Terraform Workspace-Based Real-Time Scenarios

## Scenario 1: Deploying Infrastructure in Dev, Staging, and Prod

variable "instance\_type" {  
 type = string  
}  
  
resource "aws\_instance" "my\_instance" {  
 ami = "ami-0c55b159cbfafe1f0"  
 instance\_type = var.instance\_type  
 tags = {  
 Name = terraform.workspace  
 }  
}

## Scenario 2: Provisioning S3 Buckets for Dev, Staging, and Prod

resource "aws\_s3\_bucket" "logs" {  
 bucket = "logs-${terraform.workspace}-bucket"  
}

## Scenario 3: VPC per Environment

resource "aws\_vpc" "main" {  
 cidr\_block = "10.${terraform.workspace == "prod" ? 0 : terraform.workspace == "staging" ? 1 : 2}.0.0/16"  
 tags = {  
 Name = "${terraform.workspace}-vpc"  
 }  
}

## Scenario 4: RDS per Environment with Different Sizes

resource "aws\_db\_instance" "db" {  
 identifier = "app-db-${terraform.workspace}"  
 instance\_class = terraform.workspace == "prod" ? "db.t3.large" : "db.t3.micro"  
 allocated\_storage = terraform.workspace == "prod" ? 100 : 20  
 engine = "mysql"  
 username = "admin"  
 password = "admin123"  
 skip\_final\_snapshot = true  
}

## Scenario 11: ECR Repository per Environment

resource "aws\_ecr\_repository" "app\_repo" {  
 name = "app-repo-${terraform.workspace}"  
 image\_tag\_mutability = "MUTABLE"  
 tags = {  
 Environment = terraform.workspace  
 }  
}

## Scenario 12: Elastic Load Balancer per Environment

resource "aws\_elb" "web\_elb" {  
 name = "web-elb-${terraform.workspace}"  
 availability\_zones = ["us-east-1a", "us-east-1b"]  
 listener {  
 instance\_port = 80  
 instance\_protocol = "http"  
 lb\_port = 80  
 lb\_protocol = "http"  
 }  
 instances = [aws\_instance.my\_instance.id]  
 tags = {  
 Environment = terraform.workspace  
 }  
}

## Scenario 13: CloudFront Distributions per Environment

resource "aws\_cloudfront\_distribution" "cdn" {  
 origin {  
 domain\_name = "${terraform.workspace}-static-site.s3.amazonaws.com"  
 origin\_id = "s3-origin-${terraform.workspace}"  
 }  
 enabled = true  
 default\_root\_object = "index.html"  
 default\_cache\_behavior {  
 target\_origin\_id = "s3-origin-${terraform.workspace}"  
 viewer\_protocol\_policy = "redirect-to-https"  
 allowed\_methods = ["GET", "HEAD"]  
 cached\_methods = ["GET", "HEAD"]  
 }  
 viewer\_certificate {  
 cloudfront\_default\_certificate = true  
 }  
 tags = {  
 Environment = terraform.workspace  
 }  
}

## Scenario 14: CloudTrail Logging per Environment

resource "aws\_cloudtrail" "trail" {  
 name = "trail-${terraform.workspace}"  
 s3\_bucket\_name = aws\_s3\_bucket.logs.bucket  
 include\_global\_service\_events = true  
 is\_multi\_region\_trail = true  
 enable\_logging = true  
}

## Scenario 15: Parameter Store Configs per Environment

resource "aws\_ssm\_parameter" "db\_password" {  
 name = "/${terraform.workspace}/app/db\_password"  
 type = "SecureString"  
 value = terraform.workspace == "prod" ? "ProdSecret123!" : "DevSecret123"  
}

## Scenario 16: Elastic Beanstalk Application per Environment

resource "aws\_elastic\_beanstalk\_environment" "app\_env" {  
 name = "myapp-${terraform.workspace}"  
 application = aws\_elastic\_beanstalk\_application.app.name  
 solution\_stack\_name = "64bit Amazon Linux 2 v3.3.6 running Python 3.8"  
 setting {  
 namespace = "aws:autoscaling:launchconfiguration"  
 name = "InstanceType"  
 value = terraform.workspace == "prod" ? "t3.large" : "t3.micro"  
 }  
}  
resource "aws\_elastic\_beanstalk\_application" "app" {  
 name = "myapp-${terraform.workspace}"  
}