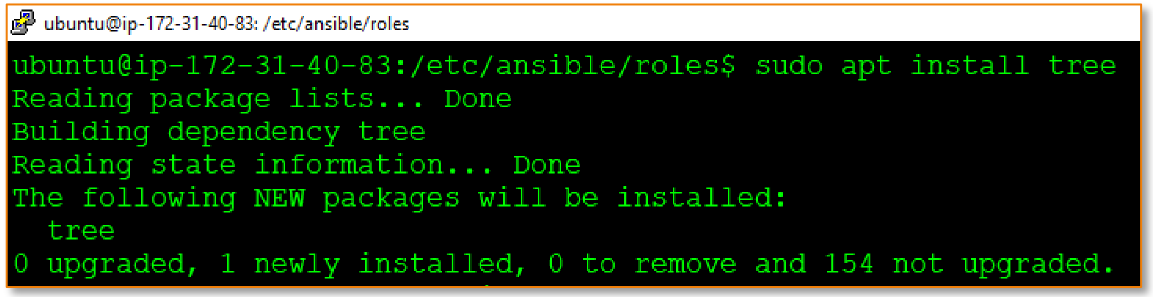

ubuntu@ip-172-31-40-83: /etc/ansible/roles

```
ubuntu@ip-172-31-40-83:~$ cd /etc/ansible/roles/
ubuntu@ip-172-31-40-83:/etc/ansible/roles$ ansible-galaxy init apache --offline
```

Creating an Ansible Role

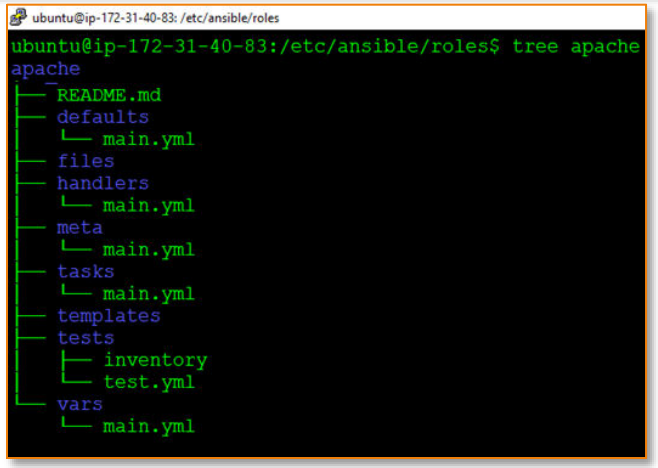
2 Install tree package using *sudo apt install tree*. Use tree command to view structure of the role

★ Use *tree <role name>* to see the role structure



ubuntu@ip-172-31-40-83: /etc/ansible/roles

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles$ sudo apt install tree
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 154 not upgraded.
```



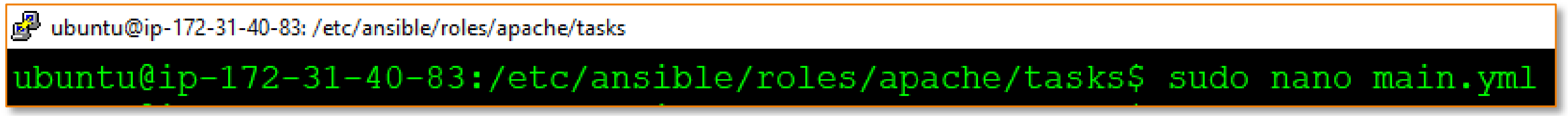
ubuntu@ip-172-31-40-83: /etc/ansible/roles

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles$ tree apache
apache
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

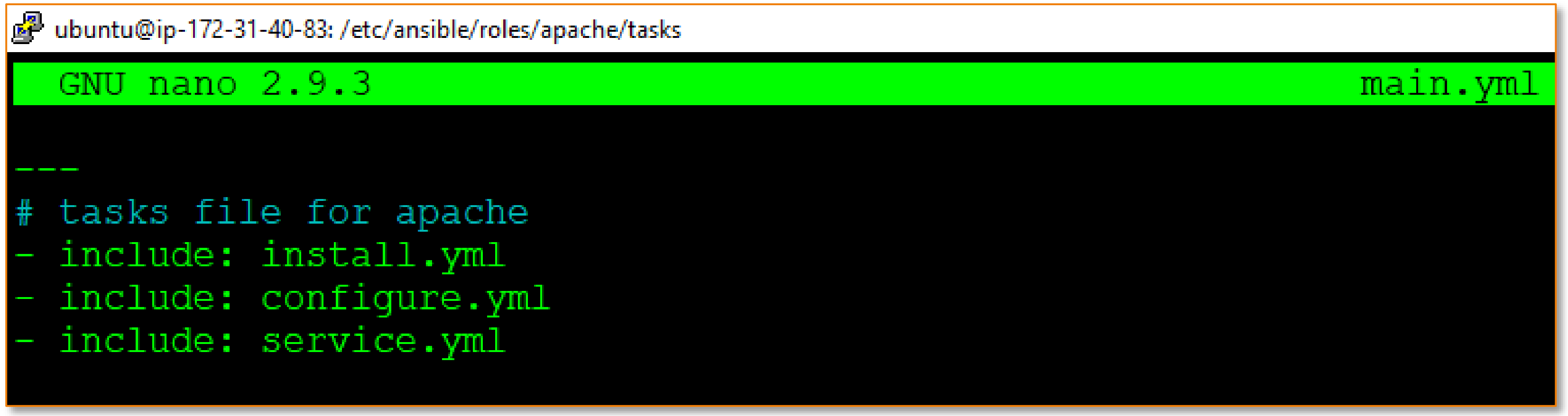
Creating an Ansible Role

3 Go inside task folder inside apache directory. Edit **main.yml** using *sudo nano main.yml*. Make changes as shown. Save and then exit.

★ Keeping install, configure and service files separately helps us reduce complexity.



```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/tasks$ sudo nano main.yml
```

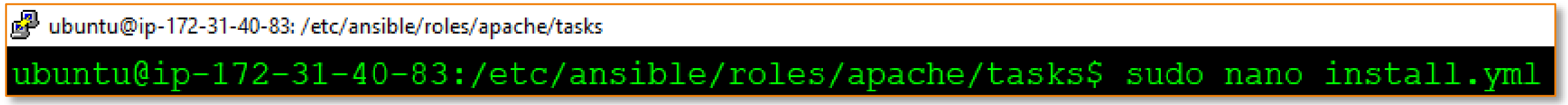


```
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks
  GNU nano 2.9.3                                              main.yml

---
# tasks file for apache
- include: install.yml
- include: configure.yml
- include: service.yml
```

Creating an Ansible Role

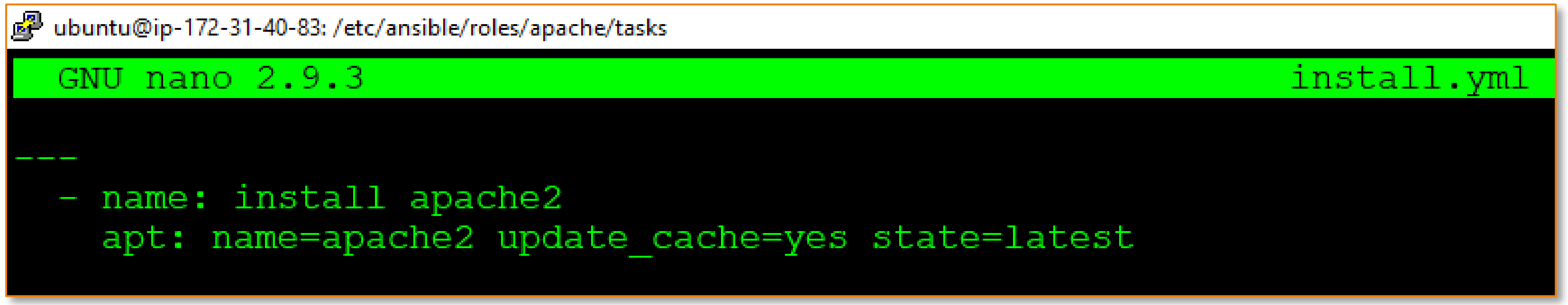
4 Create **install.yml**, **configure.yml** and **service.yml** to include in the **main.yml**

★ To install apache2 in the remote machine



ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/tasks$ sudo nano install.yml
```



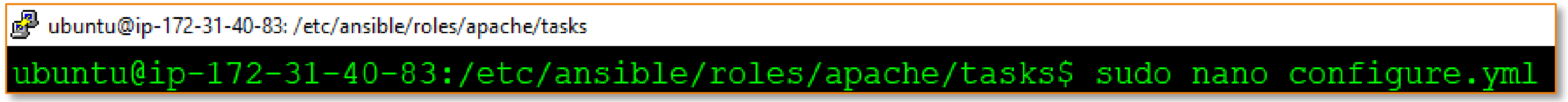
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
  GNU nano 2.9.3                                                   install.yml

---
  - name: install apache2
    apt: name=apache2 update_cache=yes state=latest
```

Creating an Ansible Role

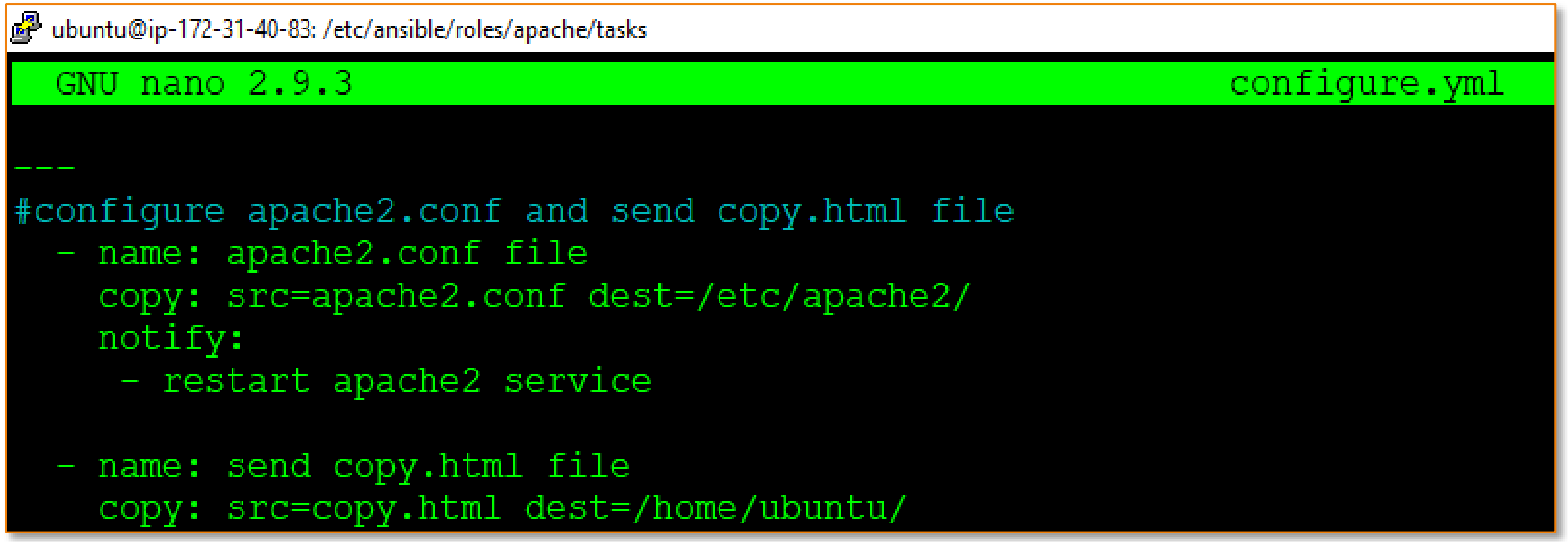
4 Create **install.yml**, **configure.yml** and **service.yml** to include in the **main.yml**

To configure the apache2.conf file and to send copy.html file to the remote machine. Add notify too, based on which handlers will get triggered



ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/tasks$ sudo nano configure.yml
```



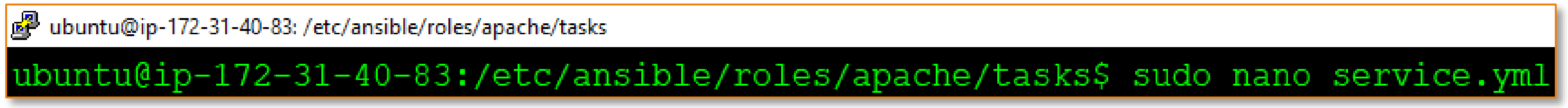
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
  GNU nano 2.9.3                                                    configure.yml

---
#configure apache2.conf and send copy.html file
  - name: apache2.conf file
    copy: src=apache2.conf dest=/etc/apache2/
    notify:
     - restart apache2 service

  - name: send copy.html file
    copy: src=copy.html dest=/home/ubuntu/
```

Creating an Ansible Role

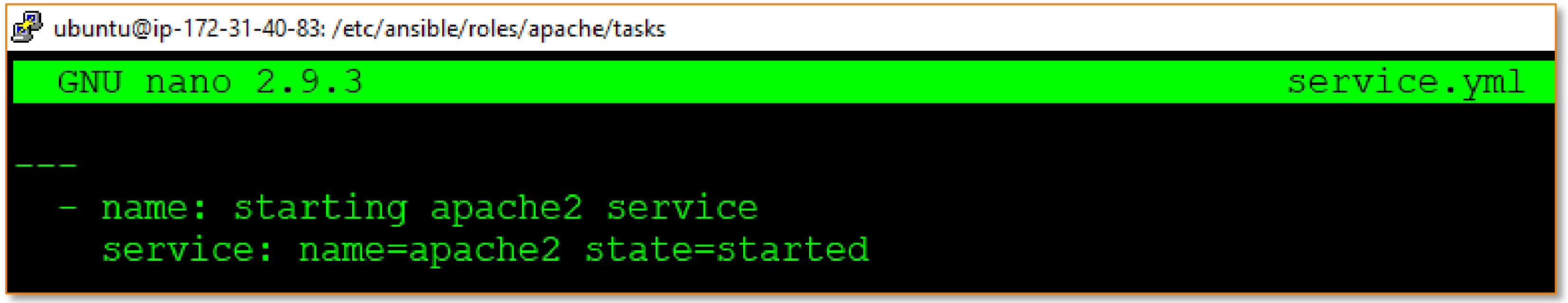
4 Create **install.yml**, **configure.yml** and **service.yml** to include in the **main.yml**

To start apache2 service in the remote machine



ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/tasks$ sudo nano service.yml
```



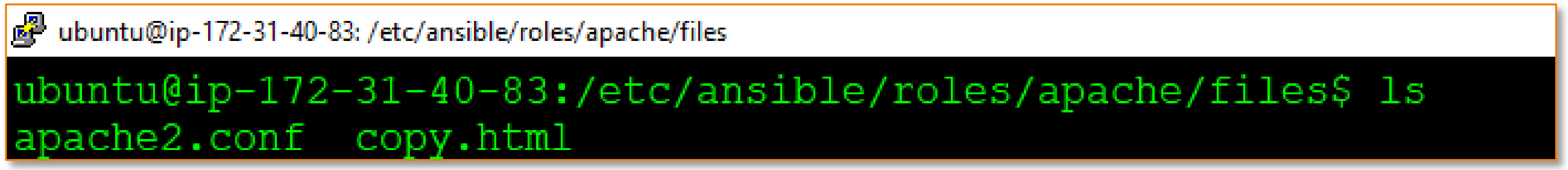
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
  GNU nano 2.9.3                                                        service.yml

---
  - name: starting apache2 service
    service: name=apache2 state=started
```

Creating an Ansible Role

5 Now go inside files. Store the files that needs to be pushed to the remote machine

★ Copy the apache2.conf file and create one html file

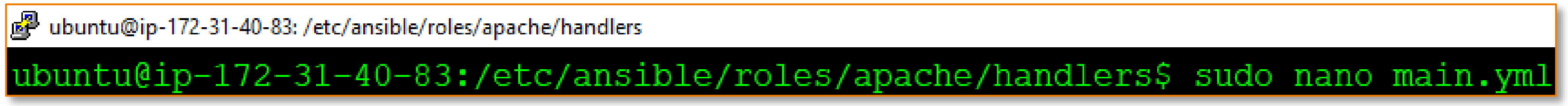

ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/files

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/files$ ls
apache2.conf  copy.html
```

Creating an Ansible Role

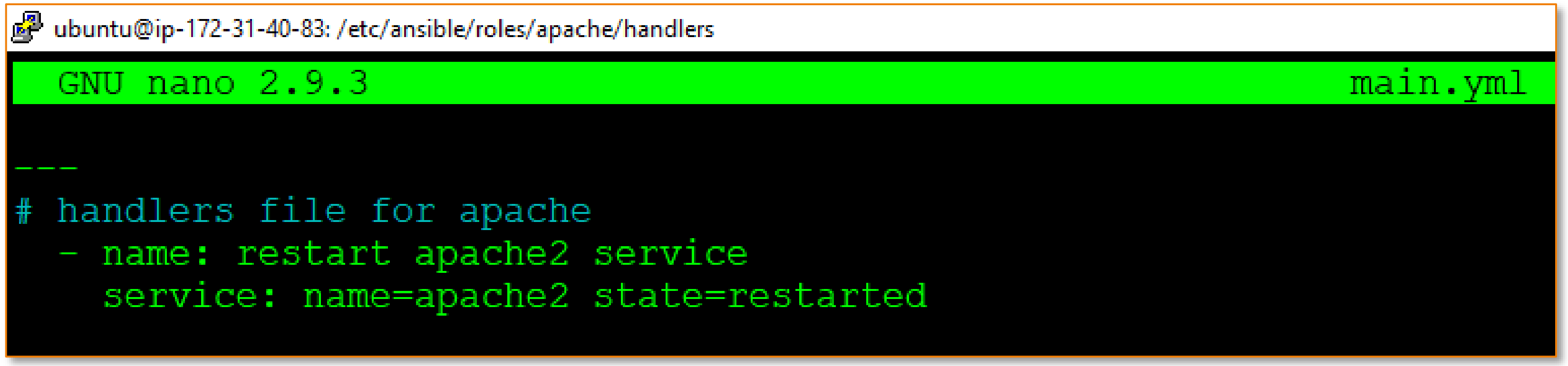
6 Go inside handlers and add the action that needs to be performed after notify from configure.yml is executed.

Once the notify gets executed restart the apache2 service



ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/handlers

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/handlers$ sudo nano main.yml
```

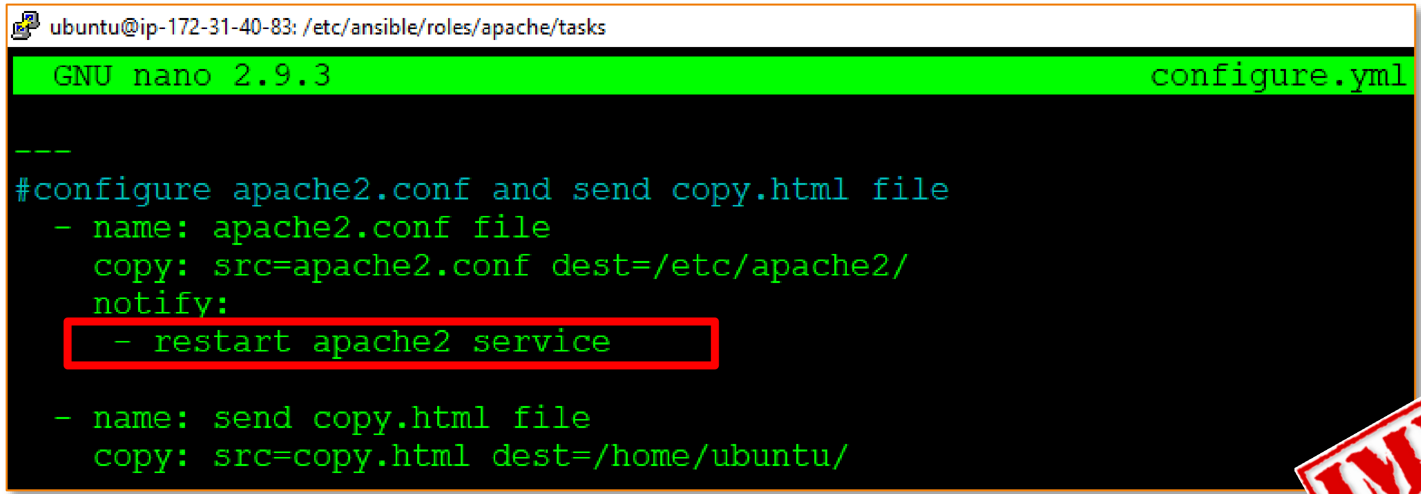


ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/handlers

```
  GNU nano 2.9.3                                                    main.yml

---
# handlers file for apache
  - name: restart apache2 service
    service: name=apache2 state=restarted
```

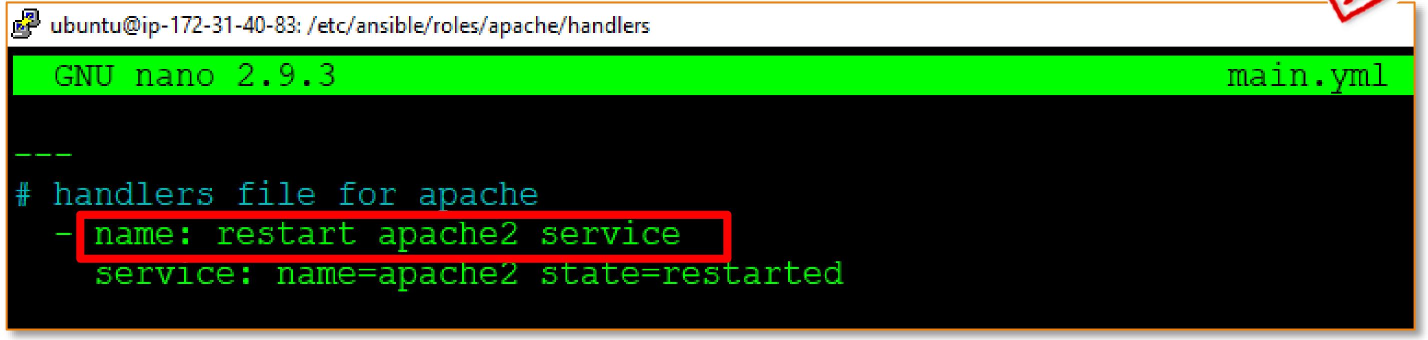
Creating an Ansible Role

Remember that notify name and handler name should match.



ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/tasks

```
  GNU nano 2.9.3                                              configure.yml

---
#configure apache2.conf and send copy.html file
  - name: apache2.conf file
    copy: src=apache2.conf dest=/etc/apache2/
    notify:
      - restart apache2 service

  - name: send copy.html file
    copy: src=copy.html dest=/home/ubuntu/
```



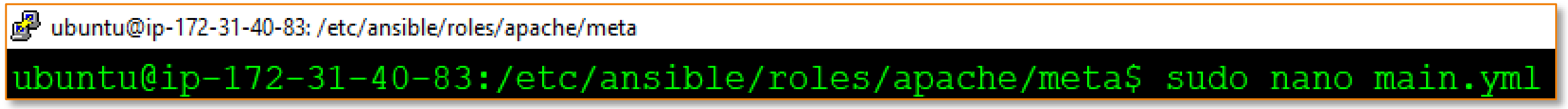
ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/handlers

```
  GNU nano 2.9.3                                                  main.yml

---
# handlers file for apache
  - name: restart apache2 service
    service: name=apache2 state=restarted
```

Creating an Ansible Role

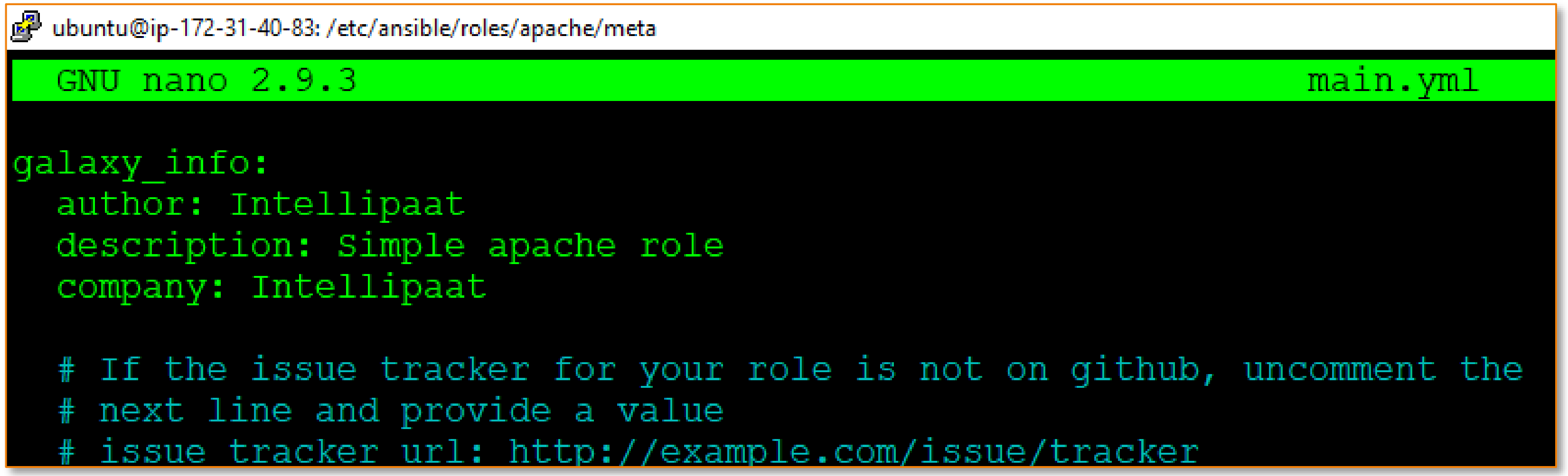
7 Go inside meta and add information related to the role

★ Add author information, role descriptions, company information etc.



ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/meta

```
ubuntu@ip-172-31-40-83:/etc/ansible/roles/apache/meta$ sudo nano main.yml
```

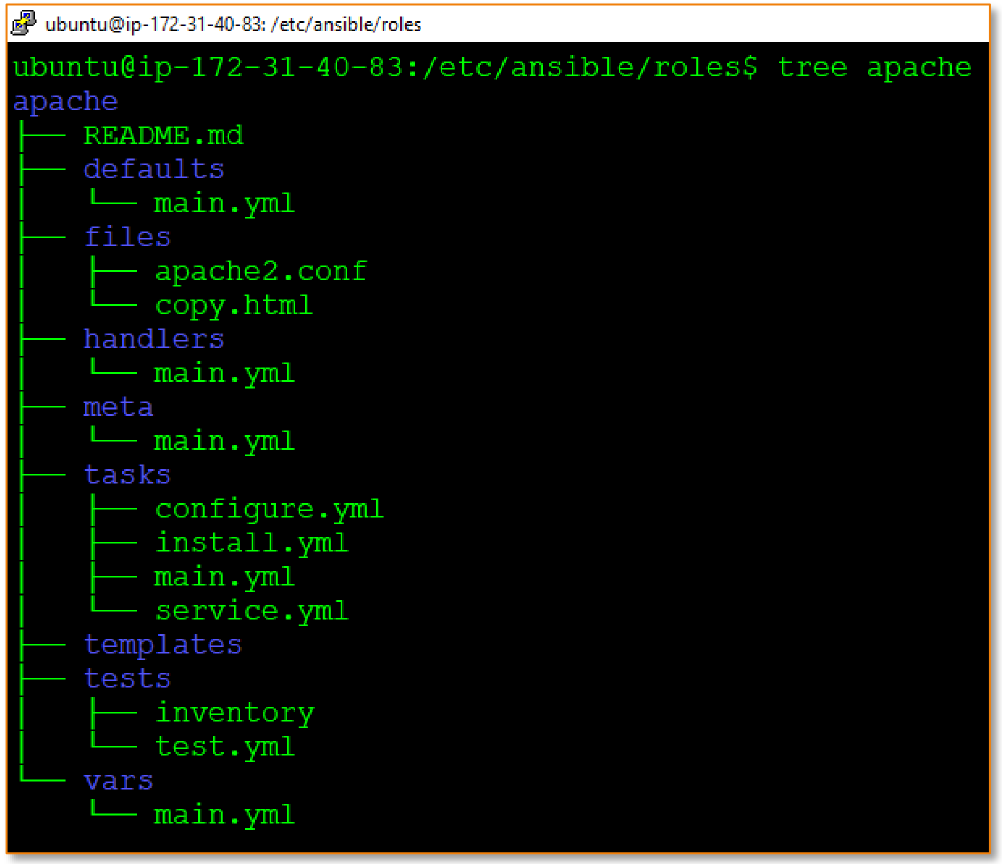


ubuntu@ip-172-31-40-83: /etc/ansible/roles/apache/meta

```
  GNU nano 2.9.3                                                main.yml

galaxy_info:
  author: Intellipaat
  description: Simple apache role
  company: Intellipaat

  # If the issue tracker for your role is not on github, uncomment the
  # next line and provide a value
  # issue_tracker_url: http://example.com/issue/tracker
```

Creating an Ansible Role

Structure of the role after adding all the required files

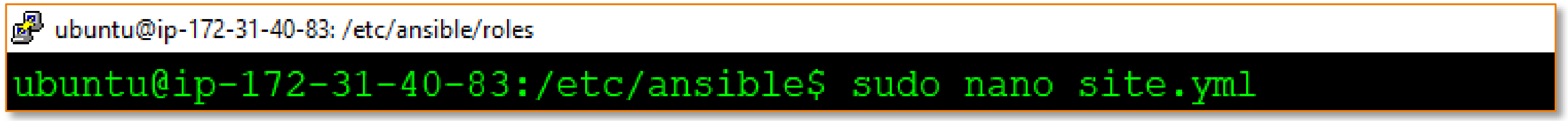


```
ubuntu@ip-172-31-40-83:/etc/ansible/roles$ tree apache
apache
├── README.md
├── defaults
│   └── main.yml
├── files
│   ├── apache2.conf
│   └── copy.html
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   ├── configure.yml
│   ├── install.yml
│   ├── main.yml
│   └── service.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Creating an Ansible Role

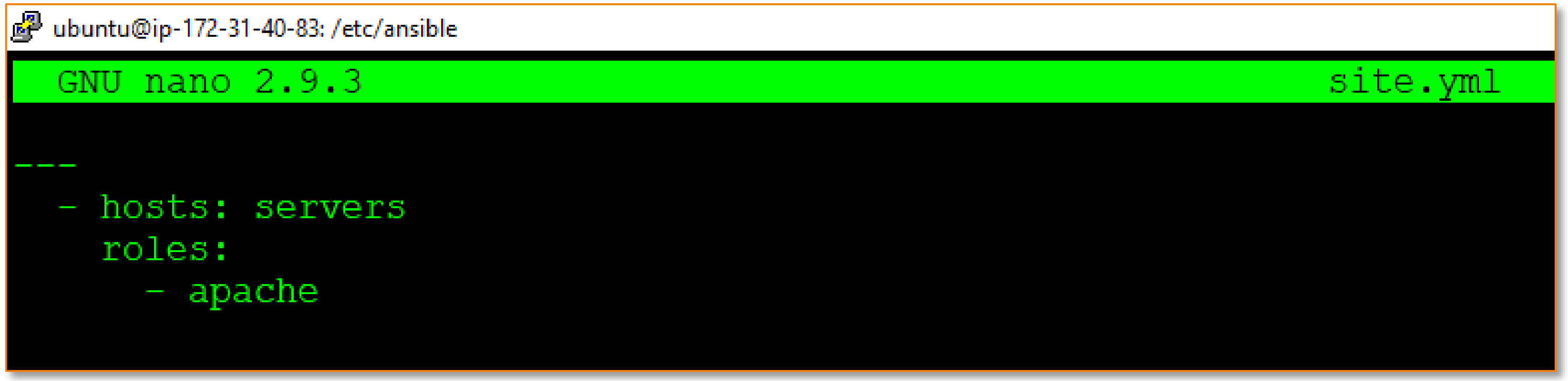
8 Go to the */etc/ansible/* and create one top level file where we can add hosts and roles to be executed

★ Execute *apache role* on the hosts that is under the group name *servers,* added in the inventory file */etc/ansible/hosts*



ubuntu@ip-172-31-40-83: /etc/ansible/roles

```
ubuntu@ip-172-31-40-83:/etc/ansible$ sudo nano site.yml
```



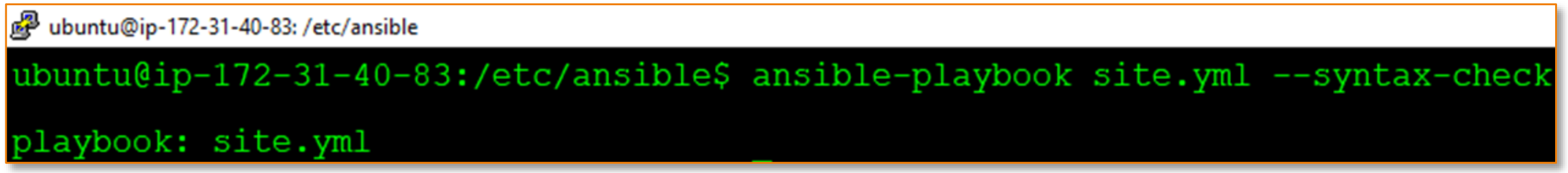
ubuntu@ip-172-31-40-83: /etc/ansible

```
  GNU nano 2.9.3                                              site.yml

---
  - hosts: servers
    roles:
      - apache
```

Creating an Ansible Role

9 Before we execute our top level yml file we will check for syntax errors.

★ Use ansible-playbook <filename.yml> --syntax-check



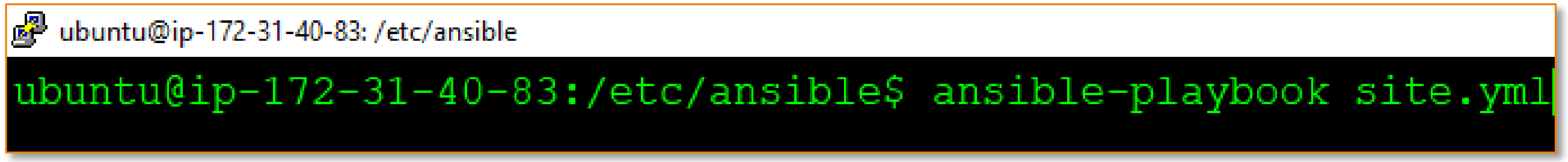
ubuntu@ip-172-31-40-83: /etc/ansible

```
ubuntu@ip-172-31-40-83:/etc/ansible$ ansible-playbook site.yml --syntax-check

playbook: site.yml
```

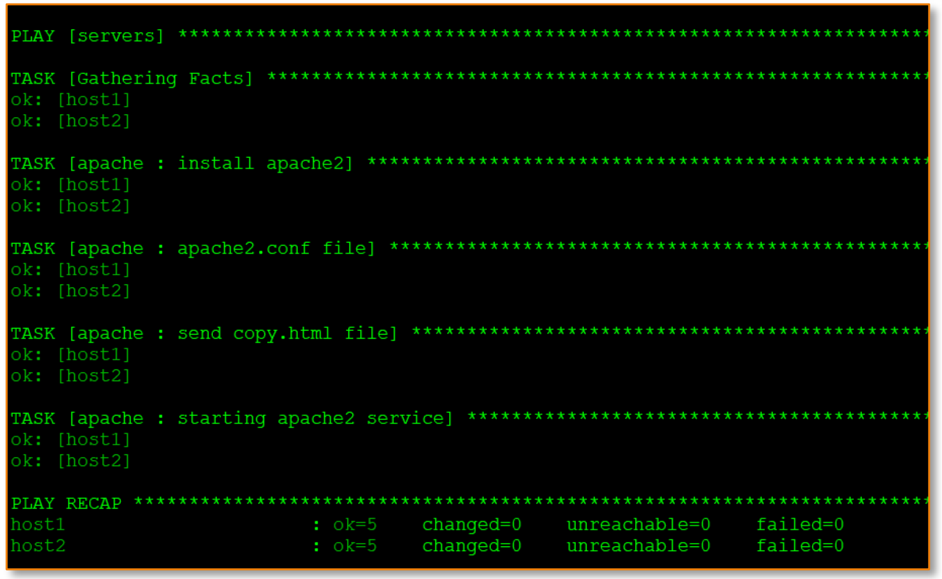
Creating an Ansible Role

10 Execute the top level yml file

★ Use ansible-playbook <filename.yml>


ubuntu@ip-172-31-40-83: /etc/ansible

```
ubuntu@ip-172-31-40-83:/etc/ansible$ ansible-playbook site.yml
```



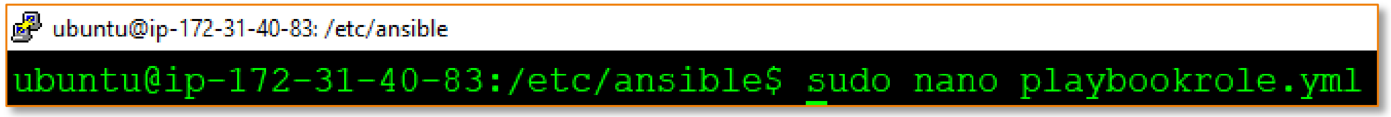
```
PLAY [servers] ********************************************************************

TASK [Gathering Facts] *************************************************************
ok: [host1]
ok: [host2]

TASK [apache : install apache2] ****************************************************
ok: [host1]
ok: [host2]

TASK [apache : apache2.conf file] **************************************************
ok: [host1]
ok: [host2]

TASK [apache : send copy.html file] ************************************************
ok: [host1]
ok: [host2]

TASK [apache : starting apache2 service] *******************************************
ok: [host1]
ok: [host2]

PLAY RECAP *************************************************************************
host1                          : ok=5    changed=0    unreachable=0    failed=0
host2                          : ok=5    changed=0    unreachable=0    failed=0
```

Using Roles in Playbook

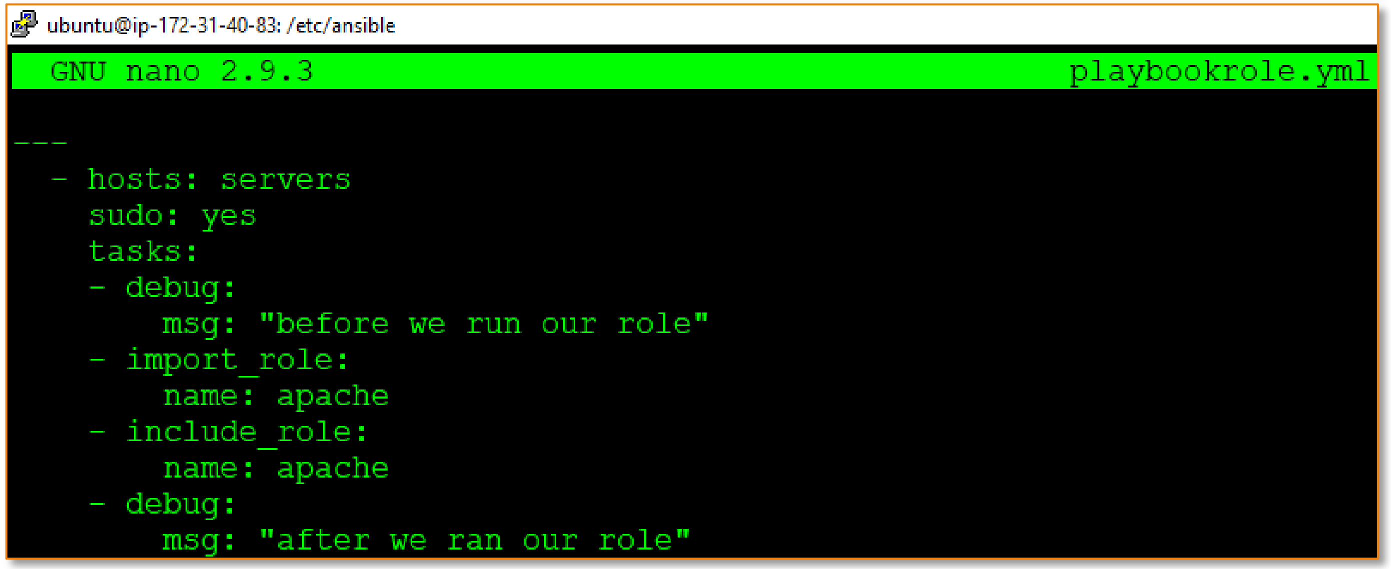
Using Roles in Playbook

To use ansible roles along with other tasks in playbook Use *import_role* and *include_role.*

Here we have created one playbook called *playbookrole.yml* to execute on *servers* along with two *debug* tasks before and after *apache* role.



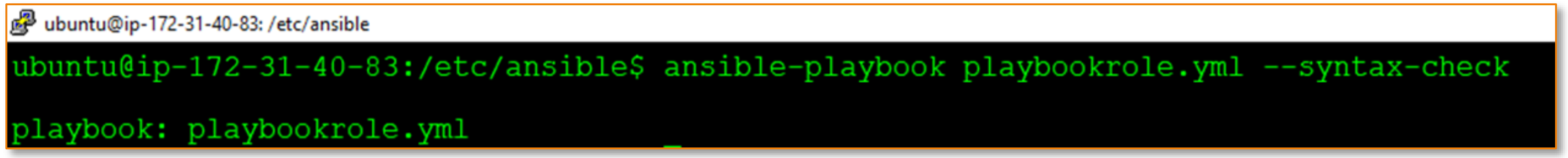
```
ubuntu@ip-172-31-40-83: /etc/ansible
ubuntu@ip-172-31-40-83:/etc/ansible$ sudo nano playbookrole.yml
```



```
ubuntu@ip-172-31-40-83: /etc/ansible
 GNU nano 2.9.3                                              playbookrole.yml

---
 - hosts: servers
   sudo: yes
   tasks:
   - debug:
       msg: "before we run our role"
   - import_role:
       name: apache
   - include_role:
       name: apache
   - debug:
       msg: "after we ran our role"
```

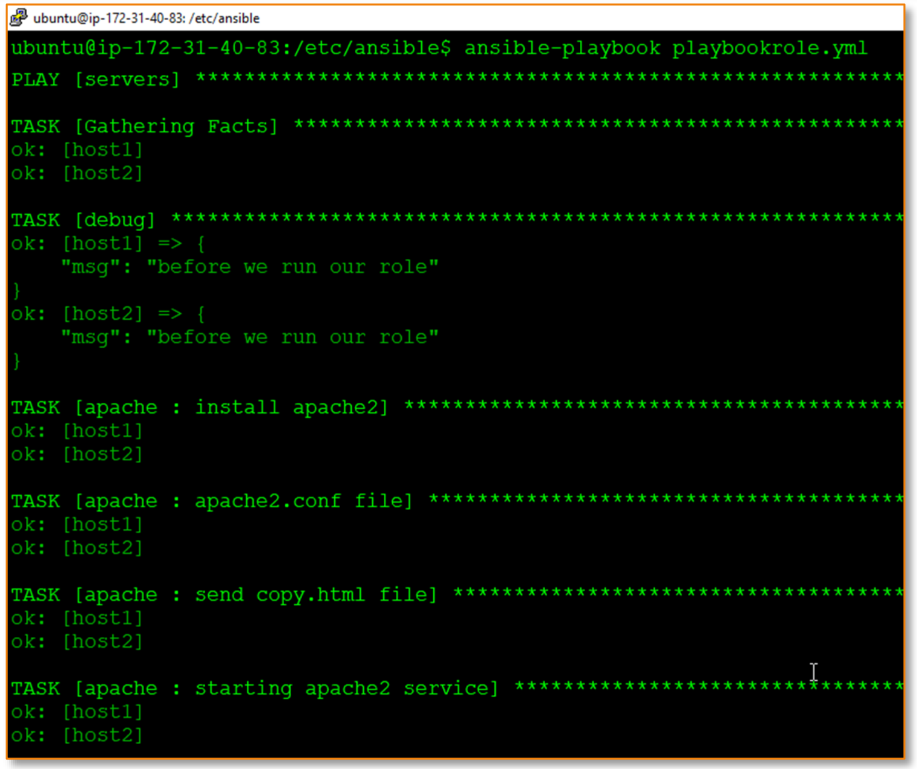
Using Roles in Playbook

Check for syntax error and execute the playbook with roles.



ubuntu@ip-172-31-40-83: /etc/ansible

```
ubuntu@ip-172-31-40-83:/etc/ansible$ ansible-playbook playbookrole.yml --syntax-check

playbook: playbookrole.yml
```



ubuntu@ip-172-31-40-83: /etc/ansible

```
ubuntu@ip-172-31-40-83:/etc/ansible$ ansible-playbook playbookrole.yml
PLAY [servers] ****************************************************************

TASK [Gathering Facts] ********************************************************
ok: [host1]
ok: [host2]

TASK [debug] ******************************************************************
ok: [host1] => {
    "msg": "before we run our role"
}
ok: [host2] => {
    "msg": "before we run our role"
}

TASK [apache : install apache2] ***********************************************
ok: [host1]
ok: [host2]

TASK [apache : apache2.conf file] *********************************************
ok: [host1]
ok: [host2]

TASK [apache : send copy.html file] *******************************************
ok: [host1]
ok: [host2]

TASK [apache : starting apache2 service] **************************************
ok: [host1]
ok: [host2]
```

Hands-on: Configuring Multiple Nodes using Ansible