# NYCPS TMS - Expanded Terraform Module Examples

This document provides more detailed, representative Terraform code examples for key AWS GovCloud services identified in the architecture. Place these within appropriately named subdirectories inside your main `modules/` directory.

**CRITICAL:** This is example code. Customize names, CIDRs, ARNs, IAM policies, security groups, instance types, KMS keys, tags, and all other configuration parameters according to NYCPS requirements, security policies, and best practices before deploying to any environment. Always run `terraform plan` and review carefully.

## Module: Secure S3 Bucket (`modules/s3\_bucket/`)

Creates a private S3 bucket with versioning, encryption, access logging, and public access block.

### `variables.tf`

variable "bucket\_name\_prefix" { description = "Prefix for the S3 bucket name (will have env/random suffix)" type = string } variable "environment\_name" { description = "Environment name (e.g., dev, prod)" type = string } variable "kms\_key\_arn" { description = "Optional: KMS Key ARN for SSE-KMS encryption" type = string default = null } variable "access\_log\_bucket\_name" { description = "Name of the S3 bucket where access logs should be delivered" type = string default = null # Set to enable logging } variable "lifecycle\_rules" { description = "List of lifecycle rule objects (e.g., for transitions to Glacier)" type = any # Complex type, define structure or use specific object type default = [] } variable "tags" { description = "Map of tags to assign to the bucket" type = map(string) default = {} }

### `main.tf`

resource "random\_id" "suffix" { byte\_length = 4 } resource "aws\_s3\_bucket" "this" { bucket = "${var.bucket\_name\_prefix}-${var.environment\_name}-${random\_id.suffix.hex}" tags = var.tags } resource "aws\_s3\_bucket\_versioning" "this" { bucket = aws\_s3\_bucket.this.id versioning\_configuration { status = "Enabled" } } resource "aws\_s3\_bucket\_server\_side\_encryption\_configuration" "this" { bucket = aws\_s3\_bucket.this.id rule { apply\_server\_side\_encryption\_by\_default { sse\_algorithm = var.kms\_key\_arn == null ? "AES256" : "aws:kms" kms\_master\_key\_id = var.kms\_key\_arn } } } resource "aws\_s3\_bucket\_public\_access\_block" "this" { bucket = aws\_s3\_bucket.this.id block\_public\_acls = true block\_public\_policy = true ignore\_public\_acls = true restrict\_public\_buckets = true } resource "aws\_s3\_bucket\_logging" "this" { count = var.access\_log\_bucket\_name == null ? 0 : 1 bucket = aws\_s3\_bucket.this.id target\_bucket = var.access\_log\_bucket\_name target\_prefix = "log/${aws\_s3\_bucket.this.bucket}/" } resource "aws\_s3\_bucket\_lifecycle\_configuration" "this" { count = length(var.lifecycle\_rules) > 0 ? 1 : 0 bucket = aws\_s3\_bucket.this.id dynamic "rule" { for\_each = var.lifecycle\_rules content { id = rule.value.id status = lookup(rule.value, "status", "Enabled") dynamic "filter" { # Optional filter block for\_each = lookup(rule.value, "filter", null) != null ? [1] : [] content { prefix = lookup(rule.value.filter, "prefix", null) # Add tags filter if needed } } dynamic "transition" { for\_each = lookup(rule.value, "transition", []) content { days = lookup(transition.value, "days", null) storage\_class = transition.value.storage\_class } } dynamic "expiration" { for\_each = lookup(rule.value, "expiration", null) != null ? [1] : [] content { days = lookup(rule.value.expiration, "days", null) # expired\_object\_delete\_marker = true # for versioning } } # Add noncurrent\_version\_transition/expiration if needed } } }

### `outputs.tf`

output "bucket\_id" { description = "The name of the bucket" value = aws\_s3\_bucket.this.id } output "bucket\_arn" { description = "The ARN of the bucket" value = aws\_s3\_bucket.this.arn }

## Module: IAM Role (`modules/iam\_role/`)

Creates a generic IAM role for an AWS service (e.g., Lambda, EC2, ECS Task) with specified policies.

### `variables.tf`

variable "role\_name" { type = string } variable "description" { type = string; default = "IAM role" } variable "assume\_role\_policy\_json" { description = "JSON policy document allowing services (e.g., lambda.amazonaws.com) to assume this role" type = string } variable "managed\_policy\_arns" { description = "List of ARNs for AWS Managed Policies to attach" type = list(string) default = [] } variable "inline\_policy\_json" { description = "Optional: JSON policy document for an inline policy" type = string default = null } variable "inline\_policy\_name" { description = "Name for the inline policy, required if inline\_policy\_json is set" type = string default = "inline-policy" } variable "tags" { type = map(string); default = {} }

### `main.tf`

resource "aws\_iam\_role" "this" { name = var.role\_name description = var.description assume\_role\_policy = var.assume\_role\_policy\_json tags = var.tags } # Attach Managed Policies resource "aws\_iam\_role\_policy\_attachment" "managed" { count = length(var.managed\_policy\_arns) role = aws\_iam\_role.this.name policy\_arn = var.managed\_policy\_arns[count.index] } # Attach Inline Policy (if provided) resource "aws\_iam\_role\_policy" "inline" { count = var.inline\_policy\_json != null ? 1 : 0 name = var.inline\_policy\_name role = aws\_iam\_role.this.id policy = var.inline\_policy\_json }

### `outputs.tf`

output "role\_arn" { description = "ARN of the created IAM role" value = aws\_iam\_role.this.arn } output "role\_name" { description = "Name of the created IAM role" value = aws\_iam\_role.this.name }

**IAM Security:** Craft `assume\_role\_policy\_json` and specific policy documents (passed via `inline\_policy\_json` or created as separate `aws\_iam\_policy` resources and passed via `managed\_policy\_arns`) with extreme care, adhering strictly to least privilege. Avoid wildcard (`\*`) permissions wherever possible.

## Module: Lambda Function (`modules/lambda\_function/`)

Creates an AWS Lambda function from a deployment package in S3.

### `variables.tf`

variable "function\_name" { type = string } variable "description" { type = string; default = "Lambda function" } variable "handler" { type = string } # e.g., "index.handler" for Node.js variable "runtime" { type = string } # e.g., "nodejs18.x", "python3.11" variable "iam\_role\_arn" { type = string } # ARN from the IAM role module variable "s3\_bucket" { type = string } # Bucket where deployment package sits variable "s3\_key" { type = string } # Key (path) to the .zip file in S3 variable "s3\_object\_version" { type = string; default = null } # Optional specific version variable "memory\_size" { type = number; default = 128 } # MB variable "timeout" { type = number; default = 30 } # Seconds variable "environment\_variables" { type = map(string); default = {} } # Non-sensitive env vars variable "vpc\_subnet\_ids" { type = list(string); default = null } # Required if accessing VPC resources variable "vpc\_security\_group\_ids" { type = list(string); default = null } # Required if accessing VPC resources variable "kms\_key\_arn\_environment" { type = string; default = null } # For encrypting env vars at rest variable "tags" { type = map(string); default = {} }

### `main.tf`

resource "aws\_lambda\_function" "this" { function\_name = var.function\_name description = var.description role = var.iam\_role\_arn handler = var.handler runtime = var.runtime memory\_size = var.memory\_size timeout = var.timeout s3\_bucket = var.s3\_bucket s3\_key = var.s3\_key s3\_object\_version= var.s3\_object\_version # source\_code\_hash = filebase64sha256("path/to/your/deployment.zip") # Use if uploading directly (less common with CI/CD) dynamic "environment" { for\_each = length(keys(var.environment\_variables)) > 0 ? [1] : [] content { variables = var.environment\_variables } } dynamic "vpc\_config" { for\_each = var.vpc\_subnet\_ids != null ? [1] : [] content { subnet\_ids = var.vpc\_subnet\_ids security\_group\_ids = var.vpc\_security\_group\_ids } } kms\_key\_arn = var.kms\_key\_arn\_environment tags = var.tags # Ensure the IAM role exists before creating the function depends\_on = [aws\_iam\_role.this] # Assuming role is defined in the same module temporarily, better pass ARN } # Optional: CloudWatch Log Group for the Lambda function resource "aws\_cloudwatch\_log\_group" "this" { name = "/aws/lambda/${var.function\_name}" retention\_in\_days = 14 # TODO: Adjust retention # kms\_key\_id = var.kms\_key\_arn # Optional: Encrypt log group }

### `outputs.tf`

output "function\_arn" { description = "ARN of the Lambda function" value = aws\_lambda\_function.this.arn } output "function\_name" { description = "Name of the Lambda function" value = aws\_lambda\_function.this.function\_name }

**Deployment Packages:** The Lambda function code needs to be packaged (e.g., as a .zip file) and uploaded to the specified S3 bucket, typically as part of your CI/CD pipeline before Terraform runs `apply`. Terraform references the S3 object.

## Module: ECS Fargate Service (`modules/ecs\_fargate\_service/`)

Creates an ECS Cluster (optional, can be shared), Task Definition, and Fargate Service with basic networking and load balancing.

### `variables.tf`

variable "service\_name" { type = string } variable "environment\_name" { type = string } variable "ecs\_cluster\_arn" { type = string } # ARN of an existing shared cluster or one created by this module variable "task\_cpu" { type = number; default = 256 } # CPU units (1024 = 1 vCPU) variable "task\_memory" { type = number; default = 512 } # Memory in MiB variable "container\_image" { type = string } # ECR image URI (e.g., ACCOUNT\_ID.dkr.ecr.REGION.amazonaws.com/repo:tag) variable "container\_port" { type = number } # Port the container listens on variable "desired\_count" { type = number; default = 2 } # Minimum number of tasks for HA variable "task\_execution\_role\_arn" { type = string } # Role for ECS agent (pull image, send logs) variable "task\_role\_arn" { type = string; default = null } # Optional role for the application inside the container variable "vpc\_id" { type = string } variable "private\_subnet\_ids" { type = list(string) } variable "security\_group\_ids" { type = list(string) } # SG allowing traffic from LB/other services variable "alb\_target\_group\_arn" { type = string; default = null } # Optional: Link to ALB Target Group variable "container\_environment\_variables" { type = list(object({ name = string, value = string })); default = [] } variable "container\_secrets" { type = list(object({ name = string, valueFrom = string })); default = [] } # For Secrets Manager/Parameter Store variable "tags" { type = map(string); default = {} }

### `main.tf`

# Optional: Create ECS Cluster if not using a shared one # resource "aws\_ecs\_cluster" "this" { ... } # CloudWatch Log Group for the container resource "aws\_cloudwatch\_log\_group" "this" { name = "/ecs/${var.service\_name}-${var.environment\_name}" retention\_in\_days = 14 # TODO: Adjust } resource "aws\_ecs\_task\_definition" "this" { family = "${var.service\_name}-${var.environment\_name}" network\_mode = "awsvpc" requires\_compatibilities = ["FARGATE"] cpu = var.task\_cpu memory = var.task\_memory execution\_role\_arn = var.task\_execution\_role\_arn task\_role\_arn = var.task\_role\_arn container\_definitions = jsonencode([ { name = var.service\_name image = var.container\_image cpu = var.task\_cpu # Often same as task CPU for single container memory = var.task\_memory # Often same as task Memory essential = true portMappings = [ { containerPort = var.container\_port hostPort = var.container\_port # HostPort must match containerPort in awsvpc mode protocol = "tcp" } ] environment = var.container\_environment\_variables secrets = var.container\_secrets logConfiguration = { logDriver = "awslogs" options = { "awslogs-group" = aws\_cloudwatch\_log\_group.this.name "awslogs-region" = data.aws\_region.current.name "awslogs-stream-prefix" = var.service\_name } } } # Add more containers if needed (e.g., sidecars) ]) tags = var.tags } resource "aws\_ecs\_service" "this" { name = "${var.service\_name}-${var.environment\_name}" cluster = var.ecs\_cluster\_arn task\_definition = aws\_ecs\_task\_definition.this.arn desired\_count = var.desired\_count launch\_type = "FARGATE" network\_configuration { subnets = var.private\_subnet\_ids security\_groups = var.security\_group\_ids assign\_public\_ip = false # Typically false for private subnets } dynamic "load\_balancer" { for\_each = var.alb\_target\_group\_arn != null ? [1] : [] content { target\_group\_arn = var.alb\_target\_group\_arn container\_name = var.service\_name container\_port = var.container\_port } } # Optional: Deployment configuration (rolling update, blue/green) # Optional: Service discovery configuration (Cloud Map) # Optional: Auto Scaling configuration (aws\_appautoscaling\_target, aws\_appautoscaling\_policy) tags = var.tags } data "aws\_region" "current" {}

### `outputs.tf`

output "service\_name" { description = "Name of the ECS service" value = aws\_ecs\_service.this.name } # ... other outputs

## Module: DynamoDB Table (`modules/dynamodb\_table/`)

Creates a DynamoDB table with basic configuration.

### `variables.tf`

variable "table\_name" { type = string } variable "billing\_mode" { type = string; default = "PAY\_PER\_REQUEST" } # Or PROVISIONED variable "hash\_key" { type = string } variable "range\_key" { type = string; default = null } variable "attributes" { # Define attributes used in keys/indexes type = list(object({ name = string type = string # S=String, N=Number, B=Binary })) } variable "enable\_pitr" { type = bool; default = true } # Point-in-time recovery variable "kms\_key\_arn" { type = string; default = null } # For CMK encryption variable "tags" { type = map(string); default = {} } # TODO: Add variables for GSIs, LSIs, Stream Specification if needed

### `main.tf`

resource "aws\_dynamodb\_table" "this" { name = var.table\_name billing\_mode = var.billing\_mode hash\_key = var.hash\_key range\_key = var.range\_key dynamic "attribute" { for\_each = var.attributes content { name = attribute.value.name type = attribute.value.type } } point\_in\_time\_recovery { enabled = var.enable\_pitr } server\_side\_encryption { enabled = true kms\_key\_arn = var.kms\_key\_arn # If null, uses AWS owned key } # TODO: Add blocks for global\_secondary\_index, local\_secondary\_index, stream\_specification # ttl { enabled = false } # Optional TTL configuration tags = var.tags }

### `outputs.tf`

output "table\_name" { value = aws\_dynamodb\_table.this.name } output "table\_arn" { value = aws\_dynamodb\_table.this.arn }

## Module: Kinesis Data Stream (`modules/kinesis\_stream/`)

Creates a Kinesis Data Stream.

### `variables.tf`

variable "stream\_name" { type = string } variable "shard\_count" { type = number; default = 1 } variable "retention\_period" { type = number; default = 24 } # Hours variable "kms\_key\_id" { type = string; default = null } # ARN for SSE-KMS variable "tags" { type = map(string); default = {} }

### `main.tf`

resource "aws\_kinesis\_stream" "this" { name = var.stream\_name shard\_count = var.shard\_count retention\_period = var.retention\_period # Use "PROVISIONED" (default) or "ON\_DEMAND" # shard\_level\_metrics = ["IncomingBytes", "OutgoingBytes"] stream\_mode\_details { stream\_mode = "PROVISIONED" # Or "ON\_DEMAND" } encryption\_type = var.kms\_key\_id == null ? "NONE" : "KMS" kms\_key\_id = var.kms\_key\_id tags = var.tags }

### `outputs.tf`

output "stream\_name" { value = aws\_kinesis\_stream.this.name } output "stream\_arn" { value = aws\_kinesis\_stream.this.arn }

## Module: SNS Topic (`modules/sns\_topic/`)

Creates an SNS topic.

### `variables.tf`

variable "topic\_name" { type = string } variable "kms\_master\_key\_id" { type = string; default = null } # Optional KMS key for SSE variable "tags" { type = map(string); default = {} }

### `main.tf`

resource "aws\_sns\_topic" "this" { name = var.topic\_name kms\_master\_key\_id = var.kms\_master\_key\_id # Use "alias/aws/sns" for AWS-managed key tags = var.tags # TODO: Add aws\_sns\_topic\_policy if specific cross-account or service access is needed }

### `outputs.tf`

output "topic\_arn" { value = aws\_sns\_topic.this.arn }

## Module: SQS Queue (`modules/sqs\_queue/`)

Creates a standard SQS queue.

### `variables.tf`

variable "queue\_name" { type = string } variable "is\_fifo\_queue" { type = bool; default = false } variable "visibility\_timeout\_seconds" { type = number; default = 30 } variable "message\_retention\_seconds" { type = number; default = 345600 } # 4 days variable "kms\_master\_key\_id" { type = string; default = null } # Use "alias/aws/sqs" or custom key ARN for SSE variable "kms\_data\_key\_reuse\_period\_seconds" { type = number; default = 300 } variable "dead\_letter\_queue\_arn" { type = string; default = null } # ARN of a DLQ variable "max\_receive\_count" { type = number; default = 5 } # Used with DLQ variable "tags" { type = map(string); default = {} }

### `main.tf`

resource "aws\_sqs\_queue" "this" { name = var.is\_fifo\_queue ? "${var.queue\_name}.fifo" : var.queue\_name fifo\_queue = var.is\_fifo\_queue content\_based\_deduplication = var.is\_fifo\_queue ? true : false # Often used with FIFO visibility\_timeout\_seconds = var.visibility\_timeout\_seconds message\_retention\_seconds = var.message\_retention\_seconds kms\_master\_key\_id = var.kms\_master\_key\_id kms\_data\_key\_reuse\_period\_seconds = var.kms\_data\_key\_reuse\_period\_seconds redrive\_policy = var.dead\_letter\_queue\_arn == null ? null : jsonencode({ deadLetterTargetArn = var.dead\_letter\_queue\_arn maxReceiveCount = var.max\_receive\_count }) tags = var.tags # TODO: Add aws\_sqs\_queue\_policy if needed }

### `outputs.tf`

output "queue\_url" { value = aws\_sqs\_queue.this.id } # Note: id attribute gives URL output "queue\_arn" { value = aws\_sqs\_queue.this.arn }

## Module: API Gateway (HTTP API Example) (`modules/api\_gateway/`)

Creates a basic HTTP API Gateway with Lambda integration.

### `variables.tf`

variable "api\_name" { type = string } variable "lambda\_integration\_uri" { type = string } # Invoke ARN of the target Lambda variable "route\_key" { type = string; default = "ANY /{proxy+}" } # Default catch-all variable "tags" { type = map(string); default = {} } # TODO: Add variables for custom domain, authorizers, CORS, logging, etc.

### `main.tf`

resource "aws\_apigatewayv2\_api" "this" { name = var.api\_name protocol\_type = "HTTP" description = "HTTP API for ${var.api\_name}" # cors\_configuration { ... } # TODO: Configure CORS if needed tags = var.tags } resource "aws\_apigatewayv2\_integration" "lambda" { api\_id = aws\_apigatewayv2\_api.this.id integration\_type = "AWS\_PROXY" integration\_method = "POST" # Always POST for Lambda proxy integration\_uri = var.lambda\_integration\_uri payload\_format\_version = "2.0" # Use payload format 2.0 for HTTP APIs } resource "aws\_apigatewayv2\_route" "proxy" { api\_id = aws\_apigatewayv2\_api.this.id route\_key = var.route\_key target = "integrations/${aws\_apigatewayv2\_integration.lambda.id}" # TODO: Add authorizer configuration if needed # authorization\_type = "JWT" / "AWS\_IAM" / "CUSTOM" # authorizer\_id = aws\_apigatewayv2\_authorizer.this.id } resource "aws\_apigatewayv2\_stage" "default" { api\_id = aws\_apigatewayv2\_api.this.id name = "$default" # Creates a default stage accessible at the base invoke URL auto\_deploy = true # TODO: Configure access logging, throttling, custom domain mapping # access\_log\_settings { ... } # default\_route\_settings { ... } # domain\_name { ... } } # Permissions for API Gateway to invoke Lambda resource "aws\_lambda\_permission" "api\_gw" { statement\_id = "AllowAPIGatewayInvoke" action = "lambda:InvokeFunction" function\_name = var.lambda\_integration\_uri # Use the function name part if using ARN principal = "apigateway.amazonaws.com" # Restrict to the specific API Gateway API source\_arn = "${aws\_apigatewayv2\_api.this.execution\_arn}/\*/\*" }

### `outputs.tf`

output "api\_endpoint" { description = "The invoke URL for the API Gateway stage" value = aws\_apigatewayv2\_stage.default.invoke\_url } output "api\_id" { description = "The ID of the API Gateway" value = aws\_apigatewayv2\_api.this.id }

## Module: CloudWatch Alarm (`modules/cloudwatch\_alarm/`)

Creates a basic CloudWatch alarm based on a metric.

### `variables.tf`

variable "alarm\_name" { type = string } variable "comparison\_operator" { type = string } # e.g., "GreaterThanOrEqualToThreshold" variable "evaluation\_periods" { type = number } # e.g., 1 variable "metric\_name" { type = string } # e.g., "Errors" variable "namespace" { type = string } # e.g., "AWS/Lambda" variable "period" { type = number } # Seconds, e.g., 300 (5 minutes) variable "statistic" { type = string } # e.g., "Sum", "Average" variable "threshold" { type = number } variable "alarm\_description" { type = string; default = null } variable "dimensions" { type = map(string); default = {} } # e.g., { FunctionName = "my-function" } variable "alarm\_actions" { type = list(string); default = [] } # List of SNS topic ARNs variable "ok\_actions" { type = list(string); default = [] } # Optional: actions when state returns to OK

### `main.tf`

resource "aws\_cloudwatch\_metric\_alarm" "this" { alarm\_name = var.alarm\_name comparison\_operator = var.comparison\_operator evaluation\_periods = var.evaluation\_periods metric\_name = var.metric\_name namespace = var.namespace period = var.period statistic = var.statistic threshold = var.threshold alarm\_description = var.alarm\_description dimensions = length(keys(var.dimensions)) > 0 ? var.dimensions : null alarm\_actions = var.alarm\_actions ok\_actions = var.ok\_actions # treat\_missing\_data = "missing" # Other options: "ignore", "breaching", "notBreaching" }

### `outputs.tf`

output "alarm\_arn" { value = aws\_cloudwatch\_metric\_alarm.this.arn } output "alarm\_name" { value = aws\_cloudwatch\_metric\_alarm.this.alarm\_name }

## Module: VPC Interface Endpoint (`modules/vpc\_endpoint/`)

Creates a VPC Interface Endpoint for accessing AWS services privately.

### `variables.tf`

variable "vpc\_id" { type = string } variable "service\_name" { type = string } # e.g., "com.amazonaws.us-gov-west-1.secretsmanager" variable "subnet\_ids" { type = list(string) } # Subnets where the ENI will be placed (usually private) variable "security\_group\_ids" { type = list(string) } # SG allowing HTTPS traffic \*to\* the endpoint ENI variable "private\_dns\_enabled" { type = bool; default = true } variable "tags" { type = map(string); default = {} }

### `main.tf`

resource "aws\_vpc\_endpoint" "this" { vpc\_id = var.vpc\_id service\_name = var.service\_name vpc\_endpoint\_type = "Interface" subnet\_ids = var.subnet\_ids security\_group\_ids = var.security\_group\_ids private\_dns\_enabled = var.private\_dns\_enabled tags = var.tags }

### `outputs.tf`

output "endpoint\_id" { value = aws\_vpc\_endpoint.this.id } output "dns\_entries" { value = aws\_vpc\_endpoint.this.dns\_entry } output "network\_interface\_ids" { value = aws\_vpc\_endpoint.this.network\_interface\_ids }

## Next Steps & Implementation

1. **Create Directories:** Set up the `modules/` and `environments/` directory structure shown in Step 1.

2. **Populate Modules:** Copy the example code above into the corresponding module directories (e.g., `modules/s3\_bucket/`, `modules/iam\_role/`).

3. **Develop Remaining Modules:** Using the provided examples as a template, develop the Terraform code (`variables.tf`, `main.tf`, `outputs.tf`) for all other required components identified in the architecture (e.g., ElastiCache, Redshift, MSK, Glue, additional IAM policies, specific Security Groups per tier, Load Balancers, CloudFront, etc.). Remember to focus on modularity and parameterization.

4. **Create Environment Files:** Create the `main.tf`, `variables.tf`, `outputs.tf`, and `.tfvars` files within each directory under `environments/` (dev, qa, prod, etc.).

5. **Instantiate Modules:** In each `environments/\*/main.tf`, add `module` blocks calling the necessary modules from your `modules/` directory, passing outputs from prerequisite modules (like networking) as inputs to dependent modules (like RDS or ECS).

6. **Define Variables:** Fill in the `.tfvars` files for each environment with the appropriate configuration values, ensuring sensitive data is handled securely (e.g., referencing Secrets Manager ARNs).

7. **Initialize, Plan, Apply (Iteratively):** Follow the workflow in Step 6, starting with the `dev` environment. Initialize, plan carefully, review the plan, and apply. Address any errors. Test the provisioned resources. Repeat iteratively as you build out more components and refine configurations.

8. **Version Control:** Commit all code frequently to your Git repository.

9. **Integrate with CI/CD:** Once the manual workflow is stable, integrate the `plan` and `apply` steps into your CI/CD pipeline for automated environment management, including appropriate approval steps for Staging and Production environments.