Project Setup & Teardown – Where's Waldo App

1. Introduction

This document provides step-by-step instructions to set up and tear down the infrastructure for the Where's Waldo App using Terraform and AWS services. It includes screenshots, placeholders, and guidance for deployment and usage.

2. Prerequisites

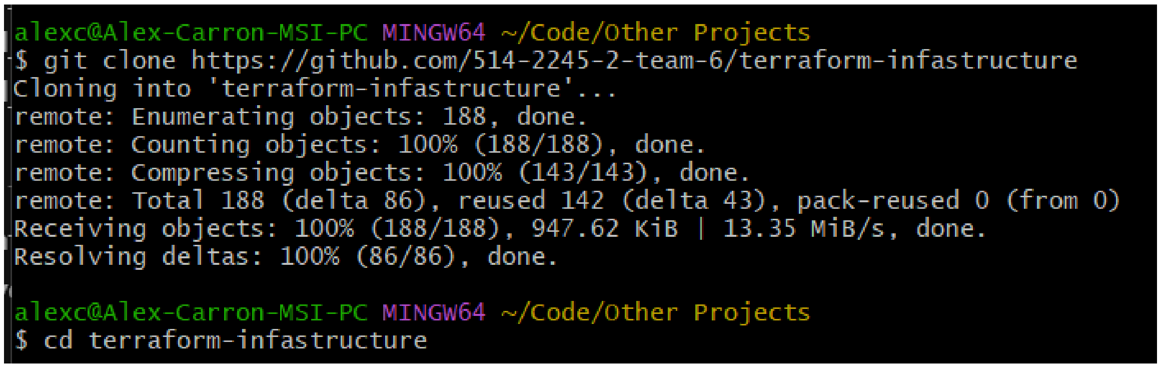
Before starting, ensure the following tools and credentials are available:

- AWS Account with required permissions
- Terraform installed
- AWS CLI installed and configured: Follow the official AWS CLI quickstart guide if you haven't installed it yet
- GitHub personal access token
- Valid email address for SNS notifications

3. Clone the Repository

Run the following commands in a terminal:

```
git clone
https://github.com/514-2245-2-team-6/terraform-infastructure

cd terraform-infastructure
```



```
alexc@Alex-Carron-MSI-PC MINGW64 ~/Code/Other Projects
$ git clone https://github.com/514-2245-2-team-6/terraform-infastructure
Cloning into 'terraform-infastructure'...
remote: Enumerating objects: 188, done.
remote: Counting objects: 100% (188/188), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 188 (delta 86), reused 142 (delta 43), pack-reused 0 (from 0)
Receiving objects: 100% (188/188), 947.62 KiB | 13.35 MiB/s, done.
Resolving deltas: 100% (86/86), done.

alexc@Alex-Carron-MSI-PC MINGW64 ~/Code/Other Projects
$ cd terraform-infastructure
```

4. Configure AWS CLI

Run the command below and provide your credentials when prompted:

```
aws configure
```

5. Configure Variables

5.1 Create a GitHub Personal Access Token

To create a GitHub personal access token with the required scopes:

1. Go to https://github.com/settings/tokens

2. Click **"Generate new token"** > **"Generate new token (classic)"**

3. Set an expiration date and token name

4. Select the following scopes:

   - `public_repo` (for public repositories)

   - `admin:repo_hook` (to allow webhook setup for automatic deployments)

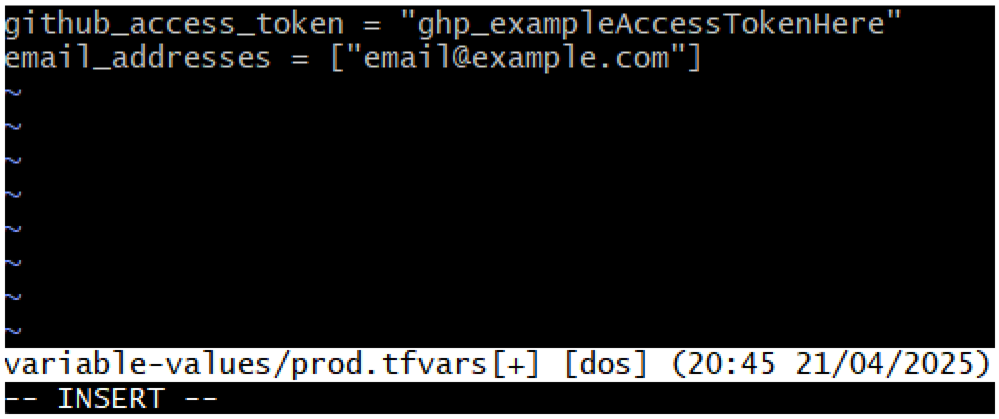
5. Click **"Generate token"** and copy the token (you won't be able to see it again)

5.2 Add Variables Values to File

Edit the file at `variable-values/prod.tfvars` and enter your GitHub access token and email address.

For example:

```
github_access_token = "ghp_exampleAccessTokenHere"
email_addresses     = ["email@example.com"]
```

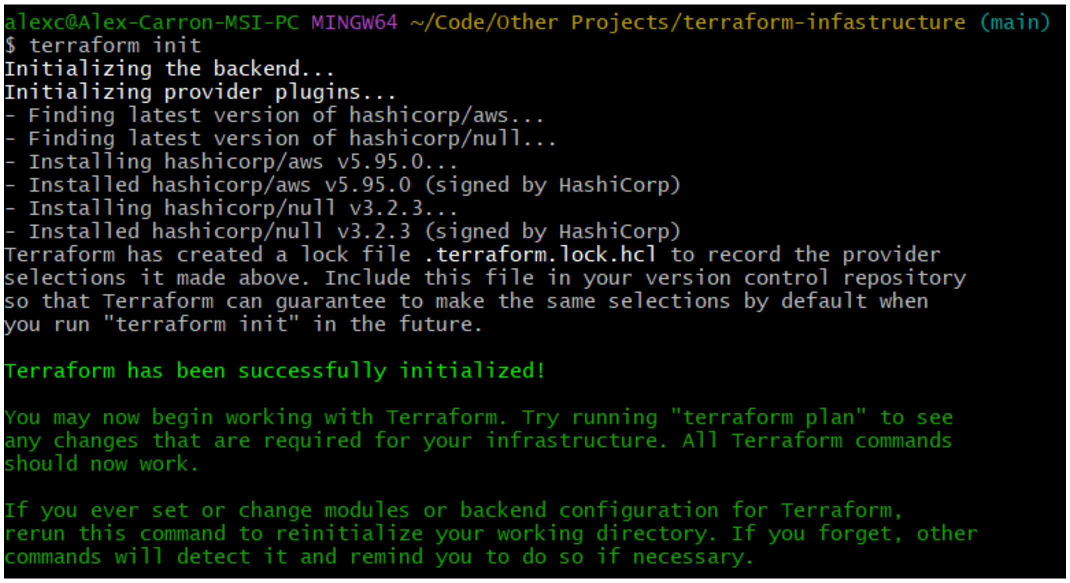


```
github_access_token = "ghp_exampleAccessTokenHere"
email_addresses = ["email@example.com"]
~
~
~
~
~
~
~
~
variable-values/prod.tfvars[+] [dos] (20:45 21/04/2025)
-- INSERT --
```

6. Initialize Terraform

Run the following command to initialize the Terraform workspace:

```
terraform init
```



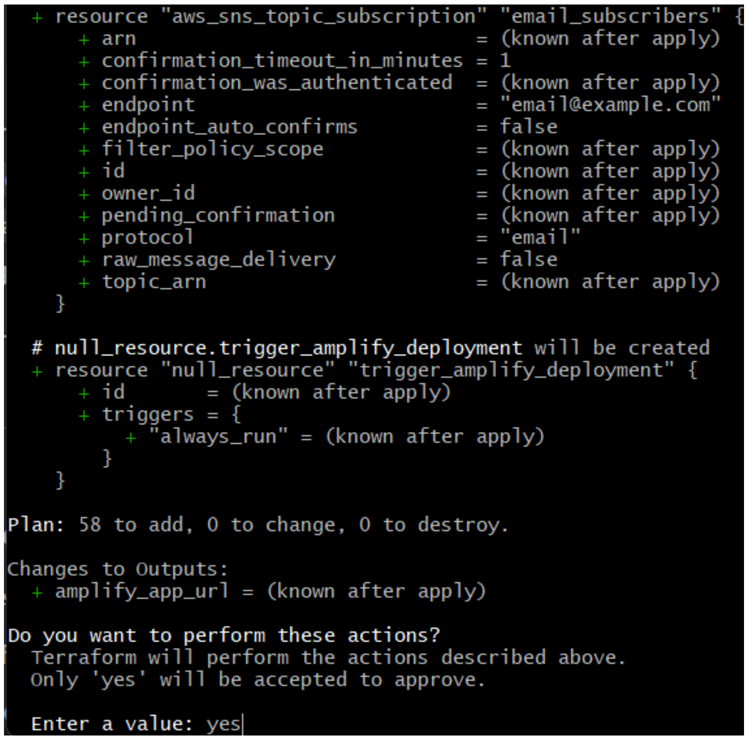
```
alexc@Alex-Carron-MSI-PC MINGW64 ~/Code/Other Projects/terraform-infastructure (main)
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/null...
- Installing hashicorp/aws v5.95.0...
- Installed hashicorp/aws v5.95.0 (signed by HashiCorp)
- Installing hashicorp/null v3.2.3...
- Installed hashicorp/null v3.2.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

7. Apply Terraform Configuration

Run this command to provision the infrastructure:

```
terraform apply -var-file="variable-values/prod.tfvars"
```

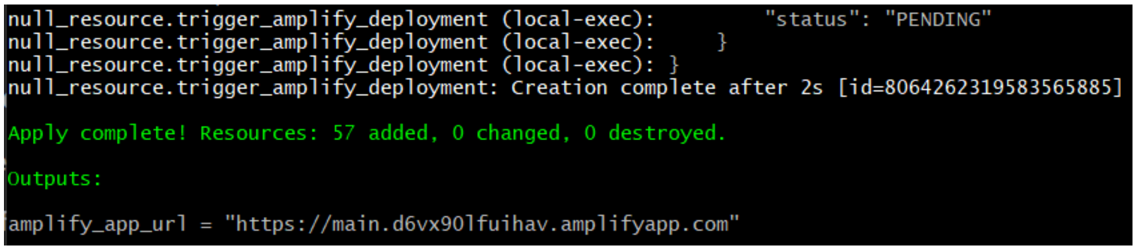
Type 'yes' when prompted to approve the plan.



```
  + resource "aws_sns_topic_subscription" "email_subscribers" {
      + arn                             = (known after apply)
      + confirmation_timeout_in_minutes = 1
      + confirmation_was_authenticated  = (known after apply)
      + endpoint                        = "email@example.com"
      + endpoint_auto_confirms          = false
      + filter_policy_scope             = (known after apply)
      + id                              = (known after apply)
      + owner_id                        = (known after apply)
      + pending_confirmation            = (known after apply)
      + protocol                        = "email"
      + raw_message_delivery            = false
      + topic_arn                       = (known after apply)
    }

  # null_resource.trigger_amplify_deployment will be created
  + resource "null_resource" "trigger_amplify_deployment" {
      + id       = (known after apply)
      + triggers = {
          + "always_run" = (known after apply)
        }
    }

Plan: 58 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + amplify_app_url = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

9. Using the Web App

After deployment, Terraform will display the app's URL. This URL points to the AWS Amplify-hosted front-end of the Where's Waldo application.



```
null_resource.trigger_amplify_deployment (local-exec):         "status": "PENDING"
null_resource.trigger_amplify_deployment (local-exec):     }
null_resource.trigger_amplify_deployment (local-exec): }
null_resource.trigger_amplify_deployment: Creation complete after 2s [id=8064262319583565885]

Apply complete! Resources: 57 added, 0 changed, 0 destroyed.

Outputs:

amplify_app_url = "https://main.d6vx90lfuihav.amplifyapp.com"
```

Although Terraform automatically configures and deploys the Amplify app, it may take 3–5 minutes for the build and deployment process to complete in the AWS Amplify console. If the app doesn't load immediately, wait a few minutes and refresh the page.

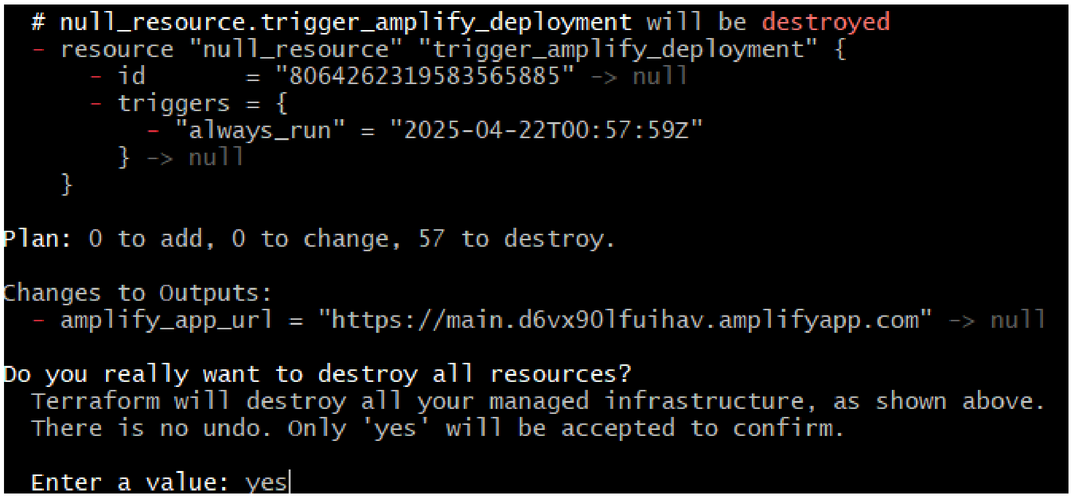
Once deployment finishes, open the provided URL in your browser to start using the app.

10. Teardown Instructions

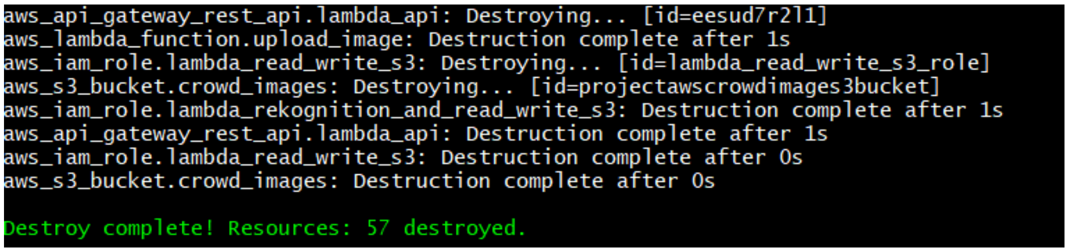
To remove all AWS resources, run:

```
terraform destroy -var-file="variable-values/prod.tfvars"
```

Type 'yes' to confirm.



```
  # null_resource.trigger_amplify_deployment will be destroyed
  - resource "null_resource" "trigger_amplify_deployment" {
      - id       = "8064262319583565885" -> null
      - triggers = {
          - "always_run" = "2025-04-22T00:57:59Z"
        } -> null
    }

Plan: 0 to add, 0 to change, 57 to destroy.

Changes to Outputs:
  - amplify_app_url = "https://main.d6vx90lfuihav.amplifyapp.com" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```



```
aws_api_gateway_rest_api.lambda_api: Destroying... [id=eesud7r2l1]
aws_lambda_function.upload_image: Destruction complete after 1s
aws_iam_role.lambda_read_write_s3: Destroying... [id=lambda_read_write_s3_role]
aws_s3_bucket.crowd_images: Destroying... [id=projectawscrowdimages3bucket]
aws_iam_role.lambda_rekognition_and_read_write_s3: Destruction complete after 1s
aws_api_gateway_rest_api.lambda_api: Destruction complete after 1s
aws_iam_role.lambda_read_write_s3: Destruction complete after 0s
aws_s3_bucket.crowd_images: Destruction complete after 0s

Destroy complete! Resources: 57 destroyed.
```