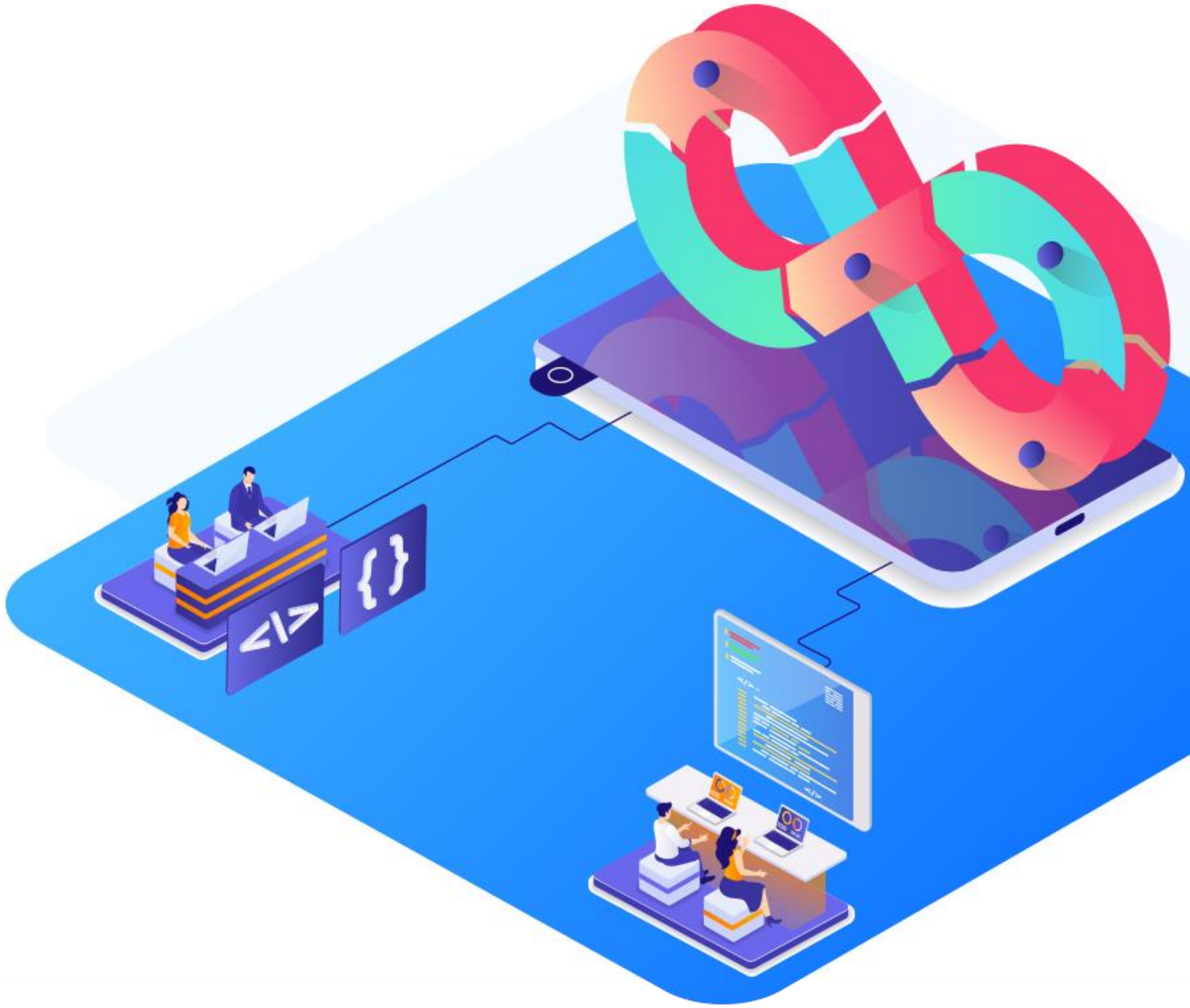
Configuration Management with Ansible and Terraform



Ansible Ad Hoc Commands

Learning Objectives

By the end of this lesson, you will be able to:

- Execute Ansible ad hoc commands for basic tasks such as file, package, and service management across multiple hosts
- Analyze the impact of ad hoc command execution on system operations to understand their performance and scalability benefits
- Optimize task efficiencies through parallel execution strategies, significantly reducing execution time on multiple servers
- Integrate advanced modules using ad-hoc commands to streamline a wide array of administrative tasks effectively

Introduction to Ad Hoc Commands

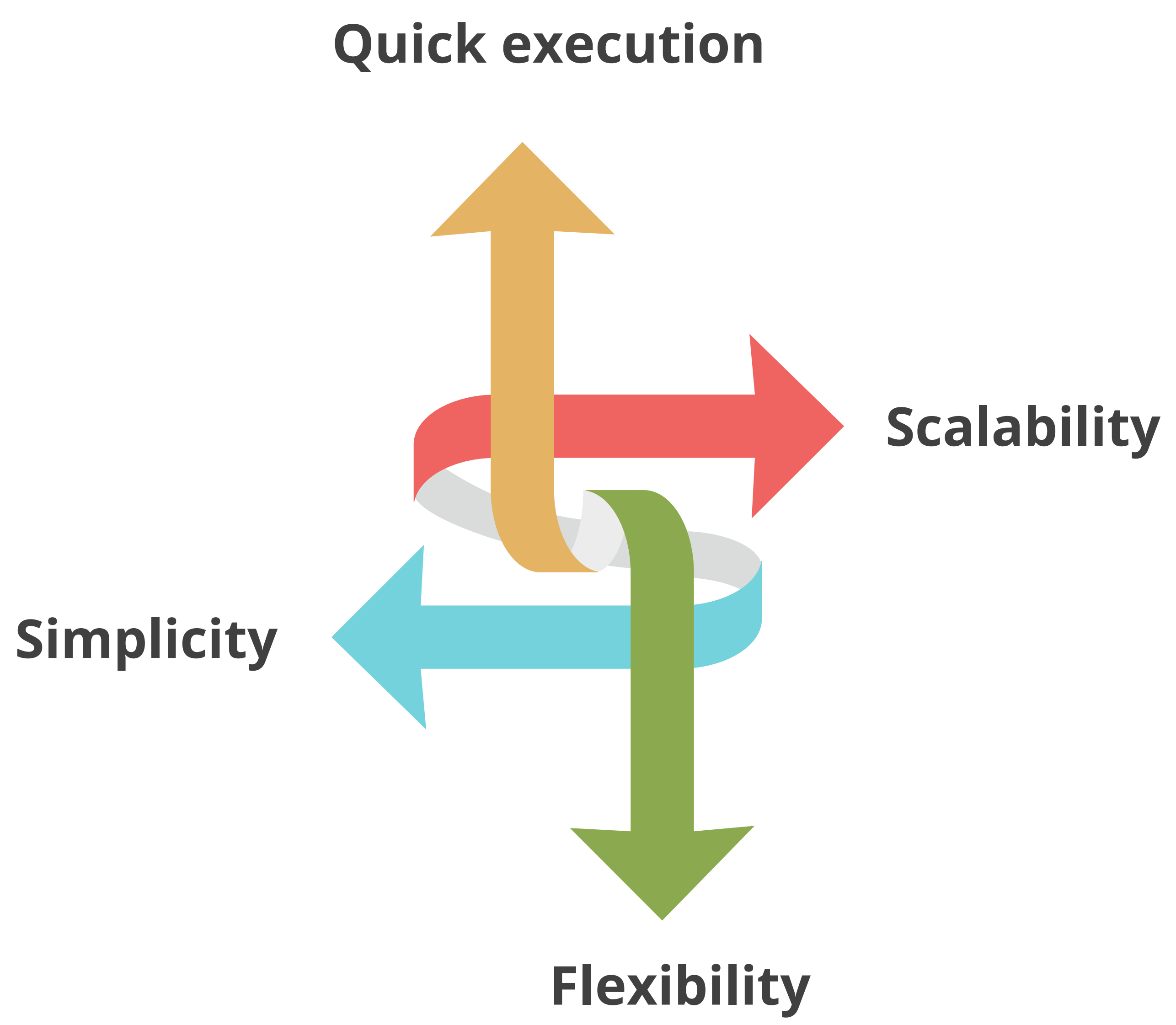
What Are Ansible Ad Hoc Commands ?

Ansible ad hoc commands are powerful one-liners used for quick tasks and can be executed without writing a playbook.

These commands offer a direct method to manage system configurations and run commands swiftly across various hosts.

Features of Ad Hoc Commands

Some attributes of ad hoc commands include:



Features of Ad Hoc Commands

Quick execution

These commands allow you to execute tasks immediately without the overhead of setting up a playbook, making them ideal for rapid, one-off tasks.

Simplicity

The ad hoc commands use simple, concise syntax that is easy to learn and remember, making them accessible even to users new to Ansible.

Features of Ad Hoc Commands

Flexibility

Ansible supports a wide range of modules that can be utilized directly in ad hoc commands. This allows for a diverse set of tasks, including the management of packages, files, services, and more.

Scalability

These commands can be used to execute tasks across multiple hosts simultaneously, making it easy to scale operations.

Why Ad Hoc Commands?

Ansible's ad hoc commands offer a quick, interactive, and effective way to complete small tasks. These are some points that describe the need for ad hoc commands:

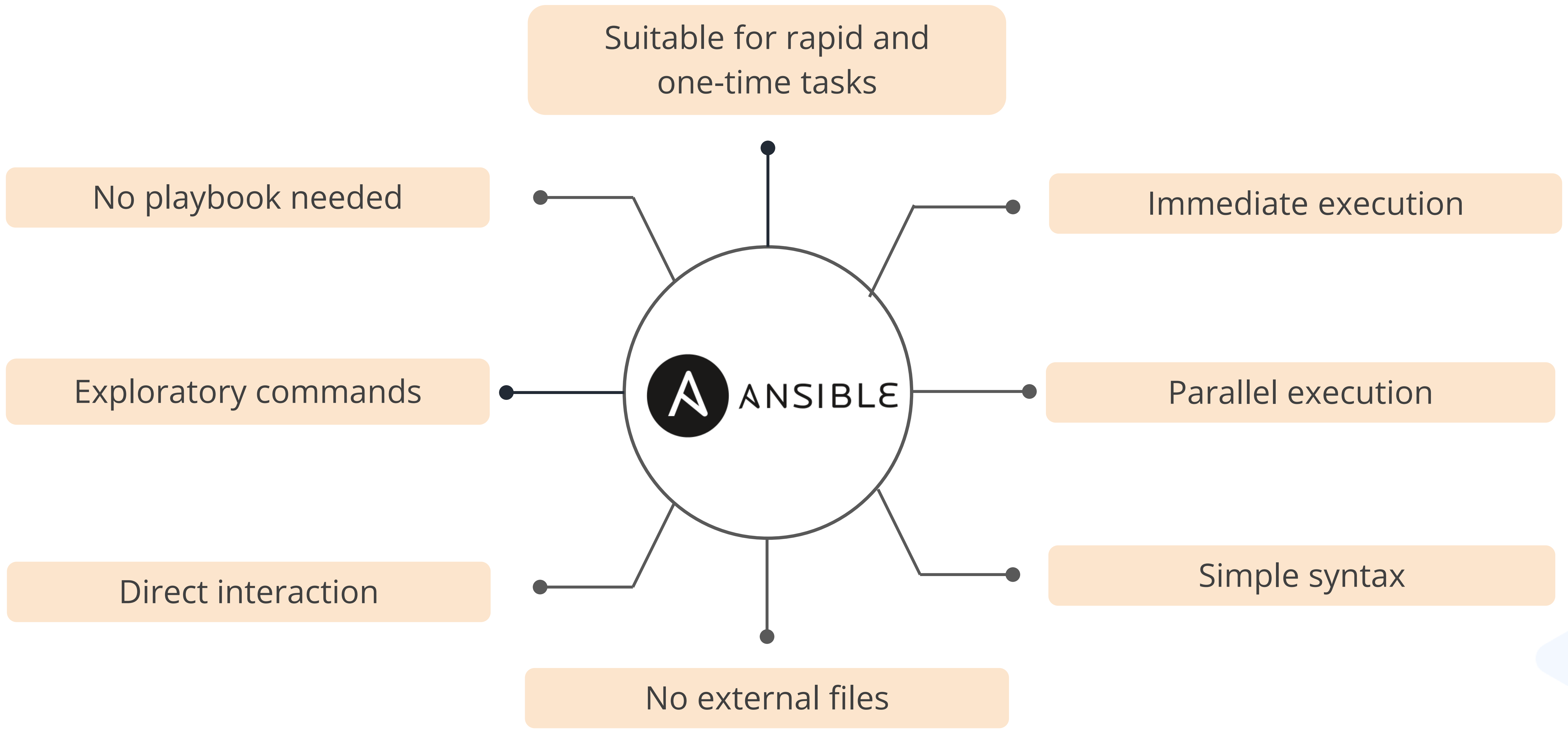
1. Ad hoc commands are great for tasks you repeat rarely.
2. They offer simplicity and immediacy, making them suitable for ad-hoc management needs.
3. They make use of the Ansible command line device to automate a single task on at least one managed node.

Syntax for Ansible Ad Hoc Commands

You can use a single command to efficiently run the syntax for Ansible ad hoc command at scale or concurrently on several hosts.

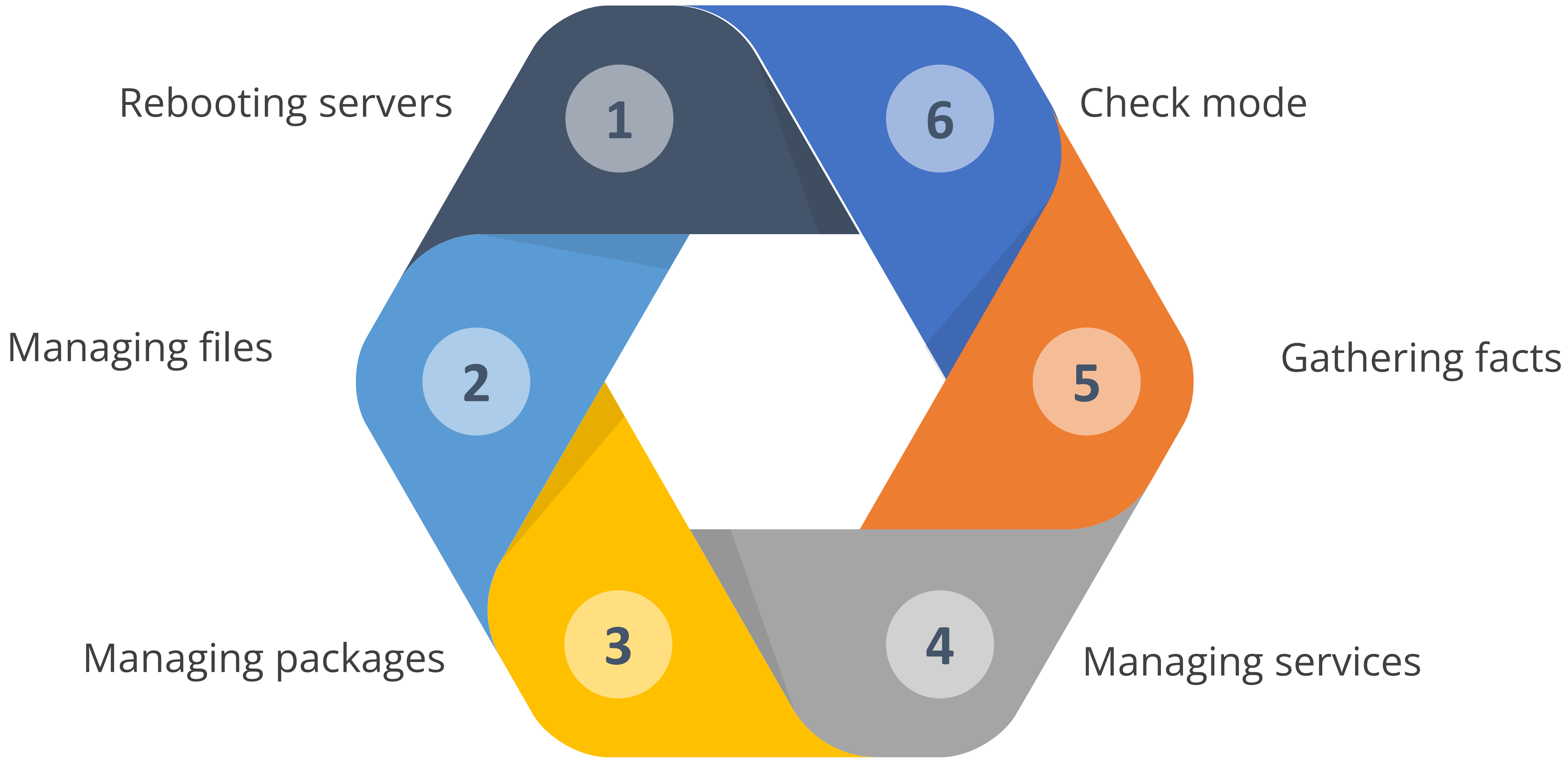
```
Ansible <host-pattern> -m <module-name> -a “<module-
command>”
```

Advantages of Ad Hoc Commands



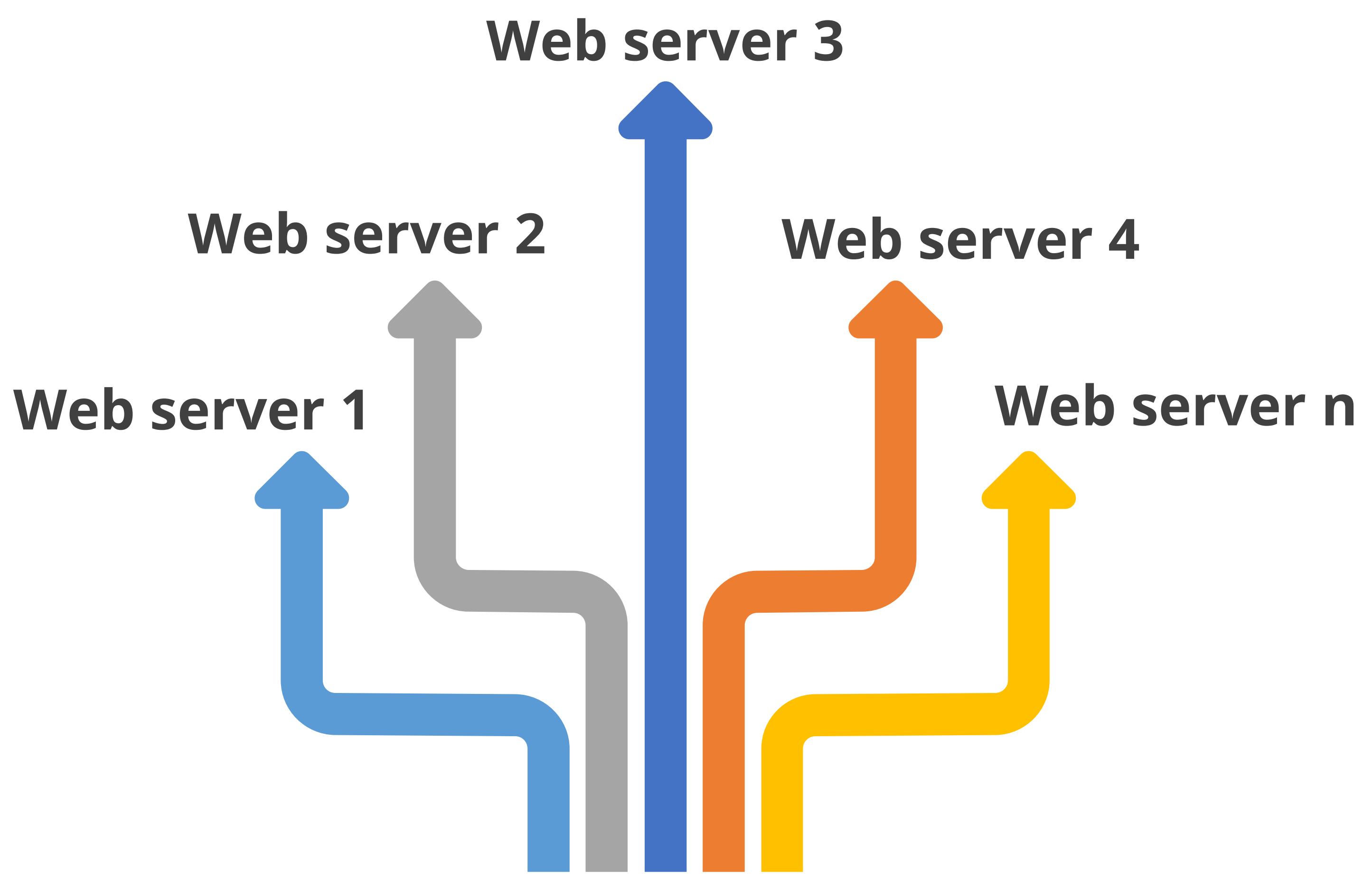
Use Cases for Ad Hoc Commands

The following are use cases available for ad hoc tasks:



Parallelism in Ansible

Parallelism helps users to create multiple resources at once while saving time.



Parallel execution makes it easier to scale operations to manage a large number of hosts or servers efficiently.

Methods to Activate Parallelism

Task-level parallelism

Execute individual tasks asynchronously and define how often to check for completion

Global parallelism

Configure the maximum number of simultaneous processes across all hosts

Play-level parallelism

Set the degree of concurrency for tasks within a specific play

Shell Commands

The shell command is used when users execute commands on remote servers. By default, all commands are executed on the **/bin/sh** shell.

- These commands are executed under the user environment, which can be controlled by various parameters.
- These commands in Ansible effectively require a careful approach to ensure the security, reliability, and maintainability of your automation scripts.
- They are versatile and often used in scenarios where built-in Ansible modules do not suffice.

Shell Commands

Here is how to run a single command using Ansible shell:

Specify the module	Begin by specifying the Ansible module you want to use, which in this case is the shell module
Define the task	Define a task in your Ansible playbook to execute the desired shell command on remote hosts
Provide the command	Provide the shell command within the task, whether a straightforward command or a complex one utilizing pipes, redirects, or additional shell functions
Specify the shell	Specify the shell you want to use for executing the command using the executable parameter. By default, Ansible uses /bin/sh.

File Transfer

It is the process of moving or managing files and directories across multiple remote hosts from a control node using Ansible's automation capabilities.

Here are the file transfer tasks and their commands:

Task	Command
To transfer a file directly to many servers	Ansible atlanta -m copy -a "src=/etc/hosts dest=/tmp/hosts"
To create directories	Ansible webservers -m file -a "dest=/srv/foo/a.txt mode=600"

File Transfer

Here are the file transfer tasks and their commands:

Task	Command
To change ownership and permissions on files	Ansible webservers -m file -a "dest=/srv/foo/b.txt mode=600 owner=mdehaan group=mdehaan"
To create directories via a particular user	Ansible webservers -m file -a "dest=/path/to/c mode=755 owner=mdehaan group=mdehaan state=directory"
To delete directories and files	Ansible webservers -m file -a "dest=/path/to/c state=absent"

Managing Packages

It is a process that automates the installation, update, and removal of software packages on remote hosts through Ansible's modular framework.

Here are various package management tasks and commands using YUM:

Task	Command
To check if a package is installed	Ansible webservers -m yum -a "name=acme state=present"
To check if a package is installed with a specific version	Ansible webservers -m yum -a "name=acme-1.5 state=present"
To check if a package has the latest version	Ansible webservers -m yum -a "name=acme state=latest"
To check if a package is not installed	Ansible webservers -m yum -a "name=acme state=absent"

Users and Groups

The user module allows easy creation and manipulation of existing user accounts, as well as the removal of user accounts that may exist.

Example for managing user accounts on remote systems:

```
Ansible all -m user -a "name=foo password=<crypted
password here>"

Ansible all -m user -a "name=foo state=absent"
```

Managing Services

These are the commands to manage services in Ansible:

Task	Command
To check if a service is started on all web servers	Ansible webservers -m service -a "name=httpd state=started"
To restart a service on all web servers	Ansible webservers -m service -a "name=httpd state=restarted"
To check if a service is stopped	Ansible webservers -m service -a "name=httpd state=stopped"

Gathering Facts

Facts are described in the playbooks section and represent discovered variables about a system. These can be used to implement conditional execution of tasks and to get ad hoc information about the system.

The syntax for gathering facts:

```
Ansible all -m setup -a
'filter=Ansible_distribution*'
```

It is also possible to filter this output to export particular facts by setting up the module.

Assisted Practice

Executing ad hoc commands

Duration: 10 minutes

Problem Statement:

You have been assigned the task of executing ad hoc commands to check memory usage, execute the command as root user on host, and create a UNIX user group.

Outcome:

You can successfully check memory usage, run commands as root, and can create a new UNIX user group on the hosts using ad hoc commands.

Note: Refer to the demo document for detailed steps

Assisted Practice: Guidelines

Steps to be followed:

1. Set up the web server
2. Execute Ansible modules on a local server
3. Execute free –m commands on remote hosts

Quick Check

You are responsible for deploying a large-scale application across multiple servers and want to ensure the deployment process is completed as quickly and efficiently as possible. Which method would you use to maximize parallel execution during this deployment?

A. Task-Level Parallelism

B. User Module

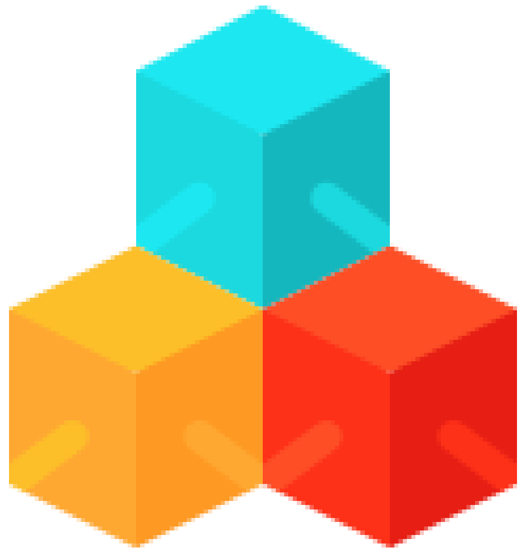
C. Play-Level Parallelism

D. Global Parallelism

Ansible Module

Introduction to Ansible Modules

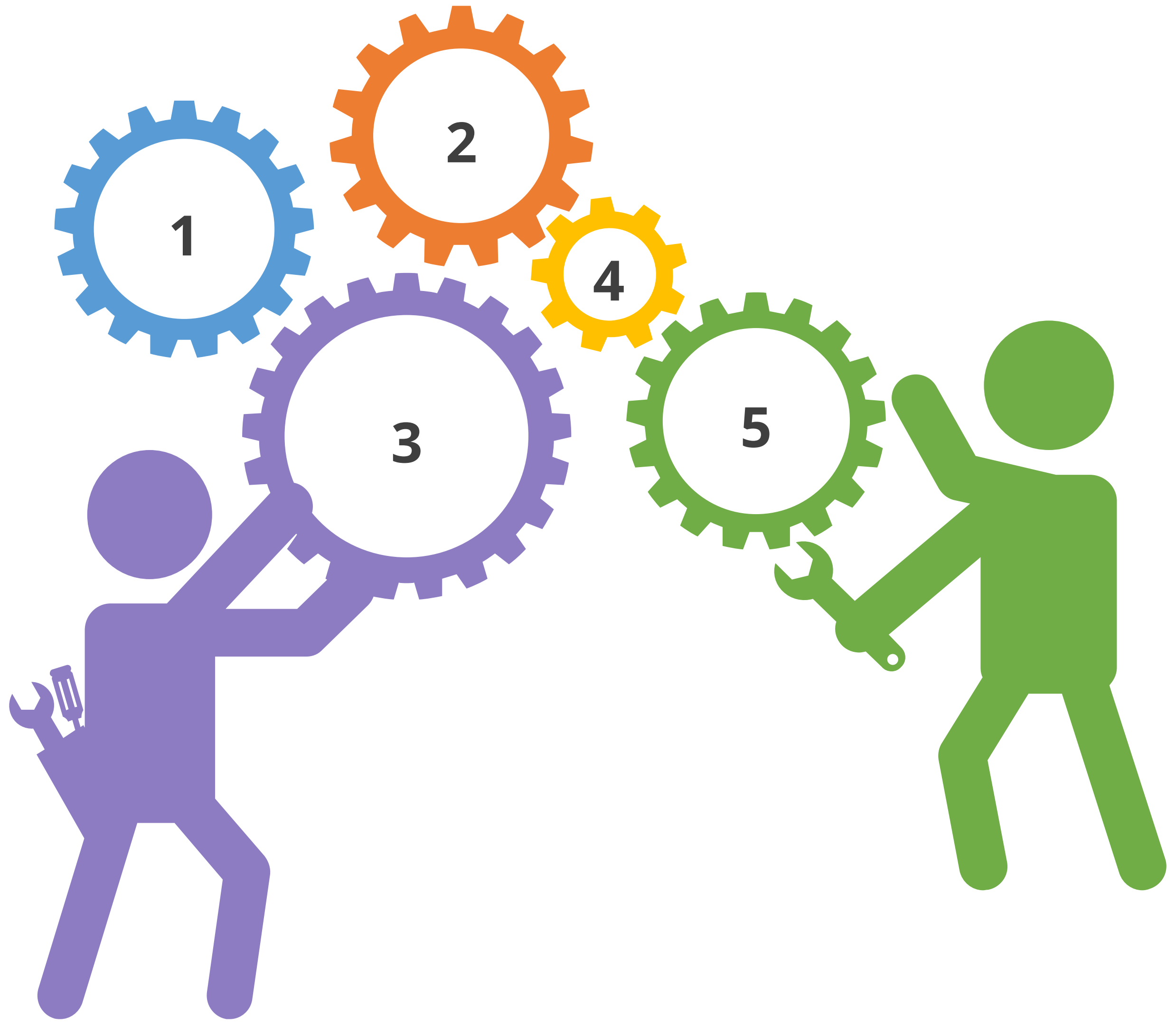
They are discrete units of code which can be used from the command line or in a playbook task.



These modules are the building blocks of Ansible automation, allowing users to define the desired state of their systems and infrastructure.

Key Features of Ansible Modules

Ansible modules play a central role in simplifying and accelerating IT automation.



Reusability

Idempotence

Consistent interface

Customizability

Error handling

Common Ansible Modules

Ansible modules are the standalone scripts that are used inside the Ansible playbook. The playbook includes a play, and a play includes different tasks.

Ping Module

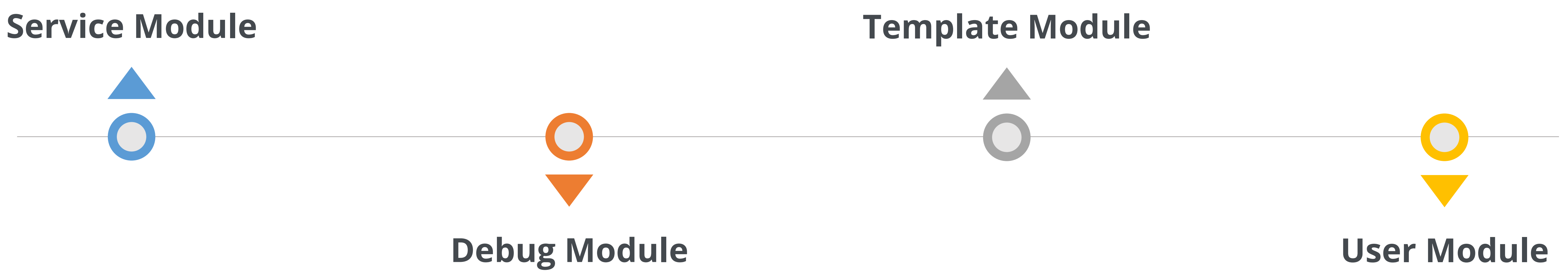
Setup Module

Copy Module

Yum Module

Common Ansible Modules

They ensure idempotency, meaning they only make necessary changes. This reliability avoids unintended modifications and supports a consistent infrastructure state.



Ping Module

Ping helps the user check whether the connection with the hosts mentioned in the inventory file is established.

Purpose

Usage

Response

Indicators

Ping Module

Ping helps the user to check whether the connection with their hosts mentioned in the inventory file is established or not.

The structure of a ping module:

```
Open command line and run:

Ansible test-servers -m ping -u ec2-user
```

Ping changes to pong if an SSH connection is established.

Setup Module

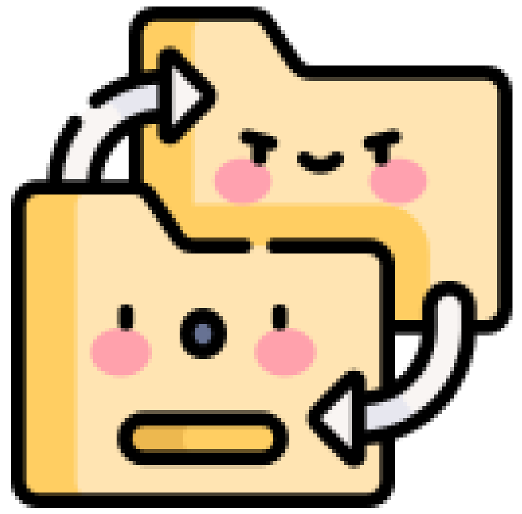
The primary purpose of the setup module is to allow the user to check all the hosts, their configurations, and comprehensive information.

The structure of a setup module:

```
Open command line

Run Ansible test-servers -m setup -u ec2-user
```

Copy Module

The copy module allows the copying of a file from the Ansible control node to the target hosts.



It provides a straightforward way to transfer files and directories, making it useful for tasks like distributing configuration files, scripts, or any other files needed on the remote hosts.

Copy Module: Syntax

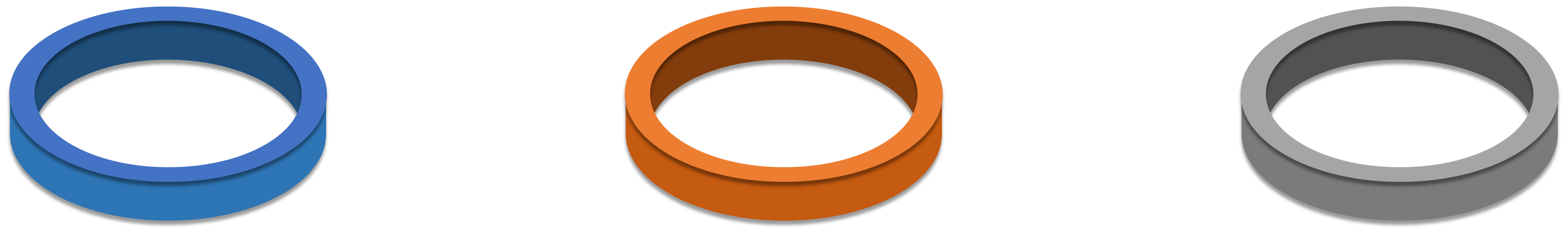
The structure of a copy module:

```
- name: Copy file or directory to remote hosts
  copy:
    src: /path/to/source
    dest: /path/to/destination
    owner: owner_name  # Optional
    group: group_name  # Optional
    mode: 0644         # Optional
    backup: yes        # Optional, creates backup (default is
no)
    remote_src: yes    # Optional, uses remote as source
(default is no)
    validate: checksum # Optional, validate file contents
(default is None)
    force: yes         # Optional, overwrite existing
destination (default is yes)
```

Yum Module

The yum module in Ansible is used for managing packages on systems that use the yum package manager.

Here are some key features of yum module:



Package management

Repository management

State management

Yum Module

Yum is a package management tool. It allows users to obtain, install, delete, query, and manage packages.

The structure of a yum module:

```
Open command line and run:

Ansible test-servers -m yum -a 'name=httpd state=present' -
become -u ec2-user
```

Shell Module

It is used to execute shell commands in the remote target machines. The shell module takes the command name and a list of space-delimited arguments.

It is useful for running commands that cannot be directly handled by Ansible's built-in modules.

Shell Module

The shell module does not execute directly on the target but in a shell environment.

The structure of a shell module:

```
Open command line and run:

Ansible test-servers -m shell -a 'ls -la' -u ec2-user
```

Service Module

It allows you to start, stop, restart, enable, or disable services using predefined service management commands specific to the operating system.

The service module allows you to manage services on remote hosts.

Service Module

The service module guarantees the status of the service that it is operating.

The structure of a service module:

```
Open command line and run:

Ansible test-servers -m service -a 'name=httpd state=started'
-become -u ec2-user
```

Debug Module

This module is part of Ansible-core and included in all Ansible installations.

This module prints statements during execution and can be used for debugging variables or expressions without necessarily halting the playbook.

Debug Module Structure

The debug module prints a message on hosts.

The structure of a debug module:

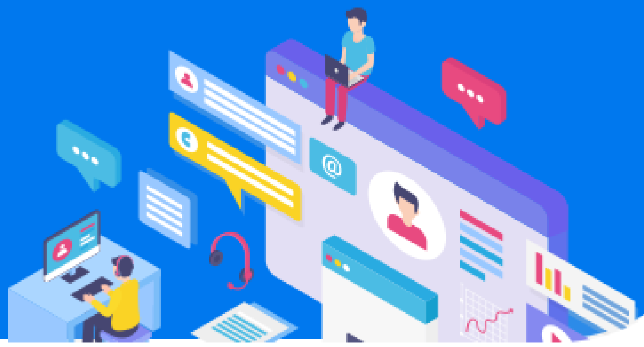
```
Open command line

Run Ansible test-servers -m debug -a 'msg=Hello' -u ec2-user
```

Template Module

With the Ansible template, users can dynamically generate text-based files using templates, variables, and facts for configuration and other purposes.

The template module allows flexible and scalable configuration management, enabling consistency across your infrastructure while adapting to each host's specific requirements.

Assisted Practice



Demonstrating the Ansible Modules **Duration: 15 Min.**

Problem statement:

You have been assigned the task of checking the different functionalities of ad hoc modules, including setting up an SSH connection, setting up and configuring different hosts, installation of service, and printing a message on hosts.

Outcome:

you can effectively manage server configurations using ansible module.

Note: Refer to the demo document for detailed steps

Assisted Practice: Guidelines

Steps to be followed:

1. Demonstrate Ansible module

Quick Check

You are an IT professional tasked with ensuring that your Ansible playbooks correctly adjust configurations based on the system's environment, such as installed packages, OS type, and available memory. Which Ansible module would you use to collect this detailed system information before executing the configuration tasks?

A. Ping module

B. User module

C. Setup module

D. Copy module

Ansible Facts

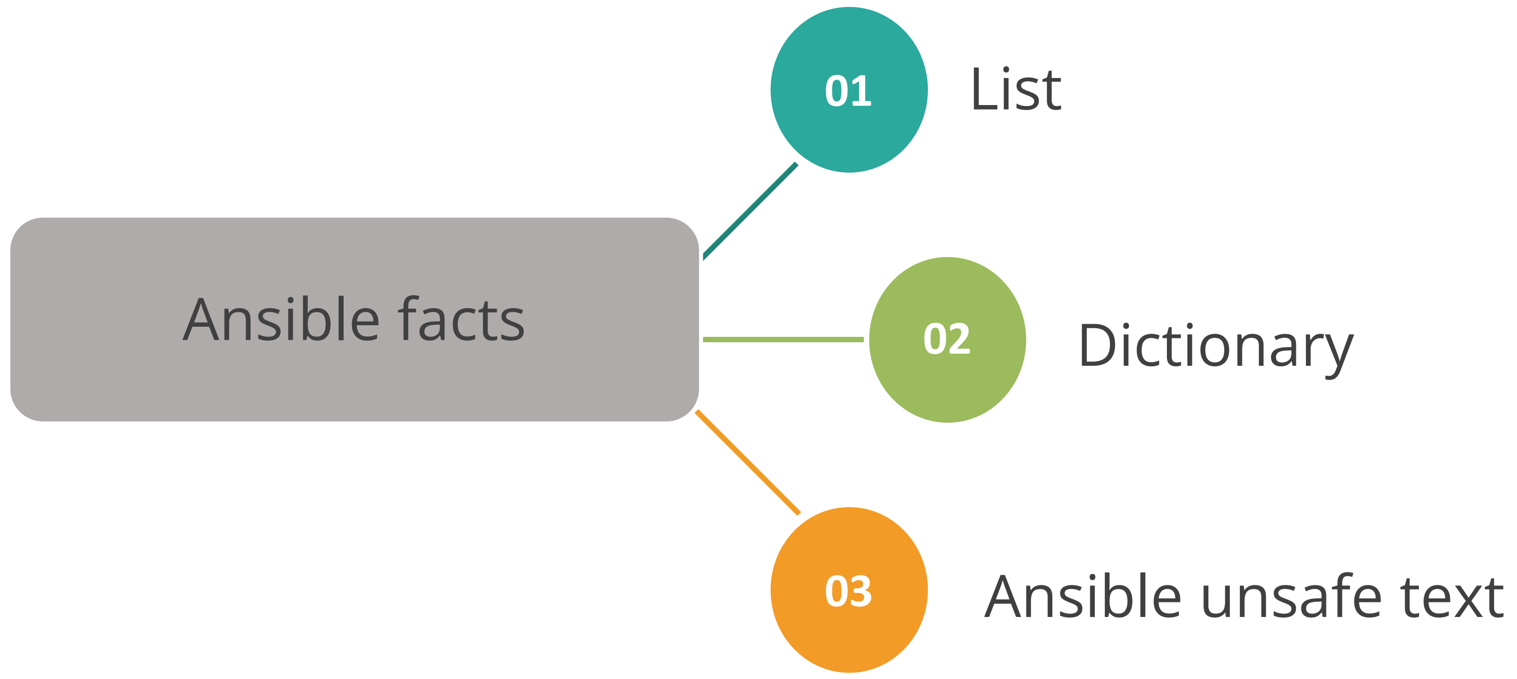
What Are Ansible Facts?

They are data related to your remote systems, including operating systems, IP addresses, and attached filesystems.

- They are stored in JSON format and are used to make important decisions about tasks based on their statistics.
- They are in an Ansible_facts variable, which is managed by Ansible Engine.
- They play a major role in syncing with hosts in accordance with real-time data.

Types of Ansible Facts

The set of facts gathered from remote hosts by Ansible can be categorized into three main types:



These categories help to understand and manage the data collected by Ansible for various configuration and automation tasks.

Custom Facts

Custom facts in Ansible allow users to extend the standard set of gathered facts by providing additional, user-defined information.

There are two types of custom facts:

Data

These facts are defined using simple text files, typically in JSON or INI format, and are static in nature.

Code

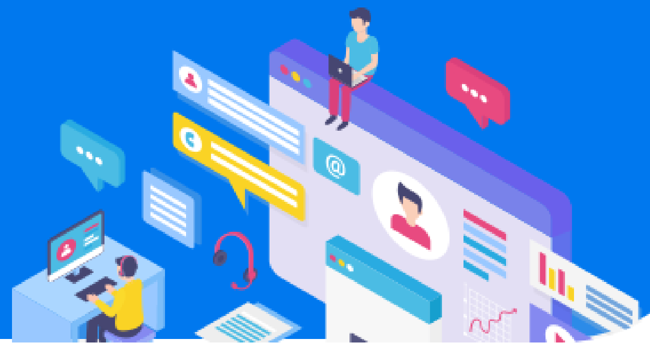
These facts are defined using executable scripts that generate dynamic information at runtime.

How to Declare Custom Facts?

Here are the steps involved in custom facts declaration:

1. Create custom facts file on control node
2. Create a playbook with two different plays
3. Run the playbook on fileservers
4. Verify Custom local facts and Samba Service

Assisted Practice



Working with Ansible Facts **Duration: 10 Min.**

Problem statement:

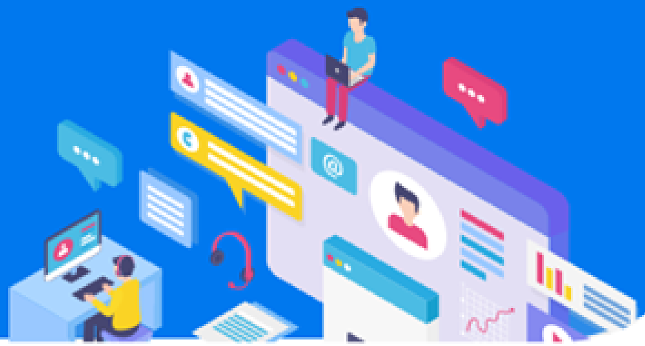
You have been assigned a task to address inconsistent configurations that cause deployment issues. Implement a standardized method for collecting and using system information to ensure reliable automation.

Outcome:

By the end of this demo, you will understand how to gather, manage, and utilize Ansible facts to streamline and enhance your automation workflows.

Note: Refer to the demo document for detailed steps

Assisted Practice: Guidelines



Steps to be followed:

1. Gather all facts
2. Filter specific facts

Quick Check

You are managing a large-scale deployment using Ansible. You need to collect and categorize various facts from remote hosts to automate configuration and reporting tasks. Which of the following types of facts collected by Ansible would you use to store the network interfaces' configurations on multiple hosts?

A. List

B. Dictionary

C. Ansible unsafe text

D. YAML

Key Takeaways

- Ansible's ad hoc commands are designed for quick tasks that don't require ongoing changes, making them straightforward and efficient.
- Ansible's parallel execution saves time and increases efficiency by executing tasks on multiple hosts simultaneously.
- Ansible offers various modules that help manage everything from software installation to file organization, ensuring tasks are handled accurately.
- Ansible categorizes gathered system data into lists, dictionaries, and unsafe text formats, simplifying the customization and automation of configurations across various environments.

Key Takeaways

- Ansible's simple command structure allows immediate task execution, speeding up operational responses.
- Ansible's diverse modules enable tailored automation for various system administration tasks, enhancing efficiency across environments.

Using Ansible to Display System Facts, Manage Files, and Services

Duration: 25 Min.

Project Agenda: To demonstrate how to use Ansible to display system facts, create directories and files, install packages, and manage services

Description: As a DevOps engineer at InnovateNow Tech, a rapidly growing tech company, you are tasked with automating the deployment of Apache web servers to handle a dynamic infrastructure that requires frequent updates and secure management of sensitive data. To do this, you are required to gather system information for all the required hosts/inventories in the network. You have decided to use Ansible facts and ad-hoc commands to accomplish this task.

LESSON-END PROJECT

Using Ansible to Display System Facts, Manage Files, and Services

Duration: 25 Min.

Perform the following:

1. Create a directory and update the playbook
2. Execute the Ansible playbook

Expected deliverables: Ansible playbook that successfully demonstrates displaying system facts, managing files, and services, including installation and execution steps documented clearly.

Thank You