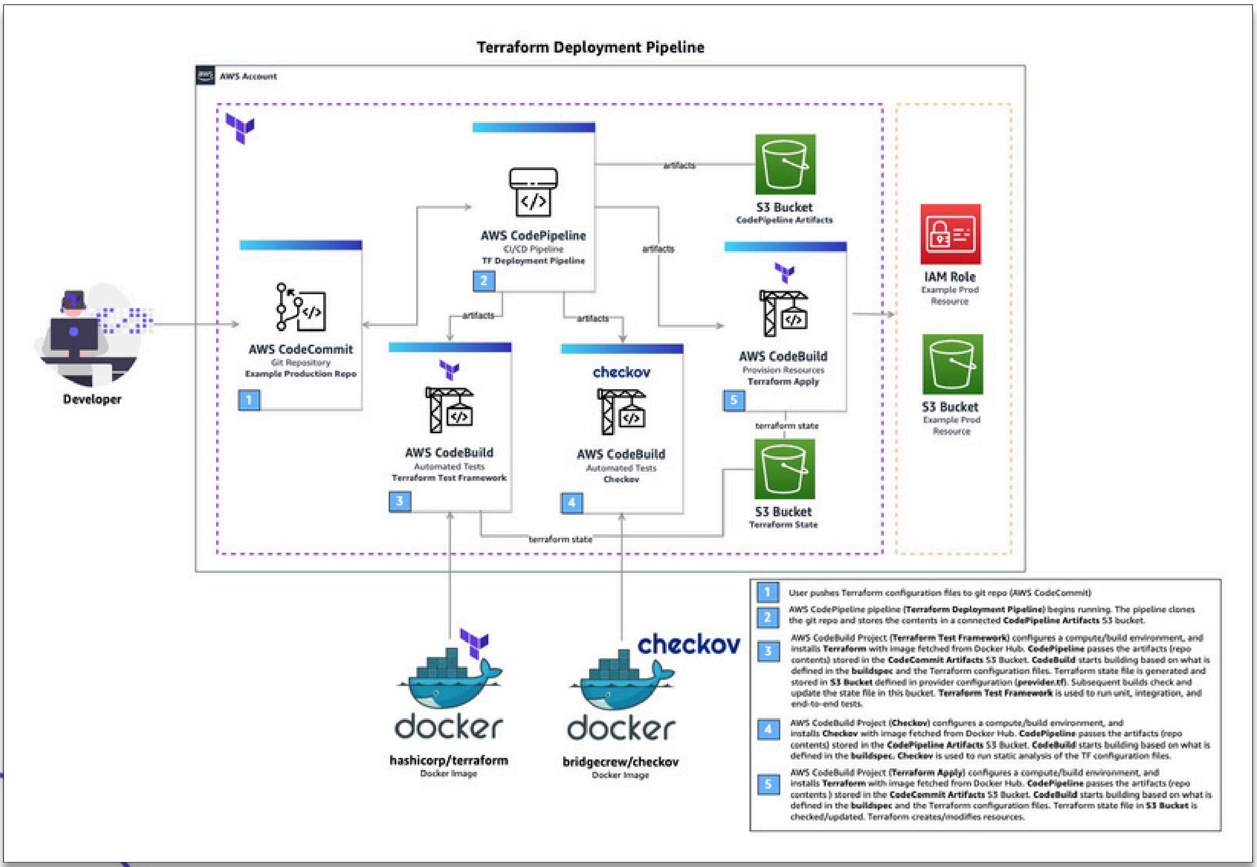
TERRAFORM CI/CD AND TESTING ON AWS

AWS Cloud Engineer



1 User pushes Terraform configuration files to git repo (AWS CodeCommit)

2 AWS CodePipeline pipeline (**Terraform Deployment Pipeline**) begins running. The pipeline clones the git repo and stores the contents in a connected **CodePipeline Artifacts** S3 bucket.

3 AWS CodeBuild Project (**Terraform Test Framework**) configures a compute/build environment, and installs **Terraform** with image fetched from Docker Hub. **CodePipeline** passes the artifacts (repo contents) stored in the **CodeCommit Artifacts** S3 Bucket. **CodeBuild** starts building based on what is defined in the **buildspec** and the Terraform configuration files. Terraform state file is generated and stored in **S3 Bucket** defined in provider configuration (**provider.tf**). Subsequent builds check and update the state file in this bucket. **Terraform Test Framework** is used to run unit, integration, and end-to-end tests.

4 AWS CodeBuild Project (**Checkov**) configures a compute/build environment, and installs **Checkov** with image fetched from Docker Hub. **CodePipeline** passes the artifacts (repo contents) stored in the **CodePipeline Artifacts** S3 Bucket. **CodeBuild** starts building based on what is defined in the **buildspec**. **Checkov** is used to run static analysis of the TF configuration files.

5 AWS CodeBuild Project (**Terraform Apply**) configures a compute/build environment, and installs **Terraform** with image fetched from Docker Hub. **CodePipeline** passes the artifacts (repo contents) stored in the **CodeCommit Artifacts** S3 Bucket. **CodeBuild** starts building based on what is defined in the **buildspec** and the Terraform configuration files. Terraform state file in **S3 Bucket** is checked/updated. Terraform creates/modifies resources.

Problem Statement

Manual Terraform deployments and lack of testing increase the risk of misconfigured infrastructure, security issues, and unscalable cloud environments. This project solves that by building a CI/CD system that automates testing and deployment of Terraform modules.

Business Impact

- Reduced risk of misconfigured infrastructure through automated testing and validation
- Improved deployment speed and consistency with full CI/CD pipeline
- Strengthened security with Checkov integration
- Centralized Terraform state storage for team collaboration using S3 + DynamoDB
- Eliminated manual provisioning, enabling repeatable, production-ready deployments

Cloud Architecture

Services Used:

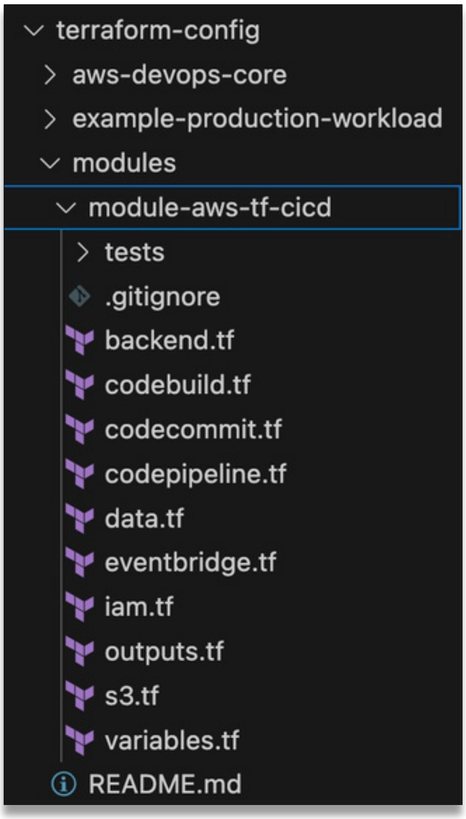
- **Terraform** – Infra automation, validation, testing
- **AWS CodePipeline** – CI/CD pipeline orchestration
- **AWS CodeBuild** – Executes Terraform tests, Checkov scans, and apply
- **S3** – Stores artifacts and remote backend state
- **DynamoDB** – State lock management
- **IAM** – Secure access roles and permissions
- **GitHub (via CodeStar Connections)** – Source repo and trigger for pipelines

Structure:

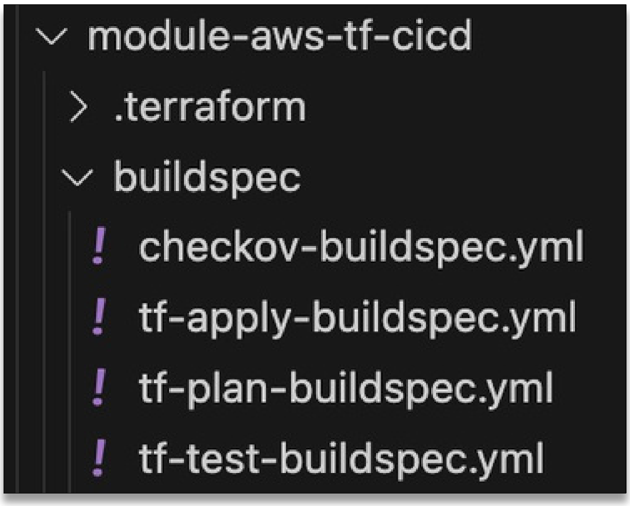
- Testing Pipeline (module validation)
- Deployment Pipeline (example workload)
- Remote state setup across two pipelines
- Refactored out CodeCommit, replaced with GitHub + CodeStar Connection

Technical Implementa?on

1. Developed a reusable Terraform module to provision AWS infrastructure using IaC best pracCces, including IAM roles, S3 buckets, and CodePipeline.

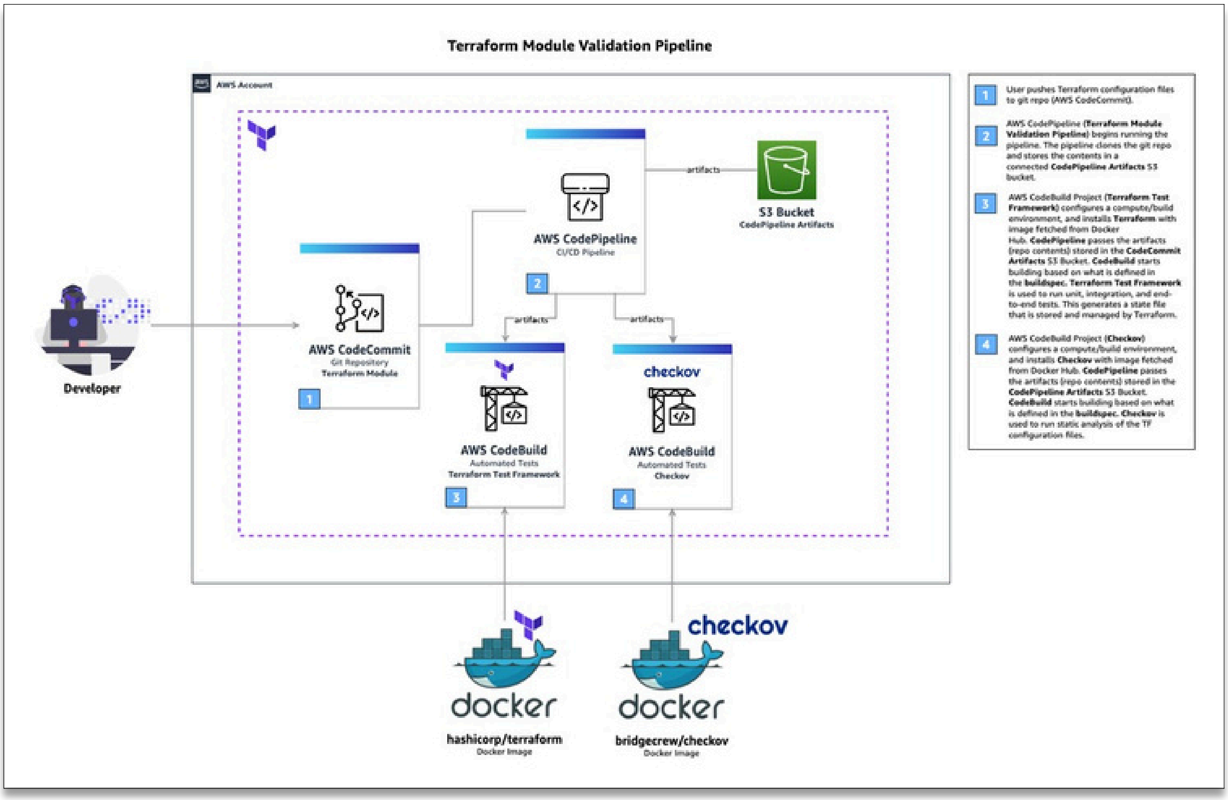


```
⌄ terraform-config
  > aws-devops-core
  > example-production-workload
  ⌄ modules
    ⌄ module-aws-tf-cicd
      > tests
        .gitignore
        backend.tf
        codebuild.tf
        codecommit.tf
        codepipeline.tf
        data.tf
        eventbridge.tf
        iam.tf
        outputs.tf
        s3.tf
        variables.tf
    ⓘ README.md
```



```
⌄ module-aws-tf-cicd
  > .terraform
  ⌄ buildspec
    ! checkov-buildspec.yml
    ! tf-apply-buildspec.yml
    ! tf-plan-buildspec.yml
    ! tf-test-buildspec.yml
```

2. Integrated Terraform Test Framework to validate module funcConality with unit, integraCon, and end-to-end tests.

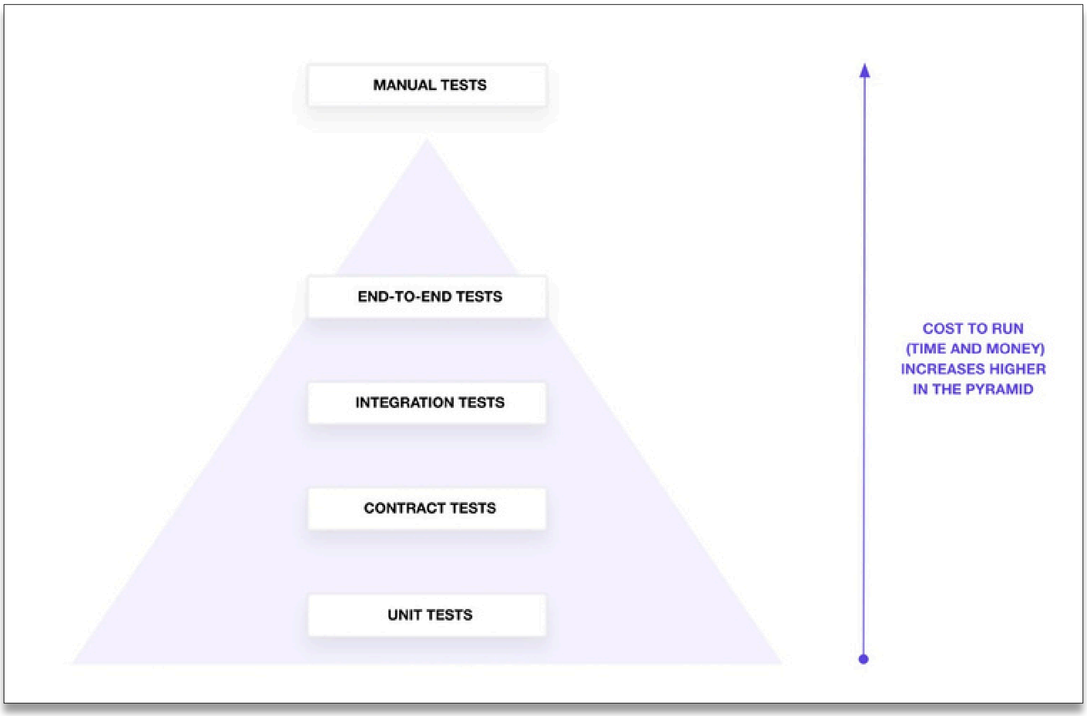


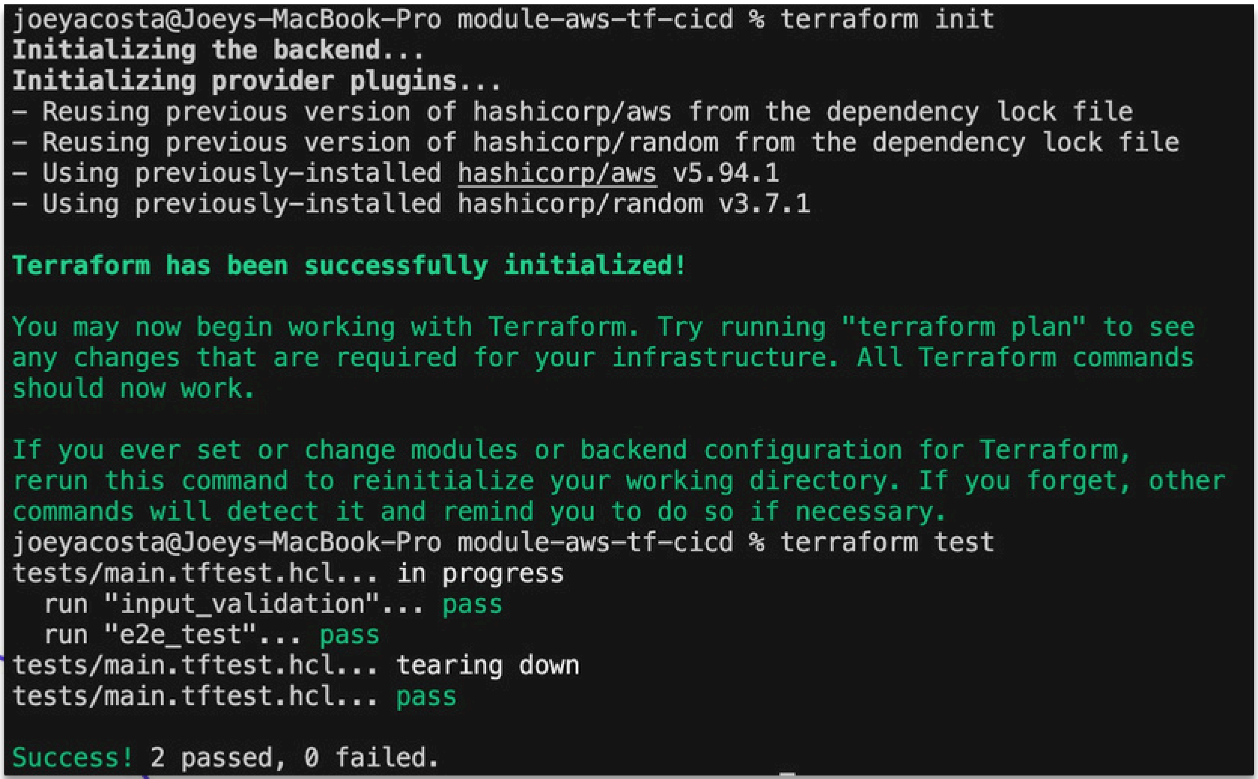
1 User pushes Terraform configuration files to git repo (AWS CodeCommit).

2 AWS CodePipeline (**Terraform Module Validation Pipeline**) begins running the pipeline. The pipeline clones the git repo and stores the contents in a connected **CodePipeline Artifacts** S3 bucket.

3 AWS CodeBuild Project (**Terraform Test Framework**) configures a compute/build environment, and installs **Terraform** with image fetched from Docker Hub. **CodePipeline** passes the artifacts (repo contents) stored in the **CodeCommit Artifacts** S3 Bucket. **CodeBuild** starts building based on what is defined in the **buildspec. Terraform Test Framework** is used to run unit, integration, and end-to-end tests. This generates a state file that is stored and managed by Terraform.

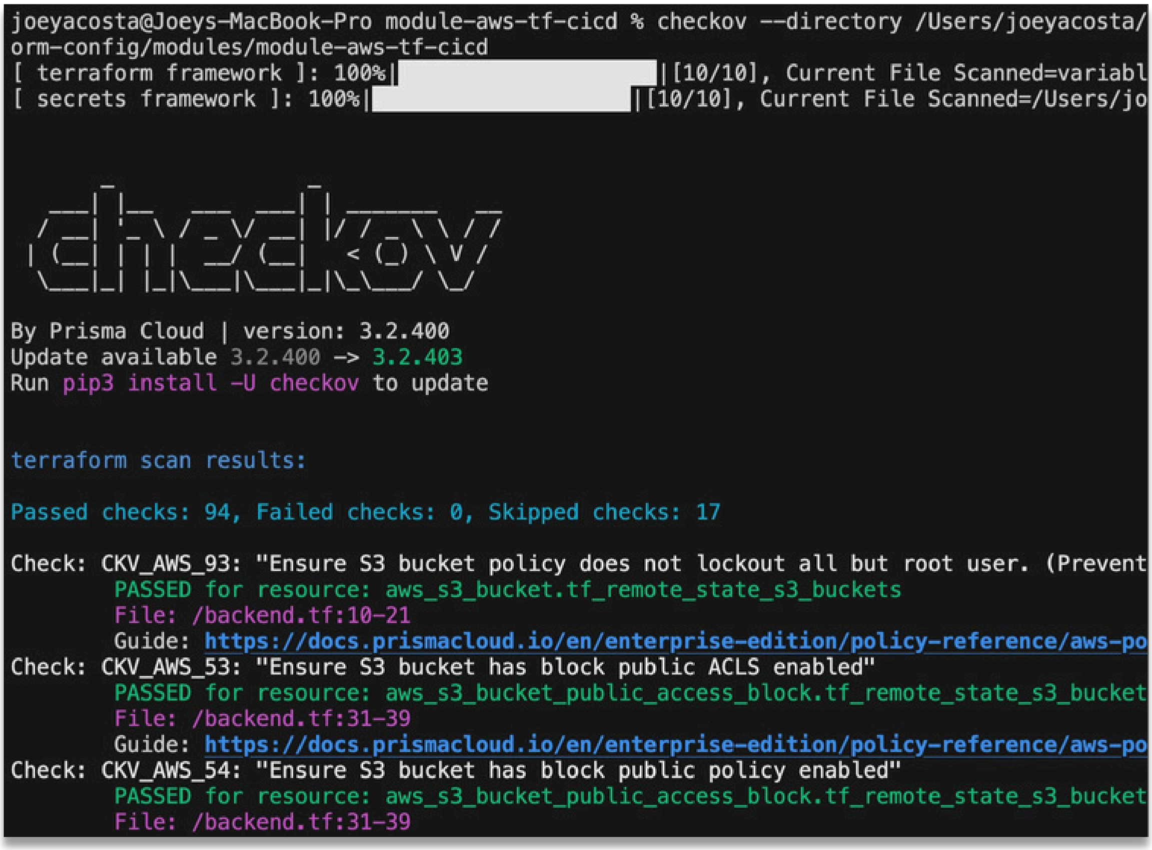
4 AWS CodeBuild Project (**Checkov**) configures a compute/build environment, and installs **Checkov** with image fetched from Docker Hub. **CodePipeline** passes the artifacts (repo contents) stored in the **CodePipeline Artifacts** S3 Bucket. **CodeBuild** starts building based on what is defined in the **buildspec. Checkov** is used to run static analysis of the TF configuration files.



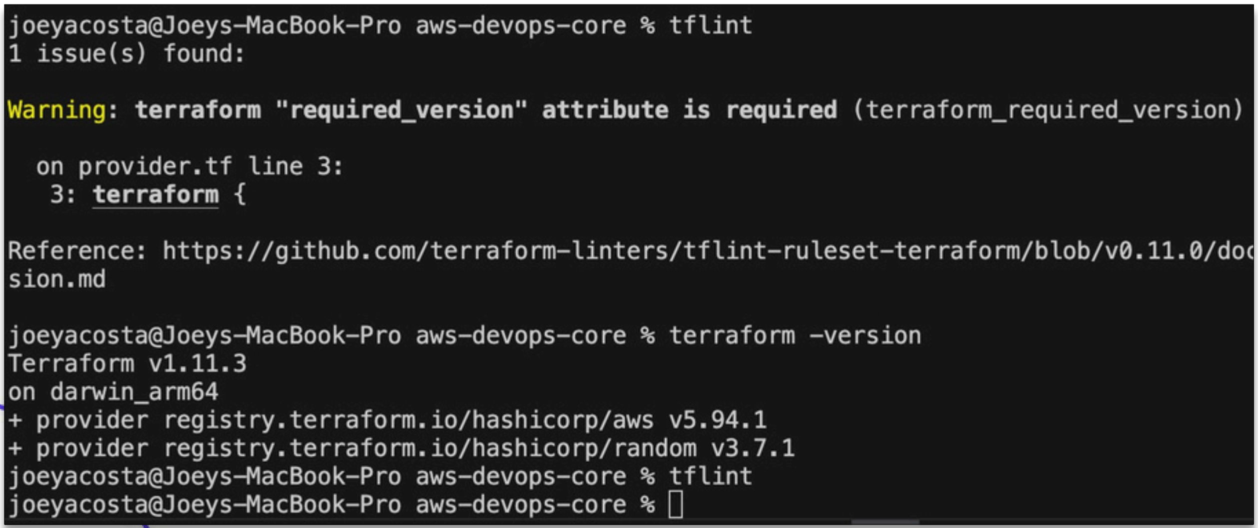


```
joeyacosta@Joeys-MacBook-Pro module-aws-tf-cicd % terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Using previously-installed hashicorp/aws v5.94.1
- Using previously-installed hashicorp/random v3.7.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
joeyacosta@Joeys-MacBook-Pro module-aws-tf-cicd % terraform test
tests/main.tftest.hcl... in progress
  run "input_validation"... pass
  run "e2e_test"... pass
tests/main.tftest.hcl... tearing down
tests/main.tftest.hcl... pass

Success! 2 passed, 0 failed.
```

3. Added security scanning with Checkov and enforced linCng with TFLint to ensure clean, secure, and compliant Terraform code.

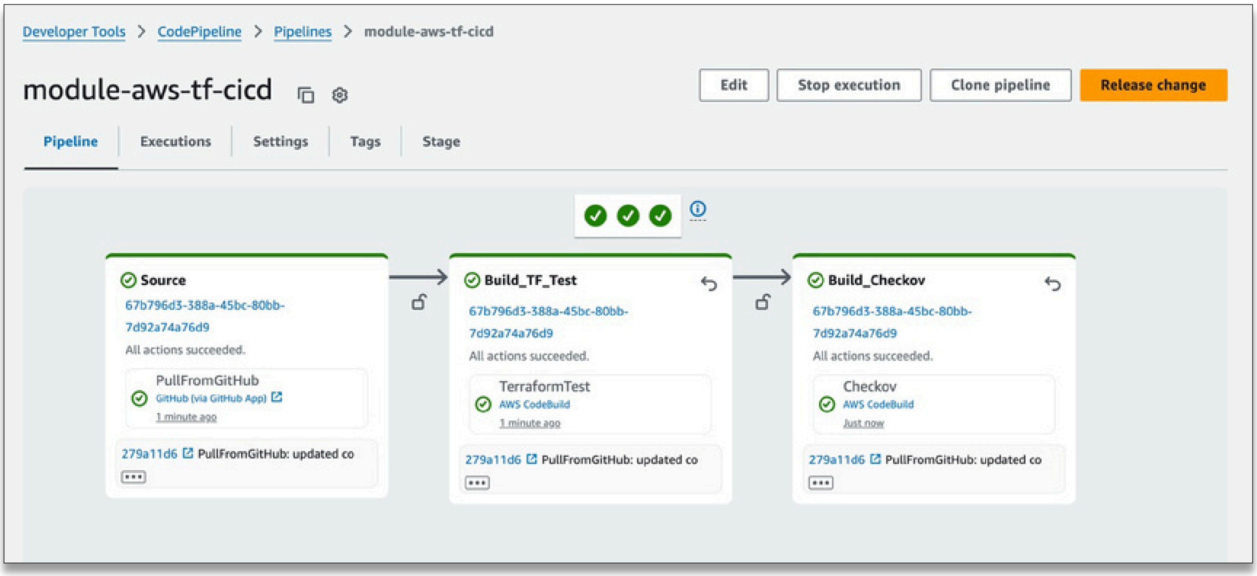


```
joeyacosta@Joeys-MacBook-Pro module-aws-tf-cicd % checkov --directory /Users/joeyacosta/
orm-config/modules/module-aws-tf-cicd
[ terraform framework ]: 100%|                    |[10/10], Current File Scanned=variabl
[ secrets framework ]: 100%|                    |[10/10], Current File Scanned=/Users/jo



       _               _
   ___| |__   ___  ___| | _______   __
  / __| '_ \ / _ \/ __| |/ / _ \ \ / /
 | (__| | | |  __/ (__|   < (_) \ V /
  \___|_| |_|\___|\___|_|\_\___/ \_/

By Prisma Cloud | version: 3.2.400
Update available 3.2.400 -> 3.2.403
Run pip3 install -U checkov to update


terraform scan results:

Passed checks: 94, Failed checks: 0, Skipped checks: 17

Check: CKV_AWS_93: "Ensure S3 bucket policy does not lockout all but root user. (Prevent
        PASSED for resource: aws_s3_bucket.tf_remote_state_s3_buckets
        File: /backend.tf:10-21
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-po
Check: CKV_AWS_53: "Ensure S3 bucket has block public ACLS enabled"
        PASSED for resource: aws_s3_bucket_public_access_block.tf_remote_state_s3_bucket
        File: /backend.tf:31-39
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-po
Check: CKV_AWS_54: "Ensure S3 bucket has block public policy enabled"
        PASSED for resource: aws_s3_bucket_public_access_block.tf_remote_state_s3_bucket
        File: /backend.tf:31-39
```



```
joeyacosta@Joeys-MacBook-Pro aws-devops-core % tflint
1 issue(s) found:

Warning: terraform "required_version" attribute is required (terraform_required_version)

  on provider.tf line 3:
   3: terraform {

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.11.0/doc
sion.md

joeyacosta@Joeys-MacBook-Pro aws-devops-core % terraform -version
Terraform v1.11.3
on darwin_arm64
+ provider registry.terraform.io/hashicorp/aws v5.94.1
+ provider registry.terraform.io/hashicorp/random v3.7.1
joeyacosta@Joeys-MacBook-Pro aws-devops-core % tflint
joeyacosta@Joeys-MacBook-Pro aws-devops-core % 
```

4. Built two automated CI/CD pipelines using Terraform, CodePipeline, and CodeBuild to test and deploy infrastructure triggered by GitHub commits.

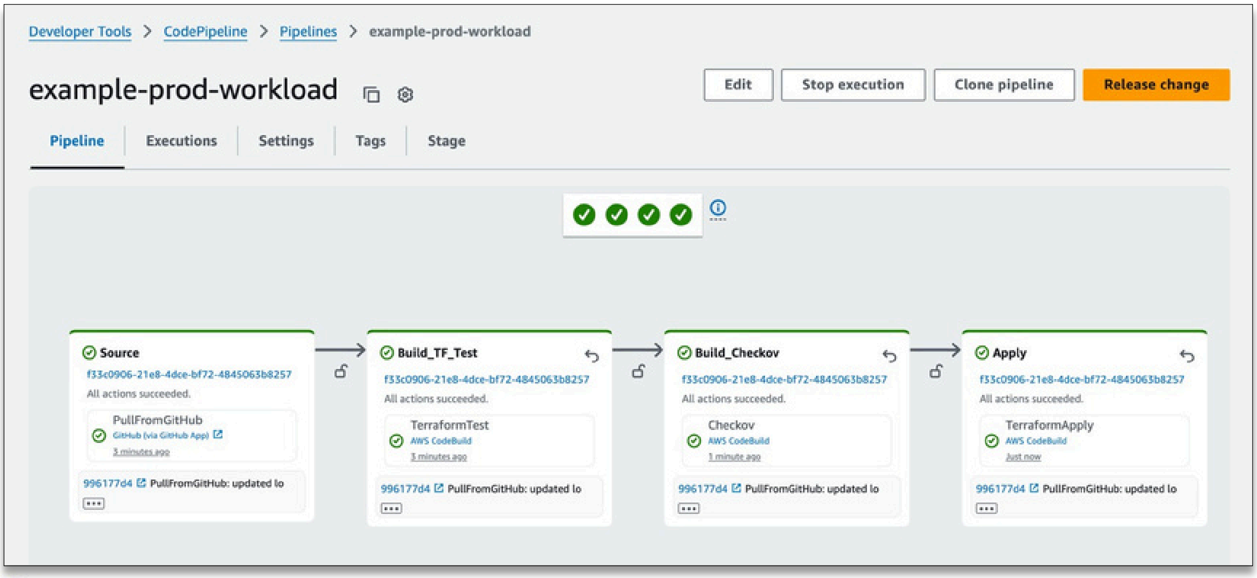


module-aws-tf-cicd

Source
67b796d3-388a-45bc-80bb-7d92a74a76d9
All actions succeeded.
PullFromGitHub
GitHub (via GitHub App)
1 minute ago
279a11d6 PullFromGitHub: updated co

Build_TF_Test
67b796d3-388a-45bc-80bb-7d92a74a76d9
All actions succeeded.
TerraformTest
AWS CodeBuild
1 minute ago
279a11d6 PullFromGitHub: updated co

Build_Checkov
67b796d3-388a-45bc-80bb-7d92a74a76d9
All actions succeeded.
Checkov
AWS CodeBuild
Just now
279a11d6 PullFromGitHub: updated co



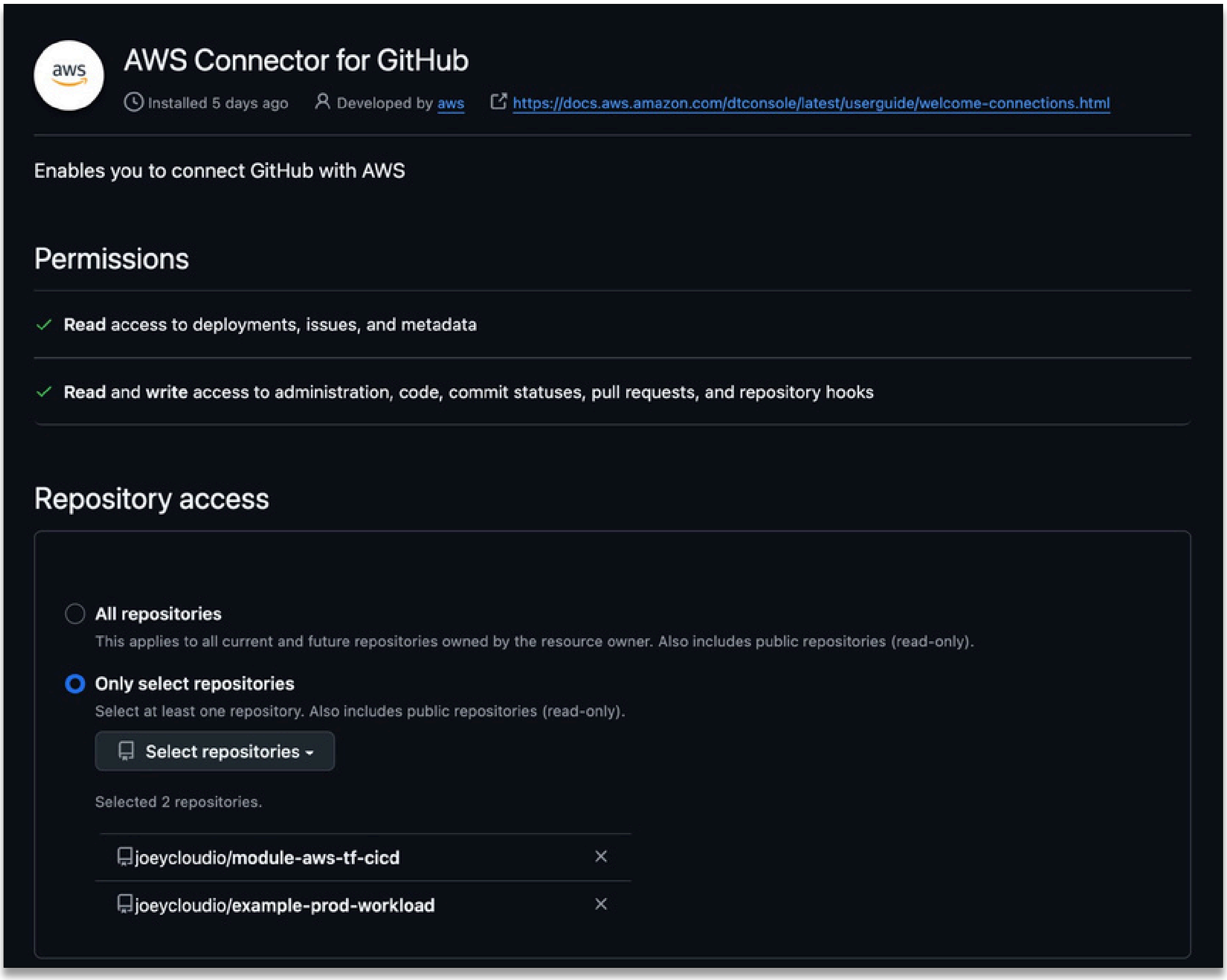
example-prod-workload

Source
f33c0906-21e8-4dce-bf72-4845063b8257
All actions succeeded.
PullFromGitHub
GitHub (via GitHub App)
3 minutes ago
996177d4 PullFromGitHub: updated lo

Build_TF_Test
f33c0906-21e8-4dce-bf72-4845063b8257
All actions succeeded.
TerraformTest
AWS CodeBuild
3 minutes ago
996177d4 PullFromGitHub: updated lo

Build_Checkov
f33c0906-21e8-4dce-bf72-4845063b8257
All actions succeeded.
Checkov
AWS CodeBuild
1 minute ago
996177d4 PullFromGitHub: updated lo

Apply
f33c0906-21e8-4dce-bf72-4845063b8257
All actions succeeded.
TerraformApply
AWS CodeBuild
Just now
996177d4 PullFromGitHub: updated lo



AWS Connector for GitHub

Installed 5 days ago · Developed by aws · https://docs.aws.amazon.com/dtconsole/latest/userguide/welcome-connections.html

Enables you to connect GitHub with AWS

Permissions

✓ **Read** access to deployments, issues, and metadata

✓ **Read** and **write** access to administration, code, commit statuses, pull requests, and repository hooks

Repository access

○ **All repositories**
This applies to all current and future repositories owned by the resource owner. Also includes public repositories (read-only).

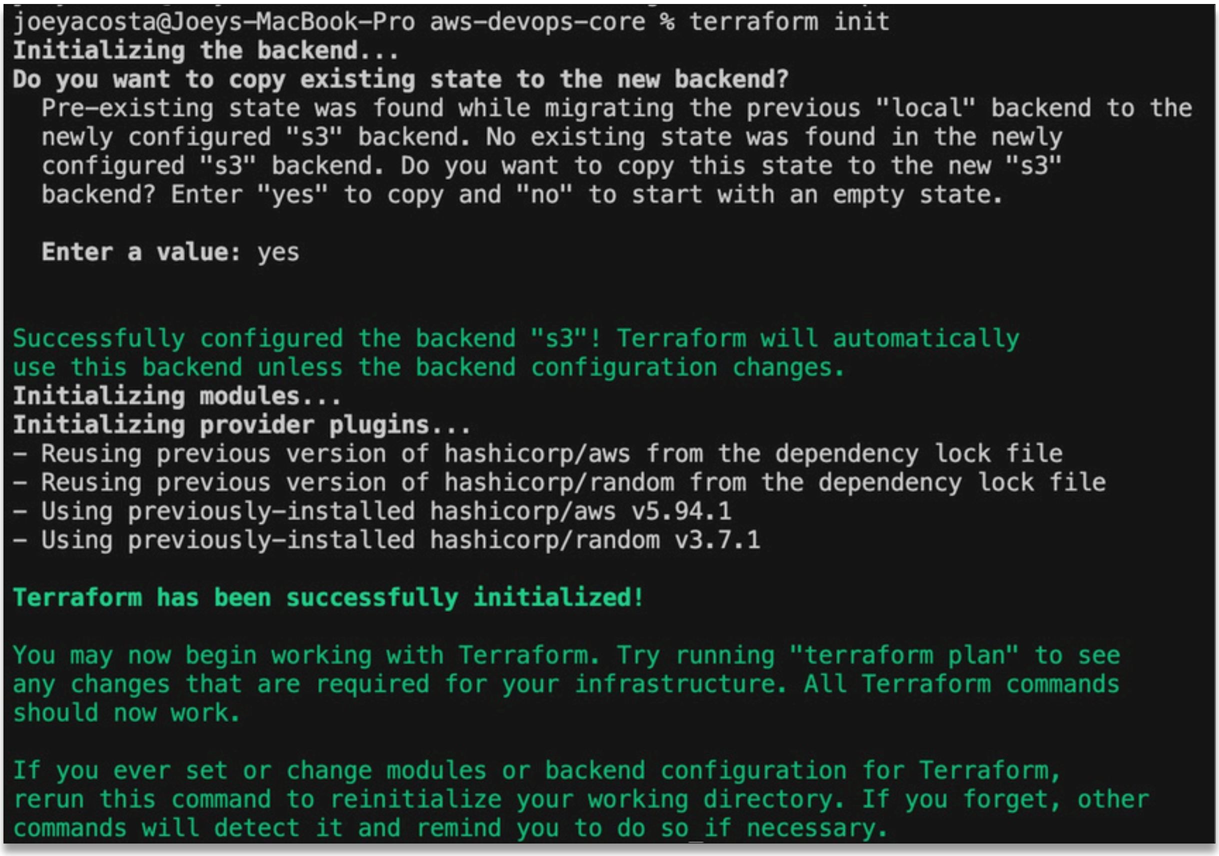
● **Only select repositories**
Select at least one repository. Also includes public repositories (read-only).

Selected 2 repositories.

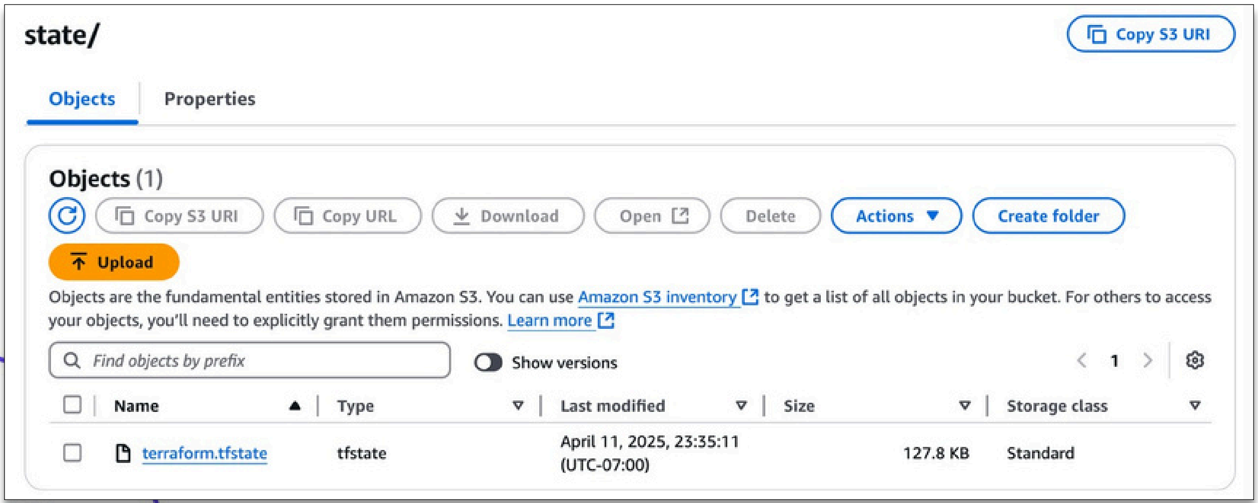
joeycloudio/**module-aws-tf-cicd** ✕

joeycloudio/**example-prod-workload** ✕

5. Configured S3 remote backend and DynamoDB state locking, enabling collaboraCon and safe, versioned state management.



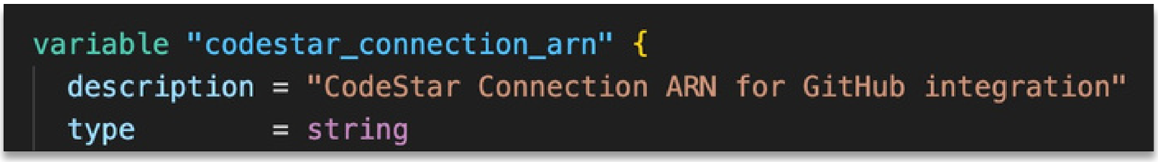
```
joeyacosta@Joeys-MacBook-Pro aws-devops-core % terraform init
Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "s3" backend. No existing state was found in the newly
  configured "s3" backend. Do you want to copy this state to the new "s3"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes


Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/random from the dependency lock file
- Using previously-installed hashicorp/aws v5.94.1
- Using previously-installed hashicorp/random v3.7.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```



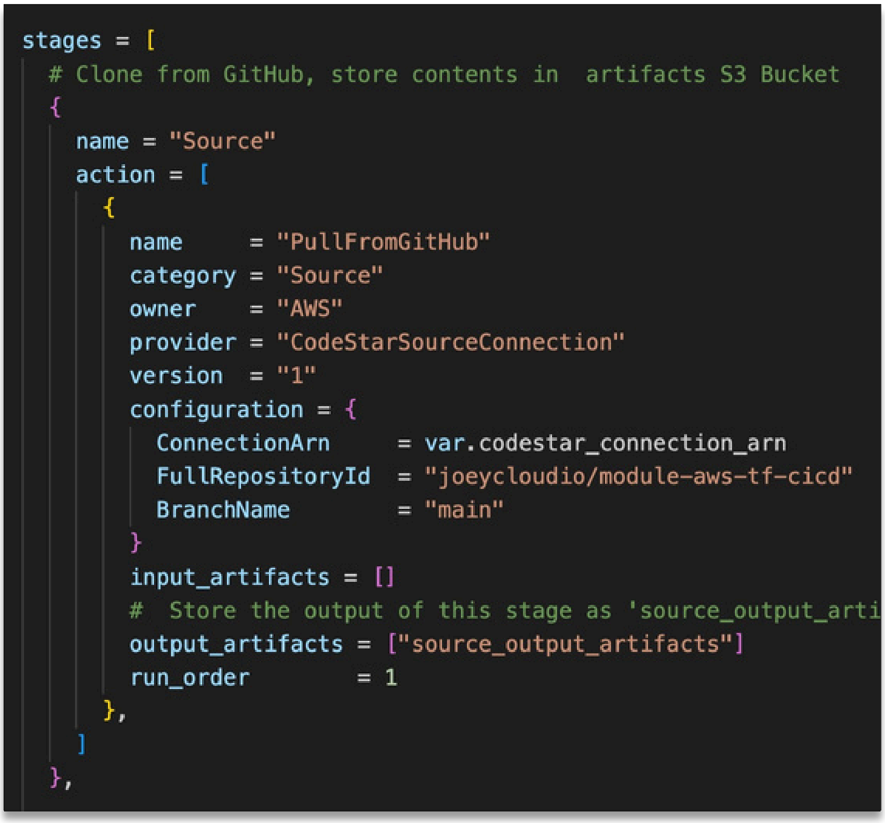
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	terraform.tfstate	tfstate	April 11, 2025, 23:35:11 (UTC-07:00)	127.8 KB	Standard

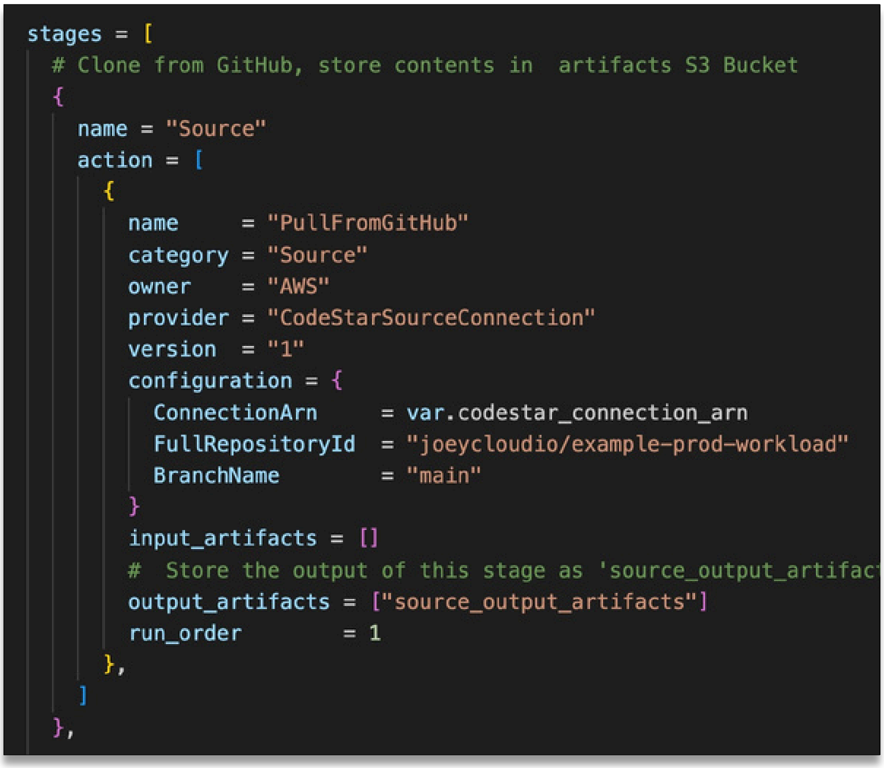
6. Refactored the project to replace AWS CodeCommit with GitHub and CodeStar ConnecCons, resolving all repo trigger issues and enabling automated pipeline execuCon.



```
variable "codestar_connection_arn" {
  description = "CodeStar Connection ARN for GitHub integration"
  type        = string
```



```
stages = [
  # Clone from GitHub, store contents in  artifacts S3 Bucket
  {
    name = "Source"
    action = [
      {
        name     = "PullFromGitHub"
        category = "Source"
        owner    = "AWS"
        provider = "CodeStarSourceConnection"
        version  = "1"
        configuration = {
          ConnectionArn     = var.codestar_connection_arn
          FullRepositoryId  = "joeycloudio/module-aws-tf-cicd"
          BranchName        = "main"
        }
        input_artifacts = []
        #  Store the output of this stage as 'source_output_arti
        output_artifacts = ["source_output_artifacts"]
        run_order        = 1
      },
    ]
  },
```



```
stages = [
  # Clone from GitHub, store contents in  artifacts S3 Bucket
  {
    name = "Source"
    action = [
      {
        name     = "PullFromGitHub"
        category = "Source"
        owner    = "AWS"
        provider = "CodeStarSourceConnection"
        version  = "1"
        configuration = {
          ConnectionArn     = var.codestar_connection_arn
          FullRepositoryId  = "joeycloudio/example-prod-workload"
          BranchName        = "main"
        }
        input_artifacts = []
        #  Store the output of this stage as 'source_output_artifac
        output_artifacts = ["source_output_artifacts"]
        run_order        = 1
      },
    ]
  },
```

Key Accomplishments

- Replaced CodeCommit with GitHub integra=on across both pipelines
- Built modular, testable Terraform infrastructure using`terraform test framework`
- Integrated security scanning (Checkov) and lin=ng (TFLint)
- Configured and tested remote S3/DynamoDB backend
- Migrated local Terraform state to remote backend with zero dri_
- Troubleshot and resolved CodePipeline ARN issues, buildspec errors, and webhook bugs
- Successfully deployed and verified 50+ AWS resources end-to-end

Key Learnings

- Real-world Terraform debugging is o_e n reverse-engineering someone else's design
- CI/CD pipelines require precise wiring: GitHub → CodeStar → CodePipeline → CodeBuild
- Remote state config must be handled carefully, especially post-deploy
- Modular Terraform doesn't mean every folder should be a root module
- Git and Terraform interac=on (rebase, state dri_, force-unlock) mager more than tutorials admit

Why This MaCers in Produc?on

This project reflects actual prac=ces used in infrastructure teams—tes=ng Terraform modules, automa=ng secure deployments, and managing state with best prac=ces. It's built for scale, collabora=on, and real-world reliability, not just a local sandbox.

Next Steps

- Add a manual approval stage + SNS no=fica=ons (op=onal stretch goal)
- Prac=ce deploying to a second AWS account via cross-account IAM roles
- Reuse modules and pipelines for future real-world Terraform repos