

Lab Instructions: Automate the packaging and deployment of files to a remote server using Fabric and shutil package.

Objective:

Automate the packaging and deployment of files to a remote server using Fabric and shutil package.

Prerequisites:

Install Python (Ignore if Already Done)

- Ensure Python 3.6 or later is installed.
- Verify installation using:
- `python3 --version`

Install Fabric (Ignore if Already Done)

- Fabric uses Paramiko for SSH automation. Install it using:
- `pip install fabric`

Create a Virtual Environment (If Not Already Created)

- To isolate your project, create a virtual environment:
- `python3 -m venv venv`
- Activate the virtual environment:
- `source venv/bin/activate` # For Linux/macOS
- `venv\Scripts\activate` # For Windows

Step 1: Create the Fabfile

1. Remove Any Previous Fabfile

Ensure no previous fabfile exists by removing it:

```
rm fabfile.py
```

2. Create a New Fabfile

Use a text editor to create a fabfile:

```
nano fabfile.py
```

Add the following Python script:

```
from fabric import task
```

```
import os
```

```
import shutil
```

```
@task
```

```
def create_archive(c, source_dir, archive_name=None):
```

```
    """Creates a zip archive including the original folder."""
```

```
    folder_name = os.path.basename(source_dir.rstrip(os.sep)) # Ensure folder name is correct
```

```
    archive_name = archive_name or f"{folder_name}.zip" # Name the archive based on the folder
```

```
    shutil.make_archive(folder_name, 'zip', root_dir=os.path.dirname(source_dir), base_dir=folder_name)
```

```
    print(f"Archive {archive_name} created successfully.")
```

```
    return archive_name
```

```
@task
```

```
def upload_and_extract(c, local_archive, remote_path):
```

```
    """Uploads the archive to a remote server and extracts it while preserving the folder name."""
```

```
    remote_archive = os.path.join(remote_path, os.path.basename(local_archive))
```

```
    print("Uploading archive...")
```

```
    c.put(local_archive, remote_archive)
```

```
    print(f"Archive uploaded to {remote_archive}")
```

```
    # Extract to remote_path while preserving the folder name
```

```
    print("Extracting archive...")
```

```
    c.run(f"unzip {remote_archive} -d {remote_path}")
```

```
    c.run(f"rm {remote_archive}") # Optional: Remove the archive after extraction
```

```
    print(f"Extraction complete. Folder structure preserved under {remote_path}.")
```

```
@task

def package_and_deploy(c, source_dir, remote_path):

    """Packages, uploads, and extracts files on the remote server."""

    archive_name = create_archive(c, source_dir)

    upload_and_extract(c, archive_name, remote_path)
```

Press **Ctrl+O** then Click **Enter** and then press **Ctrl+X** to close the file.

Step 2: List Tasks in Fabfile

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

```
fab --list
```

Expected Output: You should see the following tasks listed:

- - create_archive
- upload_and_extract
- package_and_deploy

Step 3: Execute Tasks

1. Run the full deployment task:

```
fab -H myserver --prompt-for-login-password package_and_deploy --
source-dir=./project --remote-path=/home/your_username/deployments
```

2. Replace `myserver` with your remote machine IP address, `source-dir=./project` with `source-dir=./<your_directory>` and `remote-path=/home/your_username/deployments` with `remote-path=/home/<your_remote_username>/`
3. When it prompts for a password, enter the password of your remote user.

Step 4: Verify the Output

1. Log into the remote machine and check the extracted directory:
`ls /home/<your_username>/`
2. Ensure the files are extracted correctly.

Step 5: Verify Virtual Environment

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

To deactivate when done, run:

```
deactivate
```