**Terraform Cloud + AWS Decommissioning Automation**

**Table of Contents**

1. Overview
2. Objective
3. Automation Workflow
4. Components Involved
5. Terraform Version Handling
6. EC2 Deployment Strategy
7. Key Benefits
8. Appendix

**1. Overview**

This project automates the decommissioning of AWS accounts and their associated Terraform Cloud (TFC) workspaces using a fully non-interactive Python-based solution. It ensures secure login, handles version-specific Terraform execution, and runs from a non-GUI environment like an EC2 instance.

**2. Objective**

* Automatically destroy Terraform-managed infrastructure.
* Remove associated resources from state.
* Delete Terraform Cloud workspaces.
* Close AWS accounts.
* Eliminate all manual steps and support automation via code alone.

**3. Automation Workflow**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Read input from config.json specifying orgs, workspaces, AWS account ID |
| 2 | Fetch required Terraform version from Terraform Cloud workspace metadata |
| 3 | Install and switch to correct version using tfenv |
| 4 | Configure CLI credentials automatically for Terraform |
| 5 | Initialize Terraform backend for the given workspace |
| 6 | Remove specified resources (e.g., S3) from Terraform state |
| 7 | Trigger terraform destroy run using TFC API |
| 8 | Monitor run until completion |
| 9 | Delete workspaces from Terraform Cloud |
| 10 | Optionally initiate AWS account closure (if provided) |

**4. Components Involved**

|  |  |
| --- | --- |
| **File** | **Description** |
| decommission.py | Main workflow runner, calls TFC and AWS modules |
| tfc.py | Handles Terraform Cloud backend config, CLI login, workspace actions |
| aws\_account.py | Optional module for initiating AWS account closure |
| config.json | User-defined config for org, workspace, and resources |
| user\_data | Bootstraps EC2 with required software and code for automation |

**5. Terraform Version Handling**

* Each workspace may require a different Terraform version.
* Version is fetched dynamically via TFC API.
* tfenv is used to install and switch to the required version automatically.

**6. EC2 Deployment Strategy**

* Automation runs on an EC2 instance (no GUI needed).
* EC2 instance is provisioned with:
  + Python + dependencies (requests, tfenv, etc.)
  + Terraform CLI
  + Automation code (cloned or copied)
* Everything is bootstrapped via user\_data for hands-free setup and execution.

**7. Key Benefits**

✅ Fully automated Terraform Cloud workspace and AWS resource decommissioning  
 ✅ Version-aware Terraform execution (no manual switching)  
 ✅ Secure, API-based Terraform login (no manual terraform login)  
 ✅ Cloud-native and headless deployment from EC2  
 ✅ Modular and extensible design for future use cases

**8. Appendix**

* Deployment path: /opt/aws\_decom/AWSAccountDecom/
* Code files: tfc.py, decommission.py, aws\_account.py, config.json
* Cloud-native execution enabled using EC2 user\_data and tfenv
* Logs generated with timestamps for traceability