2-Dec-2024

Internship Day - 91 Report:

Python Fabric is a library and command-line tool for streamlining the process of executing remote or local shell commands and automating tasks, particularly for system administration and deployment workflows. It enables Python developers to write simple or complex task automation scripts efficiently.

Why Do We Need Python Fabric?

1. **Task Automation:** Fabric is useful for automating repetitive system tasks like application deployment, server setup, or log analysis.
2. **Remote Command Execution:** It simplifies the process of running commands on remote machines using SSH.
3. **Multi-Host Orchestration:** Fabric can execute commands across multiple servers simultaneously, ideal for managing clusters or distributed systems.
4. **Integration with Python:** It combines Python's scripting capabilities with remote operations, allowing for complex logic in deployment or maintenance scripts.

Key Features of Python Fabric

1. Remote Execution: Execute shell commands on remote servers via SSH with ease.
2. Local Execution: Run commands locally for setup or testing purposes.
3. File Transfers: Upload and download files to and from remote servers.
4. Parallel Execution: Execute tasks on multiple servers concurrently, enhancing efficiency.
5. Custom Scripts: Write Python scripts to define reusable tasks or workflows.

6. Interactive SSH: Support for interactive SSH sessions for real-time debugging or manual intervention.
7. Secure Connection: Utilizes paramiko, a Python SSH library, ensuring encrypted and secure communication.
8. Extensibility: Easy integration with other Python libraries for extended functionality.

Typical Use Cases

1. **Application Deployment:**
    - Automating deployment pipelines for web or database applications.
2. **Server Management:**
    - Setting up or updating software packages, configurations, or services.
3. **System Monitoring:**
    - Collecting logs, monitoring disk usage, or analyzing system health.
4. **Backup and Recovery:**
    - Automating file transfers for backup purposes.
5. **Multi-Server Orchestration:**
    - Managing multiple servers with a single command or script.

3-Dec-2024

Internship Day - 92 Report:

To install Fabric **on a Linux system, follow these steps:**

python3 --version

pip3 --version

```
python3 --version
pip3 --version
```

sudo apt update

sudo apt install python3 python3-pip

```
sudo apt update
sudo apt install python3 python3-pip
```

Create a simple Python program in the terminal:

```
print("How are you")
```

Another example of Python print function:

```
print("You are available")
```

Exit terminal or file editing:

```
Ctrl+D  # Exit command in the terminal
```

Create a directory for Python scripts:

```
mkdir pythonscript
cd pythonscript
```

Create an empty Python file:

```
touch 1st.py
```

Edit the Python file using vi editor:

```
vi 1st.py
```

Inside the vi editor, add the following code:

```
#!/usr/bin/python
print("Welcome to Python")
```

Run the Python script:

```
python3 1st.py
```

EXAMPLES:

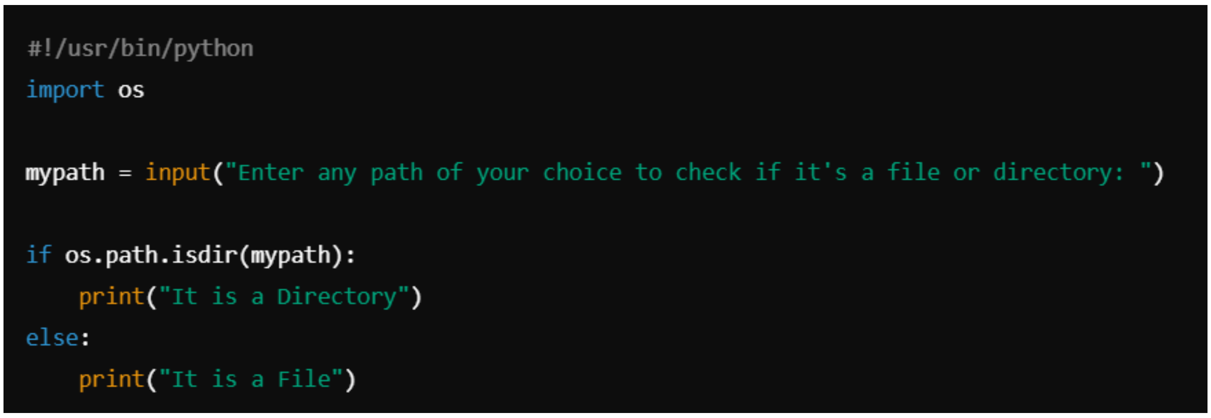
```
#!/usr/bin/python
import os

a = int(input("Enter A value: "))
b = int(input("Enter B value: "))

if a > b:
    os.system("ls")  # Executes 'ls' command in terminal to list directory contents
else:
    os.system("pwd")  # Executes 'pwd' command to print the current directory
```

- #!/usr/bin/python: This is the shebang that indicates the script should be run with Python.
- import os: The os module allows you to interact with the operating system.
- input(): Used to get input from the user, int() converts the input into an integer.
- os.system("ls"): If a is greater than b, the script will run the ls command (list the directory contents).
- os.system("pwd"): If a is not greater than b, the script runs the pwd command to print the current working directory.

EXAMPLES:



```
#!/usr/bin/python
import os

mypath = input("Enter any path of your choice to check if it's a file or directory: ")

if os.path.isdir(mypath):
    print("It is a Directory")
else:
    print("It is a File")
```

- #!/usr/bin/python: Shebang for running the script with Python.
- import os: The os module provides a way to use operating system-dependent functionality.
- input(): Prompts the user to input a path (file or directory).
- os.path.isdir(): Checks if the given path is a directory.
- If true, it prints "It is a Directory".

4-Dec-2024

Internship Day - 93 Report:

What is Ansible?

Ansible is an open-source IT automation tool used for configuration management, application deployment, task automation, and orchestration. It uses simple, human-readable YAML files (called playbooks) to define tasks and does not require agents to be installed on target machines, making it lightweight and easy to use.

Why Use Ansible?

1. **Agentless:** It uses SSH to manage nodes, so no additional software is required on the managed systems.
2. **Simple Configuration:** YAML-based configuration makes it easy to learn and use.
3. **Idempotent:** Ensures tasks are applied only when necessary, avoiding repeated changes.
4. **Efficient for Multi-Node Management:** Automates repetitive tasks across multiple systems.
5. **Wide Use Cases:** Can be used for provisioning, configuration management, application deployment, and continuous delivery.

Ansible is a powerful automation tool with a wide range of features that make it suitable for IT configuration management, deployment, and orchestration. Below are its key features:

Features of Ansible

1. Agentless Architecture

- No agents required: Ansible does not require any software or agents to be installed on managed nodes.
- Uses SSH (or WinRM for Windows) to communicate with remote systems.
- Simplifies setup and reduces resource consumption.

2. Simple and Human-Readable

- Tasks and configurations are defined in YAML files (playbooks), which are easy to read and write.
- No need for specialized coding skills; the learning curve is minimal.

3. Idempotency

- Ensures tasks are applied only if needed, avoiding unnecessary changes.
- Guarantees that the system state is consistent even if a playbook is run multiple times.

4. Cross-Platform Support

- Supports a wide variety of systems, including Linux, Windows, macOS, networking devices, and cloud platforms.
- Can manage hybrid environments seamlessly.

5. Modular and Extensible

- Comes with a wide range of built-in modules to manage tasks like package management, file operations, user creation, and service management.
- Easily extended with custom modules or third-party modules from Ansible Galaxy.

6. Declarative and Procedural Support

- Define the desired state of the system in a declarative way (e.g., "ensure Nginx is installed and running").
- Alternatively, execute procedural tasks with precise steps.

7. Scalability

- Can manage a single node or thousands of nodes efficiently.
- Supports inventory grouping for managing servers based on roles, environments, or other attributes.

8. Built-In Security

- Uses SSH (secure by design) for Linux systems and WinRM for Windows.
- Provides Ansible Vault for encrypting sensitive data like passwords and API keys.

9. Integration with DevOps and Cloud

- Supports popular DevOps tools like Jenkins, Docker, Kubernetes, Terraform, and Git.
- Integrates with cloud platforms such as AWS, Azure, Google Cloud, and OpenStack for provisioning and management.

10. Orchestration

- Handles complex workflows involving multiple servers and services.
- Allows orchestration across diverse infrastructure components, ensuring they work in harmony.

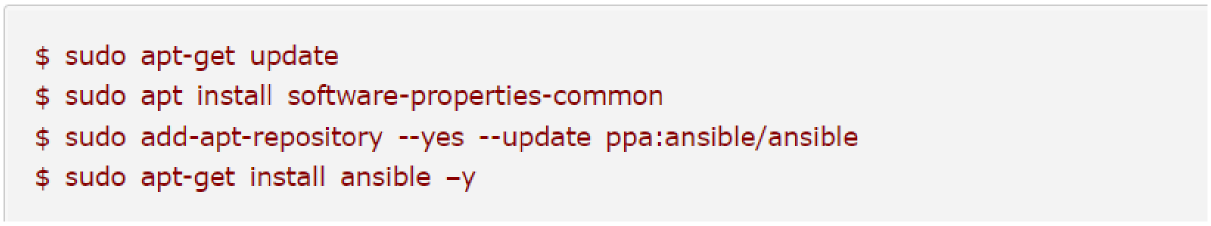
5-Dec-2024

Internship Day - 94 Report:

Installing Ansible un Ubuntu Machine:

You need to install the Ansible software on the machine that will serve as the Ansible control node. From your control node, run the following command

Commands:



```
$ sudo apt-get update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt-get install ansible –y
```

- $ sudo apt-get update
- $ sudo apt install software-properties-common
- $ sudo add-apt-repository --yes --update ppa:ansible/ansible
- $ sudo apt-get install ansible –y

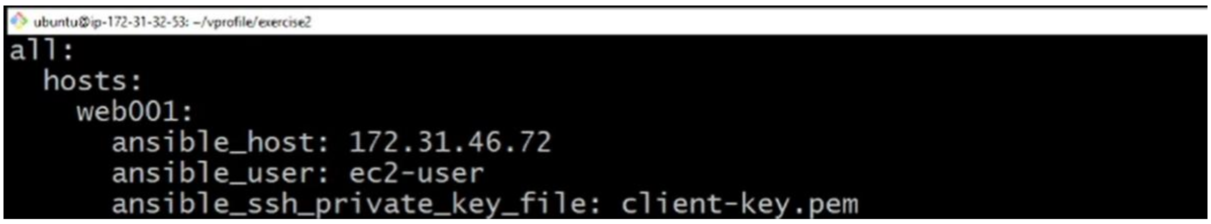
Next, check the version of the installed software using command below

- $ ansible –version

Here it should reflect the ansible version on your console.

Exercise 1:

- **Make a File:**



ubuntu@ip-172-31-32-53: ~/vprofile/exercise2

```
all:
  hosts:
    web001:
      ansible_host: 172.31.46.72
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem
```

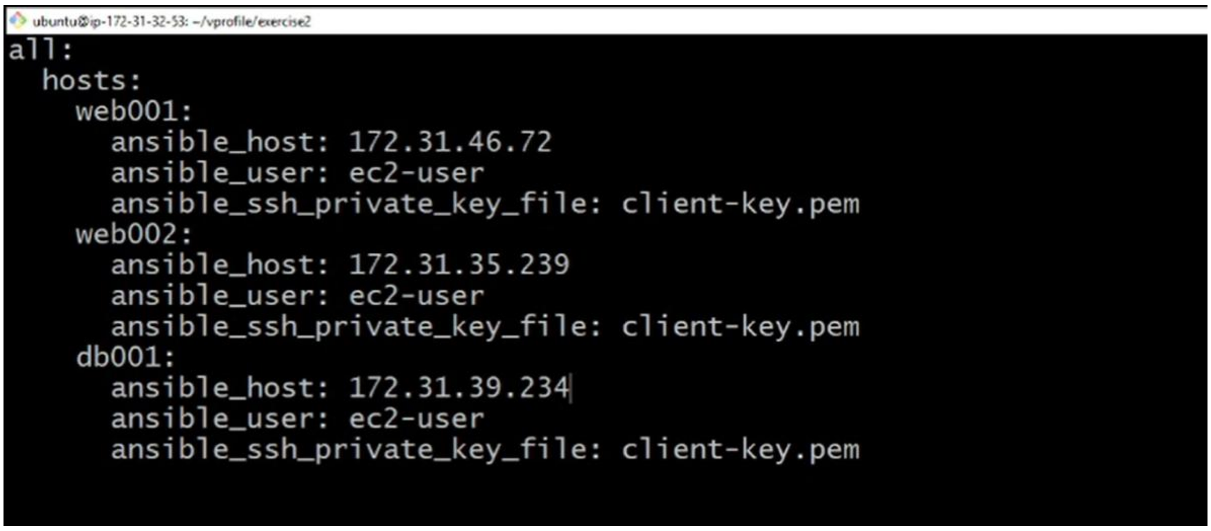
- **Run File:**

ansible web001 –m ping -i inverntory

- **Output Screen:**

Exercise 2:

- **Make a File:**



ubuntu@ip-172-31-32-53: ~/vprofile/exercise2

```
all:
  hosts:
    web001:
      ansible_host: 172.31.46.72
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem
    web002:
      ansible_host: 172.31.35.239
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem
    db001:
      ansible_host: 172.31.39.234
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem
```

- **Run File:**

ansible web001 –m ping -i inverntory

ansible * –m ping -i inverntory

ansible all –m ping -i inverntory

ansible “web*” –m ping -i inverntory

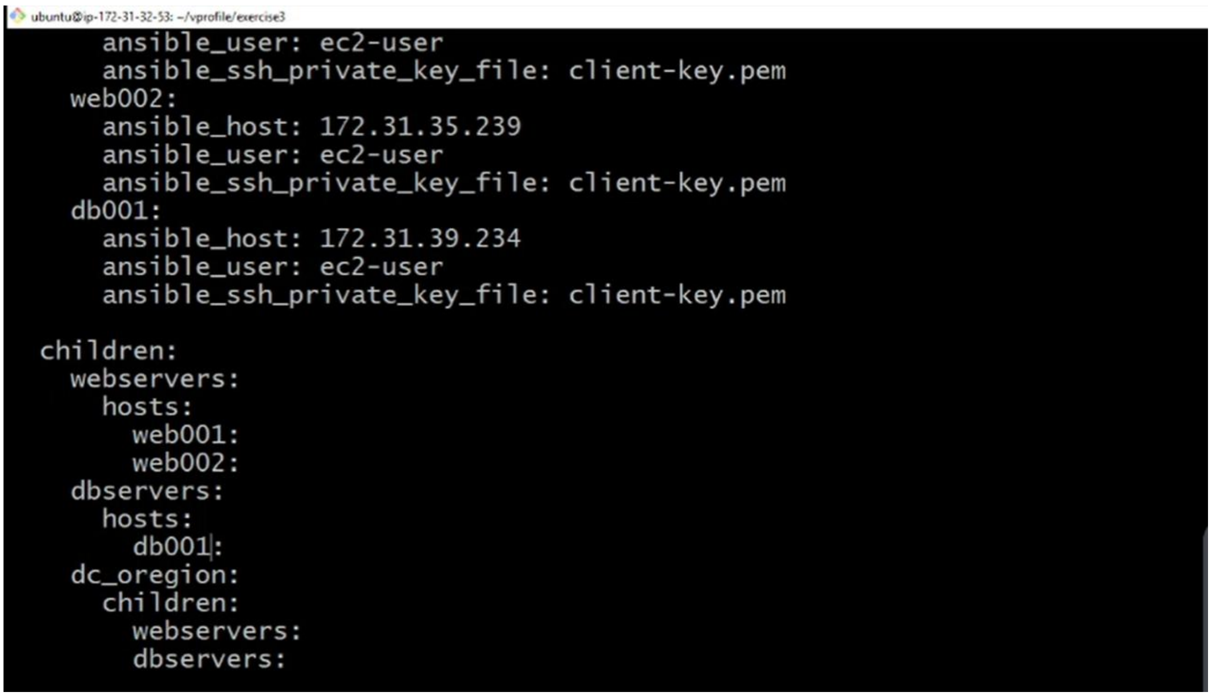
- **Output Screen:**

6-Dec-2024

Internship Day - 95 Report:

Exercise 3: (Grouping)

- **Make a File:**



ubuntu@ip-172-31-32-53: ~/vprofile/exercise3

```
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem
    web002:
      ansible_host: 172.31.35.239
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem
    db001:
      ansible_host: 172.31.39.234
      ansible_user: ec2-user
      ansible_ssh_private_key_file: client-key.pem

  children:
    webservers:
      hosts:
        web001:
        web002:
    dbservers:
      hosts:
        db001:
    dc_oregion:
      children:
        webservers:
        dbservers:
```

- **Run File:**

ansible webservers –m ping -i inverntory

ansible dbservers –m ping -i inverntory

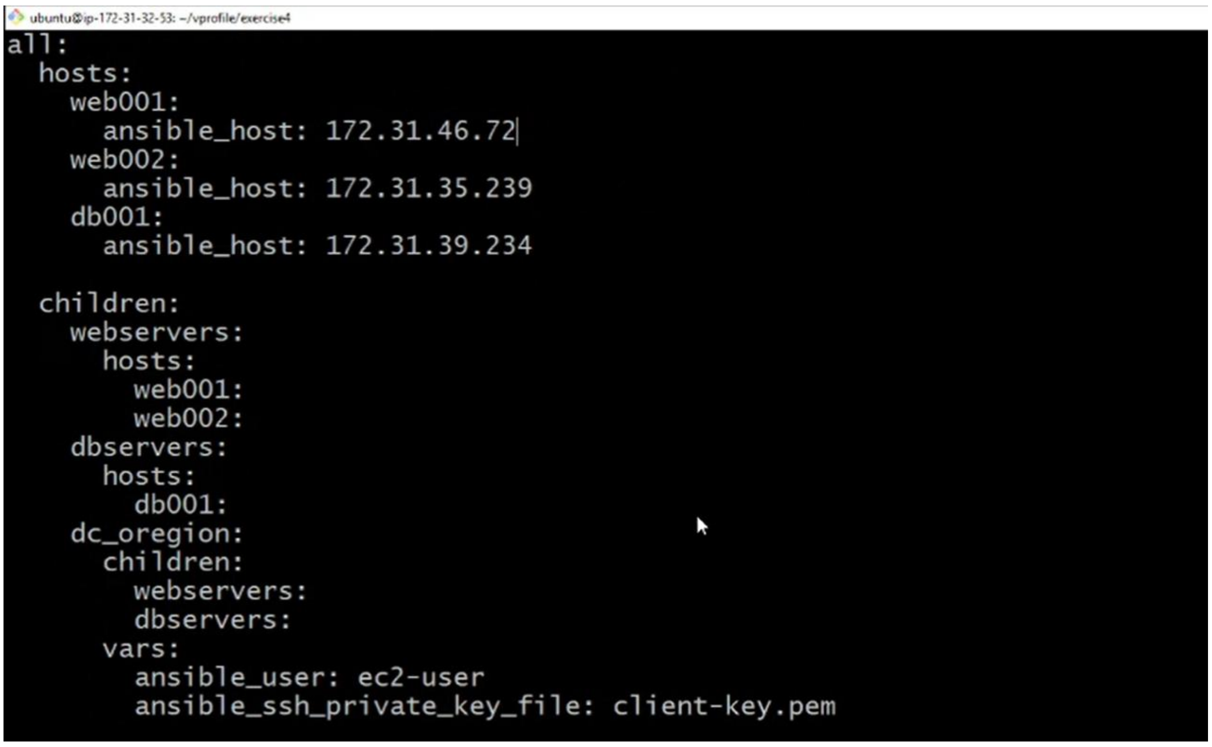
ansible dc_oregion –m ping -i inverntory

ansible “web*” –m ping -i inverntory

- **Output Screen:**

Exercise 4: (Variables)

- **Make a File:**



```
ubuntu@ip-172-31-32-53: ~/vprofile/exercise4
all:
  hosts:
    web001:
      ansible_host: 172.31.46.72
    web002:
      ansible_host: 172.31.35.239
    db001:
      ansible_host: 172.31.39.234

  children:
    webservers:
      hosts:
        web001:
        web002:
    dbservers:
      hosts:
        db001:
    dc_oregion:
      children:
        webservers:
        dbservers:
      vars:
        ansible_user: ec2-user
        ansible_ssh_private_key_file: client-key.pem
```

- **Run File:**

ansible webservers –m ping -i inverntory

ansible dbservers –m ping -i inverntory

ansible dc_oregion –m ping -i inverntory

ansible “web*” –m ping -i inverntory

- **Output Screen:**