Terraform - Academic Batch Training

References:

1. Terraform Basics Training Course Course | KodeKloud
2. HashiCorp Certified: Terraform Associate 2024

Day	Topic	Description	References
Day 1	Understanding Infrastructure as Code (IaC)	1. What is IaC (Infrastructure as Code)? 2. Different IaC Tools.	
	Introduction to Terraform	1. Overview of Terraform 2. Why Terraform?	
	Setting up environment	1. Install Terraform. 2. Setting up AWS / AZURE account. 3. Install VS Code Extensions	**Extension**: 1. https://marketplace.visualstudio.com/items?itemName=HashiCorp.terraform
	Terraform Basics	1. Understanding HCL 2. Terraform Providers and versioning 3. Input Variables 4. Output Variables 5. Resources Blocks 6. Data sources 7. locals 8. tf commands - init, plan, apply, destroy, fmt 9. terraform.tfvars file	**AWS Provider** - Terraform Registry **Azure Provider** - Terraform Registry
	Assignment 1	**AWS** - Provision a S3.	**S3** - Terraform Registry
		Azure - Provision a Storage Account.	**Storage Account** - Terraform Registry
	Terraform Resource Attributes	1. Basics of Attributes 2. Cross-Resource Attribute References	
	Terraform State	1. Terraform State 2. Desired State and Current State 3. tf state command	
	Assignment 2	**AWS** - Create a basic web server on AWS using Terraform to automate the setup of an VPC, subnet, EC2 instance and its associated security group.	**EC2** - Terraform Registry **VPC** - Terraform Registry **Subnet** - Terraform Registry **Security Group** - Terraform Registry
		Azure - Create a basic web server on AZURE using Terraform to automate the setup of an VNet, subnets, virtual machine and network security group.	**Linux VM** - Terraform Registry **Windows VM** - Terraform Registry **VNet** - Terraform Registry **Network Security Group** - Terraform Registry

Day 2	**Assignment 3**	**AWS :** • Provision a VPC with 4 subnets (2 private and 2 public) distributed across different availability zones. • Set up an Auto Scaling Group (ASG) in the public subnets to launch EC2 instances running Apache2. Define the desired capacity as 2 , minimum as 1 , and maximum number of instances as 2 in the ASG. • Create an Application Load Balancer (ALB) in the public subnets. Attach the ASG to the ALB. • Provision an RDS MySQL instance running on port 3306 in the private subnets. • Configure security groups to: ◦ Allow HTTP access (port 80) to the EC2 instances via the ALB from the internet. ◦ Allow the EC2 instances to connect to the RDS instance on port 3306. • Check the DB connectivity from EC2. **Bonus Task:** 1. Configure an autoscale rule: a. Scale out when the average CPU usage exceeds 75%. b. Scale in when the average CPU usage drops below 25%.	
		Azure : • Provision a VNet with 2 subnets. • Set up a Virtual Machine Scale Set (VMSS) with instances running Apache2. • Set up an AZURE Load Balancer in front of the VMSS to distribute incoming HTTP traffic (port 80). Attach the VMSS to the Load Balancer. • Set up an AZURE SQL Database in the private subnets. Ensure it is accessible only from the VMSS instances via the necessary SQL port (1433). • Create NSGs to allow incoming HTTP traffic (port 80) to the Load Balancer. Configure rules to allow traffic between the VMSS and the SQL Database on port 1433. Restrict all other inbound traffic. **Bonus Task:** 1. Configure auto-scaling for the VMSS based on CPU utilization:	

		a. Scale out when the average CPU utilization exceeds 75%. b. Scale in when the average CPU utilization falls below 25%.	
	Review and Feedback	1. Feedback of Mentor. Understand the best practices. 2. Make changes as suggested by Mentor.	
Day 3	State Backend	1. Remote Backends 2. State Lockings 3. Backend using S3 in AWS 4. Backend using Storage Account in AZURE	**AWS S3** - Backend Type: s3 | Terraform | HashiCorp Developer **Azure Storage Account** - Backend Type: azurerm | Terraform | HashiCorp Developer **State locking** - State: Locking | Terraform | HashiCorp Developer
	Meta-Arguments	Explore the following arguments and understand their applications: 1. depends-on 2. count 3. for_each 4. lifecycle 5. provider	1. The depends_on Meta-Argument - Configuration Language | Terraform | HashiCorp Developer 2. The count Meta-Argument - Configuration Language | Terraform | HashiCorp Developer 3. The for_each Meta-Argument - Configuration Language | Terraform | HashiCorp Developer 4. The Resource provider Meta-Argument - Configuration Language | Terraform | HashiCorp Developer 5. The lifecycle Meta-Argument - Configuration Language | Terraform | HashiCorp Developer
	Terraform Provisioners and null_resource	1. local-exec 2. remote-exec 3. file 4. null_resource	1. Provisioner: file | Terraform | HashiCorp Developer 2. Provisioner: local-exec | Terraform | HashiCorp Developer 3. Provisioner: remote-exec | Terraform | HashiCorp Developer 4. **null_resource** - Terraform Registry
	Terraform Functions, Expressions, Workspaces	• Learn different kinds of functions in terraform. Commonly used functions are listed below: ◦ lookup ◦ try ◦ merge ◦ range ◦ toset ◦ file • Learn different kinds of expressions in terraform. Commonly used expressions are listed below: ◦ For Expressions ◦ Conditional Expressions	**Functions** - Functions - Configuration Language | Terraform | HashiCorp Developer **Expressions** - Expressions - Configuration Language | Terraform | HashiCorp Developer

		◦ Dynamic Blocks ◦ String Interpolation ◦ Directives ◦ Splat Expressions ◦ Version Constraints • Workspaces	
	Terraform import, tainting	• import command • taint command • untaint command	Import - Command: import | Terraform | HashiCorp Developer taint - Command: taint | Terraform | HashiCorp Developer untaint - Command: taint | Terraform | HashiCorp Developer
	Assignment 4	**AWS** - Implement Remote Backend using S3 and state lock using DynamoDB.	**AWS S3** - Backend Type: s3 | Terraform | HashiCorp Developer **State locking** - State: Locking | Terraform | HashiCorp Developer
		Azure - Implement Remote Backend using Storage Account.	**Azure Storage Account** - Backend Type: azurerm | Terraform | HashiCorp Developer
Day 4	Assignment 5	Modify the Assignment 3 Terraform template to include the following additional requirements: 1. Create a separate module to provision VPC and subnets with environment tag value as "training". 2. Create separate module to provision an EC2 instance with possible instance name, type, ami, count, tag values etc., as module inputs. Use data source to retrieve the AMI ID. 3. Use provisioner to deploy the mysql-connection.php to the Apache2 instance /var/www/html/ with the provisioned RDS instance as part of this template **Bonus:** **Multi-Region Deployment:** • Create resources (like EC2 instances and S3 buckets) across multiple AWS regions.	
		Modify the Assignment 3 Terraform template to include the following additional requirements: 1. Create a separate module to provision VNet and subnets with environment tag value as "training". 2. Create separate module to provision an Virtual Machine with possible instance name, type, count, tag values etc., as module inputs.	

		3. Use provisioner to deploy the mysql-connection.php to the Apache2 instance /var/www/html/ with the provisioned SQL Database instance as part of this template	
	Review and Feedback	1. Feedback of Mentor. Understand the best practices. 2. Make changes as suggested by Mentor.	
Day 5	Explore	1. tf-docs 2. pre-commit-validations 3. Module versioning 4. DevSecOps Tools - CheckOv, tfsec, infracost. 5. Testing Tools - terratest, Kitchen Test	
	Tools to Explore	Opentofu, Terragrunt	Terrafgrunt - Terragrunt | OpenTofu/Terraform wrapper Opentofu