

**DATE : 07.04.2025**  
**DT/NT : DT**  
**LESSON : ANSIBLE**  
**SUBJECT: INTRODUCTION TO ANSIBLE**  
**BATCH : B 303**

**AWS-DEVOPS**



**TECHPRO**  
EDUCATION



techproeducation.com



+1 (585) 304 29 59





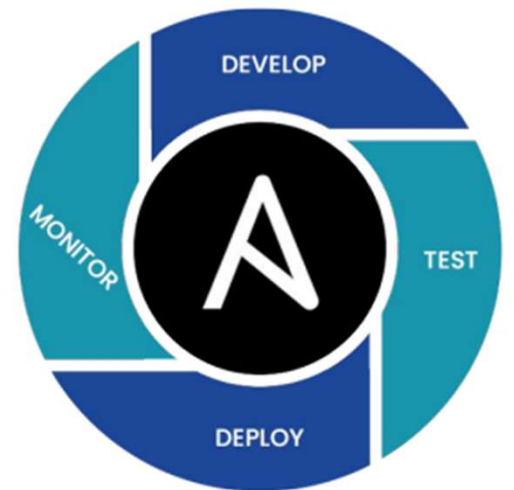
# Table of Contents

- ▶ About Ansible
- ▶ Installation
- ▶ Configuring Ansible
- ▶ Ansible Concepts
- ▶ ad-hoc Commands



## About Ansible

**Ansible is an open-source IT automation tool.** It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments.



# About Ansible



Scripts

- Time
- Coding Skills
- Maintenance



- Simple
- Powerfull
- Agentless



# About Ansible

## Scripts

```
#!/bin/bash
# Script to add a user to Linux system
if [ $(id -u) -eq 0 ]; then
    $username=johndoe
    read -s -p "Enter password : " password
    egrep "^$username" /etc/passwd >/dev/null
    if [ $? -eq 0 ]; then
        echo "$username exists!"
        exit 1
    else
        useradd -m -p $password $username
        [ $? -eq 0 ] && echo "User has been added
to system!" || echo "Failed to add a user!"
    fi
fi
```

## Playbook

```
- hosts: all_my_web_servers_in_DR
  tasks:
    - user:
        name: johndoe
```

A



# About Ansible

## SIMPLE

- Human readable automation
- No special coding skills needed
- Tasks executed in order

**Get productive quickly**

## POWERFULL

- App deployment
- Configuration management
- Workflow orchestration

**Orchestrate the app lifecycle**

## AGENTLESS

- Agentless architecture
- Uses Open SSH
- No agents to exploit or update

**More efficient & more secure**



# Installation

We can install Ansible using yum and apt package managers.

**To install on Amazon Linux :**

```
sudo amazon-linux-extras install ansible2 -y
```

**For install with apt:**

```
sudo apt-get -y install ansible
```

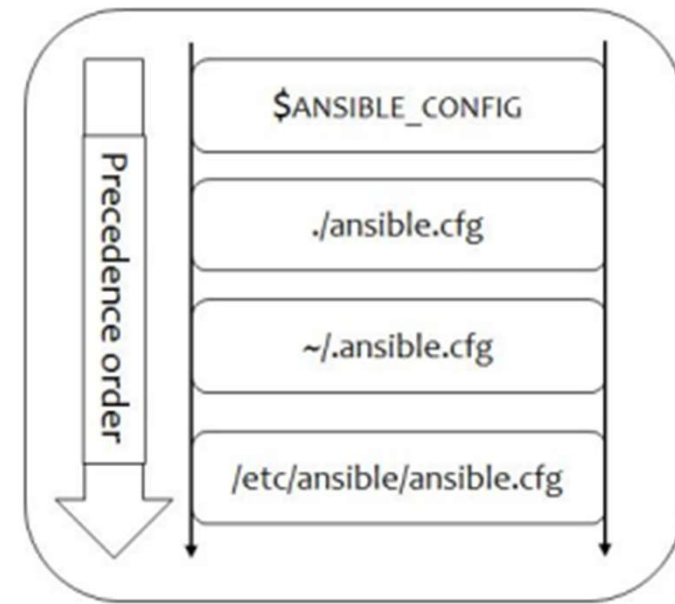
**For install with pip:**

```
pip install ansible
```



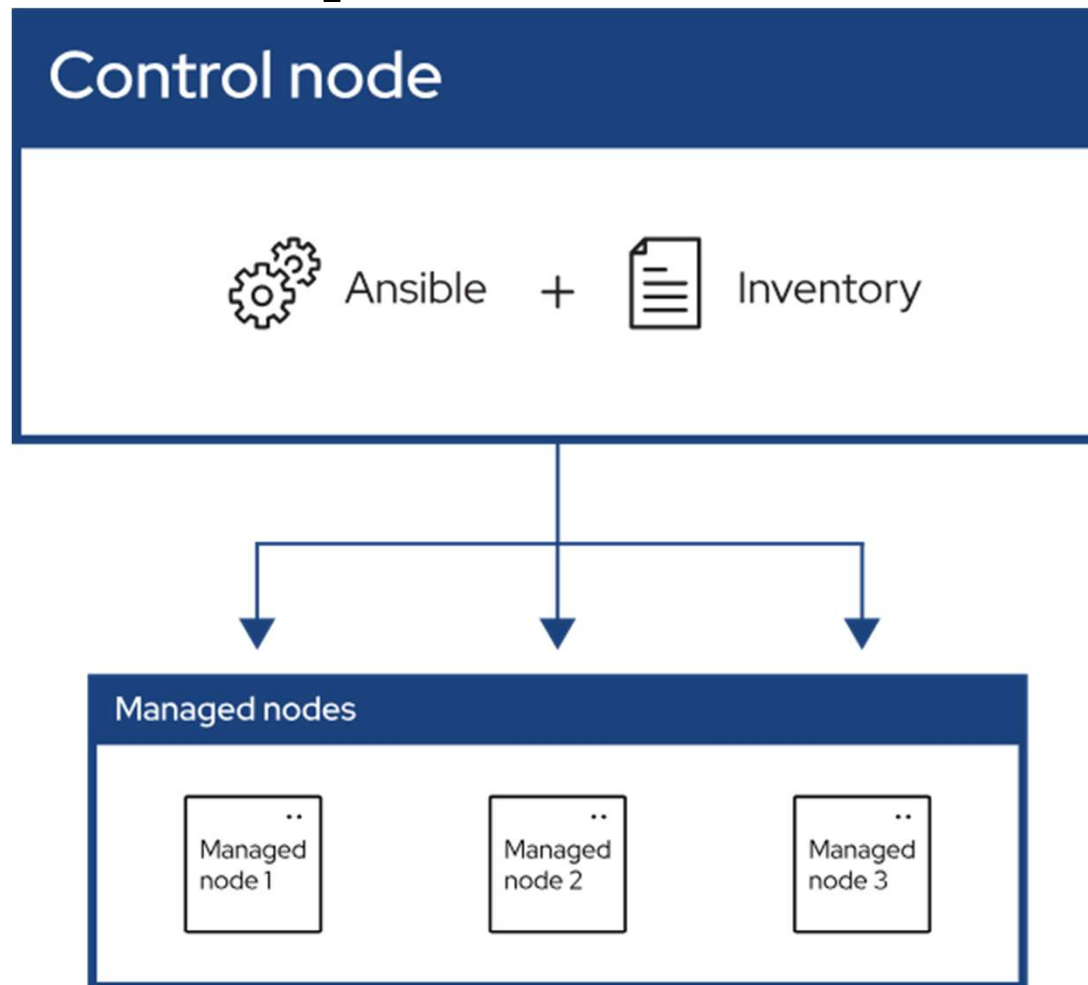
# Configuring Ansible

- Ansible supports several sources for configuring its behavior, including a file named `ansible.cfg`, environment variables, command-line options, playbook keywords, and variables.
- Certain settings in Ansible are adjustable via a configuration file (`ansible.cfg`).
- Changes can be made and used in a configuration file which will be searched for in the following order:

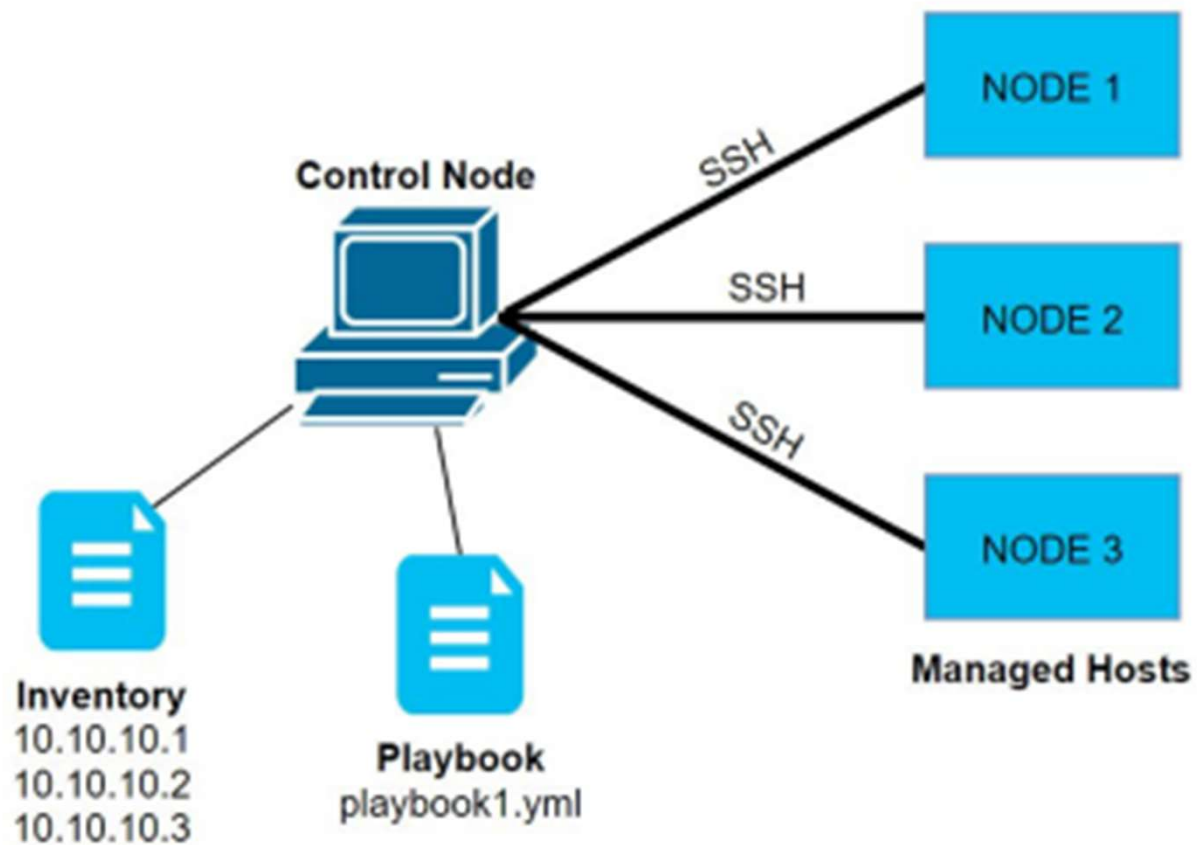




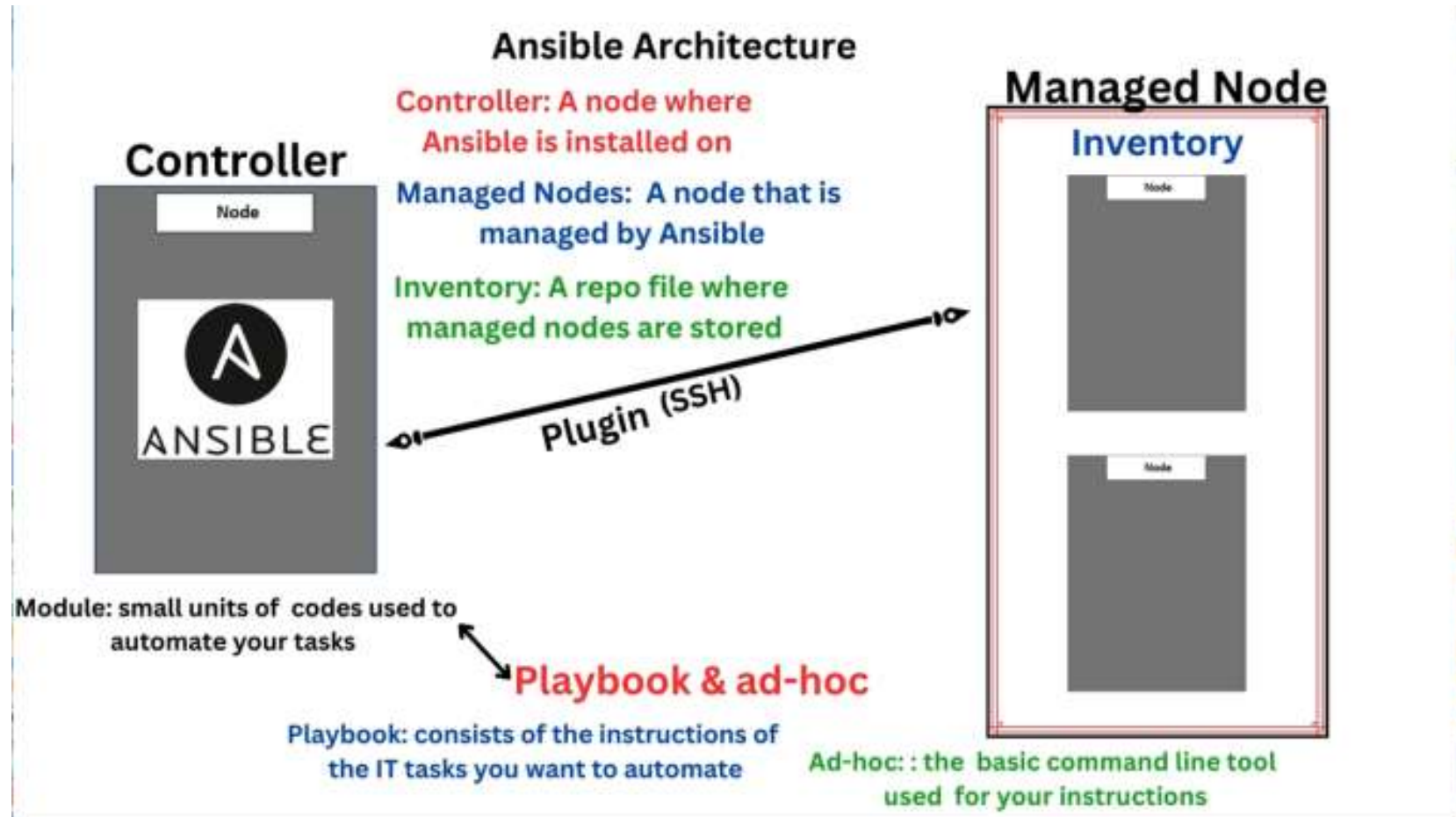
# Ansible Concepts



# Ansible Concepts



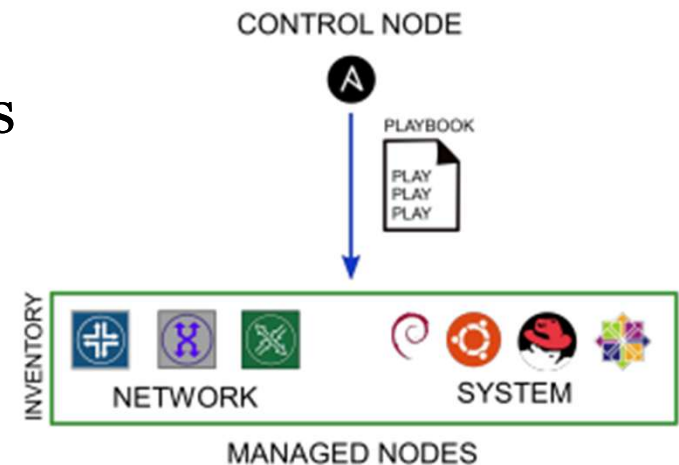
# Ansible Concepts



# Ansible Concepts

## Control node:

Any machine with Ansible installed. You can run commands and playbooks, invoking `/usr/bin/ansible` or `/usr/bin/ansible-playbook`, from any control node. You can use any computer that has Python installed on it as a control node - laptops, shared desktops, and servers can all run Ansible. However, you cannot use a Windows machine as a control node.

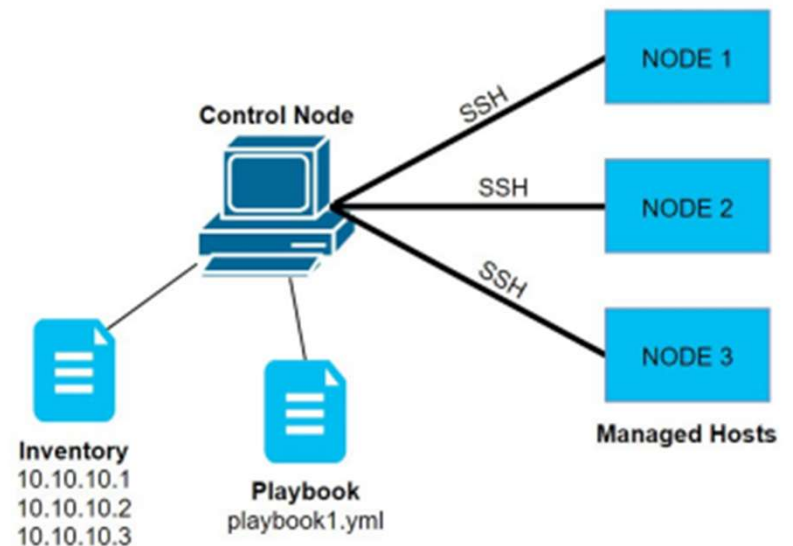


# Ansible Concepts

## Managed Nodes:

The network devices (and/or servers) you manage with Ansible.

Managed nodes are also sometimes called hosts. Ansible is not installed on managed nodes.



# Ansible Concepts

## Inventory:

A list of managed nodes. An inventory file is also sometimes called a hostfile. Your inventory can specify information like IP address for each managed node. An inventory can also organize managed nodes, creating and nesting groups for easier scaling.

### The inventory file

#### Where it is located

/etc/ansible/hosts

#### What is the format

##### [mailservers]

mail.example.com

##### [webservers]

foo.example.com ansible\_ssh\_user = user001

bar.example.com ansible\_ssh\_private\_key\_file =  
/.ssh/ansible\_key001

##### [dbservers]

one.example.com

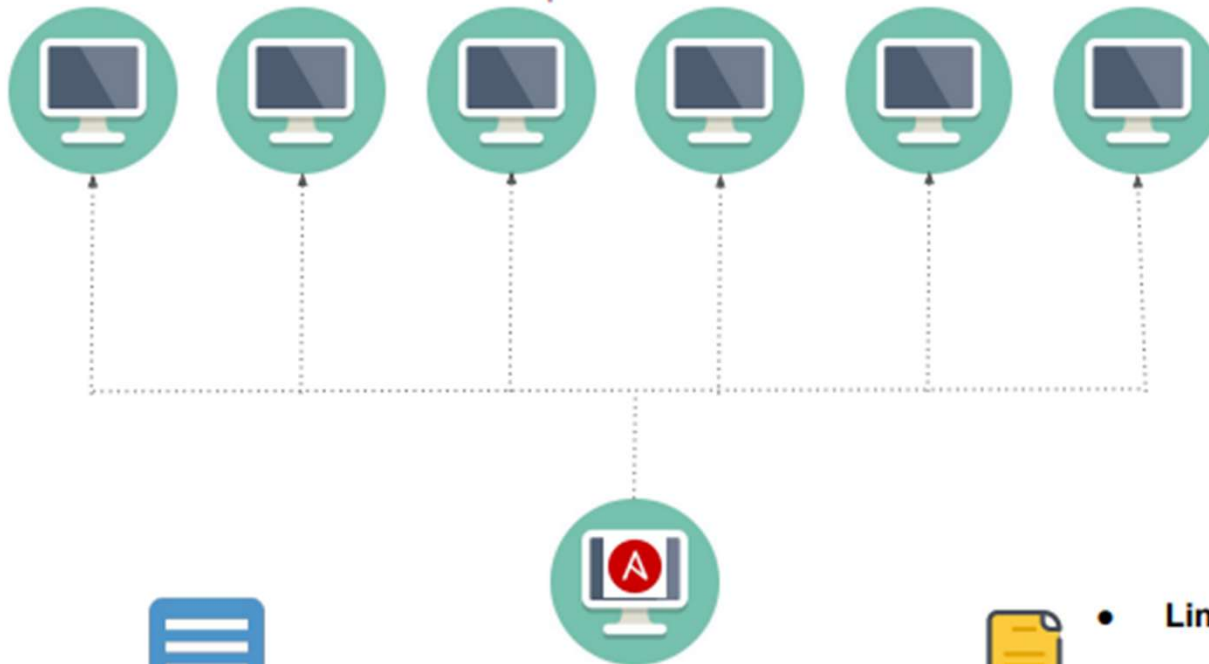
two.example.com

db-[a:f].example.com



# Ansible Concepts

Inventory



```
server1.company.com
server2.company.com

[mail]
server3.company.com
server4.company.com

[db]
server5.company.com
server6.company.com

[web]
server7.company.com
server8.company.com
```



**inventory**  
/etc/ansible/hosts



- Linux - SSH
- Windows - Powershell Remoting



- Agentless



# Ansible Concepts

Group hosts for easier inventory selection and less conditional tasks -- the more groups the better.

## WHAT

```
[db]  
db[1:4]
```

```
[web]  
web[1:4]
```

```
db1 = db, east, dev
```

## WHERE

```
[east]  
db1  
web1  
db3  
web3
```

```
[west]  
db2  
web2  
db4  
web4
```

## WHEN

```
[dev]  
db1  
web1
```

```
[test]  
db3  
web3
```

```
[prod]  
db2  
web2  
db4  
web4
```



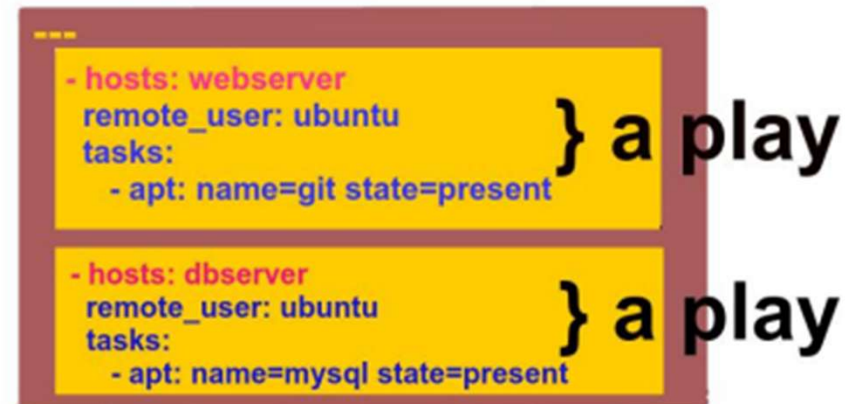


# Ansible Concepts

## Playbooks:

Ordered lists of tasks, saved so you can run those tasks in that order repeatedly. Playbooks can include variables as well as tasks. Playbooks are written in YAML and are easy to read, write, share and understand.

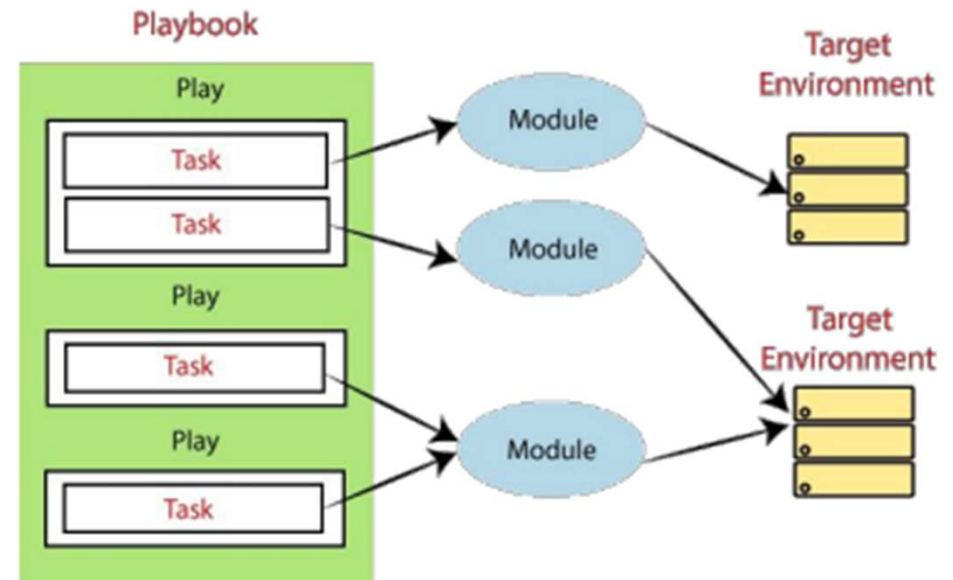
## Playbook



# Ansible Concepts

## Tasks:

The units of action in Ansible.  
You can execute a single task once with an ad-hoc command.



# Ansible Concepts

## Modules:

The units of code Ansible executes. Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device.

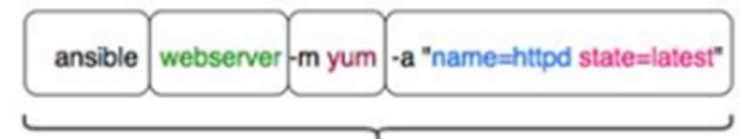
Modules	Module Categories							
System	User	Group	Iptables	Mount	Ping	Systemd	Service	Hostname
Commands	Command	Expect	Raw	Script	Shell			
Files	Acl	Archive	Find	Copy	Replace	Stat	File	Unarchive
Database	MySQL	MongoDB	MSSQL	PostgreSQL	ProxySQL	Vertica		
Cloud	Amazon	Azure	Google	Linode	Openstack	VMware	Docker	Atomic
Windows	Win_copy	Win_command	Win_msi	Win_ping	Win_rsq	Win_shell	Win_path	Win_service



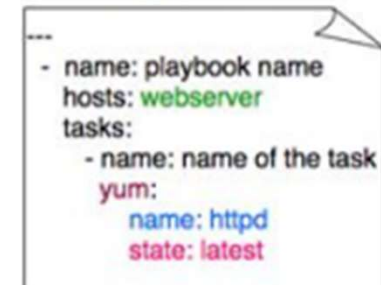
# ad-hoc Commands

- An Ansible ad-hoc command uses the `/usr/bin/ansible` command-line tool to automate a single task on one or more managed nodes.
- Ad-hoc commands are quick and easy, but they are not reusable.
- Ad-hoc commands demonstrate the simplicity and power of Ansible.
- Ad-hoc commands are great for tasks you repeat rarely.

AD HOC command



Ansible Playbook



# ad-hoc Commands

- **ansible** <inventory> -m

AD HOC command

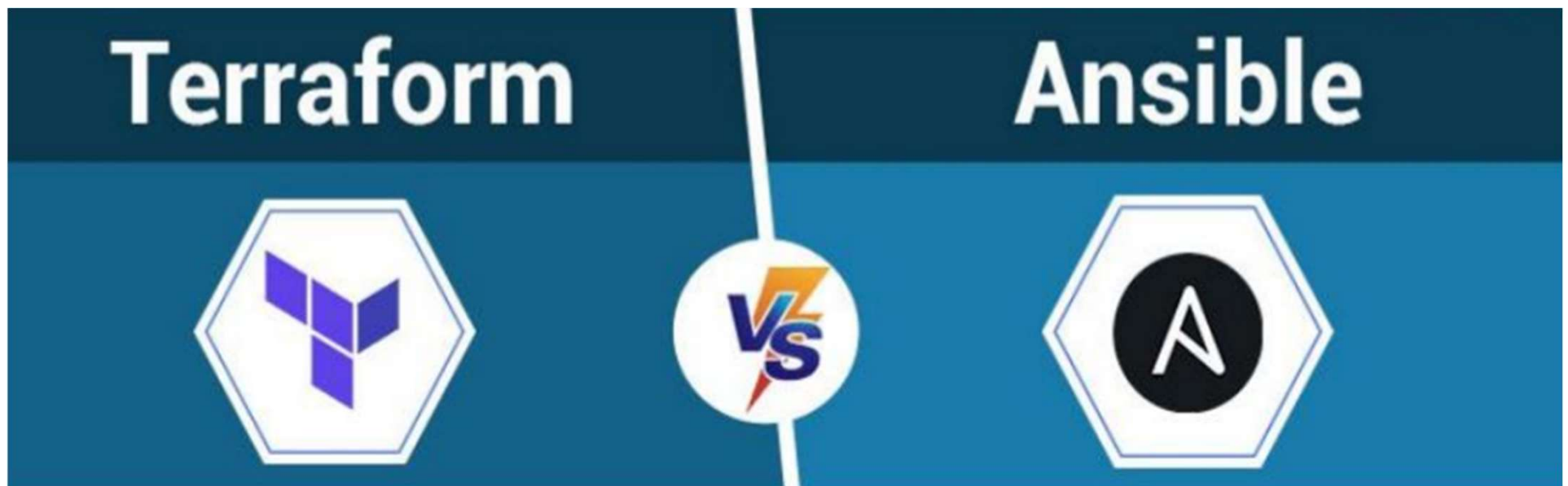
```
ansible webserver -m yum -a "name=httpd state=latest"
```

**Runs a command** or **calls a module** directly from the **command line**, no Playbook required

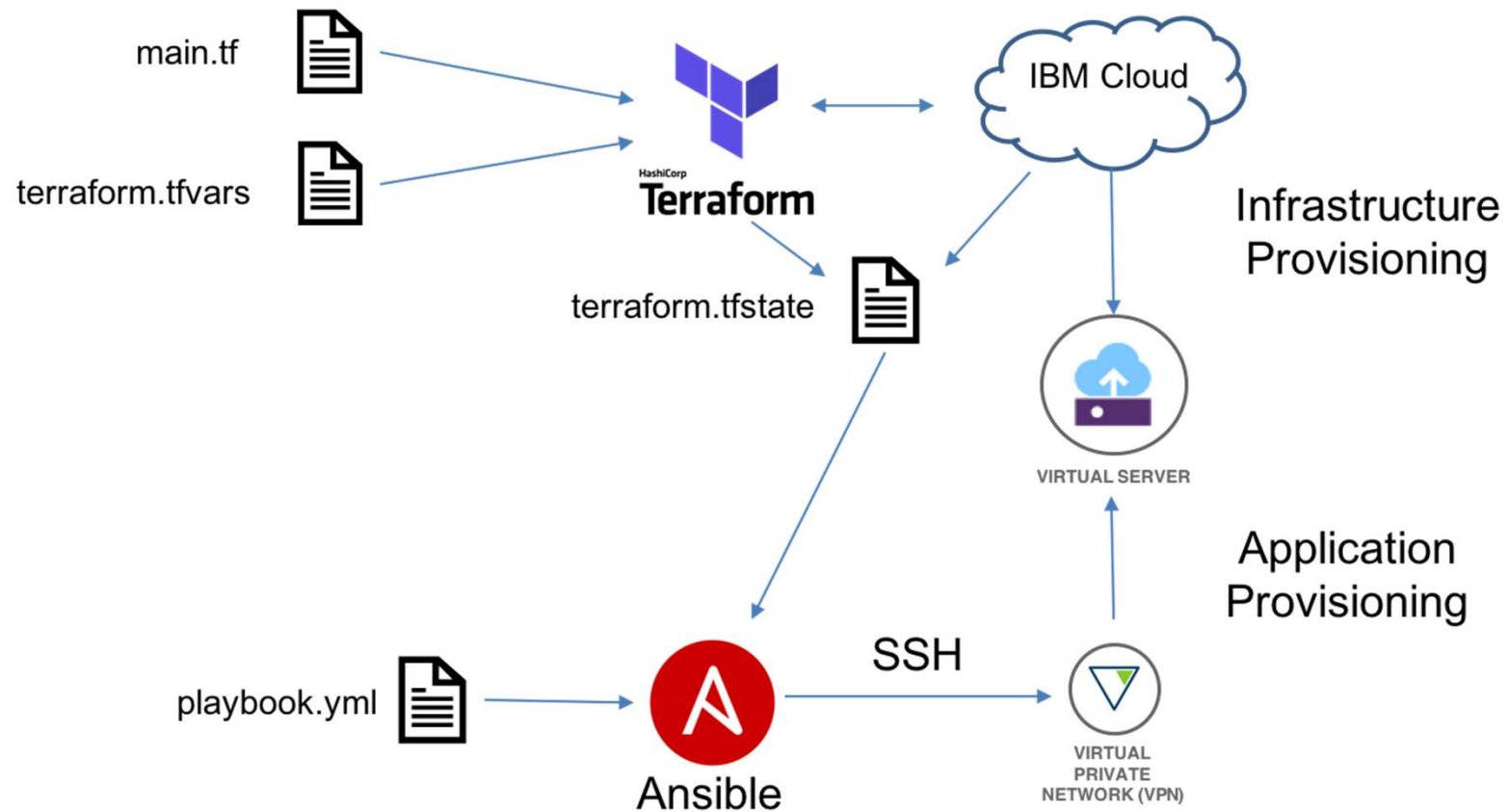
```
ansible <inventory> <options>
ansible web -a /bin/date
ansible web -m ping
ansible web -m yum -a "name=openssl state=latest"
```



## Conclusion






# Conclusion



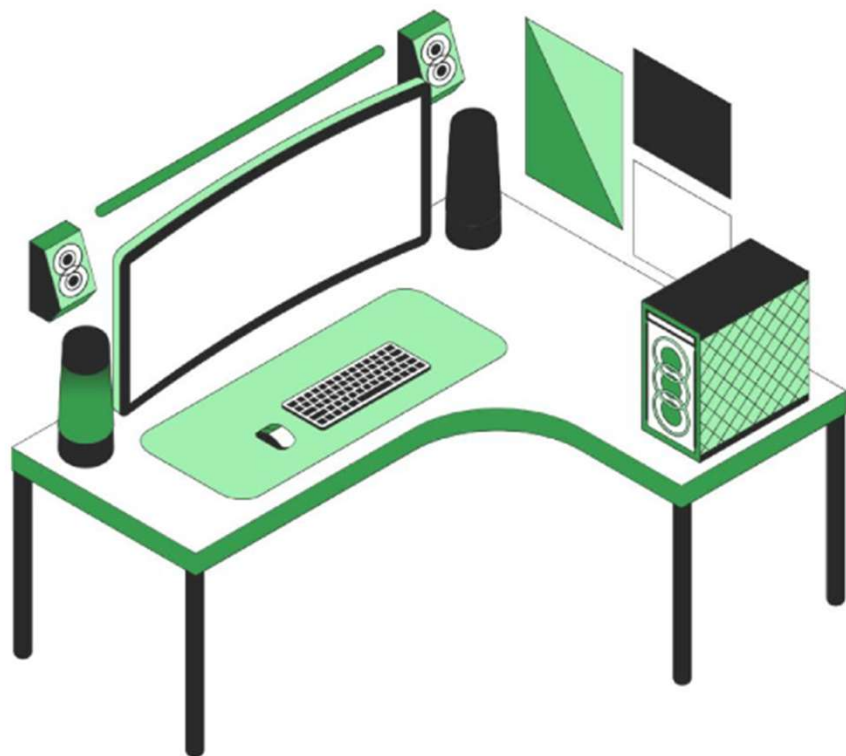


# Conclusion



 VS  VS 			
Criteria	Ansible	Puppet	Chef
Configuration Language	<ul style="list-style-type: none"> <li>- YAML (Yet Another Markup Language)</li> <li>- Python</li> </ul>	<ul style="list-style-type: none"> <li>- Puppet DSL (Domain Specific Language)</li> <li>- Embedded Ruby (ERB)</li> </ul>	Ruby DSL
Availability	Facility of Secondary instance to replace the Primary instance in case of failure	Facility of another master for replacing an active master in case of failure	Facility of backup server to replace the Chef server in case of failure
Ease of setup and installation	Easy	Time-consuming due to certificate signing between master and agent	Time-consuming and complicated due to the Chef Workstation
Ease of access	Use of SSH running system allows configuring only specific nodes	Puppet clients can only configure the client machine and puppet server for Master machine	Chef clients can only configure the client machine and Chef Server
Management	Support for pull and push configurations with faster remote execution	Use of pull configuration & the Puppet DSL language create setbacks for management	Support for pull configuration only & Ruby DSL configuration language create pitfalls for management
Scalability	Efficient scalability with installation of Python interpreter and root access SSH	Large Puppet DSL code creates problems for scalability	The massive Ruby DSL code creates difficulties for scalable deployments
Interoperability	Ansible server should be on Linux/Unix machine and client machines support Windows.	Puppet Master should be on Linux/Unix machine and Puppet Agent machines support Windows.	Support for pull configuration only & Ruby DSL configuration language create pitfalls for management
Pricing	<ul style="list-style-type: none"> <li>• Self-support package starts at \$5000 annually</li> <li>• Premium version costs \$14,000 annually for each 100 nodes</li> </ul>	<ul style="list-style-type: none"> <li>• Puppet Enterprise pricing starts at \$120 for every node annually</li> <li>• Premium plan of Puppet Enterprise is available at annual price of \$199 for each node</li> </ul>	<ul style="list-style-type: none"> <li>• The standard Hosted Chef plan starts at \$72 annually for every node</li> <li>• The Chef Automation version charges \$137 per node annually</li> </ul>
Product capabilities	<ul style="list-style-type: none"> <li>• App deployment</li> <li>• Streamlined provisioning</li> <li>• Simple orchestration</li> <li>• Integration of security and compliance policy in automated processes</li> <li>• Automated workflow for continuous delivery</li> </ul>	<ul style="list-style-type: none"> <li>• Orchestration</li> <li>• Role-based access control</li> <li>• Configuration automation</li> <li>• Automated provisioning</li> <li>• Code management</li> <li>• Node management</li> <li>• Visualization and reporting</li> </ul>	<ul style="list-style-type: none"> <li>• Automation for DevOps workflow</li> <li>• Automated workflow for continuous delivery</li> <li>• Infrastructure automation</li> <li>• Compliance and security management</li> </ul>





# Do you have any questions?

Send it to us! We hope you learned something new.