Lab Instructions: Automate Remote Operations with pexpect on Ubuntu

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. 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 pexpect (ignore if this already done):

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

pip install pexpect

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

Step 2: Execute the Python Script

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 Automate_package_download_remote_file_operations.py

Paste the following code into the file.

```
import pexpect

def ssh_connect(host, username, password):

    """Connect to a remote host via SSH using pexpect."""

    ssh_command = f"ssh {username}@{host}"
```

```
    child = pexpect.spawn(ssh_command)
    # Handle password prompt
    try:
        child.expect("password:")
        child.sendline(password)
        child.expect("[$#>] ")  # Wait for shell prompt
    except pexpect.exceptions.TIMEOUT:
        print("Connection timed out.")
        return None
    return child
def install_package(child, package_name):
    """Install a package on the remote machine using a package manager."""
    command = f"sudo apt-get install -y {package_name}"
    child.sendline(command)
    child.expect("password for")  # Sudo password prompt
    sudo_password = input("Enter sudo password: ")
    child.sendline(sudo_password)
    child.expect("[$#>] ")  # Wait for shell prompt
    print(f"Installed {package_name} on the remote machine.")

def upload_file(child, local_path, remote_path):
    """Upload a file to the remote machine using scp."""
    scp_command = f"scp {local_path} {child.before.decode().strip()}:{remote_path}"
    scp_child = pexpect.spawn(scp_command)
    # Handle password prompt
    try:
        scp_child.expect("password:")
        scp_child.sendline(password)
        scp_child.expect(pexpect.EOF)
        print(f"Uploaded {local_path} to {remote_path}.")
    except pexpect.exceptions.TIMEOUT:
```

```
        print("SCP operation timed out.")

def download_file(child, remote_path, local_path):
    """Download a file from the remote machine using scp."""
    scp_command = f"scp {child.before.decode().strip()}:{remote_path} {local_path}"
    scp_child = pexpect.spawn(scp_command)
    # Handle password prompt
    try:
        scp_child.expect("password:")
        scp_child.sendline(password)
        scp_child.expect(pexpect.EOF)
        print(f"Downloaded {remote_path} to {local_path}.")
    except pexpect.exceptions.TIMEOUT:
        print("SCP operation timed out.")

def run_remote_command(child, command):
    """Run a command on the remote machine."""
    child.sendline(command)
    child.expect("[$#>] ")  # Wait for shell prompt
    print(child.before.decode().strip())

def main():
    host = input("Enter the remote host IP or domain: ")
    username = input("Enter the SSH username: ")
    password = input("Enter the SSH password: ")
    # Connect to the remote host
    child = ssh_connect(host, username, password)
    if child is None:
        return
    # Example operations
    package_name = input("Enter the package name to install: ")
```

```
    install_package(child, package_name)

    local_file = input("Enter the local file path to upload: ")

    remote_file = input("Enter the remote file path: ")

    upload_file(child, local_file, remote_file)

    remote_file_to_download = input("Enter the remote file path to download: ")

    local_file_save_path = input("Enter the local path to save the file: ")

    download_file(child, remote_file_to_download, local_file_save_path)


    command = input("Enter a command to run on the remote machine: ")

    run_remote_command(child, command)

    # Exit SSH

    child.sendline("exit")

    child.close()

if __name__ == "__main__":

    main()
```

Execute the script using Python 3:

python3 remote_operations.py

2. Provide Input Parameters:

- Enter the remote host's IP or domain.
- Provide the SSH username and password when prompted.

3. Perform Automated Tasks:

- **Install a Package**: Enter the package name you want to install on the remote machine.
- **Upload a File**: Specify the local file path and the destination path on the remote machine.
- **Download a File**: Provide the remote file path and the destination path on your local machine.
- **Run a Command**: Enter a command to execute on the remote machine.

Step 3: Verify Operations

After each task (e.g., package installation or file transfer), verify that the operation completed successfully by checking the output. For commands, ensure the output matches the expected result.

Step 4: Exit the Remote Host

Once all tasks are completed, the script will exit the SSH session automatically.