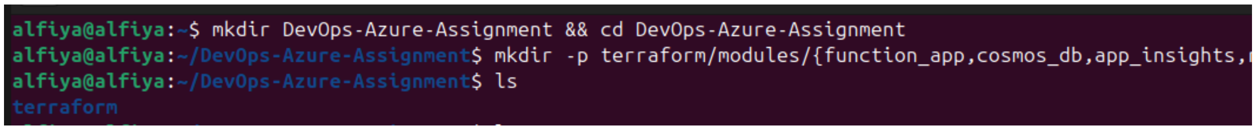
1. Scaling & Cost Optimization Strategy

   Scalable Design
   - Using Azure Functions with Premium plan for automatic scaling up to 25k+ students
   - Using Cosmos DB with autoscale feature for better performance
   - Can add Azure Front Door (CDN) to reduce latency

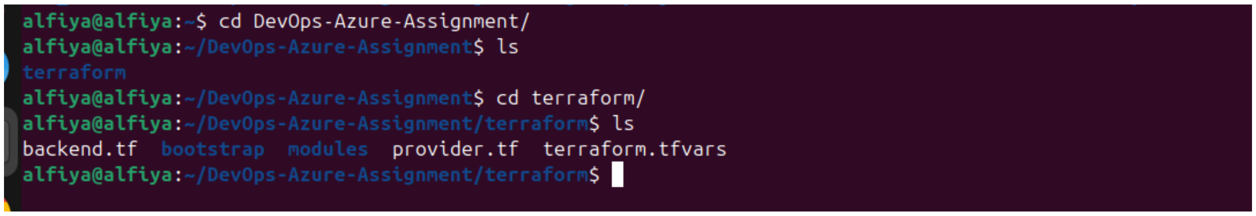
   Cost Optimization
   - Clean up unused resources using Terraform destroy and enforce resource tagging to track costs per environment.
   - Using Github Actions since its open source
   - Set Azure Monitor autoscale rules for Functions and App Service Plans.

   Steps

2. Repository Setup
   - Create a new GitHub repository with a clear and organized structure.
   - Inside the repo, add a terraform/ directory to hold backend and provider configuration.



```
alfiya@alfiya:~$ mkdir DevOps-Azure-Assignment && cd DevOps-Azure-Assignment
alfiya@alfiya:~/DevOps-Azure-Assignment$ mkdir -p terraform/modules/{function_app,cosmos_db,app_insights,
alfiya@alfiya:~/DevOps-Azure-Assignment$ ls
terraform
```



```
alfiya@alfiya:~$ cd DevOps-Azure-Assignment/
alfiya@alfiya:~/DevOps-Azure-Assignment$ ls
terraform
alfiya@alfiya:~/DevOps-Azure-Assignment$ cd terraform/
alfiya@alfiya:~/DevOps-Azure-Assignment/terraform$ ls
backend.tf  bootstrap  modules  provider.tf  terraform.tfvars
alfiya@alfiya:~/DevOps-Azure-Assignment/terraform$
```

3. Define Provider and Backend
   - In terraform/backend.tf, configure Terraform to use AzureRM as the provider and backend:

```
terraform {
  backend "azurerm" {}
}

provider "azurerm" {
  features {}
}
```

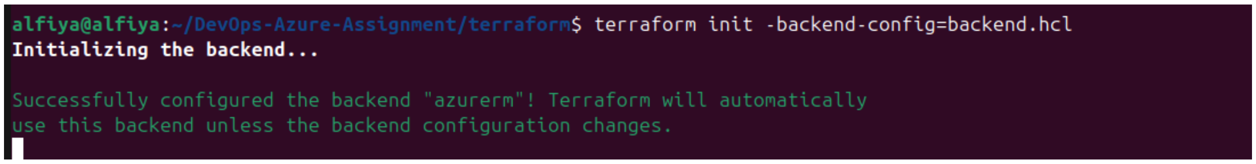
4. Create the Backend Configuration File
    - Inside the terraform/ directory, create a file called backend.hcl with the following content:

```
resource_group_name  = "tfstate-rg"
storage_account_name = "tfstateabcd1234"
container_name       = "tfstate"
key                  = "staging.terraform.tfstate"
```

5. Link Your Project to the Backend
    - Navigate to the root Terraform project directory and initialize Terraform using the backend file:

```
terraform init -backend-config=backend.hcl
```

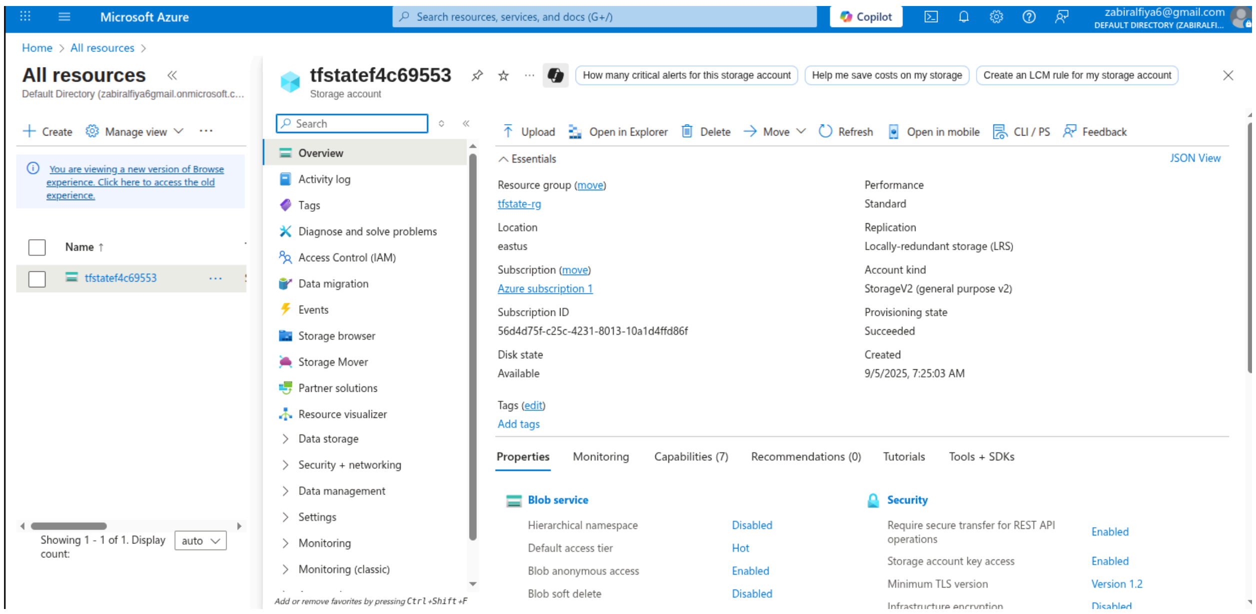
- This command migrates your local Terraform state into the Azure Storage backend.



```
alfiya@alfiya:~/DevOps-Azure-Assignment/terraform$ terraform init -backend-config=backend.hcl
Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.
```

- Output:


tfstatef4c69553
Storage account

Essentials

Resource group (move)
tfstate-rg

Location
eastus

Subscription (move)
Azure subscription 1

Subscription ID
56d4d75f-c25c-4231-8013-10a1d4ffd86f

Disk state
Available

Performance
Standard

Replication
Locally-redundant storage (LRS)

Account kind
StorageV2 (general purpose v2)

Provisioning state
Succeeded

Created
9/5/2025, 7:25:03 AM

Tags (edit)
Add tags

Blob service

Hierarchical namespace	Disabled
Default access tier	Hot
Blob anonymous access	Enabled
Blob soft delete	Disabled

Security

Require secure transfer for REST API operations	Enabled
Storage account key access	Enabled
Minimum TLS version	Version 1.2
Infrastructure encryption	Disabled

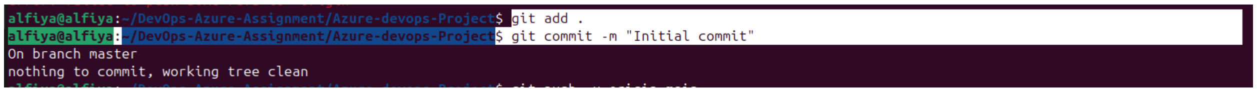
6. Run Terraform Commands
   - Once initialized, run the standard Terraform workflow:

```
Terraform init
Terraform validate
Terraform plan
Terraform apply -auto-approve
```

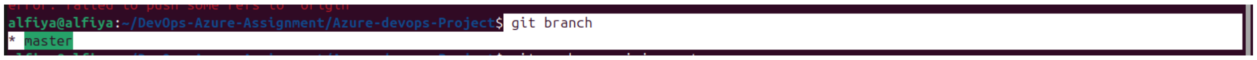
7. Push to GitHub
   - Before pushing, create a .gitignore file to exclude large or sensitive files.
   - Then run:

```
git add .
git commit -m "initial commit"
git push -u origin <branch_name>
```

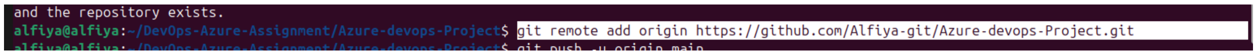
   - Outputs:



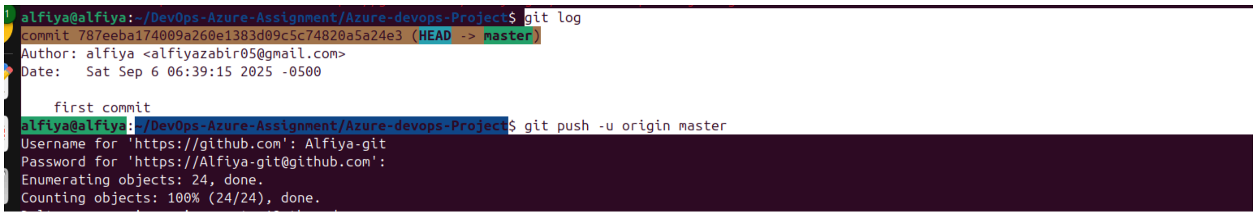
```
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git add .
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git commit -m "Initial commit"
On branch master
nothing to commit, working tree clean
```



```
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git branch
* master
```

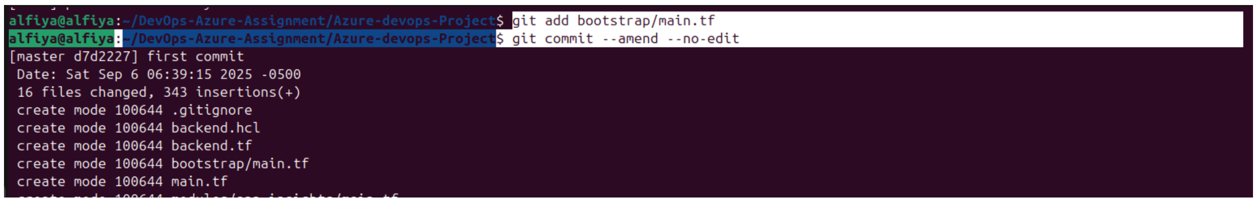


```
and the repository exists.
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git remote add origin https://github.com/Alfiya-git/Azure-devops-Project.git
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git push -u origin main
```

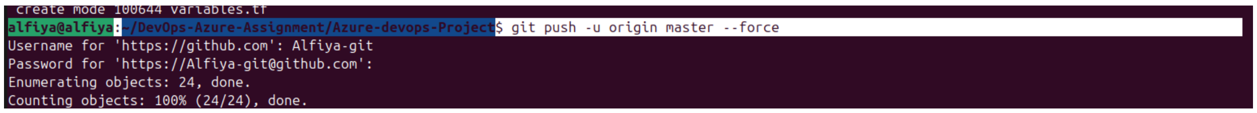


```
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git log
commit 787eeba174009a260e1383d09c5c74820a5a24e3 (HEAD -> master)
Author: alfiya <alfiyazabir05@gmail.com>
Date:   Sat Sep 6 06:39:15 2025 -0500

    first commit
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git push -u origin master
Username for 'https://github.com': Alfiya-git
Password for 'https://Alfiya-git@github.com':
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
```



```
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git add bootstrap/main.tf
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git commit --amend --no-edit
[master d7d2227] first commit
 Date: Sat Sep 6 06:39:15 2025 -0500
 16 files changed, 343 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 backend.hcl
 create mode 100644 backend.tf
 create mode 100644 bootstrap/main.tf
 create mode 100644 main.tf
```

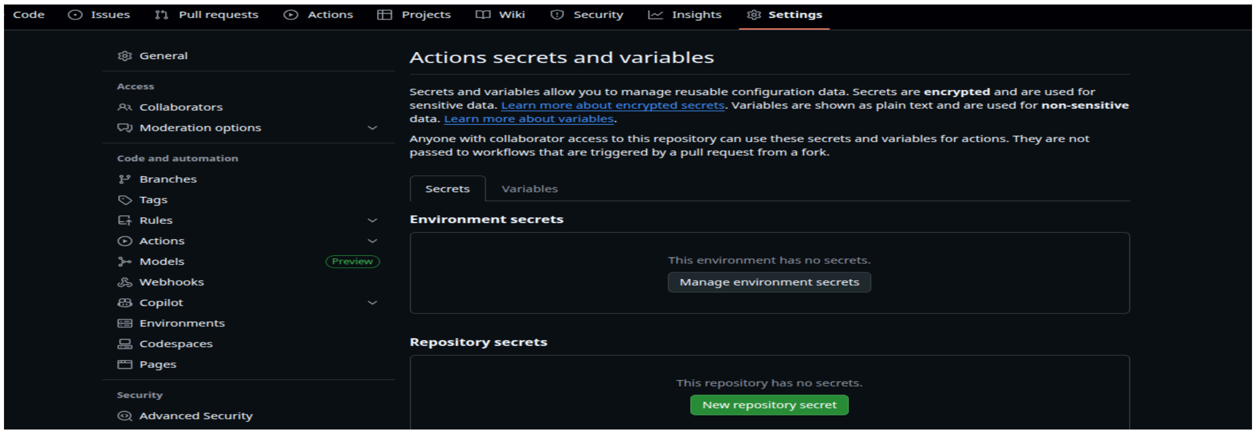


```
 create mode 100644 variables.tf
alfiya@alfiya:~/DevOps-Azure-Assignment/Azure-devops-Project$ git push -u origin master --force
Username for 'https://github.com': Alfiya-git
Password for 'https://Alfiya-git@github.com':
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
```

8. Store Azure Credentials in GitHub Secrets
    - Go to your repository → Settings → Secrets and variables → Actions.
    - Click New repository secret.
    - Name: AZURE_CREDENTIALS
    - Value: paste the JSON service principal credentials, for example:

```
{
  "clientId": "<APPLICATION_CLIENT_ID>",
  "clientSecret": "<SECRET_VALUE>",
  "subscriptionId": "56d4d75f-c25c-4231-8013-10a1d4ffd86f",
  "tenantId": "<TENANT_ID>",
  "activeDirectoryEndpointUrl":
"https://login.microsoftonline.com",
  "resourceManagerEndpointUrl":
"https://management.azure.com/",
  "activeDirectoryGraphResourceId":
"https://graph.windows.net/",
  "sqlManagementEndpointUrl":
"https://management.core.windows.net:8443/",
  "galleryEndpointUrl": "https://gallery.azure.com/",
  "managementEndpointUrl":
```

```
"https://management.core.windows.net/"
}
```



Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for **non-sensitive** data. Learn more about variables.

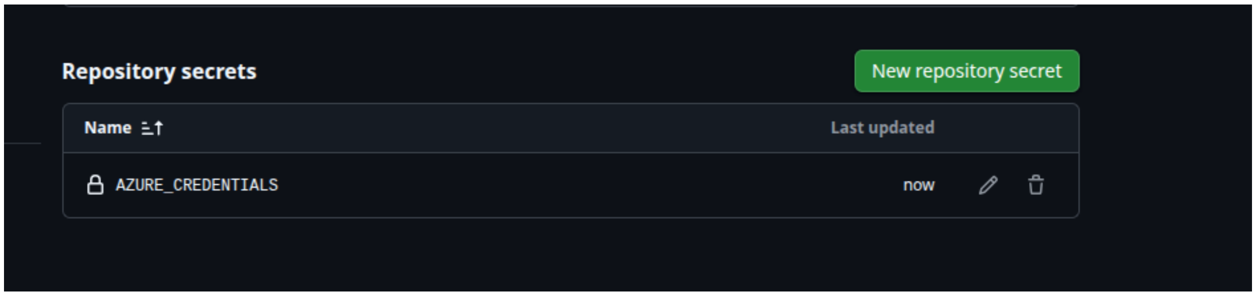
Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Environment secrets

This environment has no secrets.

Repository secrets

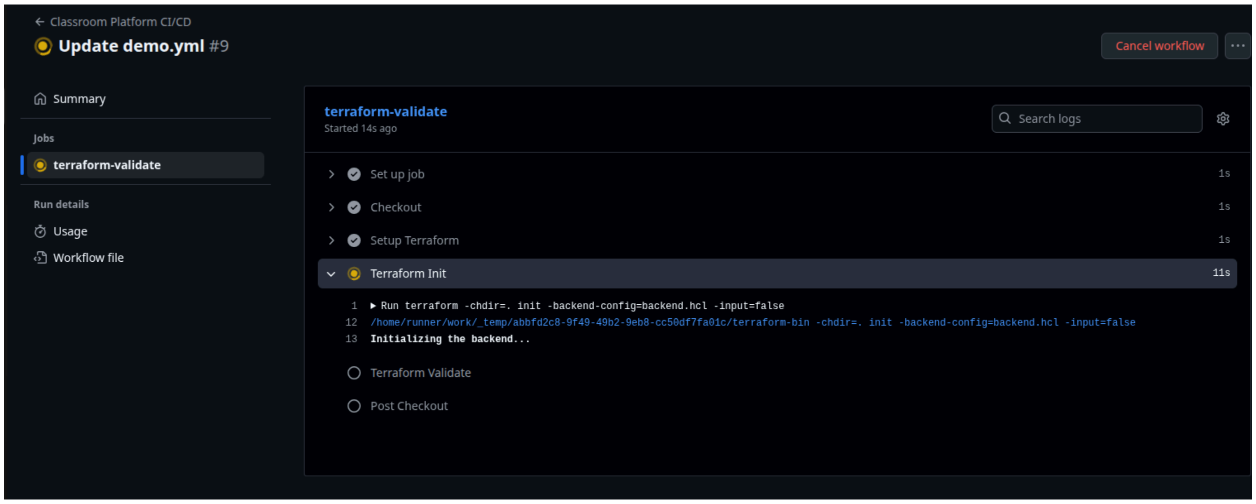
This repository has no secrets.



Repository secrets

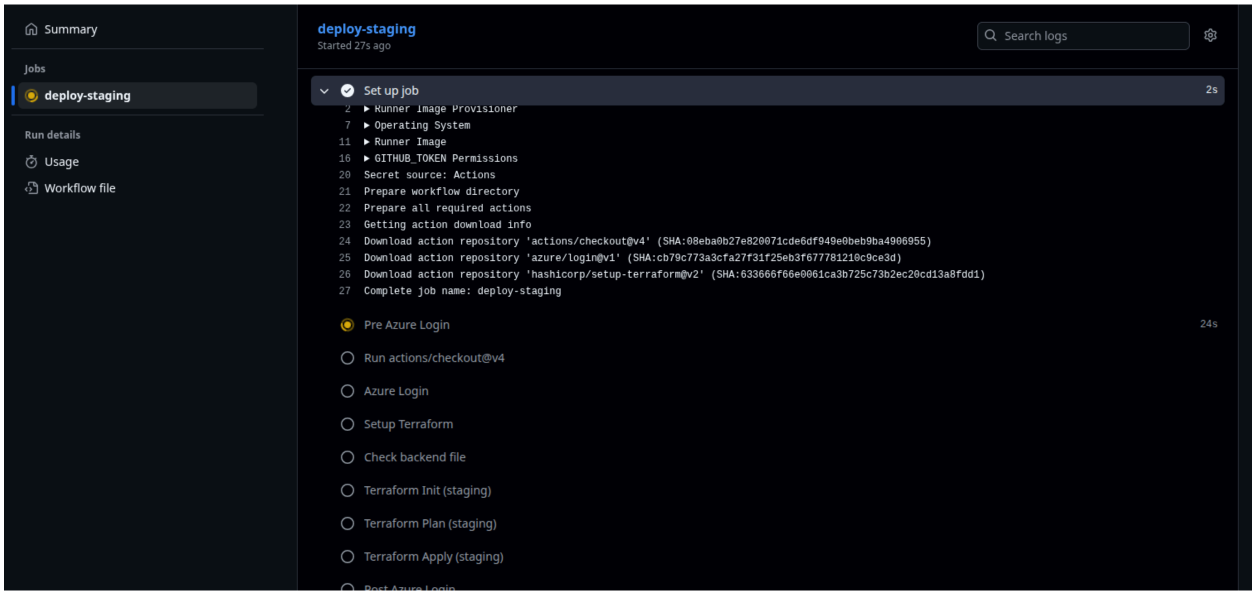
Name	Last updated
AZURE_CREDENTIALS	now

9. Configure CI/CD with GitHub Actions
    - In your repo, go to the Actions tab.
    - Set up a new workflow.
    - Define jobs for terraform init, validate, plan, and apply.
    - Reference the AZURE_CREDENTIALS secret for authentication.



terraform-validate
Started 14s ago

```
Set up job                1s
Checkout                  1s
Setup Terraform           1s
Terraform Init            11s
 1 ▶ Run terraform -chdir=. init -backend-config=backend.hcl -input=false
12 /home/runner/work/_temp/abbfd2c8-9f49-49b2-9eb8-cc50df7fa01c/terraform-bin -chdir=. init -backend-config=backend.hcl -input=false
13 Initializing the backend...
Terraform Validate
Post Checkout
```



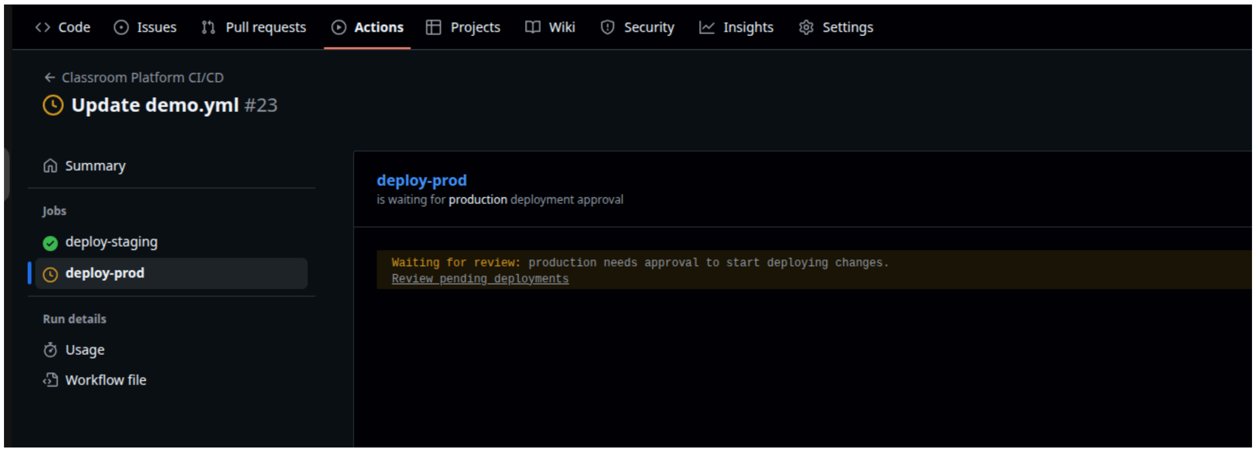
deploy-staging
Started 27s ago

Set up job 2s

```
▶ Runner Image Provisioner
▶ Operating System
▶ Runner Image
▶ GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Getting action download info
Download action repository 'actions/checkout@v4' (SHA:08eba0b27e820071cde6df949e0beb9ba4906955)
Download action repository 'azure/login@v1' (SHA:cb79c773a3cfa27f31f25eb3f677781210c9ce3d)
Download action repository 'hashicorp/setup-terraform@v2' (SHA:633666f66e0061ca3b725c73b2ec20cd13a8fdd1)
Complete job name: deploy-staging
```

Pre Azure Login 24s
Run actions/checkout@v4
Azure Login
Setup Terraform
Check backend file
Terraform Init (staging)
Terraform Plan (staging)
Terraform Apply (staging)
Post Azure Login

10. Manual Approval for Production
    - Configure environment protection in GitHub. Go to your repo → Settings → Environments → Click New environment → Name it production.

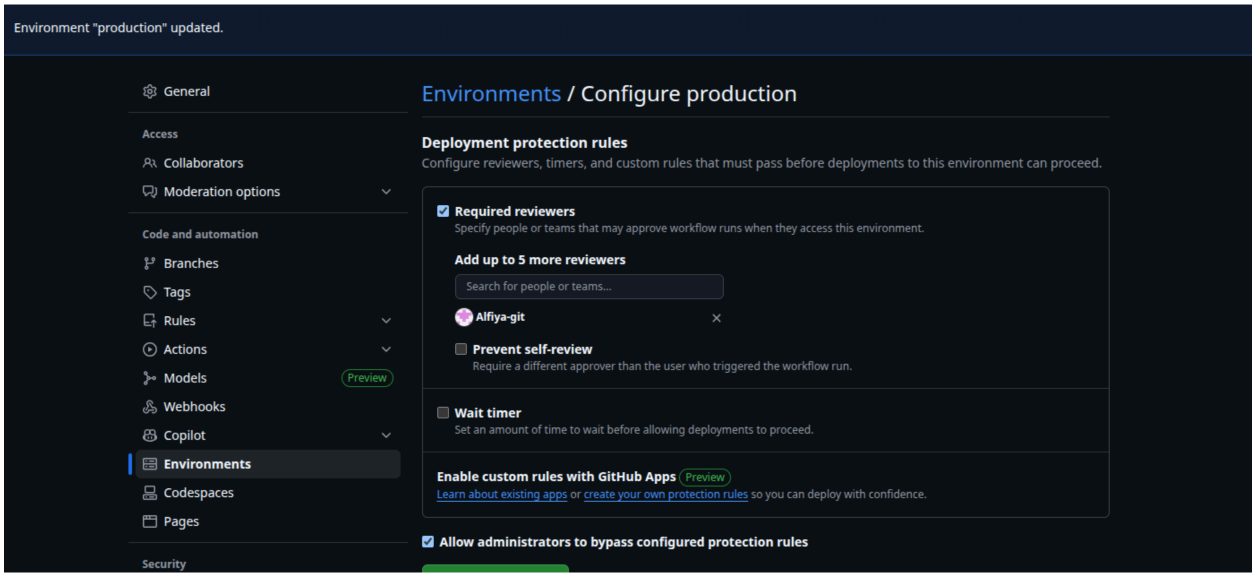


Update demo.yml #23

deploy-prod
is waiting for production deployment approval

```
Waiting for review: production needs approval to start deploying changes.
Review pending deployments
```

- Under Deployment protection rules, set Required reviewers → Choose who must approve before the job runs. select one or more GitHub users/teams who must approve and save.


Environment "production" updated.

Environments / Configure production

Deployment protection rules
Configure reviewers, timers, and custom rules that must pass before deployments to this environment can proceed.

[✓] **Required reviewers**
Specify people or teams that may approve workflow runs when they access this environment.

Add up to 5 more reviewers
Search for people or teams...
Alfiya-git

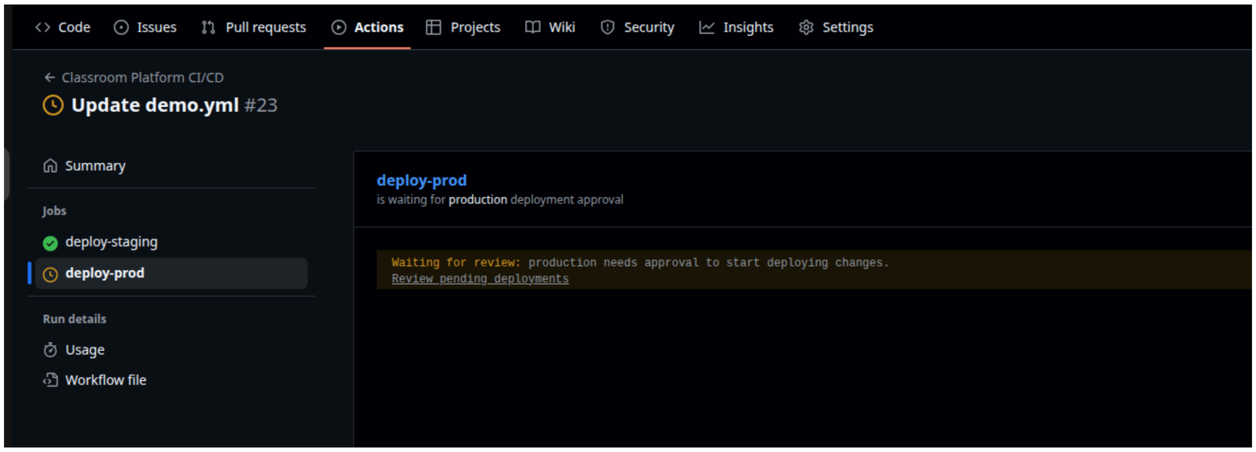
[ ] **Prevent self-review**
Require a different approver than the user who triggered the workflow run.

[ ] **Wait timer**
Set an amount of time to wait before allowing deployments to proceed.

Enable custom rules with GitHub Apps Preview
Learn about existing apps or create your own protection rules so you can deploy with confidence.

[✓] **Allow administrators to bypass configured protection rules**

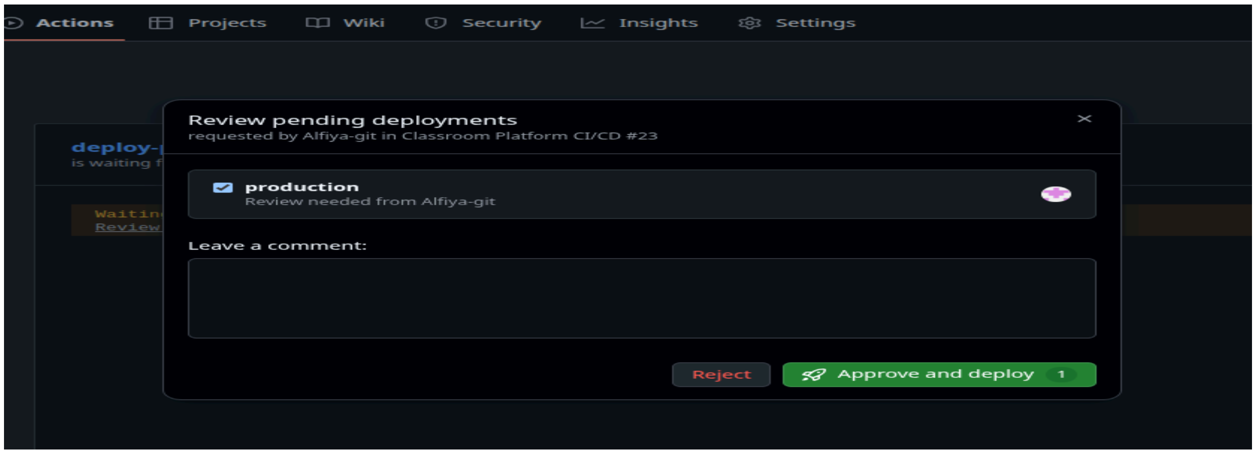
- GitHub will pause the job and show "Awaiting approval" until someone from the required reviewers list approves it.



Classroom Platform CI/CD
Update demo.yml #23

deploy-prod
is waiting for production deployment approval

```
Waiting for review: production needs approval to start deploying changes.
Review pending deployments
```

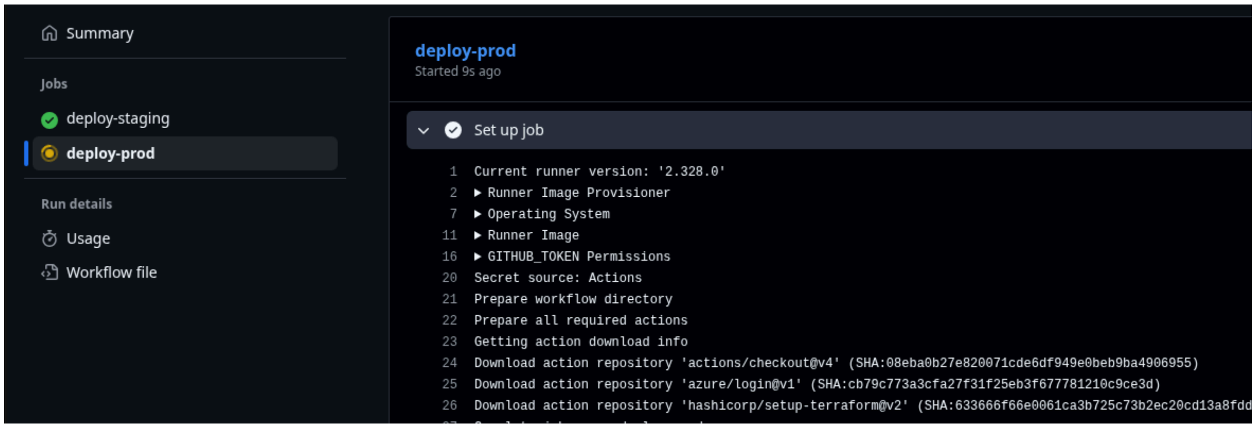


Review pending deployments
requested by Alfiya-git in Classroom Platform CI/CD #23

[✓] **production**
Review needed from Alfiya-git

Leave a comment:

- Since its approved the workflow has started to execute.



```
Current runner version: '2.328.0'
▶ Runner Image Provisioner
▶ Operating System
▶ Runner Image
▶ GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Getting action download info
Download action repository 'actions/checkout@v4' (SHA:08eba0b27e820071cde6df949e0beb9ba4906955)
Download action repository 'azure/login@v1' (SHA:cb79c773a3cfa27f31f25eb3f677781210c9ce3d)
Download action repository 'hashicorp/setup-terraform@v2' (SHA:633666f66e0061ca3b725c73b2ec20cd13a8fdd
```

11. Add a CI/CD Smoke Test Job
    - Code for smoke test

```
smoke-test:
    runs-on: ubuntu-latest
    needs: deploy-prod
    environment: production
    steps:
      - name: Run smoke test
        run: |
          response=$(curl -s -o /dev/null -w "%{http_code}"
https://classroom_app_url.com)
          if [ "$response" != "200" ]; then
            echo "Smoke test failed"
            exit 1
          fi
          echo "Smoke test passed"
```

12. Use Azure Cost Management Alerts
    - In Azure Portal go to Cost Management + Billing then Budgets.
    - Create a Budget
    - Add Alert rules and send email if you exceed thresholds.

13. FERPA / GDPR Compliance

FERPA: Student data privacy

- Only authorized roles (teacher/admin) can access sensitive records (RBAC).
- All access logged via Azure Monitor.

GDPR: Right to be forgotten & consent

- Data retention rules automatically delete expired data.
- Backups have expiration
- PII stored in encrypted form in Cosmos DB.

14. Secret Storage (Azure Key Vault)

```
resource "azurerm_key_vault" "kv" {
  name                        = "myproject-kv"
  location                    =
azurerm_resource_group.rg.location
  resource_group_name         =
azurerm_resource_group.rg.name
  tenant_id                   =
data.azurerm_client_config.current.tenant_id
  sku_name                    = "standard"

  soft_delete_retention_days  = 90
  purge_protection_enabled    = true

  access_policy {
    tenant_id =
data.azurerm_client_config.current.tenant_id
    object_id =
azurerm_user_assigned_identity.func.identity[0].principal_i
d

    secret_permissions = ["Get", "List"]
  }
}
```

Instead of writing passwords or API keys directly in code which is risky, we lock them in a Key Vault. Only approved people have the access to open that safe. If someone hacks the app, they can’t automatically see the secrets.

15. DDoS Protection & Audit Logging

```
resource "azurerm_network_ddos_protection_plan" "ddos" {
  name                = "myproject-ddos"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_monitor_diagnostic_setting" "diag" {
  name                       = "diag-logs"
  target_resource_id         =
azurerm_cosmosdb_account.db.id
  log_analytics_workspace_id =
azurerm_log_analytics_workspace.law.id

  log {
    category = "DataPlaneRequests"
    enabled  = true
  }

  metric {
    category = "AllMetrics"
    enabled  = true
  }
}
```

16. Data Retention & Backups

```
resource "azurerm_cosmosdb_account" "db" {
  name                = "myproject-cosmos"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  offer_type          = "Standard"
  kind                = "GlobalDocumentDB"

  backup {
    type                = "Continuous"
    retention_in_hours  = 240
  }
}
```