Single CodePipeline • Master-Centralized Terraform State • Cross-Account Deployments (Commercial & GovCloud)

This guide delivers a one-pipeline architecture where Terraform state (S3 + DynamoDB) lives only in the master account. The pipeline runs parallel plans and ordered applies across multiple target accounts/environments (dev, qa, nonprod, prod). Authentication chain: CodeBuild → TerraformMasterRole (master) → TerraformDeploymentRole (target). Works in both Commercial (aws) and GovCloud (aws-us-gov) using partition-aware ARNs.

# Phase 0 — Set Required Placeholders (for commands)

Set these once in your shell before running the commands (replace with your values):  
export PARTITION=aws # or aws-us-gov  
export MASTER\_ACCOUNT=111111111111 # your master/CI account  
export MASTER\_REGION=us-east-1 # or us-gov-east-1  
export DEV\_ACCOUNT=222222222222  
export QA\_ACCOUNT=333333333333  
export NONPROD\_ACCOUNT=444444444444  
export PROD\_ACCOUNT=555555555555  
  
# Repo choice (pick one)  
export REPO\_PROVIDER=CodeCommit # or CodeStar  
export CODECOMMIT\_REPO=terraform-infra  
# For CodeStar:  
export CONNECTION\_ARN=arn:aws:codestar-connections:us-east-1:${MASTER\_ACCOUNT}:connection/XXXXXXXX  
export FULL\_REPO\_ID=myorg/terraform-infra

# Phase 1 — Master Account: Central Terraform State & Artifacts

1.1 Create central Terraform state (S3 + DynamoDB) in master:

export STATE\_BUCKET=tf-state-${MASTER\_ACCOUNT}-${PARTITION}  
export LOCK\_TABLE=tf-state-locks  
  
aws s3api create-bucket \  
 --bucket ${STATE\_BUCKET} \  
 --create-bucket-configuration LocationConstraint=${MASTER\_REGION} \  
 --region ${MASTER\_REGION}  
  
aws s3api put-bucket-versioning \  
 --bucket ${STATE\_BUCKET} \  
 --versioning-configuration Status=Enabled  
  
aws s3api put-bucket-encryption \  
 --bucket ${STATE\_BUCKET} \  
 --server-side-encryption-configuration \  
 '{"Rules":[{"ApplyServerSideEncryptionByDefault":{"SSEAlgorithm":"AES256"}}]}'  
  
aws dynamodb create-table \  
 --table-name ${LOCK\_TABLE} \  
 --attribute-definitions AttributeName=LockID,AttributeType=S \  
 --key-schema AttributeName=LockID,KeyType=HASH \  
 --billing-mode PAY\_PER\_REQUEST \  
 --region ${MASTER\_REGION}

1.2 Create CodePipeline artifacts bucket in master:

export ARTIFACT\_BUCKET=pipeline-artifacts-${MASTER\_ACCOUNT}-${PARTITION}  
  
aws s3api create-bucket \  
 --bucket ${ARTIFACT\_BUCKET} \  
 --create-bucket-configuration LocationConstraint=${MASTER\_REGION} \  
 --region ${MASTER\_REGION}  
  
aws s3api put-bucket-encryption \  
 --bucket ${ARTIFACT\_BUCKET} \  
 --server-side-encryption-configuration \  
 '{"Rules":[{"ApplyServerSideEncryptionByDefault":{"SSEAlgorithm":"AES256"}}]}'

# Phase 2 — IAM Roles (Master and Targets)

2.1 Master: CodeBuild execution role (Tools-CodeBuildRole) — Trust policy:

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {"Effect":"Allow","Principal":{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}  
 ]  
}

Attach inline policy to Tools-CodeBuildRole:

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {"Sid":"CentralState","Effect":"Allow","Action":["s3:PutObject","s3:GetObject","s3:ListBucket","s3:GetBucketVersioning"],  
 "Resource":["arn:${PARTITION}:s3:::${STATE\_BUCKET}","arn:${PARTITION}:s3:::${STATE\_BUCKET}/\*"]},  
 {"Sid":"CentralLock","Effect":"Allow","Action":["dynamodb:DescribeTable","dynamodb:GetItem","dynamodb:PutItem","dynamodb:DeleteItem"],  
 "Resource":"arn:${PARTITION}:dynamodb:${MASTER\_REGION}:${MASTER\_ACCOUNT}:table/${LOCK\_TABLE}"},  
 {"Sid":"Artifacts","Effect":"Allow","Action":["s3:PutObject","s3:GetObject","s3:ListBucket","s3:GetBucketVersioning"],  
 "Resource":["arn:${PARTITION}:s3:::${ARTIFACT\_BUCKET}","arn:${PARTITION}:s3:::${ARTIFACT\_BUCKET}/\*"]},  
 {"Sid":"AssumeMaster","Effect":"Allow","Action":"sts:AssumeRole",  
 "Resource":"arn:${PARTITION}:iam::${MASTER\_ACCOUNT}:role/TerraformMasterRole"}  
 ]  
}

2.2 Master: TerraformMasterRole — Trusts CodeBuild role:

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {"Effect":"Allow","Principal":{"AWS":"arn:${PARTITION}:iam::${MASTER\_ACCOUNT}:role/Tools-CodeBuildRole"},  
 "Action":"sts:AssumeRole"}  
 ]  
}

Inline policy for TerraformMasterRole — may assume all target deployment roles:

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {"Sid":"AssumeTargets","Effect":"Allow","Action":"sts:AssumeRole","Resource":[  
 "arn:${PARTITION}:iam::${DEV\_ACCOUNT}:role/TerraformDeploymentRole",  
 "arn:${PARTITION}:iam::${QA\_ACCOUNT}:role/TerraformDeploymentRole",  
 "arn:${PARTITION}:iam::${NONPROD\_ACCOUNT}:role/TerraformDeploymentRole",  
 "arn:${PARTITION}:iam::${PROD\_ACCOUNT}:role/TerraformDeploymentRole"  
 ]}  
 ]  
}

2.3 Each target: TerraformDeploymentRole — Trusts ONLY the master role:

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {"Effect":"Allow","Principal":{"AWS":"arn:${PARTITION}:iam::${MASTER\_ACCOUNT}:role/TerraformMasterRole"},  
 "Action":"sts:AssumeRole"}  
 ]  
}

TerraformDeploymentRole permissions (start broad; tighten later):

{  
 "Version":"2012-10-17",  
 "Statement":[  
 {"Sid":"TerraformCreates","Effect":"Allow","Action":[  
 "ec2:\*","iam:\*","eks:\*","ecr:\*","elasticloadbalancing:\*","logs:\*","autoscaling:\*","kms:\*","ssm:\*","secretsmanager:\*","ecs:\*","cloudwatch:\*"  
 ],"Resource":"\*"}  
 ]  
}

# Phase 3 — Repository Layout & Terraform Code

Repo structure:  
terraform-infra/  
├─ modules/{vpc,eks,ecs}/  
│ ├─ backend.tf  
│ ├─ providers.tf  
│ └─ main.tf variables.tf outputs.tf  
├─ env/{dev,qa,nonprod,prod}/<module>.tfvars  
├─ buildspecs/{plan.yml,apply.yml}  
└─ pipelines/single-pipeline-multi-account.yaml

modules/<module>/backend.tf (state in master; injected at init):

terraform {  
 backend "s3" {  
 encrypt = true  
 # Injected via -backend-config at init:  
 # bucket = tf-state-${MASTER\_ACCOUNT}-${PARTITION}  
 # dynamodb\_table = tf-state-locks  
 # region = ${MASTER\_REGION}  
 # key = ${MODULE\_NAME}/${ENV\_NAME}.tfstate  
 }  
}

modules/<module>/providers.tf (provider assumes target role):

terraform { required\_version = ">= 1.6.0" }  
  
variable "region" { type = string }  
variable "target\_role\_arn" { type = string }  
  
provider "aws" {  
 region = var.region  
 assume\_role {  
 role\_arn = var.target\_role\_arn  
 session\_name = "tf-${var.region}"  
 }  
}

# Phase 4 — Buildspecs (Master State; Provider Assumes Target Role)

Set these environment variables in each CodeBuild project (the pipeline will supply values per environment):  
Common: PARTITION, MASTER\_ACCOUNT, MASTER\_REGION, MODULE\_NAME, MODULE\_PATH, ENV\_NAME, TARGET\_REGION  
Per environment: TF\_VAR\_target\_role\_arn = arn:${PARTITION}:iam::<ENV\_ACCOUNT\_ID>:role/TerraformDeploymentRole

buildspecs/plan.yml:

version: 0.2  
phases:  
 install:  
 commands:  
 - curl -sLo tf.zip https://releases.hashicorp.com/terraform/1.7.5/terraform\_1.7.5\_linux\_amd64.zip  
 - unzip -o tf.zip -d /usr/local/bin  
 - yum install -y jq || true  
 - terraform -version  
 pre\_build:  
 commands:  
 - MASTER\_ARN="arn:${PARTITION}:iam::${MASTER\_ACCOUNT}:role/${MASTER\_ROLE\_NAME:-TerraformMasterRole}"  
 - MC=$(aws sts assume-role --role-arn "$MASTER\_ARN" --role-session-name "tf-master-$(date +%s)" --duration-seconds 3600)  
 - export AWS\_ACCESS\_KEY\_ID=$(echo "$MC" | jq -r .Credentials.AccessKeyId)  
 - export AWS\_SECRET\_ACCESS\_KEY=$(echo "$MC" | jq -r .Credentials.SecretAccessKey)  
 - export AWS\_SESSION\_TOKEN=$(echo "$MC" | jq -r .Credentials.SessionToken)  
 - cd $MODULE\_PATH  
 - STATE\_BUCKET="tf-state-${MASTER\_ACCOUNT}-${PARTITION}"  
 - KEY="${MODULE\_NAME}/${ENV\_NAME}.tfstate"  
 - terraform init -input=false -reconfigure \  
 -backend-config="bucket=${STATE\_BUCKET}" \  
 -backend-config="dynamodb\_table=tf-state-locks" \  
 -backend-config="region=${MASTER\_REGION}" \  
 -backend-config="key=${KEY}"  
 build:  
 commands:  
 - terraform validate  
 - terraform plan -input=false -out=tfplan \  
 -var="region=${TARGET\_REGION}" \  
 -var-file="../env/${ENV\_NAME}/${MODULE\_NAME}.tfvars"  
artifacts:  
 files:  
 - $MODULE\_PATH/tfplan  
 - env/\*\*/\*  
 discard-paths: no

buildspecs/apply.yml:

version: 0.2  
phases:  
 install:  
 commands:  
 - curl -sLo tf.zip https://releases.hashicorp.com/terraform/1.7.5/terraform\_1.7.5\_linux\_amd64.zip  
 - unzip -o tf.zip -d /usr/local/bin  
 - yum install -y jq || true  
 - terraform -version  
 pre\_build:  
 commands:  
 - MASTER\_ARN="arn:${PARTITION}:iam::${MASTER\_ACCOUNT}:role/${MASTER\_ROLE\_NAME:-TerraformMasterRole}"  
 - MC=$(aws sts assume-role --role-arn "$MASTER\_ARN" --role-session-name "tf-master-$(date +%s)" --duration-seconds 3600)  
 - export AWS\_ACCESS\_KEY\_ID=$(echo "$MC" | jq -r .Credentials.AccessKeyId)  
 - export AWS\_SECRET\_ACCESS\_KEY=$(echo "$MC" | jq -r .Credentials.SecretAccessKey)  
 - export AWS\_SESSION\_TOKEN=$(echo "$MC" | jq -r .Credentials.SessionToken)  
 - cd $MODULE\_PATH  
 - STATE\_BUCKET="tf-state-${MASTER\_ACCOUNT}-${PARTITION}"  
 - KEY="${MODULE\_NAME}/${ENV\_NAME}.tfstate"  
 - terraform init -input=false -reconfigure \  
 -backend-config="bucket=${STATE\_BUCKET}" \  
 -backend-config="dynamodb\_table=tf-state-locks" \  
 -backend-config="region=${MASTER\_REGION}" \  
 -backend-config="key=${KEY}"  
 build:  
 commands:  
 - terraform apply -input=false tfplan

# Phase 5 — Single CodePipeline (Parallel Plans → Gated Applies)

CloudFormation creates one pipeline: Source → Plan (parallel) → Approval → Apply (ordered). Each CodeBuild action sets TF\_VAR\_target\_role\_arn for Terraform to assume the correct target role.

pipelines/single-pipeline-multi-account.yaml: (full template)

AWSTemplateFormatVersion: "2010-09-09"  
Description: ONE CodePipeline to plan/apply Terraform to multiple accounts with master-backed state and master->target assume  
  
Parameters:  
 Partition: { Type: String, AllowedValues: [aws, aws-us-gov] }  
 MasterAccountId: { Type: String }  
 MasterRegion: { Type: String }  
 ArtifactBucketName: { Type: String }  
  
 RepoProvider: { Type: String, AllowedValues: [CodeStar, CodeCommit], Default: CodeCommit }  
 ConnectionArn: { Type: String, Default: "" }  
 FullRepositoryId: { Type: String, Default: "" }  
 CodeCommitRepo: { Type: String, Default: "" }  
 Branch: { Type: String, Default: main }  
  
 ModuleName: { Type: String, Default: vpc }  
 ModulePath: { Type: String, Default: modules/vpc }  
  
 DevAccountId: { Type: String }  
 QaAccountId: { Type: String }  
 NonprodAccountId: { Type: String }  
 ProdAccountId: { Type: String }  
  
 DevRegion: { Type: String, Default: us-east-1 }  
 QaRegion: { Type: String, Default: us-east-1 }  
 NonprodRegion: { Type: String, Default: us-east-1 }  
 ProdRegion: { Type: String, Default: us-east-1 }  
  
 ToolsCodeBuildRoleName: { Type: String, Default: Tools-CodeBuildRole }  
 MasterRoleName: { Type: String, Default: TerraformMasterRole }  
 TargetRoleName: { Type: String, Default: TerraformDeploymentRole }  
  
Conditions:  
 UseCodeStar: !Equals [ !Ref RepoProvider, "CodeStar" ]

Resources:  
 PipelineRole:  
 Type: AWS::IAM::Role  
 Properties:  
 AssumeRolePolicyDocument:  
 Version: "2012-10-17"  
 Statement:  
 - Effect: Allow  
 Principal: { Service: codepipeline.amazonaws.com }  
 Action: sts:AssumeRole  
 Policies:  
 - PolicyName: pipeline-inline  
 PolicyDocument:  
 Version: "2012-10-17"  
 Statement:  
 - Effect: Allow  
 Action: [ "s3:PutObject","s3:GetObject","s3:ListBucket","s3:GetBucketVersioning" ]  
 Resource:  
 - !Sub "arn:${AWS::Partition}:s3:::${ArtifactBucketName}"  
 - !Sub "arn:${AWS::Partition}:s3:::${ArtifactBucketName}/\*"  
 - Effect: Allow  
 Action: [ "codebuild:StartBuild","codebuild:BatchGetBuilds" ]  
 Resource: "\*"  
 - Effect: Allow  
 Action: "codestar-connections:UseConnection"  
 Resource: !If [ UseCodeStar, !Ref ConnectionArn, !Ref "AWS::NoValue" ]  
 - Effect: Allow  
 Action: "iam:PassRole"  
 Resource: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"

PlanDev:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: plan-dev  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: dev }  
 - { Name: TARGET\_REGION, Value: !Ref DevRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${DevAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/plan.yml }  
 PlanQa:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: plan-qa  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: qa }  
 - { Name: TARGET\_REGION, Value: !Ref QaRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${QaAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/plan.yml }  
 PlanNonprod:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: plan-nonprod  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: nonprod }  
 - { Name: TARGET\_REGION, Value: !Ref NonprodRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${NonprodAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/plan.yml }  
 PlanProd:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: plan-prod  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: prod }  
 - { Name: TARGET\_REGION, Value: !Ref ProdRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${ProdAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/plan.yml }

ApplyDev:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: apply-dev  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: dev }  
 - { Name: TARGET\_REGION, Value: !Ref DevRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${DevAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/apply.yml }  
 ApplyQa:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: apply-qa  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: qa }  
 - { Name: TARGET\_REGION, Value: !Ref QaRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${QaAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/apply.yml }  
 ApplyNonprod:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: apply-nonprod  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: nonprod }  
 - { Name: TARGET\_REGION, Value: !Ref NonprodRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${NonprodAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/apply.yml }  
 ApplyProd:  
 Type: AWS::CodeBuild::Project  
 Properties:  
 Name: apply-prod  
 ServiceRole: !Sub "arn:${AWS::Partition}:iam::${MasterAccountId}:role/${ToolsCodeBuildRoleName}"  
 Artifacts: { Type: CODEPIPELINE }  
 Environment:  
 Type: LINUX\_CONTAINER  
 ComputeType: BUILD\_GENERAL1\_SMALL  
 Image: aws/codebuild/standard:7.0  
 EnvironmentVariables:  
 - { Name: PARTITION, Value: !Ref Partition }  
 - { Name: MASTER\_ACCOUNT, Value: !Ref MasterAccountId }  
 - { Name: MASTER\_REGION, Value: !Ref MasterRegion }  
 - { Name: MODULE\_NAME, Value: !Ref ModuleName }  
 - { Name: MODULE\_PATH, Value: !Ref ModulePath }  
 - { Name: ENV\_NAME, Value: prod }  
 - { Name: TARGET\_REGION, Value: !Ref ProdRegion }  
 - Name: TF\_VAR\_target\_role\_arn  
 Value: !Sub "arn:${AWS::Partition}:iam::${ProdAccountId}:role/${TargetRoleName}"  
 Source: { Type: CODEPIPELINE, BuildSpec: buildspecs/apply.yml }

Pipeline:  
 Type: AWS::CodePipeline::Pipeline  
 Properties:  
 Name: single-terraform-multi-account  
 RoleArn: !GetAtt PipelineRole.Arn  
 ArtifactStore: { Type: S3, Location: !Ref ArtifactBucketName }  
 Stages:  
 - Name: Source  
 Actions:  
 - !If  
 - UseCodeStar  
 - Name: Source  
 ActionTypeId: { Category: Source, Owner: AWS, Provider: CodeStarSourceConnection, Version: "1" }  
 Configuration:  
 ConnectionArn: !Ref ConnectionArn  
 FullRepositoryId: !Ref FullRepositoryId  
 BranchName: !Ref Branch  
 OutputArtifactFormat: CODEBUILD\_CLONE\_REF  
 OutputArtifacts: [ { Name: SourceOutput } ]  
 - Name: Source  
 ActionTypeId: { Category: Source, Owner: AWS, Provider: CodeCommit, Version: "1" }  
 Configuration:  
 RepositoryName: !Ref CodeCommitRepo  
 BranchName: !Ref Branch  
 OutputArtifacts: [ { Name: SourceOutput } ]  
 - Name: Plan  
 Actions:  
 - { Name: PlanDev, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: SourceOutput } ], OutputArtifacts: [ { Name: PlanDevOut } ], Configuration: { ProjectName: !Ref PlanDev } }  
 - { Name: PlanQa, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: SourceOutput } ], OutputArtifacts: [ { Name: PlanQaOut } ], Configuration: { ProjectName: !Ref PlanQa } }  
 - { Name: PlanNonprod, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: SourceOutput } ], OutputArtifacts: [ { Name: PlanNonprodOut } ], Configuration: { ProjectName: !Ref PlanNonprod } }  
 - { Name: PlanProd, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: SourceOutput } ], OutputArtifacts: [ { Name: PlanProdOut } ], Configuration: { ProjectName: !Ref PlanProd } }  
 - Name: Approval  
 Actions:  
 - { Name: ManualApproval, ActionTypeId: { Category: Approval, Owner: AWS, Provider: Manual, Version: "1" } }  
 - Name: Apply  
 Actions:  
 - { Name: ApplyDev, RunOrder: 1, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: PlanDevOut } ], Configuration: { ProjectName: !Ref ApplyDev } }  
 - { Name: ApplyQa, RunOrder: 2, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: PlanQaOut } ], Configuration: { ProjectName: !Ref ApplyQa } }  
 - { Name: ApplyNonprod, RunOrder: 3, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: PlanNonprodOut } ], Configuration: { ProjectName: !Ref ApplyNonprod } }  
 - { Name: ApplyProd, RunOrder: 4, ActionTypeId: { Category: Build, Owner: AWS, Provider: CodeBuild, Version: "1" }, InputArtifacts: [ { Name: PlanProdOut } ], Configuration: { ProjectName: !Ref ApplyProd } }

# Phase 6 — Deploy the Pipeline Stack (Commands)

aws cloudformation deploy \  
 --region ${MASTER\_REGION} \  
 --stack-name single-terraform-multi-account \  
 --template-file pipelines/single-pipeline-multi-account.yaml \  
 --capabilities CAPABILITY\_NAMED\_IAM \  
 --parameter-overrides \  
 Partition=${PARTITION} \  
 MasterAccountId=${MASTER\_ACCOUNT} \  
 MasterRegion=${MASTER\_REGION} \  
 ArtifactBucketName=${ARTIFACT\_BUCKET} \  
 RepoProvider=${REPO\_PROVIDER} \  
 CodeCommitRepo=${CODECOMMIT\_REPO} \  
 Branch=main \  
 ModuleName=vpc ModulePath=modules/vpc \  
 DevAccountId=${DEV\_ACCOUNT} QaAccountId=${QA\_ACCOUNT} NonprodAccountId=${NONPROD\_ACCOUNT} ProdAccountId=${PROD\_ACCOUNT} \  
 DevRegion=us-east-1 QaRegion=us-east-1 NonprodRegion=us-east-1 ProdRegion=us-east-1

For CodeStar (GitHub) instead of CodeCommit, add: RepoProvider=CodeStar ConnectionArn=${CONNECTION\_ARN} FullRepositoryId=${FULL\_REPO\_ID}

# Phase 7 — Verify, Operate, and Troubleshoot

• Trigger: commit under modules/<module>/ or env/<env>/<module>.tfvars.  
• Plans run in parallel; Approve; Applies run dev → qa → nonprod → prod.  
• Logs: terraform init points to s3://${STATE\_BUCKET} in ${MASTER\_REGION}; provider assumes the target role.  
• Add data.aws\_caller\_identity to print target account id if needed.  
  
Tips:  
• State key naming: ${MODULE\_NAME}/${ENV\_NAME}-${ENV\_ACCOUNT\_ID}.tfstate if you reuse env names across accounts.  
• ExternalId: add to provider assume\_role and target trust if your org requires it.  
• Tighten permissions over time per module (least privilege).  
• Use source path filters to limit runs to relevant modules/env files.  
• GovCloud: set Partition=aws-us-gov and prefer AWS::Partition in CloudFormation.