

✓ LAB 5 - Starter Kit and Automation

🎯 Purpose

The `starter_kit.sh` script is a shell automation tool that helps set up a basic project structure quickly and consistently. It is especially useful for data science or software projects that require organized folders from the beginning.

This script creates a folder named `project/` with three subdirectories: `scripts/`, `docs/`, and `data/`. Each subdirectory also contains a placeholder `README.md` file to describe its purpose.

By automating this repetitive task, the script saves time and reduces setup errors.

📁 What It Creates

After running the script, this folder structure is created:

```
project/  
├── scripts/  
│   └── README.md  
├── docs/  
│   └── README.md  
└── data/  
    └── README.md
```

Each `README.md` file contains a simple title indicating the folder's purpose.

🧪 Example Run

```
📁 Creating project structure...  
✓ Created project/scripts with README.md  
✓ Created project/docs with README.md  
✓ Created project/data with README.md  
  
🎉 Starter Kit Ready!
```

✓ Command:

```
./starter_kit.sh
```

📷 Output image of the code :

```
angel@angel-VirtualBox:~/scripts$ nano starter_kit.sh
angel@angel-VirtualBox:~/scripts$ chmod +x starter_kit.sh
angel@angel-VirtualBox:~/scripts$ ./starter_kit.sh
📁 Creating project structure...
✅ Created project/scripts with README.md
✅ Created project/docs with README.md
✅ Created project/data with README.md

🎉 Starter Kit Ready!
angel@angel-VirtualBox:~/scripts$
```

✅ Summary

This lab demonstrates how automation with shell scripts can improve efficiency during project setup. The `starter_kit.sh` script ensures a consistent starting structure for any new project with minimal manual effort.

?Extra Questions

Q1. What does `mkdir -p` do?

The `mkdir` command is used to create directories in Unix/Linux systems.

The `-p` flag stands for "**parents**", and it has two key benefits:

1. Creates parent directories as needed:

If the parent directories do not exist, `mkdir -p` will create them automatically.

Example:

```
mkdir -p project/scripts
```

This command will:

- Create the `project/` directory if it doesn't exist.
- Then create the `scripts/` subdirectory inside it.
- No error if directory exists:
- If the directory already exists, `mkdir -p` will not throw an error. It just moves on silently.

Q2. Why is Automation Useful in DevOps?

⚙️ What is DevOps?

DevOps is a set of practices that combines software development (**Dev**) and IT operations (**Ops**). Its goal is to shorten the software development lifecycle and deliver high-quality software continuously.

Why Automation Matters

Automation is one of the **core pillars** of DevOps. It enables faster, more consistent, and more reliable workflows by reducing manual effort.

Benefits of Automation in DevOps

1. Speed and Efficiency

- Repetitive tasks like testing, building, and deploying are done automatically.
- Saves time and reduces manual workload.

2. Consistency and Reliability

- Automation ensures the same process runs every time, reducing human errors.
- Configuration and deployment steps are repeatable and version-controlled.

3. Scalability

- Automation can handle large-scale environments effortlessly.
- Infrastructure can be replicated across multiple environments with tools like Terraform or Ansible.

4. Continuous Integration / Continuous Deployment (CI/CD)

- Automation is critical for CI/CD pipelines.
- Code changes are tested and deployed automatically, allowing for rapid and safe releases.

5. Improved Collaboration

- Dev and Ops teams share the same tools and workflows.
 - Automated pipelines reduce friction and encourage better communication.
-

Real-World Automation Examples

Task	Automated Tool Example
Code Testing	GitHub Actions, Jenkins
Deployment	Docker