🧪 Lab 18 — Using Blocks, Rescue & Always in Ansible

Author: *Sandeep Kumar Sharma*

🕑 Learning Objectives

In this lab you will learn: - What `block`, `rescue`, and `always` sections are - How to group tasks logically inside a block - How to handle failures using `rescue` - How to run tasks unconditionally using `always` - How blocks improve readability and error-handling

---

👙 Learning Outcomes

After completing this lab you will: - Write structured playbooks using blocks - Handle task failures without breaking the entire playbook - Execute cleanup tasks even when errors occur - Build more reliable and maintainable automation

---

🧰 What Is a Block in Ansible?

A **block** is a group of tasks written under a single structure. It is used for: - Grouping related tasks - Applying common directives (become, tags) - Using `rescue` and `always` sections for error-handling

---

🧰 What Is Rescue?

`rescue` runs **only when a task inside the block fails**. Useful for: - Rolling back - Retrying with alternative steps - Logging error information

---

🧰 What Is Always?

`always` runs **every time**, whether the block succeeds or fails. Useful for: - Cleanup tasks - Notifications - Status messages

---

🏛️SECTION A — Basic Block Structure (Concept)

```
block:
  - task 1
  - task 2
rescue:
  - task on failure
always:
  - task always runs
```

🛠️ Hands-On Exercise (Very Easy)

This example intentionally runs a command that will fail.

Create file:

```
nano block-example.yml
```

Add:

```
---
- name: Block, Rescue, Always demo
  hosts: dev
  become: yes

  tasks:

    - name: Demo block section
      block:
        - name: Try installing a non-existent package
          package:
            name: no-such-package
            state: present

        - name: This will not run
          debug:
            msg: "Inside block after failure"

      rescue:
        - name: Rescue task when block fails
```

```
      debug:
        msg: "Rescue: The package failed to install. Running backup steps."

    always:
      - name: Always task that runs regardless of success or failure
        debug:
          msg: "Always: Block completed (success or failure)."
```

Run:

```
ansible-playbook block-example.yml
```

Expected behavior: - Block task fails - Rescue section runs - Always section runs

---

🏛️SECTION B — Practical Example: Safe File Edit

```
nano block-edit-file.yml
```

Add:

```
---
- name: Edit file safely using block
  hosts: dev
  become: yes

  tasks:
    - name: Safe update with rescue and cleanup
      block:
        - name: Add line to configuration file
          lineinfile:
            path: /tmp/demo.conf
            line: "demo_setting=enabled"

        - name: Force a failure
          command: /bin/false

      rescue:
        - name: Rollback or notify
          debug:
            msg: "Rescue: Something went wrong while editing the file."
```

```
    always:
      - name: Cleanup confirmation
        debug:
          msg: "Always: File editing attempt finished."
```

Run:

```
ansible-playbook block-edit-file.yml
```

🧪 Hands-On Checklist

- [ ] Run the simple block-rescue-always example
- [ ] Observe rescue executing on failure
- [ ] Observe always executing every time
- [ ] Run safe file edit example

👙 Lab Summary

This lab explained block-based task grouping and how rescue and always help in structured error-handling, rollback, and cleanup operations.

Author: Sandeep Kumar Sharma