Lab Instructions: Automating Local and Remote System Queries with Fabric

1. Ensure that you are in your python virtual environment, it it is not active, use the following command to activate it:

   source my_python_env/bin/activate

Step 1: Prerequisites

1. 1. Install Python **(ignore if it is already done):**

- Ensure Python (3.6 or later) is installed on your system.

- Verify the installation:

  python3 --version

2. Install Paramiko **(ignore if this already done):**

- Paramiko is required for SSH automation. Install it using pip:

  pip install paramiko

3. Create a Virtual Environment **if it is not created** :

- Use a Python virtual environment to isolate your project.

  python3 -m venv venv

- Activate the virtual environment:

  source venv/bin/activate # For Linux/macOS

- Install fabricwithin the virtual environment:

  pip install fabric

Step 2: Creating fabfile.py in your current working directory

1. Create a Fabfile.

    1. In this step, we will create a text file called sftpupload.txt in our local machine. To do so, execute the following command:

        nano fabfile.py

```
from fabric import task, Connection
import subprocess

REMOTE_HOST = "192.168.1.166"
USERNAME = "rps"
PASSWORD = "rps@123"

@task
def query_local_and_remote_info(c):
    """Query system information for both local and remote hosts."""
    try:
        print("\n--- Local System Information ---\n")
        local_commands = [
            ("Hostname", "hostname"),
            ("Network Configuration", "ifconfig"),
            ("System Uptime", "uptime"),
            ("Current Users", "who"),
            ("Memory Usage", "free -h"),
            ("Disk Usage", "df -h"),
        ]
        for desc, command in local_commands:
            print(f"> {desc}:")
            result = subprocess.run(command, shell=True, capture_output=True, text=True)
            if result.returncode == 0:
                print(result.stdout.strip(), "\n")
            else:
                print(f"Error executing '{command}': {result.stderr.strip()}\n")
    except Exception as e:
        print(f"Error obtaining local system information: {e}")

    try:
        print("\n--- Remote System Information ---\n")
        conn = c.Connection if hasattr(c, 'connection') else Connection(
            host=REMOTE_HOST,
            user=USERNAME,
            connect_kwargs={"password": PASSWORD},
        )
        conn.open()
        remote_commands = [
            ("Hostname", "hostname"),
            ("Network Configuration", "ifconfig"),
```

```
        ("System Uptime", "uptime"),
        ("Current Users", "who"),
        ("Memory Usage", "free -h"),
        ("Disk Usage", "df -h"),
    ]
    for desc, command in remote_commands:
        print(f"> {desc}:")
        result = conn.run(command, hide=True)
        print(result.stdout.strip(), "\n")
    conn.close()
except Exception as e:
    print(f"Error obtaining remote system information: {e}")
```

Replace the placeholders `REMOTE_HOST`, `USERNAME`, and `PASSWORD` with your actual server IP, username, and password in the code."

2. Then press **Ctrl+O** to save the changes, then press **Enter**, then **Ctrl+X** to close the nano editor.

3. Now, lets verify if we have created the localfile.txt successfully, to do so, please enter the following command:

   ls

   Here, you can locate **fabfile.py** file which we created.

Step 3: List Tasks in Fabfile

1. List Tasks:

- Use the fab command to list available tasks in the fabfile.py:

fab --list

2. Expected Output:

- You should see the following task listed:

query-local-and-remote-info Query system information for both local and remote hosts.

Step 4: Execute Tasks

1. Run the Task:

- Execute the query-local-and-remote-info task:

fab query-local-and-remote-info

Step 5: Verify Virtual Environment

1. Check Active Virtual Environment:

- Ensure the virtual environment is active. The prompt should include (venv).

2. Deactivate When Done:

- Deactivate the virtual environment to exit:

deactivate

3.