**Terraform Automation HandsOn Training with AWS**

**📘 Day 1: Terraform Fundamentals & AWS Provisioning Basics (8 Hours)**

**🧠 Concepts**

* Introduction to DevOps & IaC
* What is Terraform and how it works?
* Declarative vs Imperative IaC tools
* Terraform CLI, workflow, providers, and state

**🔨 Hands-On**

* Install Terraform CLI
* Setup AWS CLI & credentials
* Create first Terraform project (S3 Bucket or EC2 instance)
* Use AWS provider
* Write and apply basic .tf files
* Understand Terraform plan/apply/destroy
* Understand resource blocks and meta arguments (count, depends\_on)
* Cleanup with destroy

**📘 Day 2: Variables, Outputs, Modules, and State Management (8 Hours)**

**🧠 Concepts**

* Input variables, local variables, outputs
* Data sources (e.g., AMI lookup, EC2 tags)
* Remote state overview and state file best practices
* Terraform backends (S3 + DynamoDB for locking)
* Reusability with modules

**🔨 Hands-On**

* Use variables with tfvars
* Output key resource values (e.g., public IP of EC2)
* Use terraform\_remote\_state
* Configure S3 as a remote backend (with DynamoDB locking)
* Break project into modules (VPC, EC2, security group)
* Reference output from modules
* Simple module registry usage

**📘 Day 3: AWS Resource Management, Monitoring & Update Automation (8 Hours)**

**🧠 Concepts**

* Infrastructure updates (patches, package upgrades)
* Basic monitoring with CloudWatch via Terraform
* AWS EC2 lifecycle (stop, update, reboot, terminate)
* Terraform provisioners (basic usage)
* Templating with templatefile function
* Managing dependencies

**🔨 Hands-On**

* Create a complete 3-tier architecture (VPC + EC2 + RDS)
* Install and update packages via remote-exec provisioner
* Use user\_data to automate package updates (Amazon Linux)
* Enable CloudWatch monitoring using aws\_cloudwatch\_log\_group, metric\_alarm
* Write alert for high CPU on EC2
* Use depends\_on and ignore\_changes effectively

**📘 Day 4: Best Practices, Collaboration, Testing & Final Project (8 Hours)**

**🧠 Concepts**

* Best practices for production-like Terraform code
* Using Terraform with Git (version control, remote modules)
* Secure variables using environment variables
* Team collaboration tips: workspaces, state isolation
* Testing basics: validate, terraform fmt, plan check
* Overview of CI/CD integration

**🔨 Hands-On**

* Use terraform fmt, validate, and plan in Git workflow
* Create separate workspaces (dev, stage, prod)
* Final project:
  + Provision a VPC + public/private subnets
  + Launch EC2 instance with updates enabled
  + Configure monitoring & alarms
  + Reuse modules
  + Use terraform.tfvars and output key values