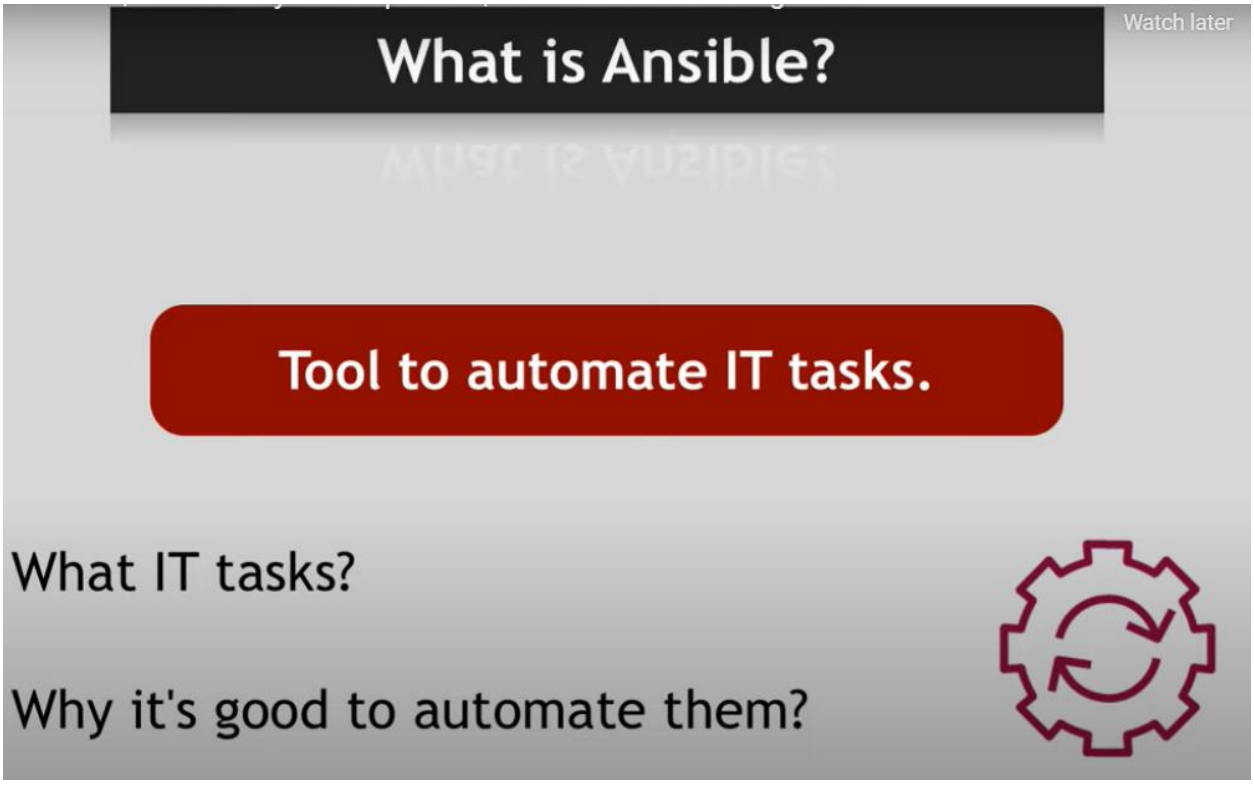
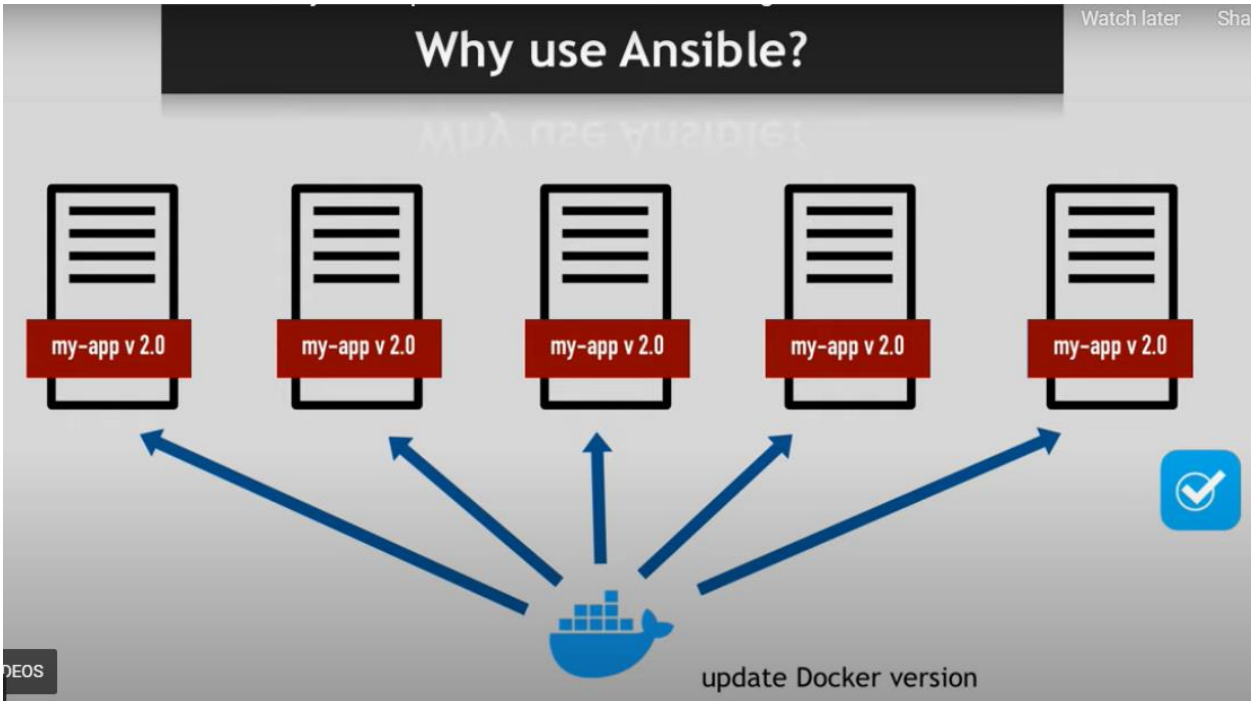
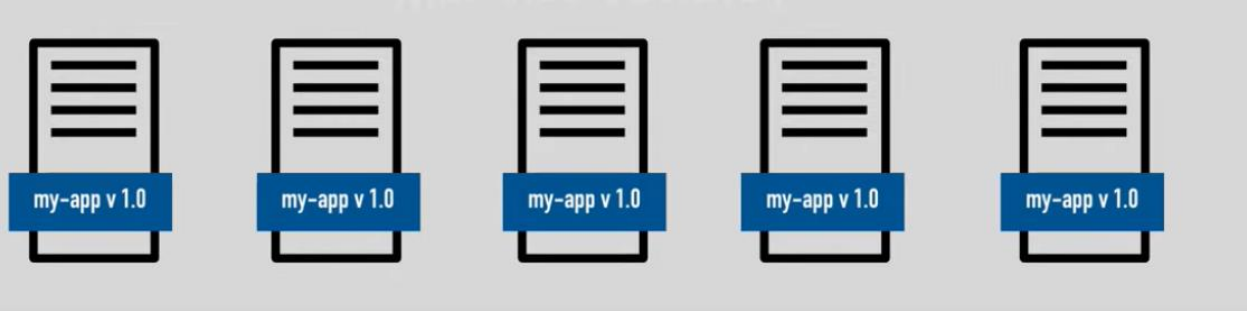


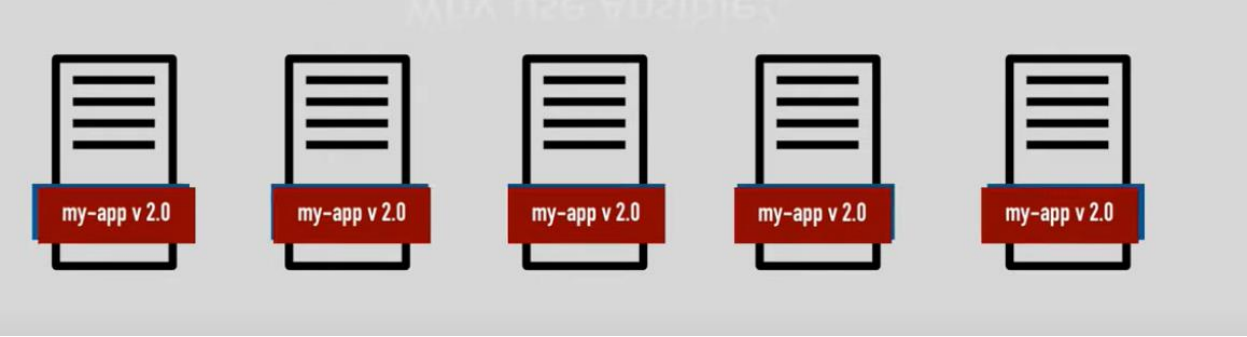
Ansible is a tool that plays an important role in automating the IT tasks.

It is important to understand the type pf IT tasks that are automated by ansible and also why it is good to automate them.

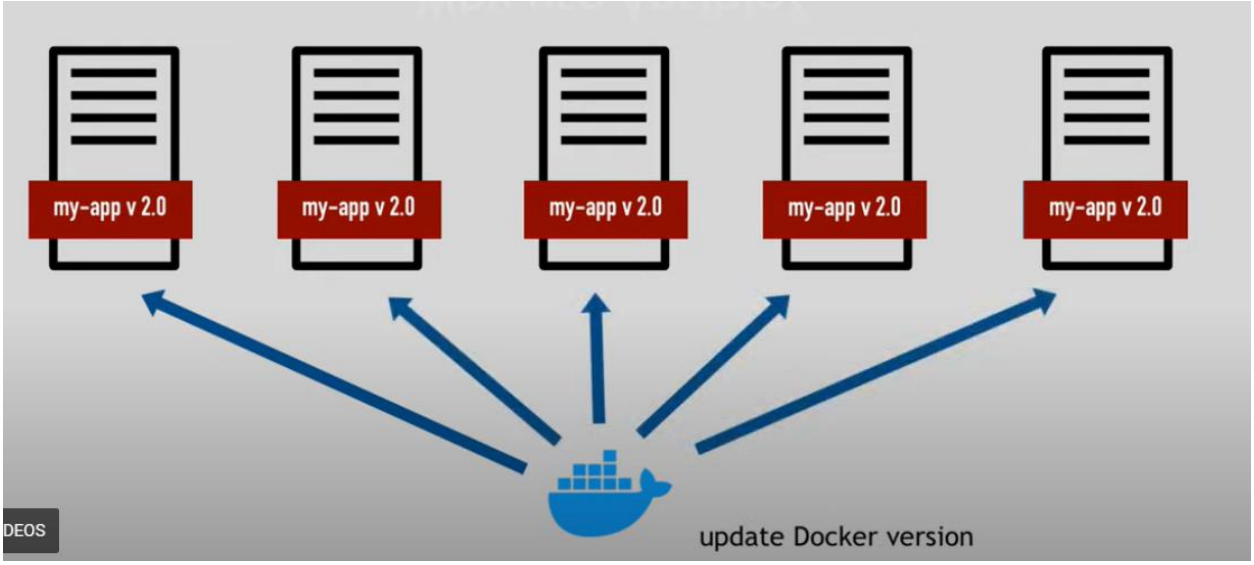




Imagine a scenario where you are running 10 different servers that have application version 1.0 running in all of them.

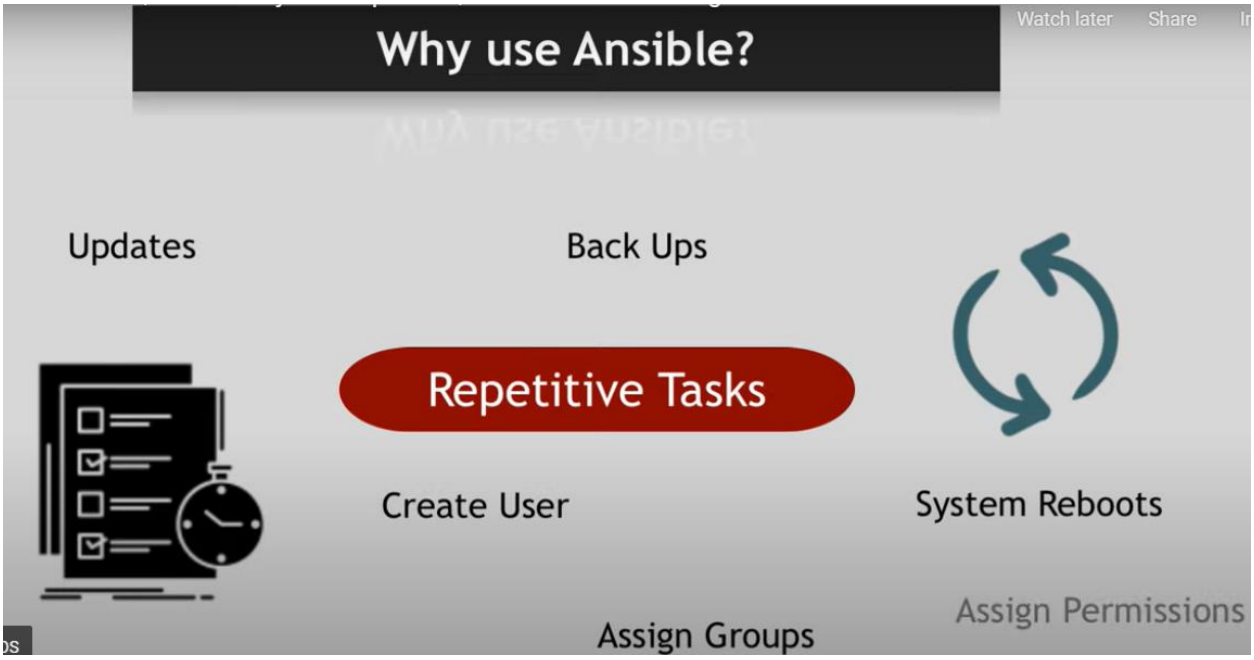


After six months your company releases another version of the application 2.0 and which mut be deployed in all the ten servers.



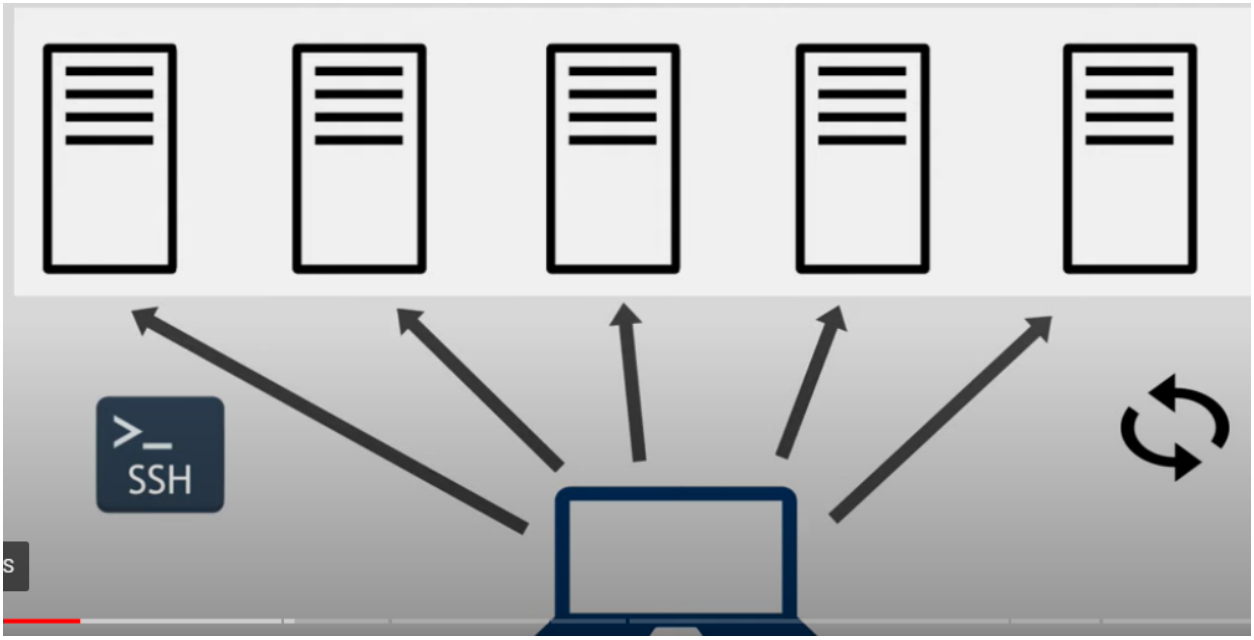
In addition to deploying the new version of the application 2.0, the manager also wants your team to update the docker version in all the 10 servers.

All these tasks can be automated through the use of ansible, and this is why we say ansible is a configuration, management and deploying tool.



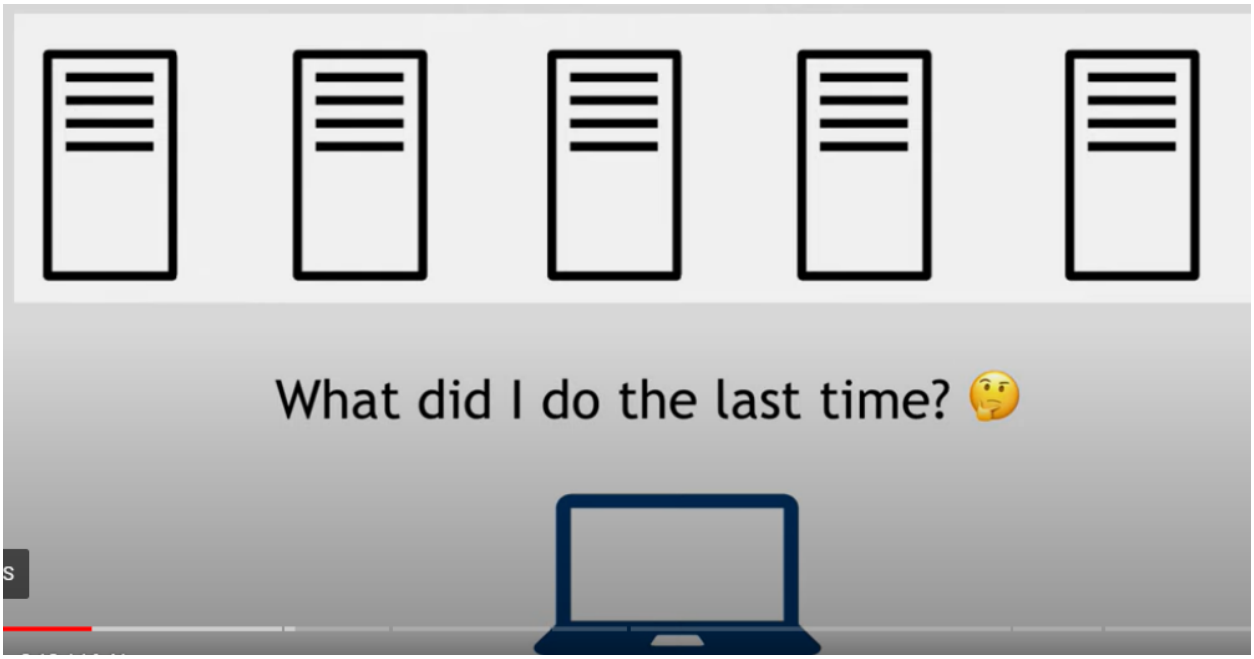
Ansible is also used to carry out all the repetitive tasks in the company.

Updates, backups, system reboots, creating user, creating groups, assigning permission to users an d groups

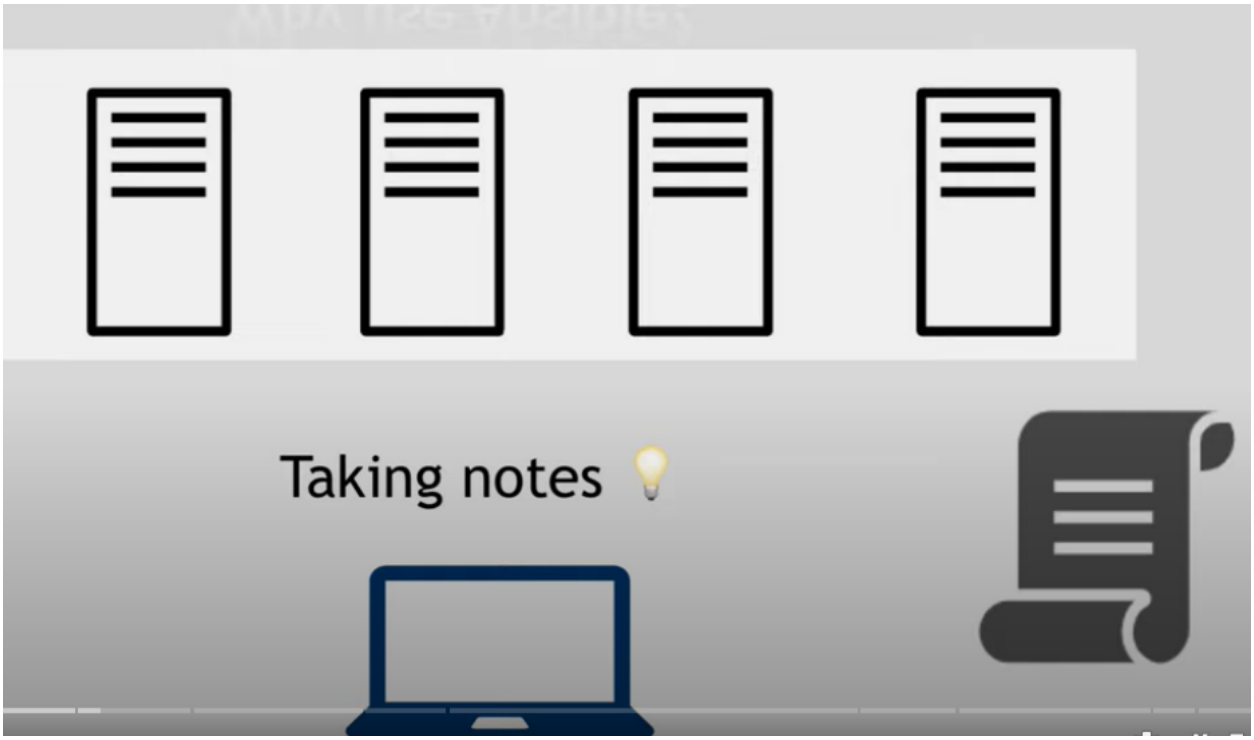


Before ansible came into existence, the whole process of updating the version 2.0 will be done manually.

This means the team will have to ssh to each server through the ansible server and manually deploy the version 2.0 on each of them until they are done with all 10 servers.



It is even more difficult especially when the job involves multiple steps and tasks because you always must remember what you did last time.



One approach that employees used to solve this problem is through taking notes and documenting all the steps and the updates in each server.

They also had to document all the steps used in deploying the updates.

For instance, imagine deploying LAMP stack in 10 centos servers manually. The team must document the following script and must write down all the steps and at the same time document all the updates in each server.

```
#!/bin/sh

        ########################################
        # Bash script to install an LAMP stack in CentOS
        # Author: Subhash (serverkaka.com)

        # Check if running as root
        if [ "$(id -u)" != "0" ]; then
           echo "This script must be run as root" 1>&2
           exit 1
        fi

        # Ask value for mysql root password
        read -p 'db_root_password [secretpasswd]: ' db_root_password
```

```
echo

# prerequisite
yum install -y wget

# Install APache
yum install -y httpd
systemctl start httpd

# Set apache autostart at system reboot
sudo systemctl enable httpd

# Allow Apache via Firewall
firewall-cmd --permanent --add-service=http
systemctl restart firewalld

# Install PHP
yum install php php-mysql php-pdo php-gd php-mbstring -y

# Install MySql
# Removing previous mysql server installation
systemctl stop mysqld.service && yum remove -y mysql-
community-server && rm -rf /var/lib/mysql && rm -rf
/var/log/mysqld.log && rm -rf /etc/my.cnf

# Installing mysql server (community edition)'
yum localinstall -y https://dev.mysql.com/get/mysql57-community-
release-el7-7.noarch.rpm
yum install -y mysql-community-server

# Starting mysql server (first run)'
systemctl enable mysqld.service
systemctl start mysqld.service
tempRootDBPass="`grep 'temporary.*root@localhost'
/var/log/mysqld.log | tail -n 1 | sed 's/.*root@localhost: //'`"

# Setting up new mysql server root password'
systemctl stop mysqld.service
rm -rf /var/lib/mysql/*logfile*
```

```
wget -O /etc/my.cnf "https://my-
site.com/downloads/mysql/512MB.cnf"
systemctl start mysqld.service
mysqladmin -u root --password="$tempRootDBPass" password
"$db_root_password"
mysql -u root --password="$db_root_password" -e <<-EOSQL
   DELETE FROM mysql.user WHERE User='';
   DROP DATABASE IF EXISTS test;
   DELETE FROM mysql.db WHERE Db='test' OR Db='test\\_%';
   DELETE FROM mysql.user where user != 'mysql.sys';
   CREATE USER 'root'@'%' IDENTIFIED BY
'${mysqlRootPass}';
   GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION;
   FLUSH PRIVILEGES;
EOSQL
systemctl status mysqld.service

# Install PhpMyAdmin
yum install -y epel-release
sudo yum -y install phpmyadmin

# Restart Apache
systemctl restart httpd

echo LAMP server installation completed, you need to configure
PhpMyAdmin for remotely access at
/etc/httpd/conf.d/phpMyAdmin.conf
```

Ansible can be used to make these processes more efficient and less time consuming.

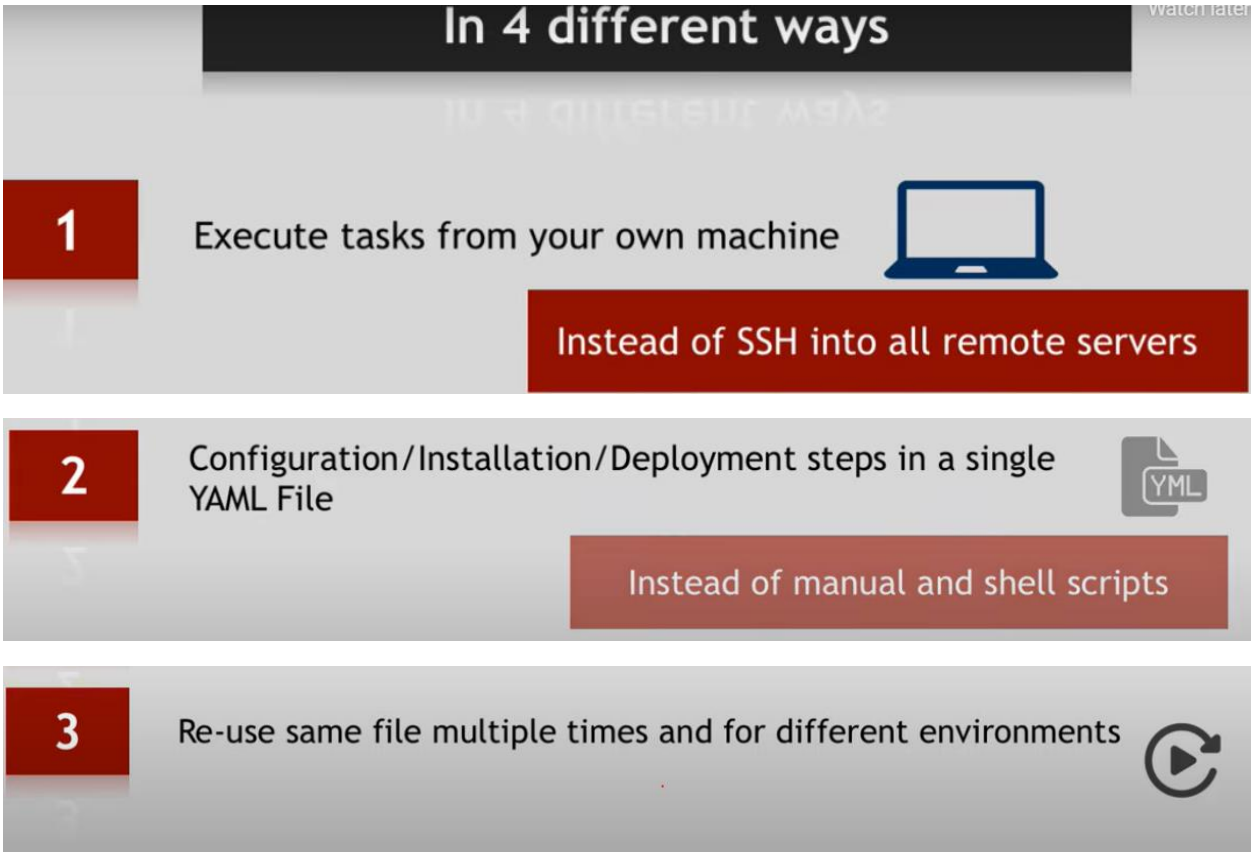
You can accomplish the same above tasks done manually with ansible using a simple playbook.

Let us look at the same process using Ansible

```
# Lamp Stack
 - hosts: localhost
 tasks:
    - name: Install LAMP stack
    become: yes
    apt:
       pkg:
       - apache2
       - mysql-client
       - php7.2
       state: present
       update_cache: yes

    - name: Start Apache2 Service
    become: yes
    service:
       name: apache2
       state: started
       enabled: yes
```

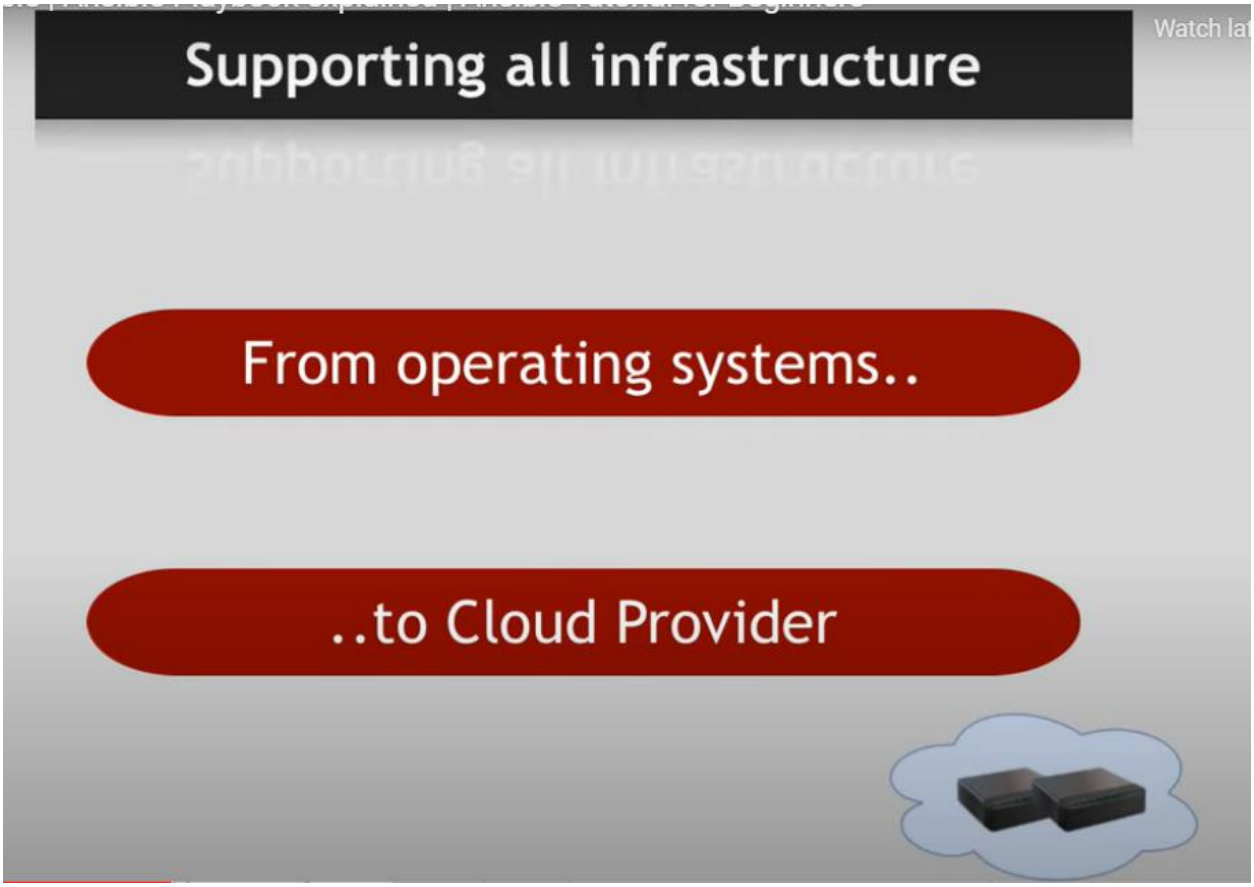
```
    - name: Download and Install Composer
    shell: |
        php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
        sudo php composer-setup.php --install-dir=/usr/local/bin --
filename=composer
        rm composer-setup.php
```

All you need to have is .yml file and an inventory file [contains the information of the servers to be updated]



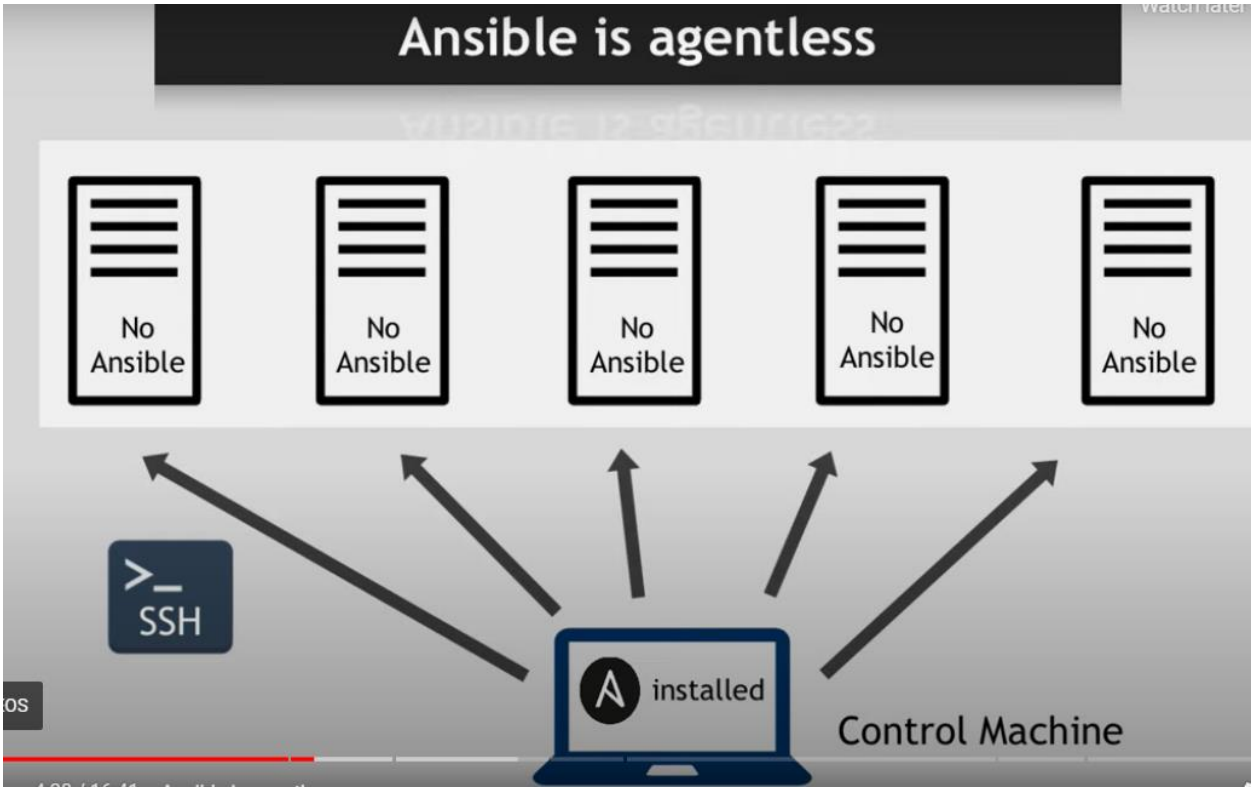
This is important in duplicating the same environment in Dev to production to ensure they both have the same updates and applications which makes the whole Devop Pipeline more efficient because everything run smoothly.

Errors are always going to be present when manually configuring, deploying, and updating the servers.



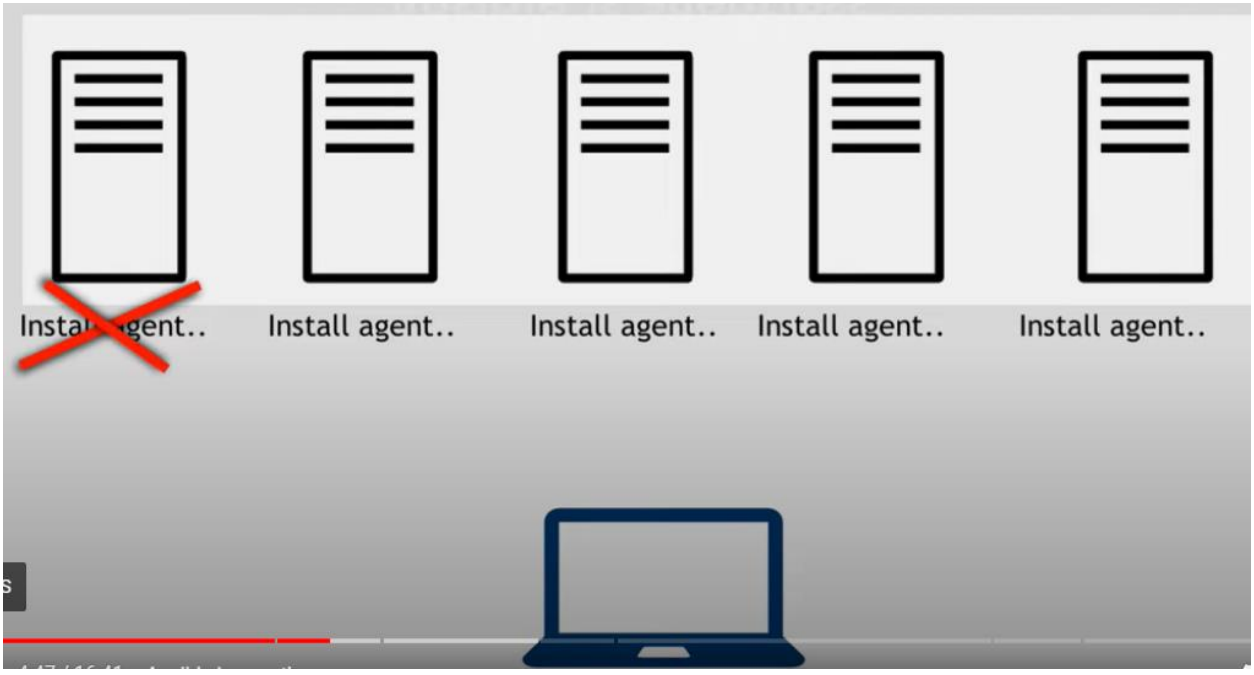
Ansible supports all the infrastructure

You can use ansible to the serves in Premise and in the cloud services such as AWS , Google Cloud, Azure

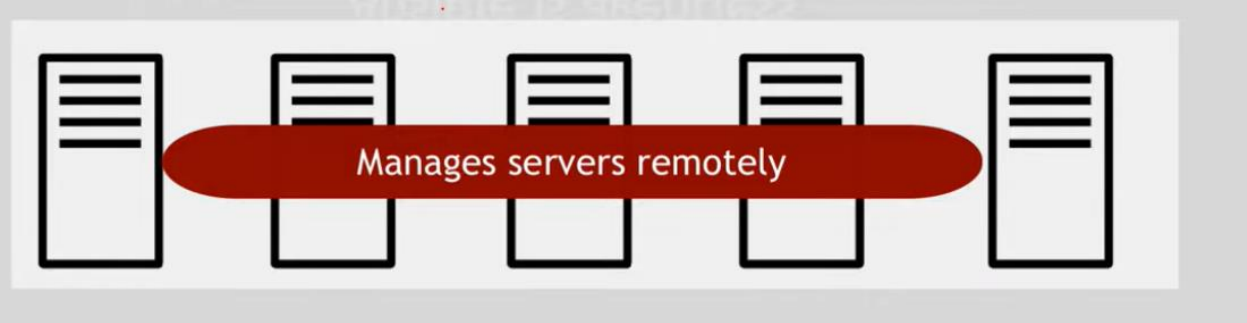


To execute the deployment/configuration file, you just need to simply ssh to the target server from your local machine that has ansible installed/ansible server and the file will be deployed in all the targeted servers.

Ansible is agentless? What does this mean



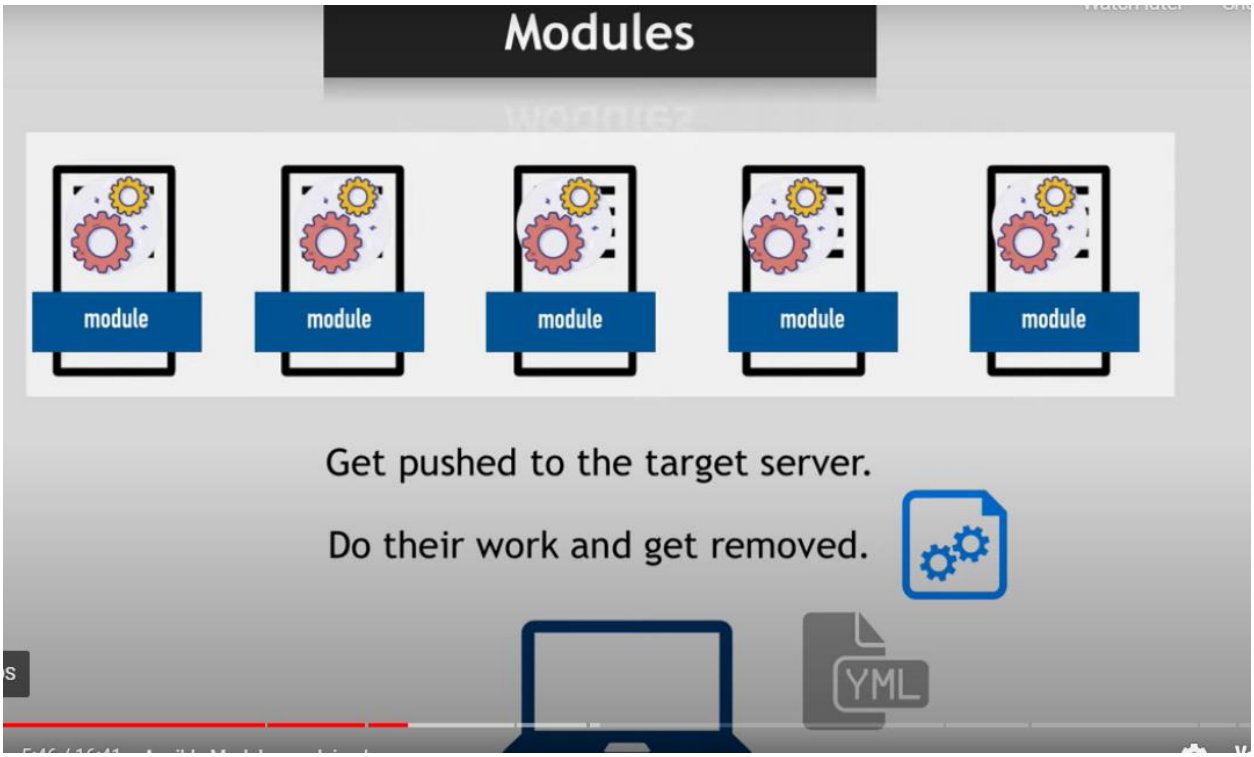
This is the unique characteristic of ansible which is different from other tools. To configure the servers, you always must install the agent for the tool in each server to enhance the process. However, for the ansible, you do not need to have an agent tool for ansible to manage the updates or deploy them into all the servers.

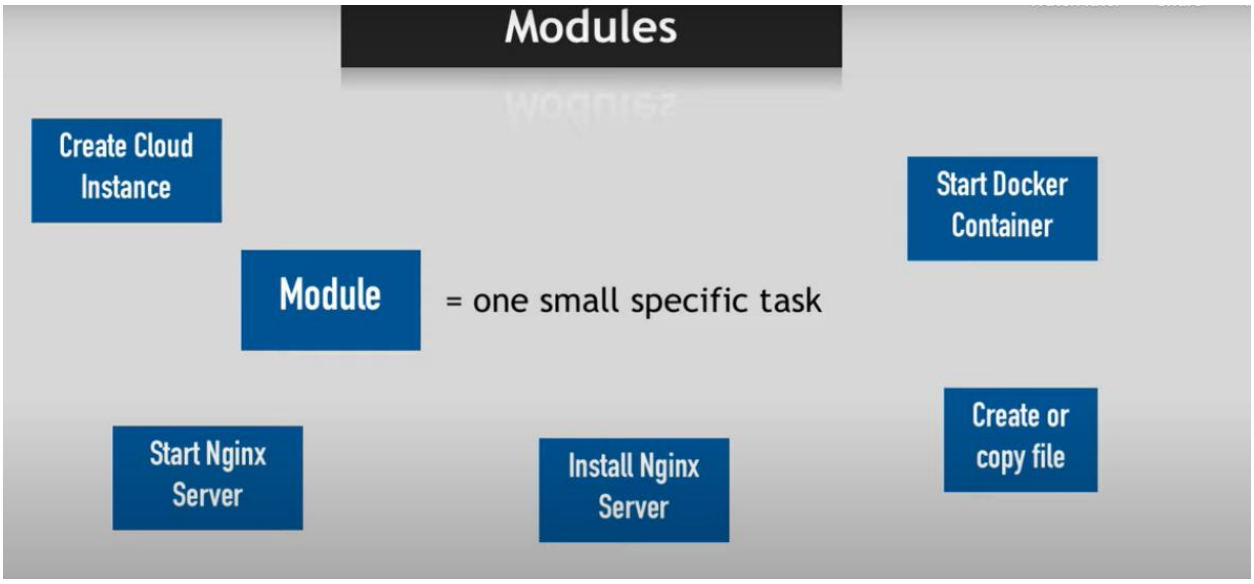


You just need to install the ansible agent in one of the machines, or even your laptop and you can manage the rest of the servers remotely.

or you can use the ansible server to manage all the centos, ubuntu etc servers that need to receive the updates through ssh/remotely.

Ansible works with Modules

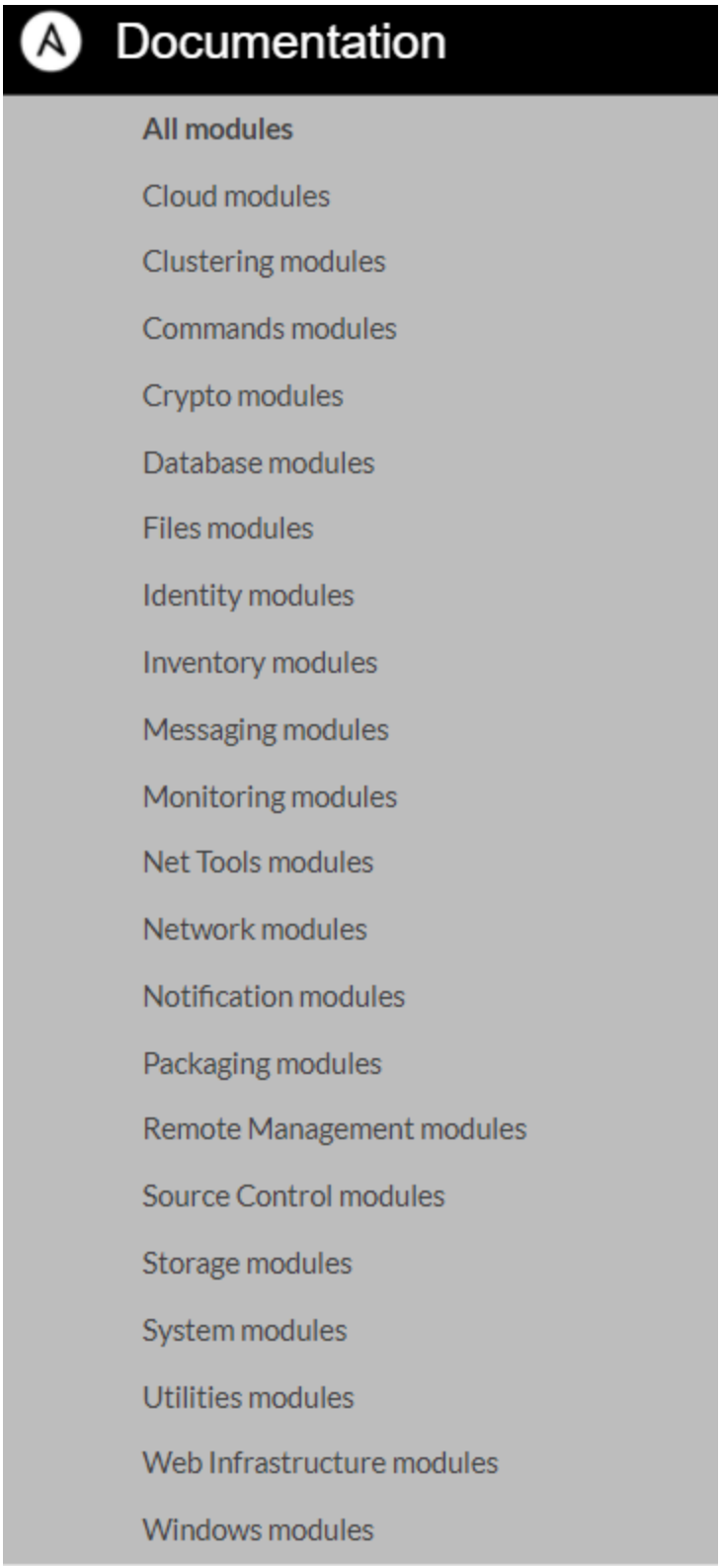


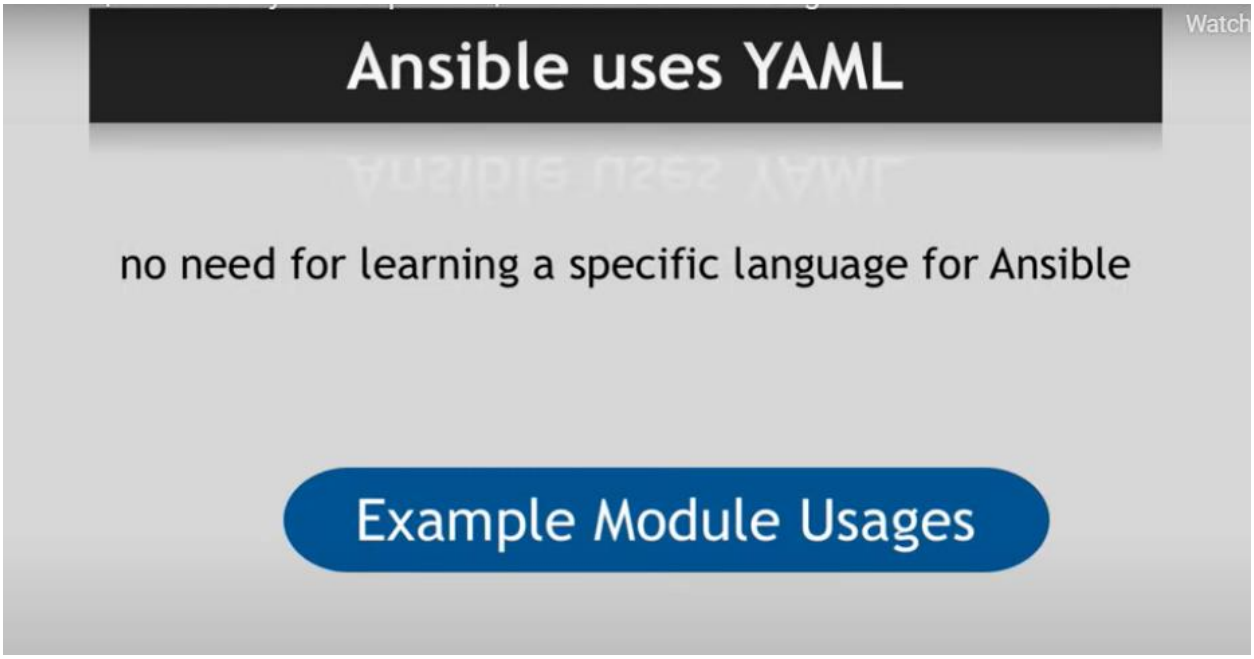


Lamp Stack

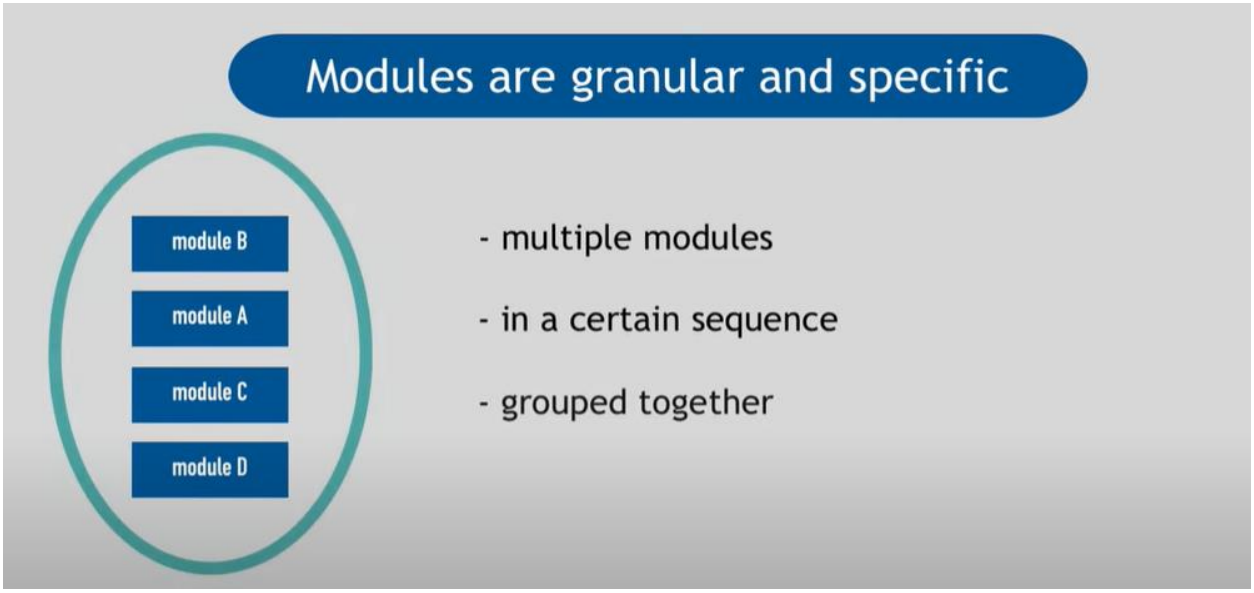
```
- hosts: localhost
tasks
   - name: Install LAMP stack – module [small packages to be installed]
   become: yes
   apt:
      pkg:
      - apache2
      - mysql-client
      - php7.2
      state: present
      update_cache: yes

   - name: Start Apache2 Service
   become: yes
   service:
      name: apache2
      state: started
      enabled: yes
```

https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html


Documentation
All modules
Cloud modules
Clustering modules
Commands modules
Crypto modules
Database modules
Files modules
Identity modules
Inventory modules
Messaging modules
Monitoring modules
Net Tools modules
Network modules
Notification modules
Packaging modules
Remote Management modules
Source Control modules
Storage modules
System modules
Utilities modules
Web Infrastructure modules
Windows modules

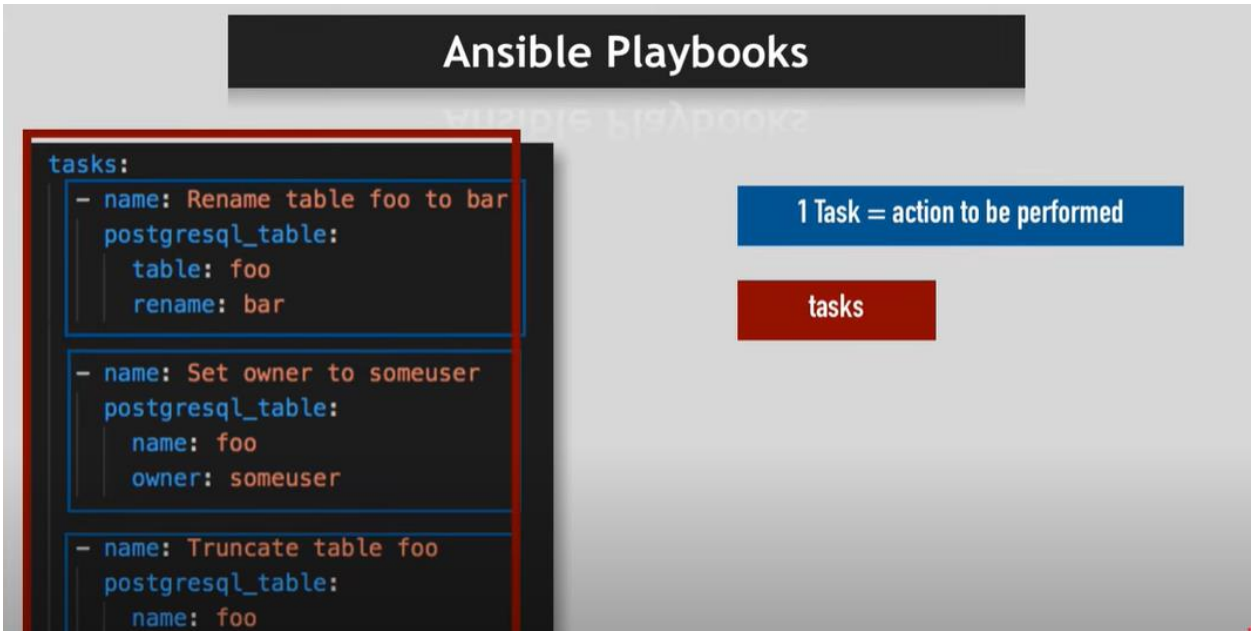


This is just like terraform. No need to learn the usage as the usage is given.

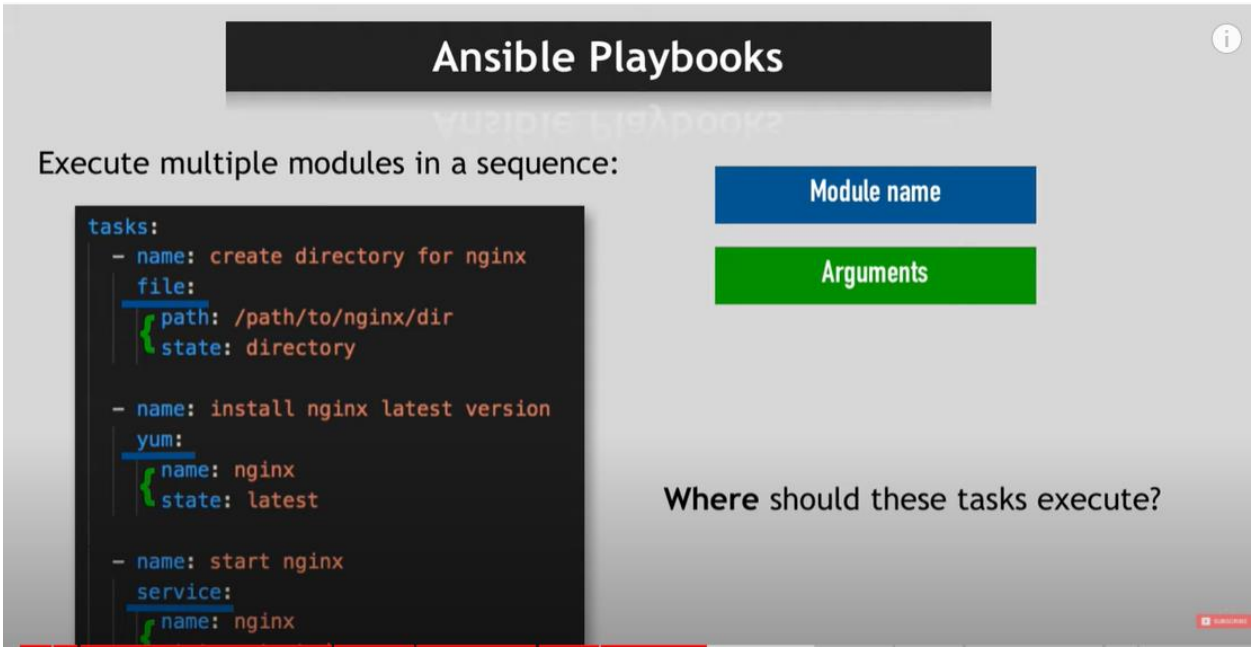


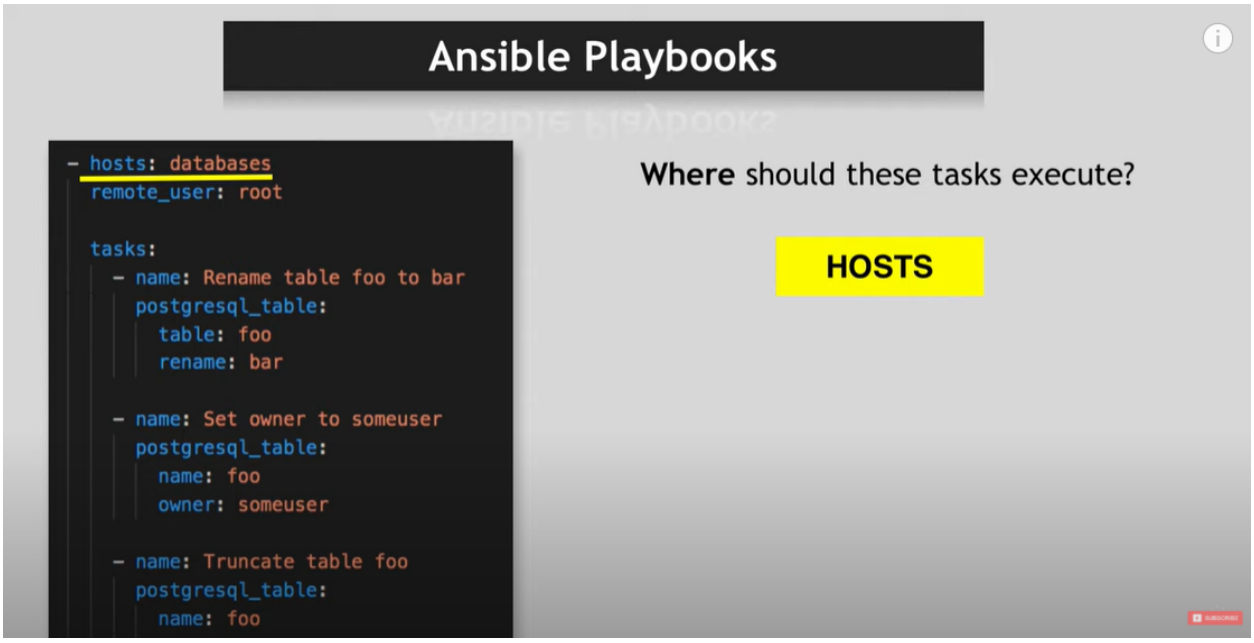
To execute complex tasks, you can include multiple modules, grouped in a specific way to accomplish a certain configuration.

IP Address: 172.17.0.2 and 172.17.0.3



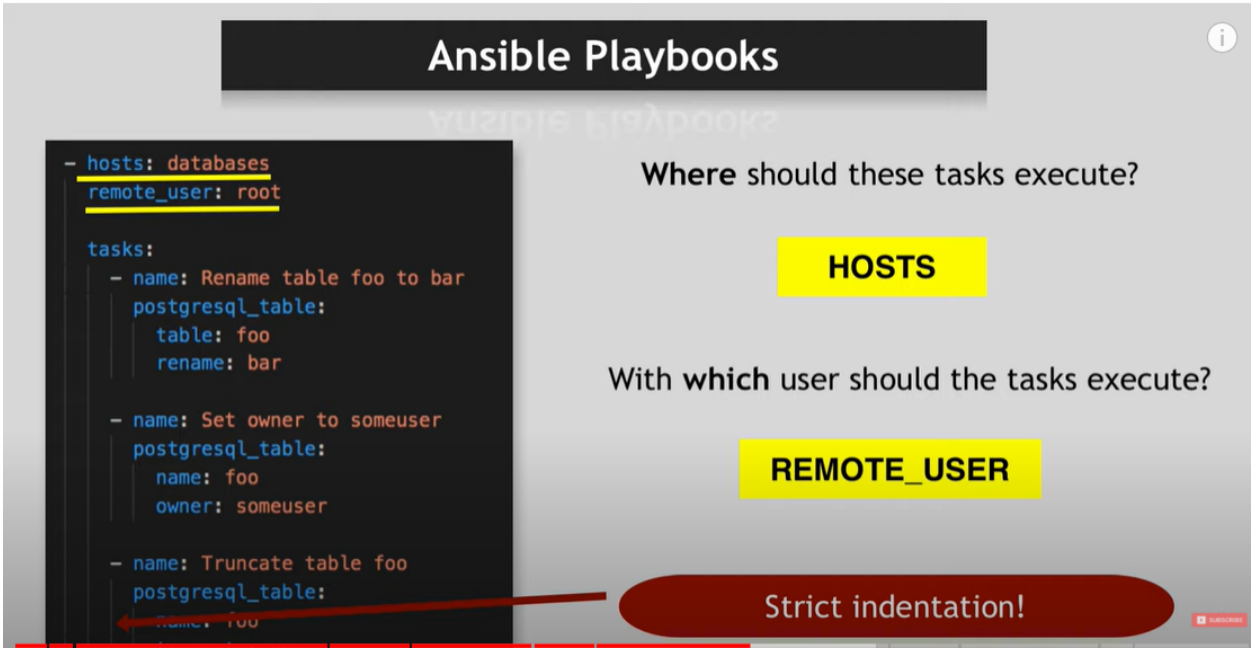
1 task = action to be performed

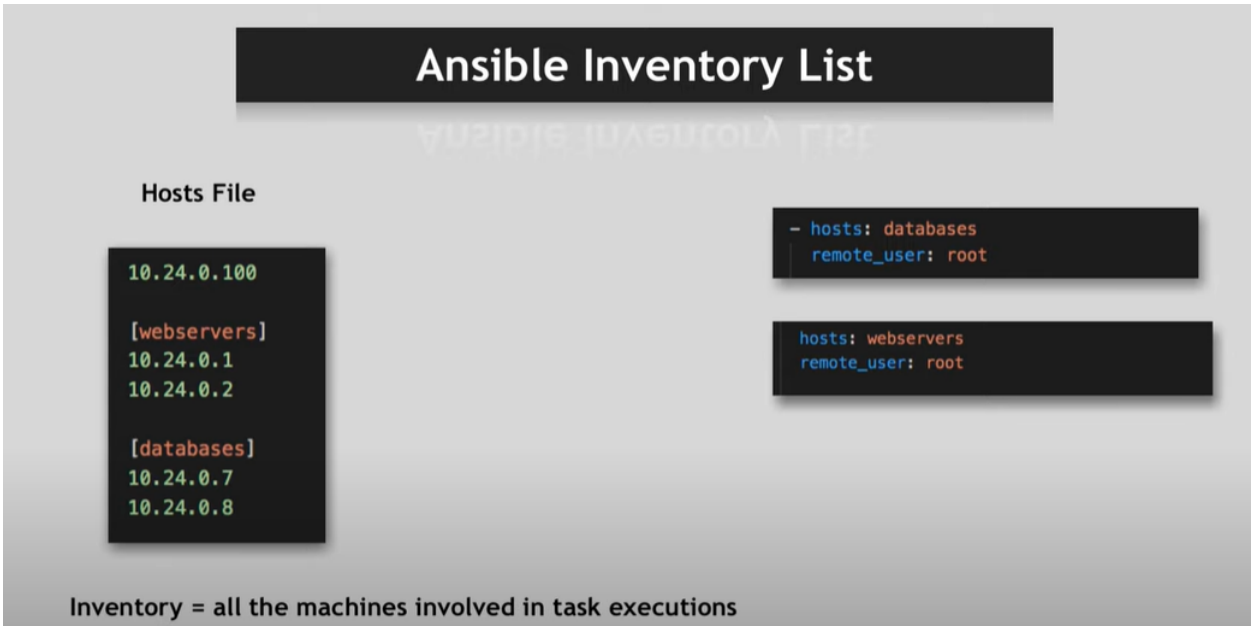




This playbook will be executed through the inventory file.

Strict indentation





Inventory file

No spaces between the square brackets

Let have a playbook and deploy it in two docker containers

ansible –version

```
[root@josh ansible]# ansible --version
ansible 2.9.25
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Nov 16 2020, 22:23:17) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]
[root@josh ansible]#
```

cd /etc/ansible/

ls

```
[root@josh ansible]# cd /etc/ansible/
[root@josh ansible]# ls
advanced-group.yml  ansible.cfg  den-inventory.yml  dev.vars.yml  hosts  pro.vars.yml  qa-inventory.yml  roles  web-deploy.yml
[root@josh ansible]#
```

touch dev.vars.yml

Put this information

[web-server]

server1 ansible_host=172.17.0.3 ansible_ssh_user=root ansible_ssh_pass=school1
{ this is hos to mask ip address}

[database-server]

server2 ansible_host=172.17.0.2 ansible_ssh_user=root ansible_ssh_pass=school1

:WQ

Let us now create a playbook

#touch web-deploy.yml

```
---
- hosts: webserver-atlanta
  user: root
  ignore_errors: yes
  tasks:
    - name: ping server
      ping:
    - name: check the date
      command: date
    - name: Run linux command
      shell: pwd
    - name: Install mysql
      yum: name=mysql state=present
- hosts: webserver
  become: yes
```

```
tasks:
  - name: Setup repo for Mysql 5.7
    yum:
      name: http://repo.mysql.com/mysql57-community-release-el7-10.noarch.rpm
      state: present

  - name: Install Packages
    yum:
      name:
      - httpd
      - mysql-community-server
      - firewalld
      state: present

  - name: Start Apache
    service:
      name: httpd
      state: started
      enabled: yes

  - name: Start Mysql
    service:
      name: mysqld
      state: started
      enabled: yes
```

```
- name: Start Firewalld
  service:
    name: firewalld
    state: started
    enabled: yes

- name: Apache to listen on 8080
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
    regexp: Listen 80
    line: Listen 8080
  notify: Restart Apache

- name: Allow 8080 port
  firewalld:
    port: 8080/tcp
    permanent: yes
    state: enabled
  notify: Restart Firewalld

- name: Disable SELinux
  selinux:
    state: disabled
```

```
  handlers:
    - name: Restart Firewalld
      service:
        name: firewalld
        state: restarted

    - name: Restart Apache
      service:
        name: httpd
        state: restartedi
```

Important commands

ansible webserver-atlanta -m ping -i dev.vars.yml

A green ping pong message should appear

With playbook

Checking the status

Ansible-playbook web-deploy.yml -I dev.vars.yml --syntax-check

Execute the play-book

Ansible-playbook web-deploy.yml -I dev.vars.yml