Main Topics Covered

- Why Automate ?
- Role of Ansible in Automation
- Components of Ansible
- A of Ansible
- Demo Requirements and Demo
- Inventory Setup Demo
- Variables Demo
- Groups of Groups, and Group Variables
- Create your own Ansible Module

ANSIBLE BASICS

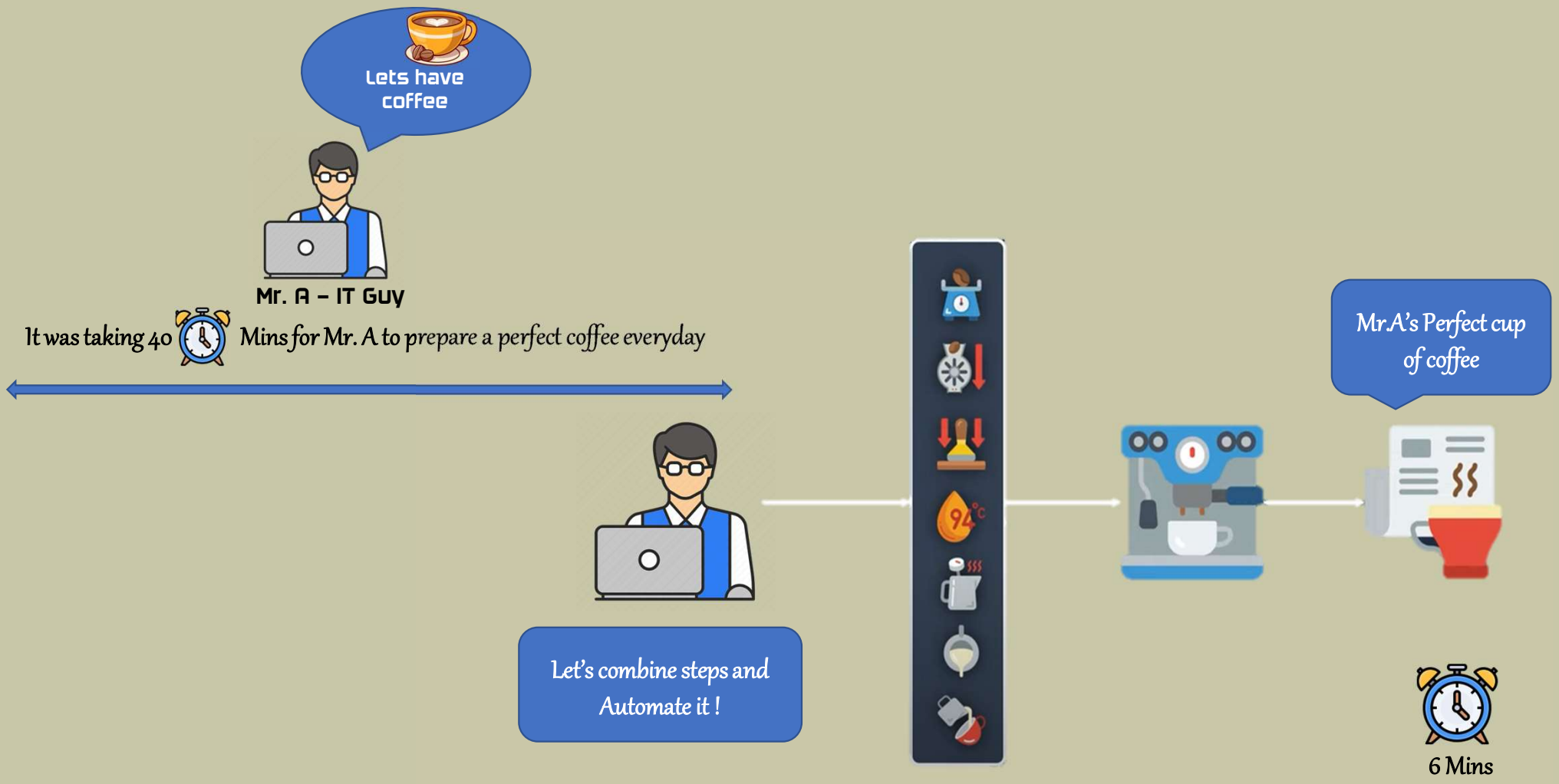
By Vivek Sheshadri

https://Blogs.decodegeek.co.in

Coforge Consultant – DevOps Services

Background : 11+ years of experience in Automation and DevSecOps

Coffee Cup Scenario

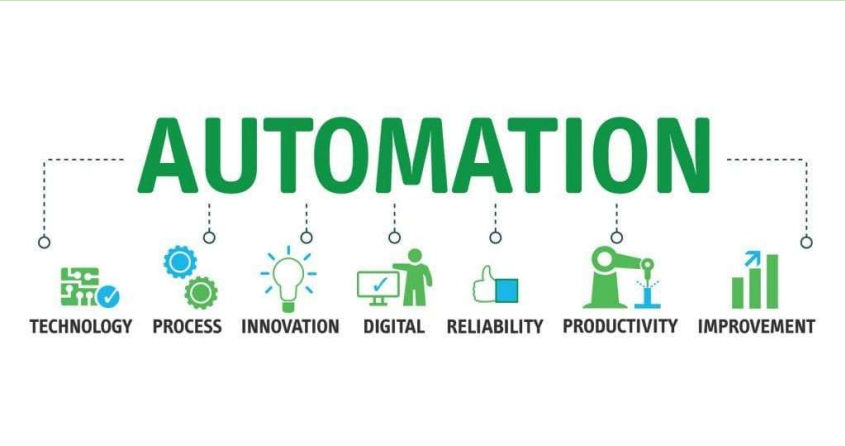


What is Configuration Management Automation ?

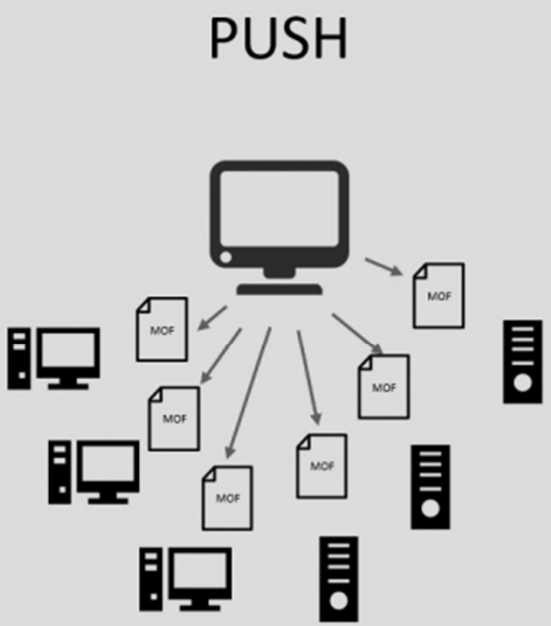
Automation **describes a wide range of technologies that reduce human intervention in processes. Human intervention is reduced by predetermining decision criteria, subprocess relationships, and related actions**

Configuration management (CM)

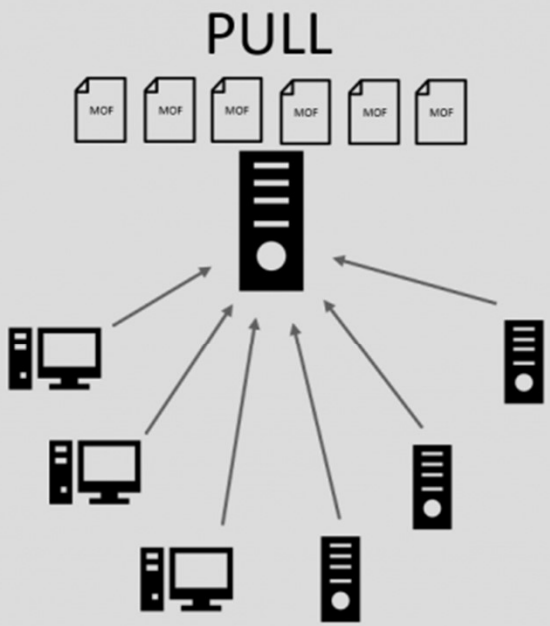
is the process of maintaining systems, such as computer hardware and software, in a desired state



Pull and Push Model

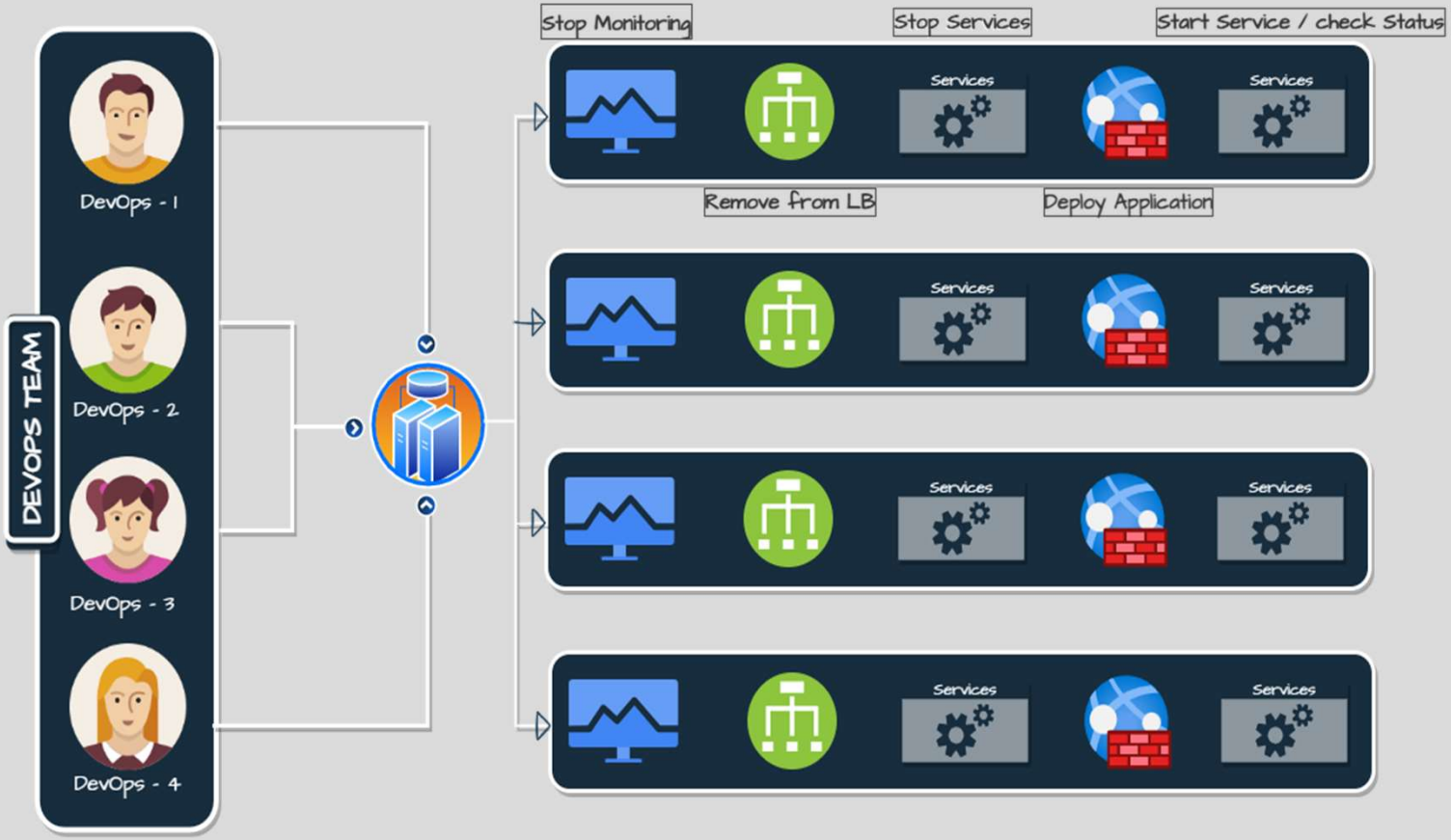


Author MOFs for every device to push out

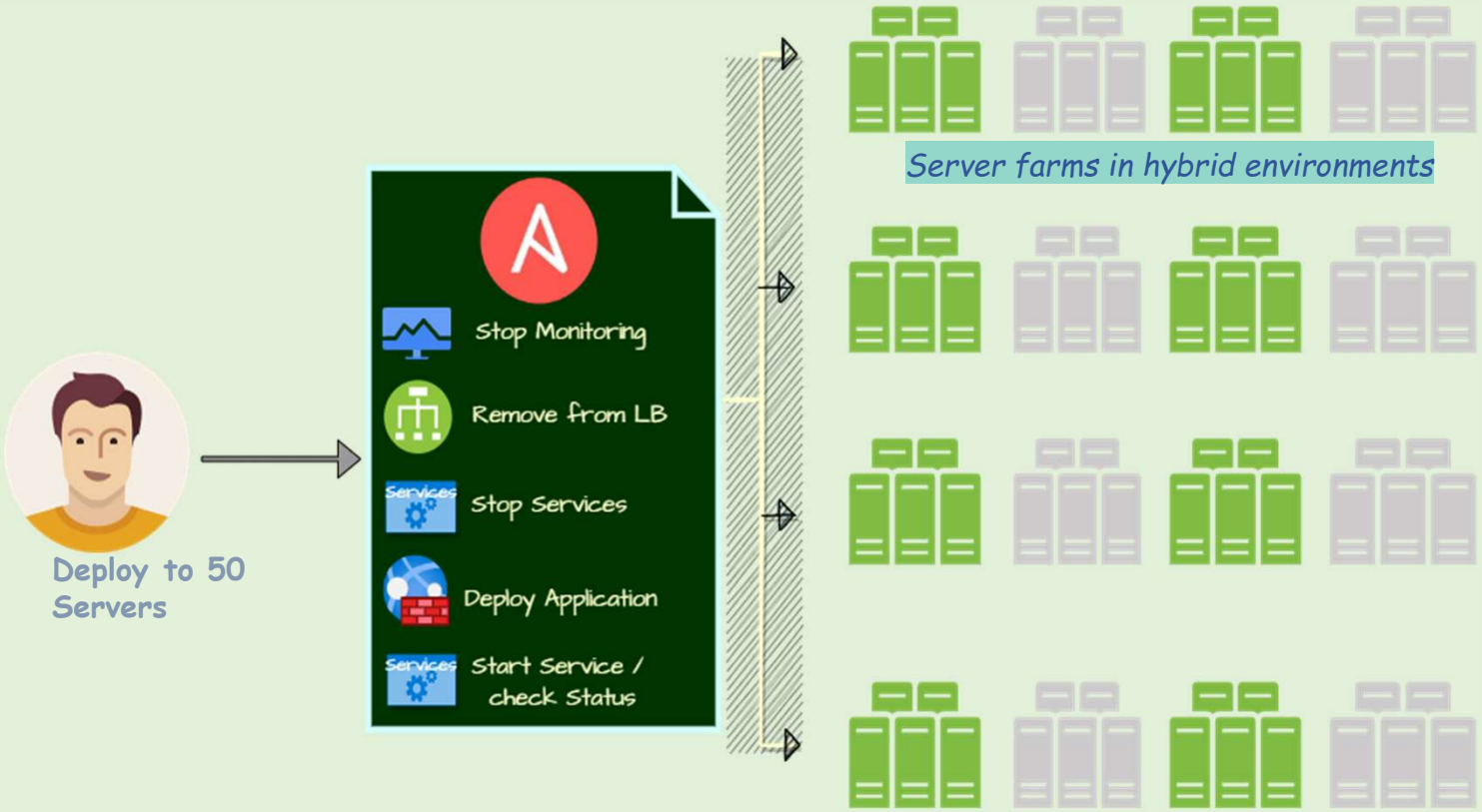


Author MOFs for every device and place on pull server

TRADITIONAL APPROCH of AUTOMATION



ANSIBLE AUTOMATION

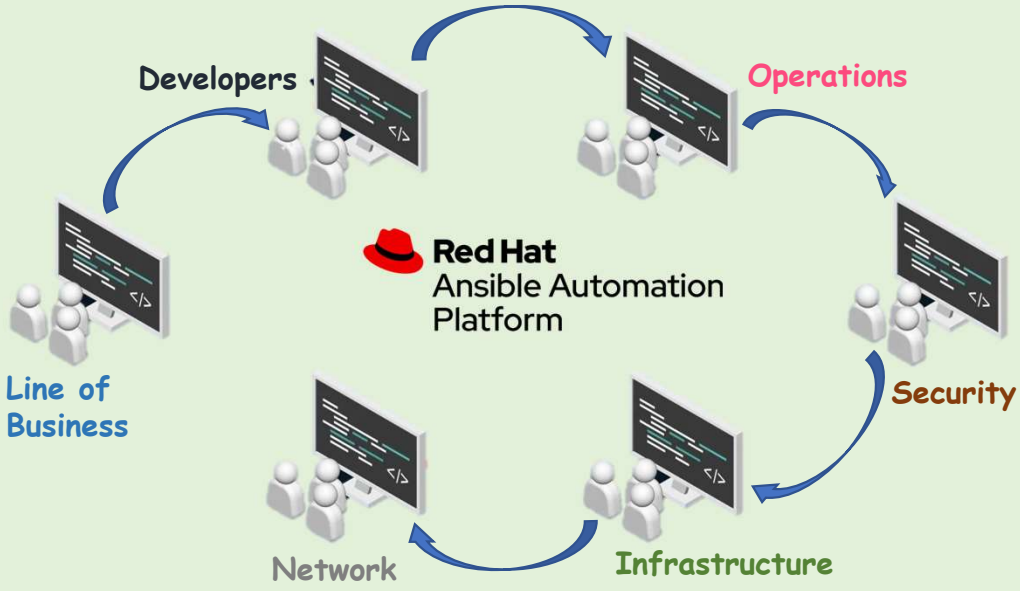


1. Ease of Use
2. Auto Repeat
3. Ease Collab

Supports :
SSH - Linux
PowerShell - Windows

ANSIBLE Definition!

Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs



ANSIBLE - Basic Concepts /Components

Control Node - The machine from which you run the Ansible CLI tools (ansible-playbook , ansible, ansible-vault and others).

Managed Nodes - Also referred to as 'hosts', these are the target devices (servers, network appliances or any computer) you aim to manage with Ansible.

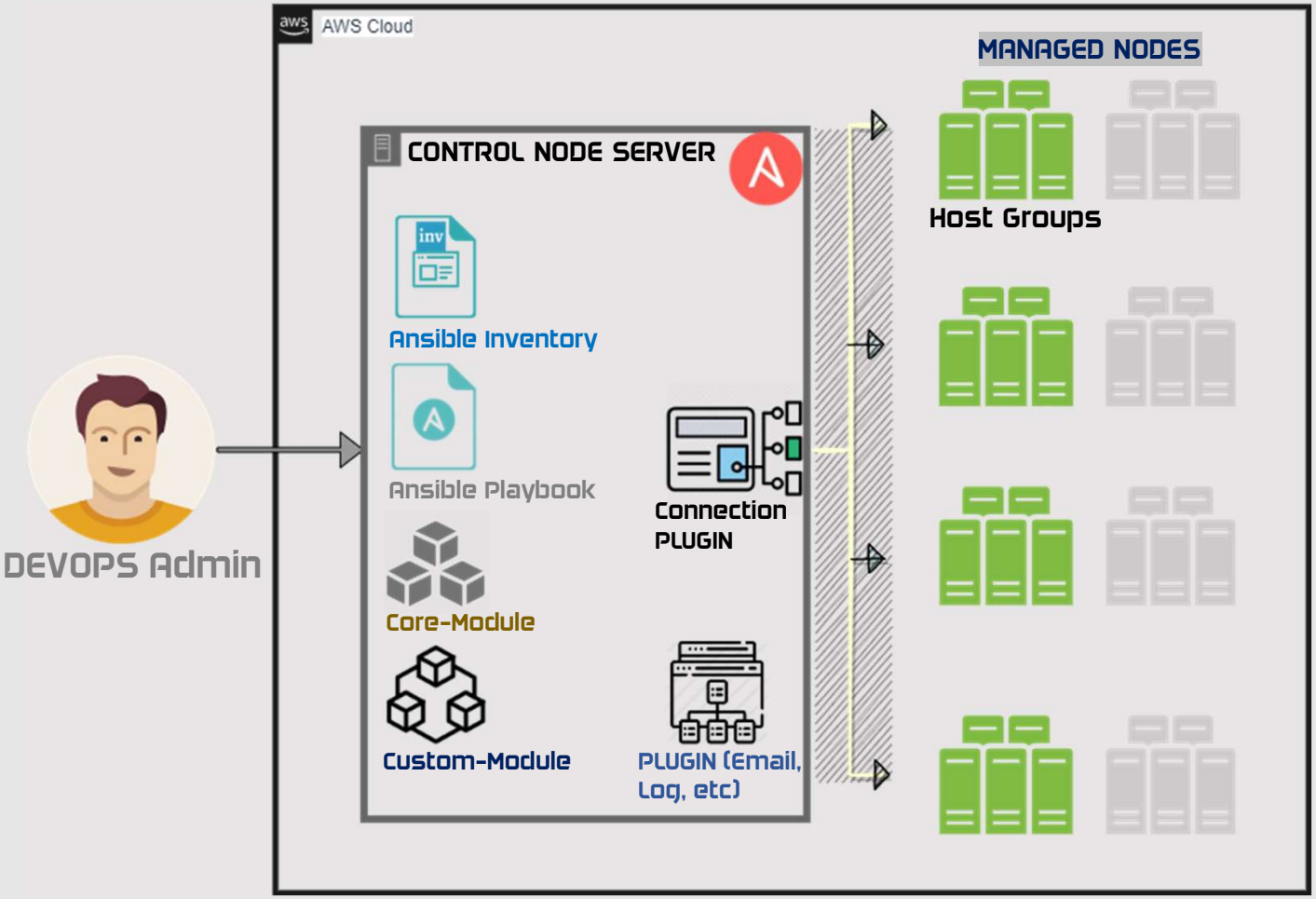
Inventory - A list of managed nodes provided by one or more 'inventory sources'. Your inventory can specify information specific to each node, like IP address.

Playbook - They contain Plays (which are the basic unit of Ansible execution). this is both an 'execution concept' and how we describe the files on which ansible-playbook operates on.

Modules - The code or binaries that Ansible copies and executes on each managed node (when needed) to accomplish the action defined in each Task. Each module has a particular use, You can invoke a single module with a task, or invoke several different modules in a playbook

Plugins - Pieces of code that expand Ansible's core capabilities, they can control how you connect to a managed node, manipulate data, and even control what is displayed in the console.

ANSIBLE Automation Architecture

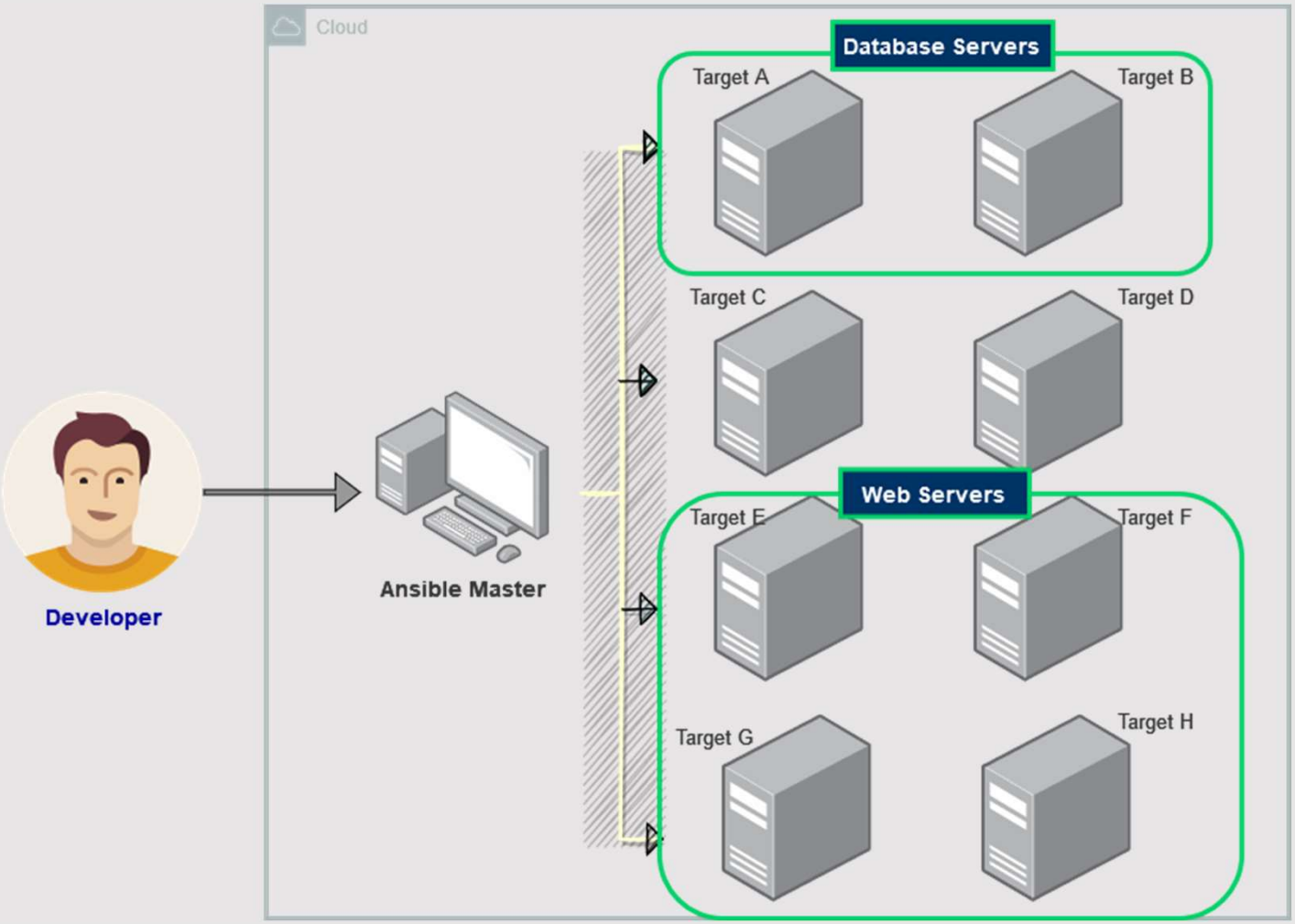


ANSIBLE –AWS Demo Setup

Name	Configuration
AWS Free tier Account	Create an Administrator Account with Full access
Ansible_Master(EC2-Instance)	Amazon Linux (free tier eligible) 64-bit Network- Allow SSH, Create key-pair (login)
Ansible_Target1(EC2-Instance)	Same as above
Ansible_Target2(EC2-Instance)	Same as above
VS Code tool	For editing the code
MobaXterm (Portable Version)	For connecting to EC2 Instances

Note: Ansible does not require any plugins or agents need to be installed in targets, so we have to do all the initial setups in Ansible-Master instance.

ANSIBLE Inventory Structure



INVENTORY FILE PATH : /etc/ansible/hosts/inventory.yml

SLNO	HOSTNAME	IP ADDRESS	USERNAME	CONNECTION TYPE	PASSWORD/SSHKEY	SERVICE
1	Target1	12.23.22.1	Ec2-user	SSH	Ec2-key.pem	WEB SERVICE
2	Target2	12.23.22.2	Ec2-user	SSH	Ec2-key.pem	WEB SERVICE
3	Target3	12.23.22.3	Ec2-user	PASSWORD	********	DB SERVICE
4	Target4	12.23.22.4	Ec2-user	PASSWORD	********	DB SERVICE
5	Target5	12.23.22.5	Ec2-user	SSH	Ec2-key.pem	WEB SERVICE

ANSIBLE Inventory Structure

INVENTORY FILE PATH: /home/ec2-user/inventory.txt

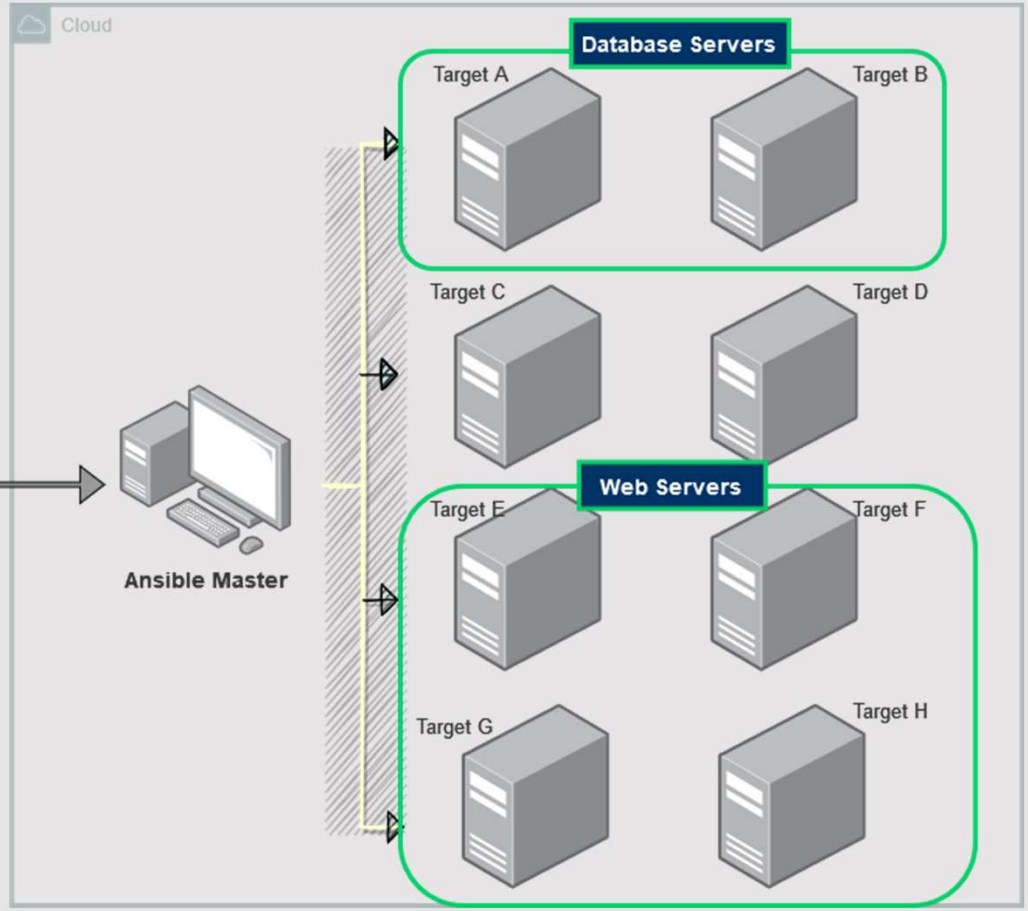
```
[web servers]
targetA.Coforge.com
targetB.Coforge.com

[database Servers]
targetE.Coforge.com
targetF.Coforge.com
targetG.Coforge.com
targetH.Coforge.com
```

```
[common]
targetA.Coforge.com
targetB.Coforge.com
targetE.Coforge.com
targetF.Coforge.com
targetG.Coforge.com
targetH.Coforge.com
```

Developer

Ansible Master

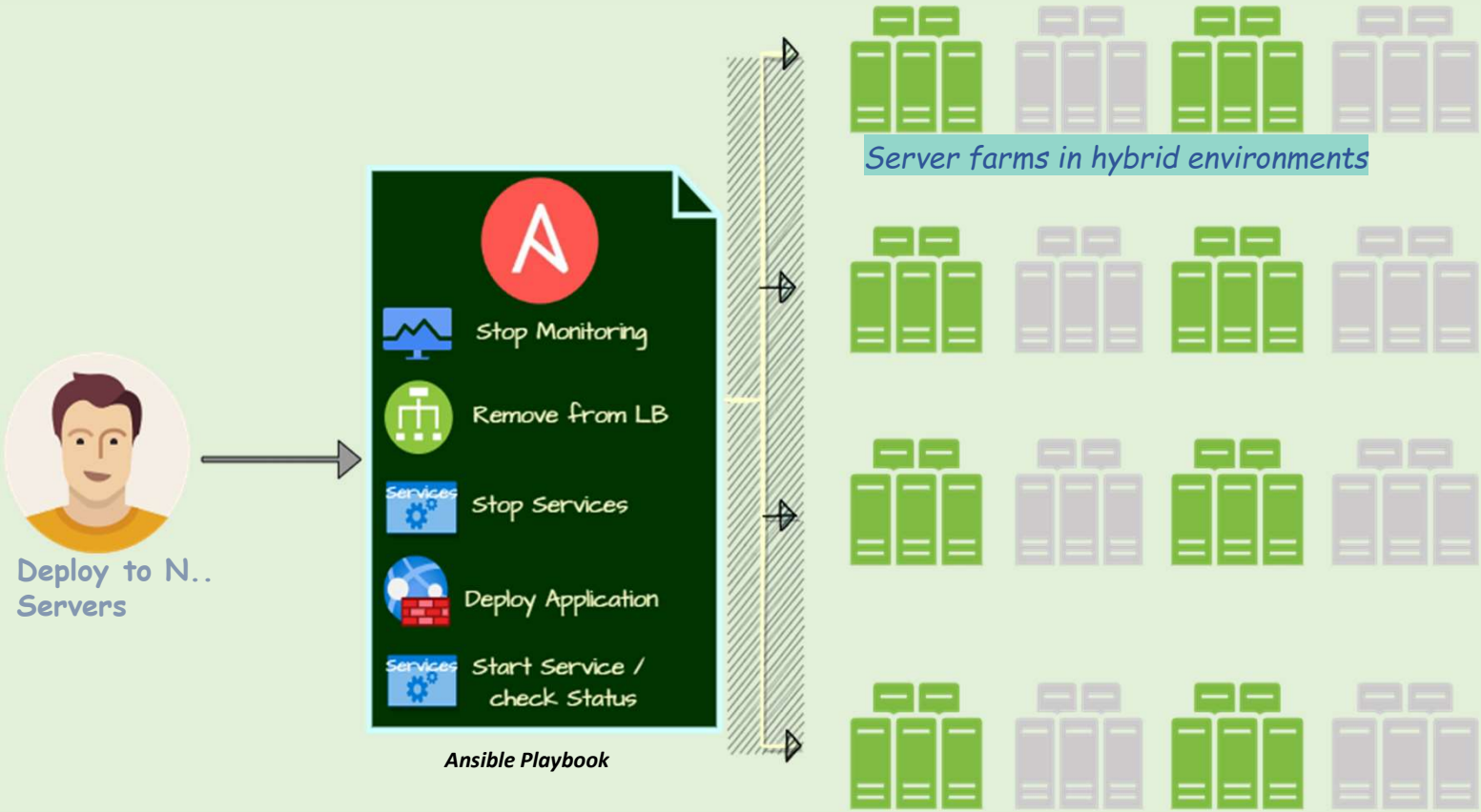


RUN WEB SERVERS: ansible “web servers” –m ping –i inventory.txt

RUN WEB SERVERS: ansible “database servers” –m ping –i inventory.txt

RUN COMMON SERVERS: ansible common –m ping –i inventory.txt

ANSIBLE PLAYBOOK



Same steps will be replicated to all servers targets with the same configuration

PLAYBOOK – List in YAML file using Coffee Cup Scenario

KEY

VALUES

Raw Materials
Coffee
Flavoured Liquid
Milk
Water
Sugar

JSON-KEY-VALUE

Raw Materials

```
{
  "raw_materials": [
    "Coffee",
    "Flavored Liquid",
    "Milk",
    "Water",
    "Sugar"
  ]
}
```

[]-LIST

YAML-KEY-VALUE

Raw Materials

```
raw_materials:
  - Coffee
  - Flavored Liquid
  - Milk
  - Water
  - Sugar
```

LIST – HYPHEN

COMBINATION OF LIST AND DICTIONARY

Coffee Making Steps		
Ingrediants	Coffee	
	Flavored Liquid	
	Milk	
	Water	
	Sugar	
Process	Grainding	
	Pressing	
	Boiling	
	Filtering	
	Mixing	
Grainding	Coffee	Sunrise
	Texture	Coarse
Boiling	Temp	92 degree C
	Flavor	Hazelnut
	Time	5 Mins
Mixing	Milk	True
	Sugar	True
	No_of_Cups	1

YAML-KEY-VALUE

Raw Materials

```
Ingredients:
  - Coffee
  - Flavoured Liquid
  - Milk
  - Water
  - Sugar

Process:
  - Grinding
  - Pressing
  - Boiling
  - Filtering
  - Mixing

Grinding:
  Coffee: Sunrise
  Texture: Coarse

Mixing:
  Milk: "True"
  Sugar: "True"
  No_of_Cups: 1
```

LIST

LIST

DICT

DICT

YAML To Json Validation

YAML Syntax in brief

Method Description	Syntax
1. To Comment a Line	# This is a sample playbook
2. Escape characters in double Quotes	“sample_text”: “Escape \t character”
3. Defining variable	file_path: “{{ variable }}”

4. **In YAML**
Ansible: "Greet: Welcome to Ansible."
	In JSON
	{
	"Ansible": "Greet: Welcome to Ansible."
	}

5. **In YAML**
Ansible: { Greet: Welcome to Ansible. }
	In JSON
	{
	"Ansible": {
	"Greet": "Welcome to Ansible."
	}
	}

What is an Ansible Module ?

Modules (also referred to as "task plugins" or "library plugins") are discrete units of code that can be used from the command line or in a playbook task

Ansible:

- Executes each module on the remote target node.
- Collects return values.

Syntax:

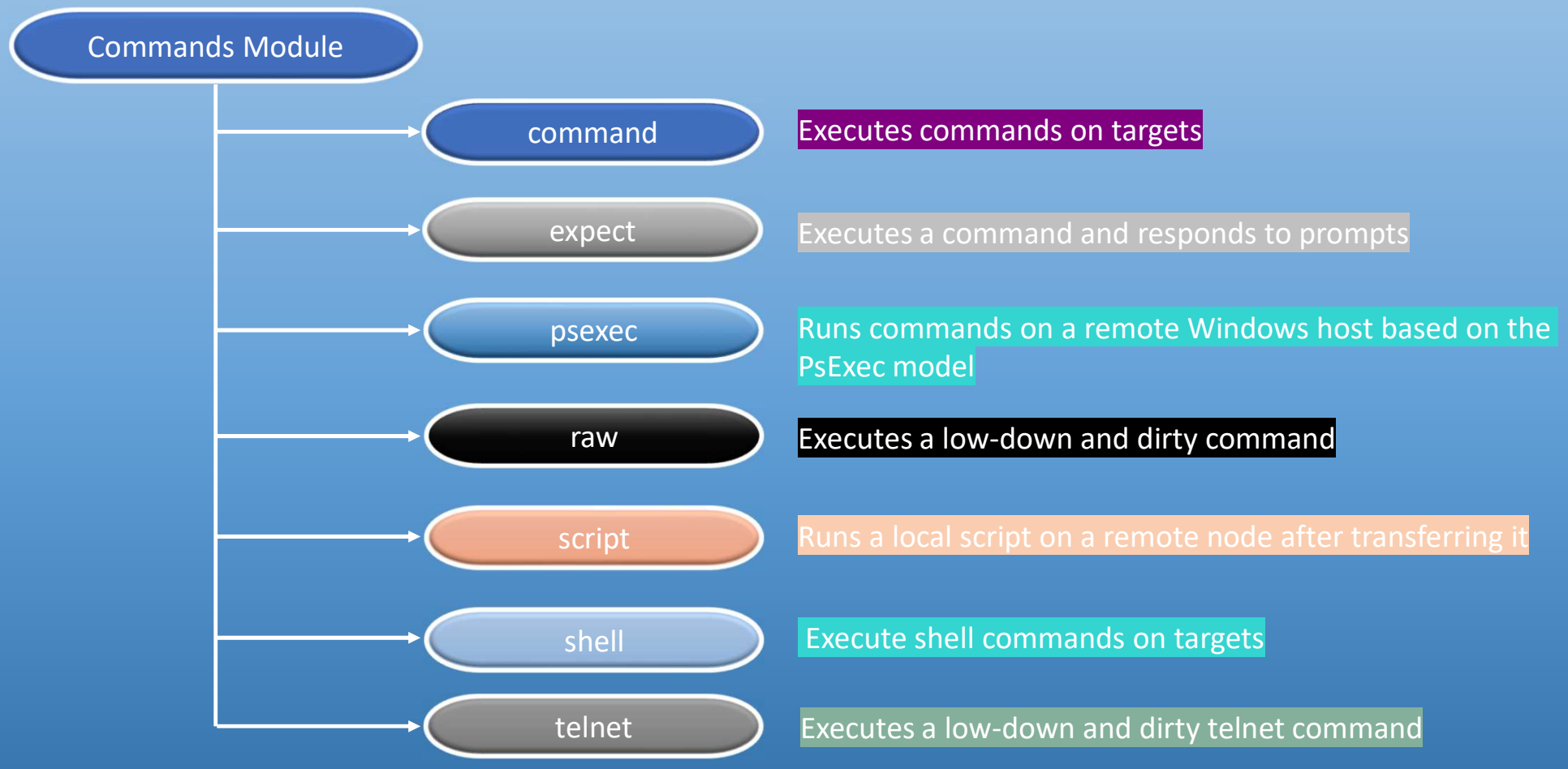
```
ansible <host or group name> -m <module_name> -a <arguments>
```

Examples:

1. ansible webservers -m service -a "name=httpd state=restarted"
2. ansible webservers -m ping
3. ansible webservers -m command -a "/sbin/reboot –t now"

ANSIBLE MODULE CATOGORIES

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


Commands Module
command
Executes commands on targets
expect
Executes a command and responds to prompts
psexec
Runs commands on a remote Windows host based on the PsExec model
raw
Executes a low-down and dirty command
script
Runs a local script on a remote node after transferring it
shell
Execute shell commands on targets
telnet
Executes a low-down and dirty telnet command

SUB-CATOGORIES - CORE MODULES

Few module examples created and maintained by Ansible core team -

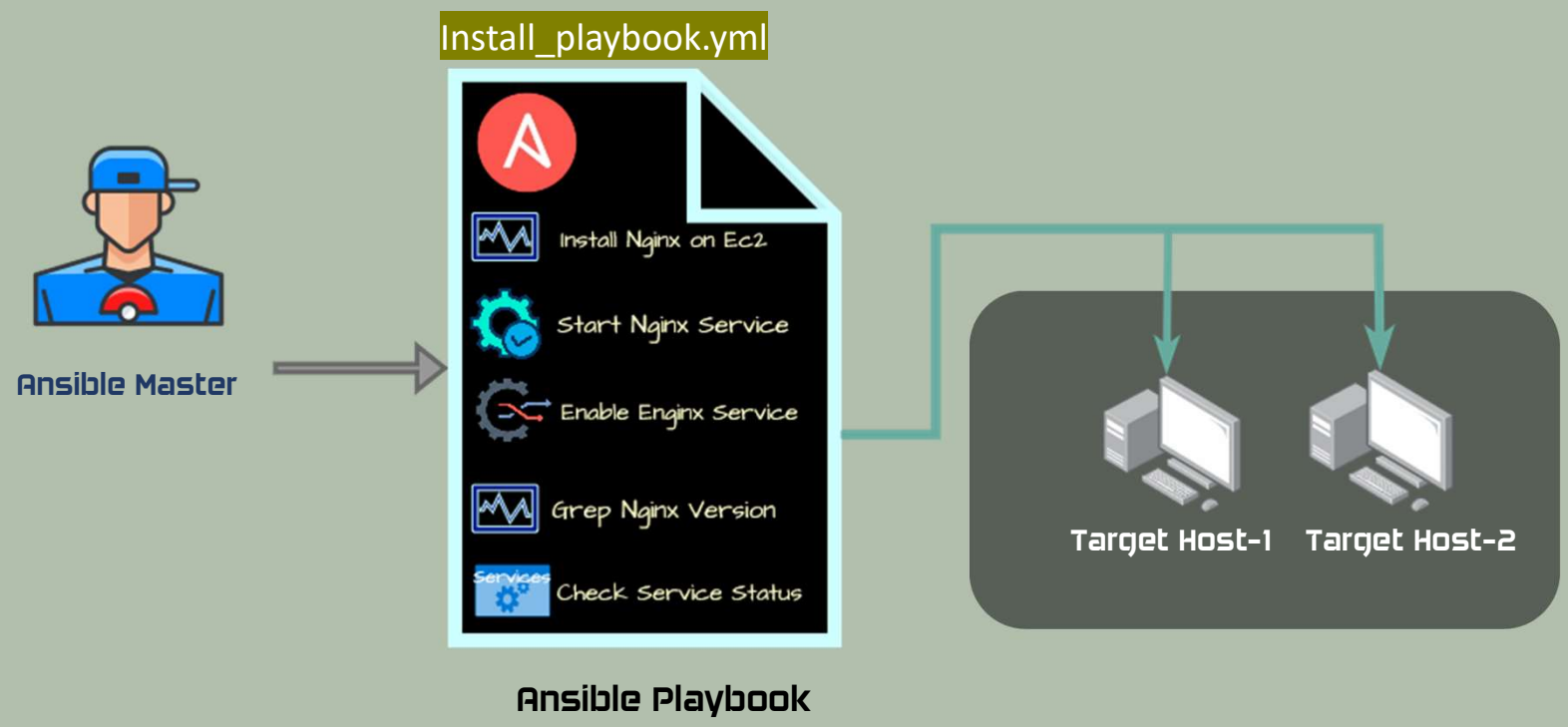
ACL Module - Sets and retrieves file ACL information

COMMAND Module - Executes a command on a remote node

Win_COMMAND Module - Executes a command on a remote Windows node

SCRIPT Module - Runs a local script on a remote node after transferring it

Configuring Ansible using Playbook



Ansible Demo

Lets Automate .....

VARIABLES

DEF: A variable is a symbol which works as a placeholder for expression or quantities that may vary or change

- Ansible uses Jinja2 expressions for variables

LINK1: Jinja2-Varaibles Templates

LINK2: Ansible-Variables

Valid Variable Rules

Valid variable names	Not valid
foo	*foo, Python keywords such as async and lambda
foo_env	playbook keywords such as environment
foo_port	foo-port, foo port, foo.port
foo5, _foo	5foo, 12

Rules for creating VALID Var Names

- Not all strings are valid Ansible variable names.
- A variable name can only include letters, numbers, and underscores.
- Python keywords or playbook keywords are not valid variable names.
- A variable name cannot begin with a number.

Defining and Referencing Variables

- Simple variables combine a variable name with a single value.

base_path: /home/ec2-user

```
- name: copy configuration file to target
  copy:
    src: '{{ base_path }}/inventory.txt'
    dest: "{{ base_path }}/inv/inventory.yml"
```

- Using Single Quote or Double Quote works fine .

Invalid format of Referencing

```
- name: copy configuration file to target
  copy:
    src: {{ base_path }}/inventory.txt
    dest: {{ base_path }}/inv/inventory.yml
```

ERROR! Syntax Error while loading YAML.

Defining and Referencing Variables

Using List of values

```
aws_region:
  - ap-south-1
  - us-west-1
  - us-east-1
```

→

```
region:"{{aws_region[0]}}"
value: ap-south-1
```

Using Key-Value as variable

```
aws_region:
  - ap-south-1: Primary
  - us-west-1: Secondary
  - us-east-1: Default
```

→

```
aws_region['ap-south-1']
aws_region.us-west-1
```

Creating Variable File

- You can define variables in reusable variables files and/or in reusable roles.

variables defined in an external file

```
---
# in the below example, this would be
vars/external_vars.yml
somevar: somevalue
password: magic
foo: host1
```

```
---
# ansible playbook example
- hosts: all
  remote_user: root
  vars:
    favcolor: blue
  vars_files:
    - /vars/external_vars.yml

  tasks:
  - name: This is just a placeholder
    ansible.builtin.command: /bin/echo '{{foo}}'
```

Host and Group Variables

- Host Variables – Assigning a variable to one Machine
- Group Variables – Assigning a variable to group of Machines

Group Variables

```
[atlanta]
host1
host2

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

Inheriting variable values

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh

[southeast:vars]
some_server=foo.southeast.example.com
base_url=/etc/ansible/hosts

[usa:children]
southeast
northeast
southwest
northwest
```

Group and Host Variables

Variables

```
[DC-1]
ansible-target-1
ansible_host=3.110.102.26

[DC-2]
ansible-target-2
ansible_host=13.126.161.14

[datacenters:children]
DC-1
DC-2

[datacenters:vars]
ansible_connection=ssh
ansible_user=ec2-user
```

Host Variables

```
[servers]
ansible-target-1 ansible_host=3.110.102.26 ansible_connection=ssh ansible_user=ec2-user
ansible-target-2 ansible_host=13.126.161.14 ansible_connection=ssh ansible_user=ec2-user
```

Runtime Variables

- You can define variables when you run your playbook by passing variables at the command line using the --extra-vars (or -e) argument.

KEY=VALUE Format

```
ansible-playbook release.yml --extra-vars "version=1.23.45 app_name=nginx"
```

json Format

```
ansible-playbook release.yml --extra-vars '{"version":"1.23.45",“app_name":“nginx"}'
```

Vars from Json or Yaml File

```
ansible-playbook release.yml --extra-vars "@some_file.json"
```

Registering Variables

- You can create variables from the output of an **Ansible task with the task keyword register**. You can use **registered variables in any later tasks** in your play.

```yaml
- hosts: web_servers

  tasks:

     - name: Run a shell command and register its output as a variable
       ansible.builtin.shell: /usr/bin/foo
       register: foo_result
       ignore_errors: true

     - name: Run a shell command using output of the previous task
       ansible.builtin.shell: /usr/bin/bar
       when: foo_result.rc == 5
```

Lets start Ansible Module Creation

Thank you all ☺

Please IM or email me personally your feedback on this session. which will definitely encourage me to make more such Sessions in future ☺