Files

https://github.com/MarylouBer/MLD_Data_Engineering_portfolio/tree/main/project14-IaC%20_CICD_GithubActions

+ The yml is in the root in the github workflow folder https://github.com/MarylouBer/MLD_Data_Engineering_portfolio/tree/main/.github !!!

Repo Architecture

```
MLD_Data_Engineering_portfolio/        <-- ROOT of Repository
│
├── .github/                           <-- MUST be at Root
│   └── workflows/
│       └── terraform.yml              <-- The Brain
│
├── project14-IaC_CICD_GithubActions/  <-- The Project Subfolder
│   ├── backend.tf
│   ├── main.tf
│   ├── providers.tf
│   ├── variables.tf
│   ├── .gitignore                     <-- Should be inside here
│   │
│   └── bootstrap/                     <-- Bootstrap is inside Project 14
│       ├── bootstrap.tf
│       └── ...
│
└── README.md                          <-- (Optional) Main Portfolio README
```

Steps

Phase 1: One-Time Setup (Bootstrap)

Goal: Create the "Foundation" (S3 Bucket & IAM Role) so the GitHub robot has a place to store its memory and permission to touch AWS.

This is deployed via your local machine

1. **Login to AWS Locally**

2. **In the terminal go to the Bootstrap folder:** `cd project14-IaC_CICD_GithubActions/bootstrap`
3. **Run Terraform Locally:** `terraform init - terraform plan - terraform apply` *(This creates the S3 Bucket and the OIDC Role). When you do terraform apply, you will get 2 outputs: Role ARN and Bucket. These are visible in your terminal and also in the automatically generated state file that is saved in your local repo*

4. **Get the Role ARN:** Copy the `output` value `"arn:aws:iam::302862750751:role/GitHubActionsTerraformRole"`
5. **Save Secret in GitHub:**
    - Go to GitHub Repo → Settings → Secrets and variables → Actions.
    - Create New Secret: `AWS_ROLE_ARN` = (Paste the ARN `"arn:aws:iam::302862750751:role/GitHubActionsTerraformRole"`

Phase 2: Configuration (Connecting the Dots)

Goal: Tell the main project where the foundation is.

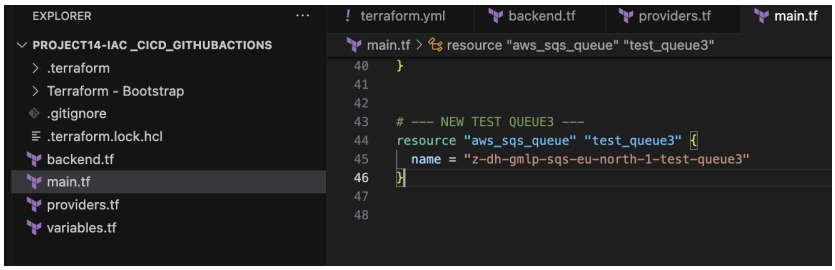
1. **Update `backend.tf`:** Open locally `project14-IaC_CICD_GithubActions/backend.tf` and make sure the `bucket` name matches the bucket you just created in Phase 1 outputs `"terraform-githubtest-tfstatefile-bucket-mld01"`
2. **Update Workflow YAML:** Ensure `.github/workflows/terraform.yml` has the correct path: `working-directory: ./project14-IaC_CICD_GithubActions.`

3. **Commit & Push:** from your terminal make sure you are in project14 directory and not in bootstrap like for phase 1. `git add . git commit -m "setup backend and workflow" git push origin main`

Phase 3: The Daily "GitOps" Workflow

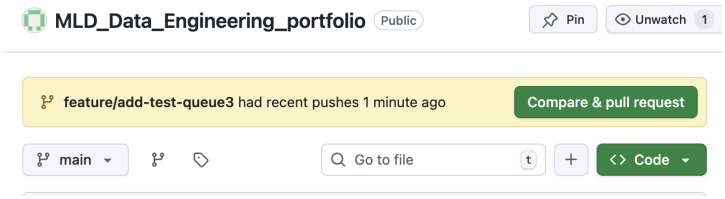
Goal: Making changes safely using the robot.

1. **Create a Branch:** Never work on `main`. To create new branch, make sure you are in the project14 directory in your terminal and enter the command `git checkout -b feature/add-sqs-queue`
2. **Write Code:** Edit `main.tf` (e.g., add a new resource).

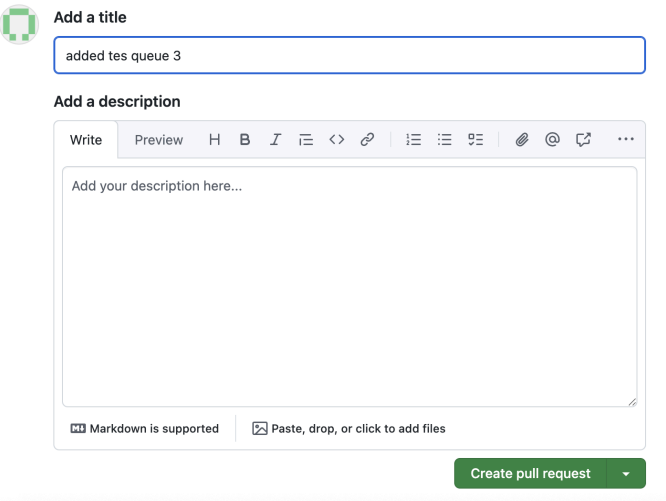


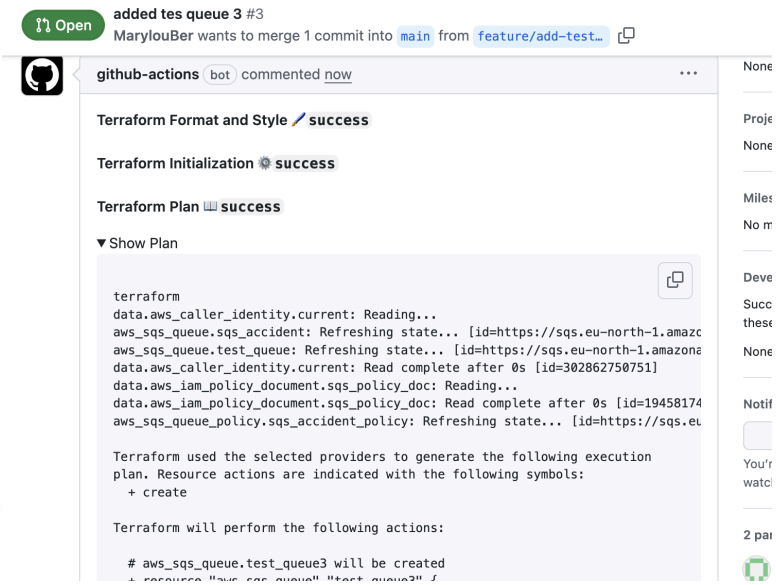
3. **Push Branch:** `git add . git commit -m "added new queue" git push origin feature/add-sqs-queue`
4. **Open Pull Request (PR):** Go to GitHub and open a PR.

Click on Compare pull request

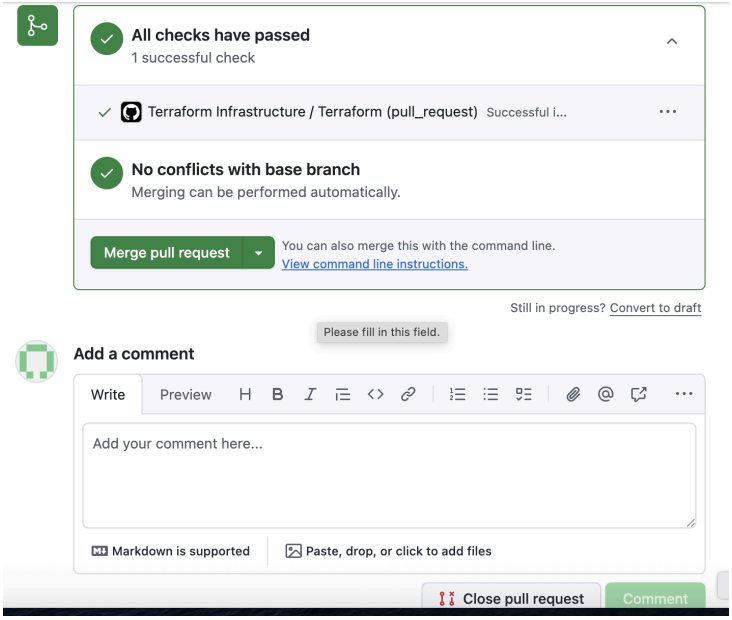


Click on Create pull request
The Robot Acts: It runs `terraform plan`.
You Review: Check the comment the robot posts. Does it say "Plan: 1 to add"?





5. **Merge:** If the plan looks correct, click **"Merge pull request"**.
    - **The Robot Acts:** It runs `terraform apply`.
    - **Done:** Your infrastructure is live on AWS (it might take some minutes to be visible on AWS)



If you don't want to merge, you can also comment and close the PR and create a new one

Phase 4: Branch Cleanup

If you are done with the merge

1. **Delete feature branch from remote repo**
   Run the command in your terminal inside the project14 directory
   git push origin --delete feature/add-test-queue

2. **Delete feature branch from local repo**
   Run the command in your terminal inside the project14 directory
   git checkout main
   git branch -D feature/add-test-queue2 feature/add-test-queue

Phase 4: Deep Cleanup

If you want to remove all resources created by this project14 from AWS

1. **Destroy Main Resources:** run `terraform destroy` locally from your laptop terminal inside the `project14` folder.
2. **Destroy Bootstrap (Final Step):** Only do this if you are **deleting the project forever**. Run `terraform destroy` inside the `bootstrap` folder on your terminal

Note:
If we want to implement a safe and more automated way to destroy all resources created:

To implement a **Safe Destroy Workflow**, you should create a completely separate GitHub Action that **only runs manually** (not on push) and requires a **confirmation password** to execute.

This prevents accidental deletion of your infrastructure.

1. Create the Destroy File

Create a new file in your repo: `.github/workflows/destroy.yml`

2. The Code

Copy and paste this code. I have added a safety check where you **must** type the word `destroy` in a popup box, otherwise the robot will refuse to run.

YAML

```yaml
name: "Terraform Destroy (Manual)"

on:
  workflow_dispatch:  # <--- This makes it a manual button click only
    inputs:
      confirmation:
        description: 'Type "destroy" to confirm deletion'
        required: true
        default: ''

permissions:
  id-token: write   # Required for OIDC authentication
  contents: read

jobs:
  destroy:
    name: "Destroy Infrastructure"
    runs-on: ubuntu-latest
    # SAFETY CHECK: Stop immediately if the user didn't type "destroy"
    if: ${{ github.event.inputs.confirmation == 'destroy' }}

    defaults:
      run:
        shell: bash
        working-directory: ./project14-IaC_CICD_GithubActions

    steps:
      - name: Checkout Code
        uses: actions/checkout@v4

      - name: Configure AWS Credentials (OIDC)
        uses: aws-actions/configure-aws-credentials@v3
        with:
          role-to-assume: ${{ secrets.AWS_ROLE_ARN }}
          aws-region: us-east-1 # Change if you use a different region

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v3

      - name: Terraform Init
        run: terraform init

      - name: Terraform Destroy
        run: terraform destroy -auto-approve
```

3. How to Use It (The Workflow)

Once you push this file to GitHub, here is how you run it:

1. Go to your GitHub Repository page.
2. Click the **Actions** tab at the top.
3. On the left sidebar, click **"Terraform Destroy (Manual)"**.
4. You will see a blue banner that says **"Run workflow"**. Click it.
5. A box will appear asking for input.
    - If you leave it blank: **Nothing happens.**
    - If you type `delete`: **Nothing happens.**
    - If you type `destroy`: **The robot wakes up and deletes everything.**

4. Why this is "Safe"

- **Manual Only:** It will never run automatically when you push code.
- **Input Guard:** The `if: ${{ github.event.inputs.confirmation == 'destroy' }}` line ensures that even if you accidentally click the button, the job cancels itself unless you explicitly typed the password.

⚠️ Important Note

This workflow destroys the **Project 14 resources** (SQS, Lambda, etc.). It does **not** destroy the Bootstrap (S3 Bucket & IAM Role). You should keep the Bootstrap alive so you can re-deploy Project 14 later if you want to!