

HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)

I am in here, as my website files like, home.html, are in here but you asked me to make directory in
cd "/d/4th Semester/DevOps - Lab/Semester
Project/DevOpsZaid"
like main.tf, providers.tf are here

So should I go back to previous folder?

I am creating AWS task IAM user, create user, for automation git and docker tasks using AWS terraform, now i have created user, and copied paste the secret key access,id,etc and region,etc. now it is asking me to create directory for terraform as
"mkdir terraform-s3
cd terraform-s3
"

Ah! I see exactly what's happening. Let me clarify the **directory structure and workflow** so it's clear. Don't worry, it's common to get confused when mixing Terraform files and your website code.

1 Current situation

- Your **website files** (like `home.html`) are in:

swift

`/d/4th Semester/DevOps - Lab/Semester Project/DevOpsZaid`

- Your **Terraform files** (`main.tf`, `providers.tf`) are also in the same folder

Copy code

This can work, but it's **better to separate Terraform files** from your website code so things don't get messy.

2 Recommended structure

lua

```
DevOpsZaid/
|
|   website/           <-- your HTML, CSS, JS files
|       └── home.html
|
└── terraform/        <-- all Terraform stuff
    ├── main.tf
    ├── providers.tf
    └── terraform-s3/  <-- optional subfolder for S3 setup
```

So yes, you **should create a separate folder for Terraform**, like `terraform-s3` or just `terraform`, inside your main project folder.

 Copy code

3 Commands to create Terraform folder

From your main project folder:

bash

```
mkdir terraform-s3
cd terraform-s3
```

Now you're **inside Terraform folder**, and this is where you can put `main.tf`,  `providers.tf`, etc.

4 Why separate?

1. Keeps website files untouched.
2. Terraform state files (`terraform.tfstate`) won't clutter your website code.

3. Easier to manage infrastructure and deployments.

💡 Tip: You can keep `terraform-s3` just for S3 bucket tasks, and later make other folders like `terraform-ec2`, etc.

If you want, I can **write the exact folder setup for your project** so your website and Terraform stay clean and organized.

Do you want me to do that?



```
resource "aws_s3_bucket" "website_bucket" {
    bucket = "devopszaid-static-site-12345"
}

resource "aws_s3_bucket_public_access_block" "public_access" {
    bucket = aws_s3_bucket.website_bucket.id

    block_public_acls  = false
    block_public_policy = false
    ignore_public_acls = false
    restrict_public_buckets = false
}

resource "aws_s3_bucket_website_configuration" "website" {
    bucket = aws_s3_bucket.website_bucket.id

    index_document {
        suffix = "home.html"
    }
}
```

```
resource "aws_s3_bucket_policy" "public_policy" {
    bucket = aws_s3_bucket.website_bucket.id

    policy = jsonencode({
        Version = "2012-10-17"
        Statement = [
            Sid      = "PublicReadGetObject"
            Effect   = "Allow"
            Principal = "*"
            Action   = "s3:GetObject"
            Resource = "${aws_s3_bucket.website_bucket.arn}/*"
        ]
    })
}
```

Change bucket name to something globally unique
Example:

devopszaid-static-site-ad-2026

Perfect! The only change you need is to make the S3 bucket name **globally unique**, because AWS S3 bucket names are shared across all AWS accounts worldwide.

Here's your updated Terraform snippet with the new bucket name:

hcl

 Copy code

```
resource "aws_s3_bucket" "website_bucket" {
    bucket = "devopszaid-static-site-ad-2026"
}

resource "aws_s3_bucket_public_access_block" "public_access" {
    bucket = aws_s3_bucket.website_bucket.id

    block_public_acls      = false
    block_public_policy     = false
```

```
ignore_public_acls      = false
restrict_public_buckets = false
}

resource "aws_s3_bucket_website_configuration" "website" {
  bucket = aws_s3_bucket.website_bucket.id

  index_document {
    suffix = "home.html"
  }
}

resource "aws_s3_bucket_policy" "public_policy" {
  bucket = aws_s3_bucket.website_bucket.id

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid      = "PublicReadGetObject"
        Effect   = "Allow"
        Principal = "*"
        Action   = "s3:GetObject"
        Resource = "${aws_s3_bucket.website_bucket.arn}/*"
      }
    ]
  })
}
```

✓ Now your bucket name `devopszaid-static-site-ad-2026` is likely **globally unique** and ready to deploy.

If you want, I can **write the exact Terraform workflow** to create the S3 bucket, upload your website files (`home.html`), and make it publicly accessible step by step. This way, it'll work perfectly.

Do you want me to do that next?



```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)
$ aws s3 ls
2025-12-15 18:34:54 github-devops-terraform-aws-tfstates
2025-12-15 11:06:41 zaid-website-devops
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)
$ cd..
bash: cd..: command not found
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)
$ cd.
bash: cd.: command not found
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)
$ mkdir terraform-s3
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)
$ cd terraform-s3
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid/terraform-s3 (main)
$ pwd
/d/4th Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid/terraform-s3
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
```

```
Project/DevOpsZaid/DevOpsZaid/terraform-s3 (main)
```

```
$ notepad provider.tf
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
```

```
Semester/DevOps - Lab/Semester
```

```
Project/DevOpsZaid/DevOpsZaid/terraform-s3 (main)
```

```
$ terraform init
```

```
Terraform initialized in an empty directory!
```

The directory has no Terraform configuration files. You may begin working with Terraform immediately by creating Terraform configuration files.

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
```

```
Semester/DevOps - Lab/Semester
```

```
Project/DevOpsZaid/DevOpsZaid/terraform-s3 (main)
```

```
$
```

I have created both providers.tf, and main.tf file

Ah! I see exactly what happened here. The key point is: **Terraform didn't detect your files** when you ran `terraform init`. That usually happens if:

1. The files aren't saved yet in the folder.
2. The filenames are wrong (e.g., `provider.tf` instead of `providers.tf`).
3. The files are empty.

Let's fix this step by step.

1 Check the files exist and are correctly named

Run:

```
bash
```

 Copy code

```
ls
```

You should see something like:

`css` Copy code`main.tf``providers.tf`

-  Make sure it's `providers.tf` **not** `provider.tf` unless you really named it `provider.tf` —Terraform doesn't care about the name too much, but consistency is good.
-

2 Make sure the files are not empty

Open them in Notepad or VS Code and make sure:

- `providers.tf` has your AWS provider configuration, e.g.:

`hcl` Copy code

```
provider "aws" {
  region      = "us-east-1"
  access_key  = "YOUR_ACCESS_KEY"
  secret_key  = "YOUR_SECRET_KEY"
}
```

- `main.tf` has your S3 bucket code:

`hcl` Copy code

```
resource "aws_s3_bucket" "website_bucket" {
  bucket = "devopszaid-static-site-ad-2026"
}
# ... rest of your S3 code ...
```

3 Run Terraform commands

1. Initialize Terraform (to download AWS provider plugin):

`bash` Copy code

```
terraform init
```

You should now see output like:

```
nginx
```

 Copy code

```
Initializing the backend...
Initializing provider plugins...
Terraform has been successfully initialized!
```

2. Validate configuration (checks for syntax errors):

```
bash
```

 Copy code

```
terraform validate
```

3. Preview changes (shows what Terraform will create in AWS):

```
bash
```

 Copy code

```
terraform plan
```

4. Apply changes (creates S3 bucket):

```
bash
```

 Copy code

```
terraform apply
```

It will ask for confirmation (`yes`), then create your S3 bucket.

Tip:

If you're in Git Bash (MINGW64), the commands `cd..` and `cd.` don't work. Use:

```
bash
```

 Copy code

```
cd ..
```

to go one folder up

and

```
bash
```

 Copy code

```
cd .
```

to stay in the **current folder** (usually not needed).

If you want, I can give you a **ready-to-use** `providers.tf` and `main.tf` for your S3 static website so you can just copy-paste and it will work.

Do you want me to do that?



it's in previous folder

Ah! Got it — your `main.tf` and `providers.tf` are **still in the previous folder** (`DevOpsZaid`), not in `terraform-s3` where you're currently running Terraform.

That's why `terraform init` said:

Terraform initialized in an empty directory!



You have **two options**:

Option 1: Move the Terraform files into the folder you created

From your current folder (`terraform-s3`), go **one folder up** and move them:

bash

Copy code

```
cd ..  
mv main.tf terraform-s3/  
mv providers.tf terraform-s3/
```

Then go back to `terraform-s3` and run:

Copy code

```
bash
```

```
cd terraform-s3
terraform init
terraform validate
terraform plan
terraform apply
```

Option 2: Run Terraform from the folder where files currently are

If you want, you can just **stay in the previous folder** where `main.tf` and `providers.tf` are and run:

```
bash
```

 Copy code

```
terraform init
terraform validate
terraform plan
terraform apply
```

 My recommendation: **Option 1** — keep all Terraform files in `terraform-s3`. This keeps your project organized and separates your website code from infrastructure code.

I can also give you the **exact commands** to move your files and run Terraform safely. Do you want me to do that?



Look, this is my file structure of main folder DevOpsZaid, inside it also a DevOpsZaid named folder created; .git, .github, DevOpsZaid, terraform-s3, these are folders main.tf, providers.tf are files .gitignore is also a file,

And my website is in DevOpsZaid folder

Ah! Perfect — now I completely understand your setup. Let me summarize your **current folder structure**:

lua

 Copy code

```
DevOpsZaid/          <-- main folder (Git repo root)
|
├── .git/
├── .github/
├── DevOpsZaid/      <-- website files (home.html, etc.)
├── terraform-s3/    <-- folder you just created for Terraform tasks
├── main.tf           <-- currently in the root folder
├── providers.tf       <-- currently in the root folder
└── .gitignore
```

 The problem: `main.tf` and `providers.tf` are in the **root folder**, but you're running Terraform inside `terraform-s3`. Terraform only reads `.tf` files in the **current folder**, so it thinks the folder is empty.

How to fix

You need to **move the Terraform files into `terraform-s3`** so everything related to AWS Terraform stays in that folder.

Run these commands from the **root folder** (`DevOpsZaid`):

bash

 Copy code

```
mv main.tf terraform-s3/
mv providers.tf terraform-s3/
```

Then go into the folder:

bash

 Copy code

```
cd terraform-s3
```

Check your files:

bash

 Copy code

```
ls
```

You should now see:

css

 Copy code

```
main.tf
```

```
providers.tf
```

Now initialize Terraform:

bash

 Copy code

```
terraform init  
terraform validate  
terraform plan  
terraform apply
```

Best practice for your project:

lua

 Copy code

```
DevOpsZaid/  
|  
|__ DevOpsZaid/          <-- website files (home.html)  
|__ terraform-s3/        <-- all Terraform files here  
|   |__ main.tf  
|   |__ providers.tf  
|__ .gitignore  
└__ ... other folders/files
```

This keeps **website code and Terraform code completely separate**, making your project clean and manageable.

If you want, I can give you the full ready-to-use `main.tf` and `providers.tf` in this folder so you can immediately create the S3 static website.

Do you want me to do that?



so those files should be inside `terraform-s3`, or can i keep them here, because I have typed `terraform init`, in main folder, and it created a file, `.terraform`

Ah! Now it's clear. Let me explain carefully:

1 Terraform behavior

- Terraform **only looks at `.tf` files in the folder you run it from.**
 - When you ran `terraform init` in the **main folder**, Terraform created the `.terraform` folder **there**. That's fine, but:
 - Terraform will now try to manage whatever `.tf` files exist in that folder.
 - If your intention is for this folder to only have Terraform files for your S3 website, it's **mixing things** with your Git repo root, which can get messy.
-

2 Options you have

Option A: Keep files in the main folder

- Pros: Already initialized, `.terraform` is there.
- Cons: Your website files and Terraform files are in the same folder. Future Terraform state files (`terraform.tfstate`) will appear in the root — messy. Not ideal for bigger projects.

Option B: Move `.tf` files into `terraform-s3`

- Pros: Clean separation of website code vs Terraform code.
- Recommended for professional practice, easier management.
- You would need to:

bash

 Copy code

```
mv main.tf terraform-s3/  
mv providers.tf terraform-s3/  
cd terraform-s3  
terraform init  
terraform plan  
terraform apply
```

- This creates a **new .terraform folder inside terraform-s3** (clean, isolated).
- Your root folder stays clean, only website code lives there.

Recommendation

Go with **Option B** — move the files to `terraform-s3`. It's cleaner, and you won't accidentally mix Terraform state with your website code.

If you want, I can give you the **exact commands to move everything, reinitialize Terraform in `terraform-s3`, and apply your S3 website** so it works perfectly.

Do you want me to do that?



so you mean i move `main.tf`,`providers.tf` file inside `terraform-s3`? will it cause issue?

Exactly — yes, **move `main.tf` and `providers.tf` into `terraform-s3`**. 

No, it **will not cause any issue** as long as:

1. You move the files **before running** `terraform apply`.
 2. You run Terraform **inside** `terraform-s3` after moving them.
-

Step-by-step safe way:

From your **root folder** (`DevOpsZaid`):

bash

 Copy code

```
mv main.tf terraform-s3/
mv providers.tf terraform-s3/
cd terraform-s3
terraform init
terraform validate
terraform plan
terraform apply
```

- `mv` just moves the files — it **does not delete content**.
 - Terraform will now see the `.tf` files in the folder where you want them.
 - Any state files (`terraform.tfstate`) will also stay in this folder, keeping your root folder clean.
-

 **Tip:** Your `.terraform` folder in the root is now irrelevant. Once you move the files and re-run `terraform init` in `terraform-s3`, a new `.terraform` folder will be created there.

If you want, I can also **give the exact ready-to-use `providers.tf` and `main.tf`** for your S3 static website so you can just move them and run `terraform apply` successfully.

Do you want me to do that?



i have deleted the `.terraform` file, and then moved `main.tf`, `providers.tf` into `terraform-s3`, but when came back to

main folder, there existed a file with name
".terraform.lock.hcl"

What should I do with it?
i will type terraform init command again

Ah! Perfect — now you're back on track. Let me explain clearly about

.terraform.lock.hcl :

1 What is .terraform.lock.hcl ?

- This file is **automatically generated by Terraform** to lock the versions of providers you're using.
 - It ensures that Terraform always uses the same provider versions across `init`, `plan`, and `apply`.
 - It doesn't contain sensitive info, but it is specific to the folder where it was generated.
-

2 What to do now

Since you moved your `.tf` files to `terraform-s3`, the old `.terraform.lock.hcl` in the root folder is no longer needed.

You have two choices:

1. Delete it — safe, won't break anything:

bash

 Copy code

```
rm .terraform.lock.hcl
```

2. Or leave it — harmless, but it won't be used because your Terraform files are now in `terraform-s3`.
-

3 Next step

Now go into your Terraform folder:

```
bash
```

 Copy code

```
cd terraform-s3
```

Run:

```
bash
```

 Copy code

```
terraform init
```

- This will recreate `.terraform` and `.terraform.lock.hcl` inside `terraform-s3`.
 - Terraform will now manage your S3 bucket from the correct folder.
-

Summary:

- Root folder: `.terraform.lock.hcl` — **can delete**.
 - Terraform folder: `terraform-s3` — run `terraform init` here, `.terraform.lock.hcl` will be created automatically.
-

If you want, I can write the **exact commands you should type now from start to finish** to safely initialize and deploy your S3 static website in the new folder.

Do you want me to do that?



```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid/terraform-s3 (main)
$ terraform init
Terraform initialized in an empty directory!
```

The directory has no Terraform configuration files. You may begin working with Terraform immediately by creating Terraform configuration files.

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid/terraform-s3
$ cd ..
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/DevOpsZaid (main)
$ cd ..
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
(main)
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.27.0...
- Installed hashicorp/aws v6.27.0 (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.

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th  
Semester/DevOps - Lab/Semester Project/DevOpsZaid  
(main)  
$ terraform init  
Terraform initialized in an empty directory!
```

The directory has no Terraform configuration files. You may begin working with Terraform immediately by creating Terraform configuration files.

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th  
Semester/DevOps - Lab/Semester Project/DevOpsZaid  
(main)  
$
```

Ah! I see exactly what's happening here — it's a **folder confusion issue**. Let me explain carefully.

1 What just happened

- You ran `terraform init` **inside** `terraform-s3`, but it said:

nginx

 Copy code

Terraform initialized in an empty directory!

The directory has **no** Terraform configuration files.

 This means `main.tf` **and** `providers.tf` **are not actually in** `terraform-s3`, even though you thought you moved them.

- Then you went **back to the parent folder** (`DevOpsZaid`) and ran `terraform init` there. Now Terraform sees the `.tf` files and initializes successfully.
-

2 Why it happened

- You likely ran `mv main.tf terraform-s3/` and `mv providers.tf terraform-s3/` from the wrong folder, or the command didn't execute correctly.
 - So `terraform-s3` is still empty, which is why Terraform says "empty directory".
-

3 How to fix (step-by-step, safely)

1. Check where the files actually are:

bash

 Copy code

```
cd /d/4th Semester/DevOps - Lab/Semester Project/DevOpsZaid/DevOpsZaid  
ls
```

You should see something like:

css

 Copy code

```
main.tf  
providers.tf  
DevOpsZaid/  
terraform-s3/  
.gitignore
```

2. Move them into `terraform-s3` (run from this folder, the root of your project):

bash

 Copy code

```
mv main.tf terraform-s3/  
mv providers.tf terraform-s3/
```

3. Go into the Terraform folder:

bash

 Copy code

```
cd terraform-s3  
ls
```

You should now see:

css

 Copy code

main.tf
providers.tf

4. Initialize Terraform here:

bash

Copy code

terraform init

- This will create .terraform and .terraform.lock.hcl **inside** terraform-s3 , cleanly separated from your website code.

5. Then run:

bash

Copy code

terraform validate
terraform plan
terraform apply

Key points

- Always run Terraform **from the folder where your .tf files are located**.
 - The .terraform folder and .terraform.lock.hcl should live **in the same folder as the .tf files**.
 - Keep Terraform files **separate from your website folder** (DevOpsZaid) to avoid clutter.
-

If you want, I can give you a **ready-to-copy command set** that will safely move your files, initialize Terraform, and deploy your S3 static site without errors.

Do you want me to do that?



tell me inshort, what i do now

In short:

1. Go to your main project folder (where `main.tf` and `providers.tf` currently are):

bash

 Copy code

```
cd /d/4th Semester/DevOps - Lab/Semester Project/DevOpsZaid/DevOpsZaid
```

2. Move Terraform files into `terraform-s3` :

bash

 Copy code

```
mv main.tf terraform-s3/
```

```
mv providers.tf terraform-s3/
```

3. Go into `terraform-s3` :

bash

 Copy code

```
cd terraform-s3
```

4. Initialize Terraform here:

bash

 Copy code

```
terraform init
```

```
terraform validate
```

```
terraform plan
```

```
terraform apply
```

 After this, your S3 static website will be managed from the `terraform-s3` folder, keeping everything clean.

No need to touch `.terraform.lock.hcl` in the root — it's fine to delete it.



cmd commands to get all files, folder inside a folder, I am in this folder;

D:\4th Semester\DevOps - Lab\Semester Project\DevOpsZaid\DevOpsZaid

it has my folders and files, cmd command to open them

Ah! You just want to **see all files and folders** in the current directory from CMD or Git Bash. Here's the simple way:

1 In Windows CMD:

cmd

 Copy code

dir

- Shows all files and folders in the current folder.
- To see **everything including hidden files**:

cmd

 Copy code

dir /a

2 In Git Bash / MINGW64 (looks like your terminal):

bash

 Copy code

ls

- Shows files and folders.
- To see **everything including hidden files**:

bash

 Copy code

ls -a

- To see a **detailed list** with sizes, dates:

bash

 Copy code

```
ls -la
```

3 To open a file or folder from CMD / Git Bash

- Open a **folder** in File Explorer:

```
cmd
```

 Copy code

```
start .
```

- Open a **file** in default program (e.g., Notepad for .txt or .tf):

```
cmd
```

 Copy code

```
notepad filename.tf
```

Replace `filename.tf` with your file name.

If you want, I can give you a **single command to open all files and folders in File Explorer from your current folder**, so you can click and see everything easily.

Do you want me to do that?



Alright AD — take a breath 😊

I hear you **clearly**: you don't want *assumptions*, you don't want *guessing*, and you **definitely don't want that red ✗ GitHub Actions trauma again**.
So this time we'll do it **slow, boring, safe, and 100% controlled**.

I will tell you **exactly**:

* **WHAT to do**

* **WHERE to do it** (local PC, AWS Console, GitHub)

* **WHAT you should see**

* **WHY we are doing it**

No skipping. No “you should already know”.

We will **deploy your static website to AWS S3 using Terraform**, **without GitHub Actions first**.

👉 CI/CD comes **after** things work.

🧠 IMPORTANT RESET (PLEASE READ)

Your **previous failures** happened because:

- * Terraform + GitHub Actions + S3 were mixed **at once**
- * Secrets, permissions, paths were wrong
- * Debugging all together is painful

✅ This time strategy:

1. **Terraform locally** (NO GitHub Actions)
2. Make S3 website work
3. Only then (optional) automate

🏗️ WHAT WE ARE BUILDING

Your PC

↓ Terraform

AWS S3 Bucket

↓

Static Website (HTML/CSS/JS)

NO Docker

NO GitHub Actions

NO CI/CD

👉 First make **AWS deployment work**

STEP 0: WHAT YOU MUST HAVE (CHECK ONLY)

On your LOCAL MACHINE

Open **Git Bash** and run:

```
bash  
terraform -version  
aws --version
```

If **either command fails**, STOP and tell me.

STEP 1: CREATE AWS IAM USER (AWS CONSOLE)

 **WHERE:** Browser → AWS Console

 **WHY:** Terraform needs permission to create S3

1. Go to:

 [https://console.aws.amazon.com/]
(https://console.aws.amazon.com/)

2. Search → **IAM**

3. Click **Users** → **Create user**

* User name:

terraform-user

4. Click **Next**

5. Permissions → **Attach policies directly**

 Select:

AmazonS3FullAccess

(ONLY S3 for now — safe & simple)

6. Create user

 STEP 1.1: CREATE ACCESS KEYS

- * Click the user **terraform-user**
- * Go to **Security credentials**
- * Scroll → **Access keys**
- * Click **Create access key**
- * Select:

Command Line Interface (CLI)

- * Create

 **COPY & SAVE**:

- * Access Key ID
- * Secret Access Key

 If you lose them, you must recreate.

 STEP 2: CONFIGURE AWS CLI (LOCAL MACHINE)

 **WHERE:** Git Bash

Run:

```
bash
aws configure
```

Enter **exactly like this**:

```
text
AWS Access Key ID:      <PASTE ACCESS KEY>
AWS Secret Access Key: <PASTE SECRET KEY>
Default region name:   us-east-1
Default output format: json
```

Verify:

```
bash
aws s3 ls
```

- ✓ If no error → AWS access works
- ✗ If error → STOP and paste it

✓ STEP 3: PREPARE TERRAFORM FOLDER (LOCAL MACHINE)

💡 **WHERE:** Your project root (NOT website folder)

Go to:

```
bash
cd "/d/4th Semester/DevOps - Lab/Semester Projec
```

Create new folder:

```
bash
mkdir terraform-s3
```

```
cd terraform-s3
```

Check:

```
bash  
pwd
```

You should be inside:

```
terraform-s3
```

```
#  STEP 4: CREATE TERRAFORM FILES (NO GUESSING)
```

```
## 4.1 Create provider.tf
```

```
bash  
notepad provider.tf
```

Paste:

```
hcl  
provider "aws" {  
    region = "us-east-1"  
}
```

Save → Close

```
## 4.2 Create main.tf
```

```
bash  
notepad main.tf
```

Paste **THIS EXACT CODE**:

```
hcl
resource "aws_s3_bucket" "website_bucket" {
    bucket = "devopszaid-static-site-12345"
}

resource "aws_s3_bucket_public_access_block" "public_access_block" {
    bucket = aws_s3_bucket.website_bucket.id

    block_public_acls      = false
    block_public_policy     = false
    ignore_public_acls     = false
    restrict_public_buckets = false
}

resource "aws_s3_bucket_website_configuration" "website_configuration" {
    bucket = aws_s3_bucket.website_bucket.id

    index_document {
        suffix = "home.html"
    }
}

resource "aws_s3_bucket_policy" "public_policy" {
    bucket = aws_s3_bucket.website_bucket.id

    policy = jsonencode({
        Version = "2012-10-17"
        Statement = [
            {
                Sid          = "PublicReadGetObject"
                Effect       = "Allow"
                Principal    = "*"
                Action       = "s3:GetObject"
                Resource    = "${aws_s3_bucket.website_bucket.id}/*"
            }
        ]
    })
}
```

```
})  
}
```

⚠ Change bucket name to something **globally unique**
Example:

devopszaid-static-site-ad-2026

STEP 5: INITIALIZE TERRAFORM

💡 **WHERE:** terraform-s3 folder

```
bash  
terraform init
```

Expected:

Terraform has been successfully initialized!

✗ If error → paste it

STEP 6: APPLY TERRAFORM (CREATE S3)

```
bash  
terraform apply
```

Type:

```
text  
yes
```

Expected:

Apply complete!

 STEP 7: UPLOAD WEBSITE FILES (MANUAL, SAFE)

 **WHERE:** AWS Console → S3

1. Open **S3**
2. Click your bucket
3. Click **Upload**
4. Upload:

- * home.html
- * about.html
- * services.html
- * products.html
- * contact.html
- * style.css
- * script.js
- * images/

 Upload **contents**, not the folder itself

 STEP 8: OPEN WEBSITE 

 **WHERE:** AWS Console

- * Go to **S3 → your bucket**
- * Open **Properties**
- * Scroll → **Static website hosting**
- * Copy **Bucket website endpoint**

Open in browser 

🧠 WHY YOUR PREVIOUS GITHUB ACTION FAILED

From your screenshot:

- * Terraform ran in GitHub Actions
- * AWS credentials were missing / wrong
- * S3 upload step failed
- * No local testing = blind debugging

 **Now we fixed the foundation**

➡️ SOON NEXT (ONLY IF YOU WANT)

After this works perfectly, we can:

1. Add **Terraform S3 upload automation**
2. Add **GitHub Actions safely**
3. Add **CloudFront (HTTPS)**
4. Add **Docker later**

🔔 IMPORTANT QUESTION (ANSWER THIS)

 **Did your S3 website open successfully in browser?**

Reply with:

- * **"YES, website opened"**
- or
- * Paste **exact error you see**

We won't move forward until this is 100% green 

in the 7th step, i have all these files in my folder, which files & folders should i upload now?

Perfect, AD 😊 — now we are at **Step 7**, uploading your website files to the S3 bucket. Let's do this precisely.

What you upload to S3

You should upload **all your website files** that are needed to render the site correctly. Based on your folder, this usually includes:

1. HTML files

- `home.html` ← your **index page**
- Any other pages: `about.html` , `services.html` , `contact.html` , etc.

2. CSS files

- `style.css` or any other CSS files

3. JS files

- `script.js` or any other JavaScript files your site uses

4. Images / assets

- `images/` folder (or `img/` if that's your folder)
 - Any other folders like `fonts/` , `videos/` if your site uses them
-

Important notes for S3

- **Upload contents, not the folder itself**
 - For example: inside `images/` , select all images and upload, **don't upload the parent images folder as a zip**
 - **Keep folder structure same as your HTML expects**
 - If `home.html` links to `images/logo.png` , you need `images/logo.png` in S3 as well
 - **Set files as public** (or bucket policy already makes them public)
-

Quick checklist for upload

If your folder looks like this:

arduino

 Copy code

```
home.html  
about.html  
contact.html  
style.css  
script.js  
images/
```

Then upload everything:

- home.html
 - about.html
 - contact.html
 - style.css
 - script.js
 - **All contents of images/ (maintain folder structure)**
-

Once uploaded, you can go to **Properties** → **Static website hosting** → **Endpoint** and open your website.

If you want, I can also write a **step-by-step “exact S3 upload plan”** so you don’t miss a single file or folder. This is what I usually give students to make the site work **first try**.

Do you want me to do that?



tell me just their names in short

Got it AD 😊 — in short, upload these **files/folders**:

arduino

 Copy code

```
home.html  
about.html  
services.html  
products.html  
contact.html  
style.css  
script.js  
images/ (all images inside)
```

 That's all you need to make your static website work.



What if I upload images folder?

Perfect question, AD 😎

If you **upload the images/ folder**, that's actually **exactly what you should do**, as long as:

1. Your HTML/CSS references the images correctly

- Example: ``
- Then S3 must have `images/logo.png` (so uploading the whole folder preserves the paths)

2. You don't zip the folder

- Upload the folder itself, S3 will keep the structure
- If you zip it and upload, your site won't find the images

 **Result:** Your HTML will correctly load all images if the folder structure matches.

So yes — **upload the images/ folder as it is**, it's safe and correct.

If you want, I can tell you a **super-safe way to check all links/images will load in S3 before opening the website**. Do you want me to do that?



After step 7, can i click upload, or just leave after uploading into s3 bucket?

After **Step 7**, yes — you **must click the Upload button in the S3 console** to actually put the files into the bucket.

Here's exactly what happens:

1. **Select files/folders** → `home.html`, other HTML pages, `style.css`, `script.js`, images/ folder
2. **Click "Upload"** → it will open the upload wizard
3. **Next/Next/Upload** → S3 uploads the files to the bucket

✖ Just dragging or selecting in the console **does nothing until you click Upload**.

✓ After upload completes, your files are in S3 and the bucket can serve your website.

Once uploaded, you can go to **Properties** → **Static website hosting** → **Bucket website endpoint** to open your website in the browser.

Do you want me to tell the **exact safe settings for the upload wizard** so nothing breaks?



Alright AD — take a breath 😊

I hear you **clearly**: you don't want ***assumptions***, you don't want ***guessing***, and you **definitely don't want that red ✖ GitHub Actions trauma again****.

So this time we'll do it **slow, boring, safe, and 100% controlled****.

I will tell you **exactly****:

- * **WHAT to do**
- * **WHERE to do it** (local PC, AWS Console, GitHub)
- * **WHAT you should see**
- * **WHY we are doing it**

No skipping. No “you should already know”.

We will **deploy your static website to AWS S3 using Terraform**, **without GitHub Actions first**.

👉 CI/CD comes **after** things work.

💡 IMPORTANT RESET (PLEASE READ)

Your **previous failures** happened because:

- * Terraform + GitHub Actions + S3 were mixed **at once**
- * Secrets, permissions, paths were wrong
- * Debugging all together is painful

✅ This time strategy:

1. **Terraform locally** (NO GitHub Actions)
2. Make S3 website work
3. Only then (optional) automate

🏗️ WHAT WE ARE BUILDING

Your PC

↓ Terraform

AWS S3 Bucket

↓

Static Website (HTML/CSS/JS)

NO Docker

NO GitHub Actions

NO CI/CD

👉 First make **AWS deployment work**

 STEP 0: WHAT YOU MUST HAVE (CHECK ONLY)

On your LOCAL MACHINE

Open **Git Bash** and run:

```
bash  
terraform -version  
aws --version
```

If **either command fails**, STOP and tell me.

 STEP 1: CREATE AWS IAM USER (AWS CONSOLE)

🔑 **WHERE:** Browser → AWS Console

🔑 **WHY:** Terraform needs permission to create S3

1. Go to:

👉 [https://console.aws.amazon.com/]
(https://console.aws.amazon.com/)

2. Search → **IAM**

3. Click **Users** → **Create user**

* User name:

terraform-user

4. Click **Next**

5. Permissions → **Attach policies directly**

Select:

AmazonS3FullAccess

(ONLY S3 for now — safe & simple)

6. Create user

 STEP 1.1: CREATE ACCESS KEYS

- * Click the user terraform-user
- * Go to **Security credentials**
- * Scroll → **Access keys**
- * Click **Create access key**
- * Select:

Command Line Interface (CLI)

- * Create

 **COPY & SAVE**:

- * Access Key ID
- * Secret Access Key

 If you lose them, you must recreate.

STEP 2: CONFIGURE AWS CLI (LOCAL MACHINE)

 **WHERE:** Git Bash

Run:

```
bash
aws configure
```

Enter **exactly like this**:

```
text
AWS Access Key ID:      <PASTE ACCESS KEY>
AWS Secret Access Key: <PASTE SECRET KEY>
Default region name:   us-east-1
Default output format: json
```

Verify:

```
bash
aws s3 ls
```

- If no error → AWS access works
- If error → STOP and paste it

```
#  STEP 3: PREPARE TERRAFORM FOLDER (LOCAL
MACHINE)
```

 **WHERE:** Your project root (NOT website folder)

Go to:

```
bash
cd "/d/4th Semester/DevOps - Lab/Semester Project"
```

Create new folder:

```
bash
mkdir terraform-s3
cd terraform-s3
```

Check:

```
bash
pwd
```

You should be inside:

```
terraform-s3
```

 STEP 4: CREATE TERRAFORM FILES (NO GUESSING)

4.1 Create provider.tf

```
bash
notepad provider.tf
```

Paste:

```
hcl
provider "aws" {
    region = "us-east-1"
}
```

Save → Close

4.2 Create main.tf

```
bash  
notepad main.tf
```

Paste **THIS EXACT CODE**:

```
hcl  
resource "aws_s3_bucket" "website_bucket" {  
    bucket = "devopszaid-static-site-12345"  
}  
  
resource "aws_s3_bucket_public_access_block" "pbl"  
bucket = aws_s3_bucket.website_bucket.id  
  
    block_public_acls    = false  
    block_public_policy = false  
    ignore_public_acls  = false  
    restrict_public_buckets = false  
}  
  
resource "aws_s3_bucket_website_configuration" "  
bucket = aws_s3_bucket.website_bucket.id  
  
    index_document {  
        suffix = "home.html"  
    }  
}  
  
resource "aws_s3_bucket_policy" "public_policy"  
bucket = aws_s3_bucket.website_bucket.id  
  
policy = jsonencode({  
    Version = "2012-10-17"  
    Statement = [{  
        Sid      = "PublicReadGetObject"  
        Effect   = "Allow"  
        Principal = "*"  
        Action   = "s3:GetObject"  
        Resource  = "${aws_s3_bucket.website_bucket.id}/*"  
    }]
```

```
})  
}
```

⚠ Change bucket name to something **globally unique**
Example:

```
devopszaid-static-site-ad-2026
```

STEP 5: INITIALIZE TERRAFORM

💡 **WHERE:** terraform-s3 folder

```
bash  
terraform init
```

Expected:

Terraform has been successfully initialized!

✗ If error → paste it

STEP 6: APPLY TERRAFORM (CREATE S3)

```
bash  
terraform apply
```

Type:

text
yes

Expected:

Apply complete!

STEP 7: UPLOAD WEBSITE FILES (MANUAL, SAFE)

WHERE: AWS Console → S3

1. Open **S3**
2. Click your bucket
3. Click **Upload**
4. Upload:

- * home.html
- * about.html
- * services.html
- * products.html
- * contact.html
- * style.css
- * script.js
- * images/

Upload **contents**, not the folder itself

STEP 8: OPEN WEBSITE

WHERE: AWS Console

- * Go to **S3 → your bucket**
- * Open **Properties**
- * Scroll → **Static website hosting**
- * Copy **Bucket website endpoint**

Open in browser 

🧠 WHY YOUR PREVIOUS GITHUB ACTION FAILED

From your screenshot:

- * Terraform ran in GitHub Actions
- * AWS credentials were missing / wrong
- * S3 upload step failed
- * No local testing = blind debugging

 **Now we fixed the foundation**

➡️ SOON NEXT (ONLY IF YOU WANT)

After this works perfectly, we can:

1. Add **Terraform S3 upload automation**
2. Add **GitHub Actions safely**
3. Add **CloudFront (HTTPS)**
4. Add **Docker later**

🔔 IMPORTANT QUESTION (ANSWER THIS)

 **Did your S3 website open successfully in browser?**

Reply with:

- * **"YES, website opened"**
- or
- * Paste **exact error you see**

We won't move forward until this is 100% green 

I have did all of these, now Tell me what will I do, will I do these tasks for automation of my website from github, like i push on github, it does everything automatically.
"NEXT (ONLY IF YOU WANT)

After this works perfectly, we can:

Add Terraform S3 upload automation

Add GitHub Actions safely

Add CloudFront (HTTPS)

Add Docker later"

if yes, then write shortly yes, and then guide me step by step as you did guide me for all these tasks in simple step by step, telling me each and everything as I am dumb.

{"I gave you all above prompt before, but writing these bracket prompt now after editing, to tell you that I already had a website which I pushed on github, and docker, now will i need to create new repository or previous one as my home.html,etc are in DevOpsZaid folder below .github/workflows. Following is my github repo, i couldn't paste screenshot, that's why pasted the text;

Skip to content

Navigation Menu

AlyanShahbaz2022

DevOpsZaid

Type / to search

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights
Settings
Owner avatar
DevOpsZaid
Public
AlyanShahbaz2022/DevOpsZaid
Go to file
t
Name
233023-git
233023-git
Fix upload Website to S3 step in Terraform workflow
512fcd4
. .
2 weeks ago
.github/workflows
Fix upload Website to S3 step in Terraform workflow
2 weeks ago
DevOpsZaid
my first commit
3 weeks ago
.gitignore
my first commit
3 weeks ago
main.tf
my first commit
3 weeks ago
providers.tf
my first commit
3 weeks ago
Repository files navigation
README
Add a README
Help people interested in this repository understand your project.

About
DevOps Semester Project Zaid

Activity

Stars

0 stars

Watchers

0 watching

Forks

0 forks

Report repository

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors

3

@233023-git

233023-git

@Hanzai003

Hanzai003

@AlyanShahbaz2022

AlyanShahbaz2022

Languages

HCL

100.0%

Footer

© 2026 GitHub, Inc.

Footer navigation

Terms

Privacy

Security

Status

Community

Docs

Contact

Manage cookies

Do not share my personal information

Chat

New Conversation

 Explain a complex thing

Explain Artificial Intelligence so that I can explain it to my six-year-old child.

 Get suggestions and create new ideas

Please give me the best 10 travel ideas around the world

 Translate, summarize, fix grammar and more...

Translate "I love you" French

GPT-4o Mini

Hello, how can I help you today?

slope of perpendicular lines

GPT-4o Mini

The slopes of perpendicular lines have a special relationship in the context of coordinate geometry. If two lines are perpendicular, the product of their slopes is equal to -1.

If you have a line with a slope (m_1), then the slope (m_2) of a line that is perpendicular to it can be found using the following formula:

$$\begin{bmatrix} m_1 \cdot m_2 = -1 \\ \end{bmatrix}$$

This can also be rewritten to find (m_2):

$$\begin{bmatrix} m_2 = -\frac{1}{m_1} \\ \end{bmatrix}$$

For example:

If one line has a slope of 3, then the slope of a line that is perpendicular to it would be ($-\frac{1}{3}$).

Conversely, if one line has a slope of -2, then the slope of any line that is perpendicular to it would be ($\frac{1}{2}$).

This relationship holds true as long as neither line is vertical (undefined slope) or horizontal (zero slope), since vertical and horizontal lines are always mutually perpendicular but do not have defined numerical slopes in this context.

AITOPIA
coin image
10
Upgrade

Name
Latest commit

233023-git
Fix upload Website to S3 step in Terraform workflow
512fcd4

History
7 Commits
.github/workflows
Fix upload Website to S3 step in Terraform workflow
DevOpsZaid

Make a Review & Earn Credit ❤️
Chat
Ask

```
Search
Write
Image
ChatFile
Vision
Agent
Full Page
Invite & Earn
"}
now guide me step by step"}
```

Alright AD — I hear you loud and clear 😎

Yes — we can now automate your **website deployment from GitHub to AWS S3** using **Terraform + GitHub Actions**. We'll go **step by step**, slowly, safely, and I'll tell you **exactly what to do, where, and why**.

No skipping. No assumptions. Let's go.

STEP 0: PRE-REQUISITES (CHECK)

Before automation, make sure:

1. Your **S3 bucket works manually** (Step 7 and 8 done).
2. Your **Terraform code works locally** (`terraform apply` runs and creates S3).
3. Your **AWS CLI credentials** exist and have `AmazonS3FullAccess`.
4. Your **website files are in GitHub repo** (`home.html`, `style.css`, `script.js`, `images/` all committed).

If all above is , we move forward.

STEP 1: Decide GitHub Repo

You do **NOT** need a new repository.

Use your **existing repo** (`DevOpsZaid`) because:

- `.github/workflows/` already exists

- Your `home.html` etc. are already committed

We will add one workflow YAML to automate Terraform & S3 deployment.

STEP 2: Create GitHub Secrets

We never put AWS keys in code. Instead, store them as **GitHub Secrets**.

💡 WHERE: GitHub → DevOpsZaid → Settings → Secrets and variables → Actions → New repository secret

Create these secrets:

Secret Name	Value
AWS_ACCESS_KEY_ID	(from your Terraform IAM user)
AWS_SECRET_ACCESS_KEY	(from your Terraform IAM user)
AWS_REGION	us-east-1 (or your region)
BUCKET_NAME	devopszaid-static-site-ad-2026 (your bucket)

These will be used inside the GitHub Actions workflow, never hardcoded.

STEP 3: Create GitHub Actions Workflow

1. Go to your repo: `.github/workflows/`
2. Create a new file (name it e.g.):

 Copy code

`terraform-s3-deploy.yml`

3. Paste the following workflow (I explain each step):

`yaml`

 Copy code

`name: Terraform S3 Deploy`

```
on:
  push:
    branches:
      - main    # trigger on push to main branch

jobs:
  deploy:
    name: Deploy Website
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repo
        uses: actions/checkout@v3

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2
        with:
          terraform_version: 1.7.7    # use your installed version

      - name: Terraform Init
        run: terraform init
        working-directory: terraform-s3

      - name: Terraform Apply
        run: terraform apply -auto-approve
        working-directory: terraform-s3
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          AWS_REGION: ${{ secrets.AWS_REGION }}

      - name: Upload Website Files to S3
        run: |
          aws s3 sync DevOpsZaid/ s3://${{ secrets.BUCKET_NAME }} --delete
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          AWS_REGION: ${{ secrets.AWS_REGION }}
```

STEP 4: Commit & Push Workflow

bash

 Copy code

```
git add .github/workflows/terraform-s3-deploy.yml  
git commit -m "Add GitHub Actions workflow to deploy website"  
git push origin main
```

 This will trigger the workflow **automatically**.

STEP 5: What happens when workflow runs

1. GitHub checks out your code
 2. Sets up Terraform
 3. Runs `terraform init` in `terraform-s3`
 4. Runs `terraform apply` (creates S3 bucket if needed)
 5. Syncs your **website files** from `DevOpsZaid/ folder` into the S3 bucket
 6. Website is live immediately after
-

STEP 6: Verify

1. Go to **Actions tab** in GitHub → check workflow run → must show 
 2. Go to **AWS S3** → **bucket** → **Properties** → **Static website hosting** → **endpoint**
 3. Open URL → website should reflect **latest commit**
-

STEP 7: Notes / Tips

- Workflow is safe: uses **secrets**, no hardcoded AWS keys
 - If you add new pages/images → push → workflow auto-updates S3
 - You **don't need a new repository**, your existing `DevOpsZaid` repo is enough
 - **Folder structure matters**: files in `DevOpsZaid/` will mirror in S3
-

AD  — this is the **exact step-by-step setup** for automation.

Next, I can give you a **super simplified command + workflow version** that will:

- Automatically deploy S3
- Upload website
- Do **everything with one commit**

Do you want me to do that?



When i enter
url"https://github.com/AlyanShahbaz2022/DevOpsZaid/se
ttings/secrets/actions"

It shows fucking following error, i typed url directly because
clicking settings gives me 404 error on my system, but
when i click on my collaborator's system, it's not causing
errors. and he's not with me now.

"

500 Error

Looks like something went wrong!

We track these errors automatically, but if the problem
persists feel free to contact us. In the meantime, try
refreshing.

Contact Support — GitHub Status — @githubstatus

Chat

New Conversation



Explain a complex thing

Explain Artificial Intelligence so that I can explain it to my
six-year-old child.

💡 Get suggestions and create new ideas

Please give me the best 10 travel ideas around the world

💡 Translate, summarize, fix grammar and more...

Translate "I love you" French

GPT-4o Mini

Hello, how can I help you today?

slope of perpendicular lines

GPT-4o Mini

The slopes of perpendicular lines have a special relationship in the context of coordinate geometry. If two lines are perpendicular, the product of their slopes is equal to -1.

If you have a line with a slope (m_1), then the slope (m_2) of a line that is perpendicular to it can be found using the following formula:

$$\begin{bmatrix} m_1 \cdot m_2 = -1 \\ \end{bmatrix}$$

This can also be rewritten to find (m_2):

$$\begin{bmatrix} m_2 = -\frac{1}{m_1} \\ \end{bmatrix}$$

For example:

If one line has a slope of 3, then the slope of a line that is perpendicular to it would be ($-\frac{1}{3}$).

Conversely, if one line has a slope of -2, then the slope of any line that is perpendicular to it would be ($\frac{1}{2}$).

This relationship holds true as long as neither line is vertical (undefined slope) or horizontal (zero slope), since vertical and horizontal lines are always mutually

perpendicular but do not have defined numerical slopes in this context.

AITOPIA
coin image
10
Upgrade

Ask me anything...

Make a Review & Earn Credit ❤️

Chat
Ask
Search
Write
Image
ChatFile
Vision
Agent
Full Page
Invite & Earn
"

Ah AD 😅 — I see exactly what's happening.

That **500 error** is **not your fault** — it's a GitHub internal server error. It sometimes happens if:

- GitHub's servers are having a temporary issue
- Your browser session is acting weird (cookies / cache / login state)
- The repo is public or you don't have full admin access in that session

Here's how to fix it **so you can add secrets**:

Step 1: Make sure you are logged in as the repo owner/admin

- You **must have Admin access** to DevOpsZaid to see Secrets → Actions
 - If you are a collaborator but **not admin**, you cannot see/add secrets
-

Step 2: Try a different browser / incognito mode

- Open **Chrome Incognito** (or any private window)
- Go directly to:

bash

 Copy code

<https://github.com/AlyanShahbaz2022/DevOpsZaid/settings/secrets/actions>

Step 3: Clear cache / cookies

- Sometimes old sessions cause 500 errors
 - Clear cache → log in again → try the URL
-

Step 4: Wait / retry

- GitHub sometimes has **temporary outages**
 - Check status: <https://www.githubstatus.com/>
 - Retry in a few minutes
-

Step 5: If all else fails

- Ask your collaborator (who can access it) to **add the secrets** for now:

nginx

 Copy code

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY

AWS_REGION

BUCKET_NAME

Once secrets exist in repo, **your workflow will run automatically** even if you cannot add them right now.

AD, the bottom line: **500 error = GitHub issue, not your setup.**

You can either:

1. Retry with incognito or clear cache
 2. Or wait / ask collaborator to add secrets
-

If you want, I can **write the exact commands + workflow snippet** you can push **immediately**, so the repo will be ready to deploy as soon as the secrets exist — no manual editing needed.

Do you want me to do that?



i am a collaborator

Ah — that explains everything, AD 😊

Since you're a **collaborator**, you do NOT have admin access to the repo.

- That's why you **get 404 / 500 errors** when trying to access:

bash

Copy code

<https://github.com/AlyanShahbaz2022/DevOpsZaid/settings/secrets/actions>

- Only the **repo owner or someone with Admin rights** can add **GitHub Secrets**.
-



What you can do now:

1. Ask the **repo owner** (or whoever has Admin access) to add the following **Secrets**:

Secret Name	Value
AWS_ACCESS_KEY_ID	(from your Terraform IAM user)
AWS_SECRET_ACCESS_KEY	(from your Terraform IAM user)
AWS_REGION	us-east-1 (or your region)
BUCKET_NAME	devopszaid-static-site-ad-2026

2. Once they add them, you can push your workflow and it will run automatically.
 3. You do not need to create a new repo, your current DevOpsZaid repo is fine.
-

 Important: As a collaborator, you can still push workflow YAML files into .github/workflows/ and your workflow will run once the secrets exist.

If you want, I can write the exact workflow YAML and Git commands you can push now, so it's ready to deploy automatically as soon as the owner adds the secrets.

Do you want me to do that?



can he make me owner also?

Ah AD 😊 — yes, but there's a distinction:

1 Repo Owner vs Collaborator

- **Owner** (or Admin) = full control: can add/remove secrets, manage settings, change collaborators, delete repo, etc.
- **Collaborator** = can push code, open PRs, run workflows (if secrets exist), but cannot manage repo settings or secrets.

2 Can your collaborator make you owner?

- If the repo is under their personal account → No, they cannot transfer “owner” rights to a collaborator in that repo.
- They can do two things instead:
 1. Give you Admin access:
 - Go to Settings → Manage Access → Invite collaborator → set role = Admin
 - Admin access allows you to add secrets and manage workflows
 2. Transfer the repository to your account:
 - Then you become the new owner
 - After transfer, all collaborators remain but you control the repo

Best option for you

- Ask the current owner to make you Admin (not full owner, but enough to add secrets)
 - Once Admin → you can create GitHub Secrets → your workflow runs automatically
-

If you want, I can write a step-by-step guide your owner can follow to make you Admin so you can proceed immediately.

Do you want me to do that?



i want to be full owner, and he can make me, tell me process in easy, and ready to paste whatsapp format

Got you AD 😎 — here's an easy-to-send WhatsApp message you can copy-paste to your collaborator so they can transfer full ownership of the repo to you:

WhatsApp Message (ready to send):

Hey [Name], I want to be the full owner of our GitHub repo `DevOpsZaid`. Can you transfer it to me?

Here's how you can do it:

1. "Go to the repo → Settings → scroll down to Danger Zone → Transfer ownership."
2. "Click Transfer."
3. "Enter my GitHub username: `AD_KHAN` (replace with your GitHub username)."
4. "Type the repo name `DevOpsZaid` to confirm."
5. "Click I understand, transfer this repository."

After this, I'll become the full owner and can manage secrets, workflows, and settings.

💡 Notes to include if you want:

- All collaborators will remain in the repo after transfer.
 - After transfer, the old owner will still be a collaborator unless removed.
-

If you want, I can also make a shorter "super casual WhatsApp version" that's even faster to send 😊

Do you want me to do that?



in username i write "Hanzai003", then what?

Exactly AD 😊 — here's what happens next if you put **Hanzai003** (the current owner) as the username during transfer:

⚠ Wait — actually, for transfer, you don't put their username — you put your GitHub username (the person who will become the new owner).

So the steps are:

Correct Steps (after clicking Transfer Ownership)

1. Click Settings → Danger Zone → Transfer ownership
2. In the first field, GitHub asks:
"New owner's GitHub username or organization name"
 - Here you write your **username** (for example: AD_KHAN)
3. In the second field, GitHub asks:
"Type the repository name to confirm"
 - Here write exactly: DevOpsZaid
4. Click I understand, transfer this repository

 After that:

- You become **full owner** of DevOpsZaid
 - The current owner (Hanzai003) stays as a collaborator unless removed
 - You can now **add secrets, manage workflows, everything**
-

If you want, I can write a **ready-to-send WhatsApp message** with the correct username **filled in** for your collaborator so they can do it in **one go**.

Do you want me to do that?



after searching my username "Hanzai003" then tried
"https://github.com/Hanzai003"
but my username not showing
tell me in short and easy

Ah AD 😊 — here's the **short and easy version**:

1. For repo transfer, you need your GitHub **username** — not the owner's.

- Example: if your GitHub profile URL is <https://github.com/AD-KHAN>, then your username is AD-KHAN

2. Steps your collaborator should follow:

- Go to Repo → Settings → Danger Zone → Transfer ownership
- New owner's GitHub username: AD-KHAN (your username)
- Type the repo name to confirm: DevOpsZaid
- Click I understand, transfer this repository

 After this, you will be full owner and can manage everything.

If you want, I can make a WhatsApp-ready 1-line message your collaborator can just read and do it immediately.

Do you want me to do that?



fucking my usrnname is "Hanzai003"

Ahhh 😊 got it AD — your username is actually Hanzai003 . Perfect, that makes it simple.

Here's the short, easy, ready-to-send WhatsApp message you can give to your collaborator to transfer the repo to you:

WhatsApp Message:

Hey [Name], can you transfer the GitHub repo DevOpsZaid to me so I can be the full owner?

Steps:

1. "Go to the repo → Settings → Danger Zone → Transfer ownership"
2. "In New owner's GitHub username, type: Hanzai003 "
3. "In Type the repository name to confirm, type: DevOpsZaid "

4. “Click I understand, transfer this repository”

After this, I'll be the full owner and can manage secrets, workflows, etc.

- That's it. Nothing else needed.

If you want, I can also give a **super-short one-line WhatsApp version** that's even faster to send 😊.

Do you want me to do that?



he did, but still I can't access settings

Ah AD 😊 — I see what's happening. Don't worry, this is normal.

Even if the repo was **transferred to you**, there's a small chance:

1. GitHub needs you to accept the transfer.

- Check your email or GitHub notifications — there should be a **repo transfer confirmation**.
- You must **accept it**, otherwise you are not yet the full owner.

2. If you **already accepted** and still can't access settings:

- Make sure you are **logged into the same account** (Hanzai003) that now owns the repo.
- Sometimes you may still be viewing GitHub in a cached session; try **log out → log back in**.
- Try **Incognito / Private Window** and open:

bash

Copy code

<https://github.com/Hanzai003/DevOpsZaid/settings/secrets/actions>

3. If it still doesn't work → GitHub might be slow to update ownership.

- Give it a few minutes and refresh.

💡 Quick test:

- Go to <https://github.com/Hanzai003/DevOpsZaid>
 - Do you see “**Settings**” tab on top menu of the repo?
 - If yes → you are full owner
 - If no → you either haven’t accepted transfer OR still logged into another account
-

If you want, I can give a **step-by-step checklist right now** to make sure you get **full owner access immediately**, including fixing session/login issues.

Do you want me to do that?



Where is request? so i accept it

Ah AD 😅 — GitHub sends a **repo transfer request automatically** to the account you’re transferring to. Here’s how to find and accept it:

Step 1: Check your email

- GitHub sends an email to **your account email** (the one for Hanzai003)
- Subject looks like:

arduino

 Copy code

Repository transfer request: DevOpsZaid

- Open it → click **Accept transfer**
-

Step 2: Check GitHub notifications (in case you missed email)

1. Log in to GitHub as Hanzai003
 2. Click  (notifications) in the top right
 3. Look for "Repository transfer requested" → click Accept
-

Step 3: After accepting

- You are now **full owner**
 - Go to the repo → **Settings tab** will appear
 - You can now add **Secrets** → **Actions**, manage workflows, everything
-

 **Tip:** If you don't see the request:

- Make sure you're **logged in to the correct account** (Hanzai003)
 - Check **email spam/junk folder**
-

If you want, I can write a **super quick "WhatsApp message you can send to your other collaborator"** in case the request didn't appear and you need them to resend the transfer.

Do you want me to do that?



I checked email and click url now what?

"Skip to content

Navigation Menu

Dashboard

Type / to search

Moving repository to Hanzai003/DevOpsZaid. This may take a few minutes.

Top repositories

Find a repository...

DevOpsZaid

Hanzai003/DevOpsZaid
devops_demo
Hanzai003/devops_demo
Revive-Fitness
Hanzai003/Revive-Fitness
Celestial-Cuisine
Hanzai003/Celestial-Cuisine
Netflix-Clone
Hanzai003/Netflix-Clone
Utility-Store-Super-Market-Management-System
Hanzai003/Utility-Store-Super-Market-Management-System
Airline-3
Hanzai003/Airline-3
Dashboard
Home
Ask anything
Chat Commands

Task
New

Create issue

Write code

Git

Pull requests
Feed
Trending repositories

See more
@code-yeongyu profile
code-yeongyu/oh-my-opencode
#1 OpenCode Plugin- Battery included. ASYNC
SUBAGENTS (YES LIKE CLAUDE CODE) · Curated agents with proper models · Crafted tools like LSP/AST included · Curated MCPs · Claude Code Compatible Layer ...
TypeScript 6.2k

@steveyegge profile
steveyegge/gastown
Gas Town - multi-agent workspace manager
Go 1.2k
Recommended for you
@cloudposse profile
cloudposse/terraform-provider-utils
The Cloud Posse Terraform Provider for various utilities
(e.g. deep merging, stack configuration management)
Go 109
Footer
© 2026 GitHub, Inc.
Footer navigation
Terms
Privacy
Security
Status
Community
Docs
Contact
Manage cookies
Do not share my personal information
Latest from our changelog
3 days ago
Reduced pricing for GitHub-hosted runners usage
2 weeks ago
Improved performance for GitHub Actions workflows page
2 weeks ago
Control who can request apps for your organization now
in public preview
2 weeks ago
You can now require reviews before closing Dependabot
alerts with delegated alert dismissal
View changelog →
Chat

New Conversation

 Explain a complex thing

Explain Artificial Intelligence so that I can explain it to my six-year-old child.

 Get suggestions and create new ideas

Please give me the best 10 travel ideas around the world

 Translate, summarize, fix grammar and more...

Translate "I love you" French

GPT-4o Mini

Hello, how can I help you today?

slope of perpendicular lines

GPT-4o Mini

The slopes of perpendicular lines have a special relationship in the context of coordinate geometry. If two lines are perpendicular, the product of their slopes is equal to -1.

If you have a line with a slope (m_1), then the slope (m_2) of a line that is perpendicular to it can be found using the following formula:

$$\begin{bmatrix} m_1 \cdot m_2 = -1 \\ \end{bmatrix}$$

This can also be rewritten to find (m_2):

$$\begin{bmatrix} m_2 = -\frac{1}{m_1} \\ \end{bmatrix}$$

For example:

If one line has a slope of 3, then the slope of a line that is perpendicular to it would be ($-\frac{1}{3}$).

Conversely, if one line has a slope of -2, then the slope of

any line that is perpendicular to it would be ($\frac{1}{2}$).

This relationship holds true as long as neither line is vertical (undefined slope) or horizontal (zero slope), since vertical and horizontal lines are always mutually perpendicular but do not have defined numerical slopes in this context.

AITOPIA
coin image
10
Upgrade

Ask me anything...

Make a Review & Earn Credit ❤️
Chat
Ask
Search
Write
Image
ChatFile
Vision
Agent
Full Page
Invite & Earn "

Perfect AD 😊 — that page means the **transfer is in progress**.

Here's what you do next:

Step 1: Wait a few minutes

- GitHub says:

css

 Copy code

Moving repository **to** Hanzai003/DevOpsZaid. This may take **a** few minutes.

- Don't close the browser, just wait 1–5 minutes.
-

Step 2: Verify the transfer

1. After a few minutes, go to your **GitHub profile** (<https://github.com/Hanzai003>)
 2. Check your repositories — you should see `DevOpsZaid` under your account now.
 3. Open it — you should see the **Settings tab** appear on top menu.
-

Step 3: Confirm ownership

- Go to:

bash

 Copy code

<https://github.com/Hanzai003/DevOpsZaid/settings/secrets/actions>

- Now you **should be able to access it**.

 If the Settings tab appears → you are now **full owner**

 If still missing → log out → log back in → refresh

After this, you can:

- Add **AWS secrets**
 - Push workflow for **automation of website deployment**
 - Manage everything as full owner
-

If you want, I can write a **step-by-step plan for GitHub Actions workflow** so that **pushing to your repo automatically deploys your website to S3** — ready for you to just copy-

paste.

Do you want me to do that next?



Alright AD — I hear you loud and clear 😎

Yes ✅ — we can now automate your **website deployment from GitHub to AWS S3** using **Terraform + GitHub Actions**. We'll go **step by step**, slowly, safely, and I'll tell you **exactly what to do, where, and why**.

No skipping. No assumptions. Let's go.

```
# **STEP 0: PRE-REQUISITES (CHECK)**
```

Before automation, make sure:

1. Your **S3 bucket works** manually (Step 7 and 8 done ✅).
2. Your **Terraform code works locally** (`terraform apply` runs and creates S3).
3. Your **AWS CLI credentials** exist and have `AmazonS3FullAccess`.
4. Your **website files are in GitHub repo** (`home.html`, `style.css`, `script.js`, `images/` all committed).

If all above is ✅, we move forward.

```
# **STEP 1: Decide GitHub Repo**
```

You **do NOT need a new repository**.

Use your **existing repo** (DevOpsZaid) because:

- * .github/workflows/ already exists
- * Your home.html etc. are already committed

We will **add one workflow YAML** to automate Terraform & S3 deployment.

STEP 2: Create GitHub Secrets

We **never put AWS keys in code**. Instead, store them as **GitHub Secrets**.

🔑 **WHERE:** GitHub → DevOpsZaid → Settings → Secrets and variables → Actions → New repository secret

Create these secrets:

Secret Name	Value
AWS_ACCESS_KEY_ID	(from your Terraform IAM user)
AWS_SECRET_ACCESS_KEY	(from your Terraform IAM user)
AWS_REGION	us-east-1 (or your region)
BUCKET_NAME	devopszaid-static-site-ad-2026 (your bucket)

These will be used **inside the GitHub Actions workflow**, never hardcoded.

STEP 3: Create GitHub Actions Workflow

1. **Go to your repo:** .github/workflows/

2. **Create a new file** (name it e.g.):

terraform-s3-deploy.yaml

3. **Paste the following workflow** (I explain each step):

```
yaml
name: Terraform S3 Deploy

on:
  push:
    branches:
      - main    # trigger on push to main branch

jobs:
  deploy:
    name: Deploy Website
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repo
        uses: actions/checkout@v3

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2
        with:
          terraform_version: 1.7.7    # use your installed version

      - name: Terraform Init
        run: terraform init
        working-directory: terraform-s3

      - name: Terraform Apply
        run: terraform apply -auto-approve
        working-directory: terraform-s3
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          AWS_REGION: ${{ secrets.AWS_REGION }}
```

```
- name: Upload Website Files to S3
  run: |
    aws s3 sync DevOpsZaid/ s3://${{ secrets.AWS_ACCESS_KEY_ID }} ${{
    env:
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
      AWS_REGION: ${{ secrets.AWS_REGION }}
```

STEP 4: Commit & Push Workflow

```
bash
git add .github/workflows/terraform-s3-deploy.yaml
git commit -m "Add GitHub Actions workflow to deploy website"
git push origin main
```

 This will trigger the workflow **automatically**.

STEP 5: What happens when workflow runs

1. GitHub checks out your code
2. Sets up Terraform
3. Runs `terraform init` in `terraform-s3`
4. Runs `terraform apply` (creates S3 bucket if needed)
5. Syncs your **website files from `DevOpsZaid/` folder** into the S3 bucket
6. Website is live immediately after

STEP 6: Verify

1. Go to **Actions tab** in GitHub → check workflow run
→ must show 
2. Go to **AWS S3 → bucket → Properties → Static website hosting → endpoint**
3. Open URL → website should reflect **latest commit**

STEP 7: Notes / Tips

- * Workflow is safe: uses **secrets**, no hardcoded AWS keys
- * If you add new pages/images → push → workflow auto-updates S3
- * You **don't need a new repository**, your existing DevOpsZaid repo is enough
- * **Folder structure matters:** files in DevOpsZaid/ will mirror in S3

AD  — this is the **exact step-by-step setup** for automation.

Next, I can give you a **super simplified command + workflow version** that will:

- * Automatically deploy S3
- * Upload website
- * Do **everything with one commit**

Do you want me to do that?

i accidentally deleted .github/workflows for Step 3

Ah AD 😅 — don't panic. Deleting .github/workflows just removed the folder where workflows live, your code and everything else is safe.

Here's how to fix it **step by step**:

Step 1: Recreate the folder

From your project root (`DevOpsZaid` folder):

bash

 Copy code

```
mkdir -p .github/workflows
```

- `-p` ensures both `.github` and `workflows` are created if missing
 - This recreates the folder structure exactly like GitHub expects
-

Step 2: Recreate your workflow file

bash

 Copy code

```
notepad .github/workflows/terraform-s3-deploy.yml
```

Paste the **workflow YAML** I gave you earlier:

yaml

 Copy code

```
name: Terraform S3 Deploy
```

on:

```
  push:  
    branches:  
      - main
```

jobs:

```
  deploy:  
    name: Deploy Website  
    runs-on: ubuntu-latest
```

steps:

```
  - name: Checkout repo  
    uses: actions/checkout@v3
```

```

- name: Setup Terraform
  uses: hashicorp/setup-terraform@v2
  with:
    terraform_version: 1.7.7

- name: Terraform Init
  run: terraform init
  working-directory: terraform-s3

- name: Terraform Apply
  run: terraform apply -auto-approve
  working-directory: terraform-s3
  env:
    AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
    AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
    AWS_REGION: ${{ secrets.AWS_REGION }}

- name: Upload Website Files to S3
  run: |
    aws s3 sync DevOpsZaid/ s3://${{ secrets.BUCKET_NAME }} --delete
  env:
    AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
    AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
    AWS_REGION: ${{ secrets.AWS_REGION }}

```

Save → close.

Step 3: Commit & push the workflow

bash

 Copy code

```

git add .github/workflows/terraform-s3-deploy.yml
git commit -m "Recreate workflow after accidental deletion"
git push origin main

```

Step 4: Verify

1. Go to GitHub → Actions tab

2. The workflow should trigger automatically on push → see  if it works
 3. Website should deploy again to your S3 bucket
-

AD 😎 — After this, everything will be back like before, workflow working, S3 updates automatically.

If you want, I can also give a super short “one-shot restore + deploy” command you can just run next time to fix workflow issues in one go.

Do you want me to do that?



Where will I create the file, in main folder or where? also where to run these following commands then?

```
""git add .github/workflows/terraform-s3-deploy.yml  
git commit -m "Recreate workflow after accidental  
deletion"  
git push origin main  
""
```

Perfect AD 😎 — let's make this crystal clear:

Where to create the workflow file

- Go to your project root folder:

makefile

 Copy code

D:\4th Semester\DevOps - Lab\Semester Project\DevOpsZaid\DevOpsZaid

- Inside this folder, create the folder structure:

bash

 Copy code

```
.github/workflows/
```

- Then create the workflow file inside it:

 Copy code

```
terraform-s3-deploy.yml
```

So full path will be:

```
makefile
```

 Copy code

```
D:\4th Semester\DevOps - Lab\Semester Project\DevOpsZaid\DevOpsZaid\.github\workf
```



Where to run the Git commands

- Open Git Bash in the same folder, i.e., your project root (DevOpsZaid)
- Then run:

```
bash
```

 Copy code

```
git add .github/workflows/terraform-s3-deploy.yml  
git commit -m "Recreate workflow after accidental deletion"  
git push origin main
```

- This adds the workflow, commits it, and pushes it to GitHub.
- After push → GitHub Actions will trigger automatically.

Summary:

1. File location: DevOpsZaid/.github/workflows/terraform-s3-deploy.yml
2. Commands: run inside DevOpsZaid folder, where .git exists

If you want, I can give you the exact one-liner commands to recreate folder, file, and push all in one go so nothing breaks.

Do you want me to do that?



But i created .github/workflows/ in the main folder,
"D:\4th Semester\DevOps - Lab\Semester
Project\DevOpsZaid"

Ah AD 😊 — that's the small issue:

You created .github/workflows/ in the **parent folder** (DevOpsZaid), but your **Git repository root** (where .git is) is actually inside:

makefile

Copy code

D:\4th Semester\DevOps - Lab\Semester Project\DevOpsZaid\DevOpsZaid

Git only tracks files **inside the folder where .git exists.**

So your .github/workflows/ folder **must be inside this folder**, not the parent.

What to do now

1. Move the folder to the correct location:

bash

Copy code

`mv "D:/4th Semester/DevOps - Lab/Semester Project/DevOpsZaid/.github" "D:/4th Sem`

-
- Now the full path will be:

makefile

Copy code

D:\4th Semester\DevOps - Lab\Semester Project\DevOpsZaid\DevOpsZaid\.github\workf

2. Go to Git repository root:

bash

 Copy code

```
cd "D:/4th Semester/DevOps - Lab/Semester Project/DevOpsZaid/DevOpsZaid"
```

3. Run Git commands:

bash

 Copy code

```
git add .github/workflows/terraform-s3-deploy.yml  
git commit -m "Fix workflow location"  
git push origin main
```

After this, GitHub Actions will detect the workflow correctly and trigger it on push.

AD 😎 — if you want, I can give a super short one-shot command set that moves the folder, creates workflow, and pushes it all in one go.

Do you want me to do that?



.git is in main folder, and in sub DevOpsZaid folder also where there are home.html,etc files. Like my website is there, outside in main folder, it's DevOpsZaid's folder to represent the website, terraform-s3 folder.git folders

Ahhh AD 😆 — now it's crystal clear. Let me summarize your setup carefully:

yaml

 Copy code

```
Main folder: DevOpsZaid (has .git, terraform-s3, main.tf, provider.tf)  
Subfolder: DevOpsZaid (inside main) → contains website files: home.html, style.cs  
Other folder: terraform-s3 → contains main.tf, provider.tf for Terraform
```

So your Git repository root is **main folder**, not the subfolder. 

What this means for GitHub Actions

1. `.github/workflows/` must be **in the main folder**, i.e.:

makefile

 Copy code

```
D:\4th Semester\DevOps - Lab\Semester Project\DevOpsZaid\.github\workflows\terra
```

- 
2. In the workflow, when you upload website files, you need to **point to the subfolder** that actually has the website:

yaml

 Copy code

```
aws s3 sync DevOpsZaid/ s3://${{ secrets.BUCKET_NAME }} --delete
```

- `DevOpsZaid/` here refers to the subfolder with `home.html`, `style.css`, etc. 

3. **Git commands** also run from the **main folder**, because that's where `.git` exists:

bash

 Copy code

```
cd "D:/4th Semester/DevOps - Lab/Semester Project/DevOpsZaid"
git add .github/workflows/terraform-s3-deploy.yml
git commit -m "Recreate workflow after accidental deletion"
git push origin main
```

Summary for your situation

Thing	Location
<code>.git</code>	Main folder
<code>terraform-s3</code> folder	Main folder
Website files (<code>home.html</code> , etc)	Subfolder <code>DevOpsZaid/</code>
<code>.github/workflows</code>	Must be in main folder

Thing**Location**

Workflow YAML reference to files

Use DevOpsZaid/ in aws s3 sync

 This setup is correct — workflow will now see both Terraform folder and website files.

If you want, I can **rewrite the workflow YAML exactly for your setup**, so you just paste it and it works perfectly with your folder structure.

Do you want me to do that?



```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester
Project/DevOpsZaid/terraform-s3 (main)
$ cd ..
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
(main)
$ git add .github/workflows/terraform-s3-deploy.yml
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
(main)
$ git commit -m "Recreate workflow after accidental
deletion"
[main 2227910] Recreate workflow after accidental
deletion
1 file changed, 40 insertions(+)
create mode 100644 .github/workflows/terraform-s3-
deploy.yml
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
```

```
(main)
$ git push origin main
To https://github.com/AlyanShahbaz2022/DevOpsZaid.git
  ! [rejected]      main -> main (fetch first)
error: failed to push some refs to
'https://github.com/AlyanShahbaz2022/DevOpsZaid.g
it'
hint: Updates were rejected because the remote contains
work that you do not
hint: have locally. This is usually caused by another
repository pushing to
hint: the same ref. If you want to integrate the remote
changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help'
for details.
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
(main)
$
```

Ah AD 😅 — classic “remote has changes you don’t have locally” issue. Don’t panic, we can fix this **safely** without breaking anything.

Since your remote repo has commits that your local repo doesn’t, you need to **pull first** and then push.

Step 1: Pull remote changes safely

bash

 Copy code

```
git pull origin main --rebase
```

- `--rebase` keeps your commit (the workflow) on top of the remote changes.
- If there are **conflicts**, Git will tell you which files.

You resolve conflicts by editing the file, then:

bash

```
git add <file>
git rebase --continue
```

Step 2: Push your changes

bash

Copy code

```
git push origin main
```

This will now succeed because your local repo is fully in sync with remote.

TL;DR for you:

1. `git pull origin main --rebase`
2. Resolve any conflicts (if prompted)
3. `git push origin main`

After this, your **workflow will be live on GitHub** and Actions will trigger on push.

If you want, I can write the **exact commands for your setup** so you just copy-paste everything safely.

Do you want me to do that?

...

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
(main)
$ git pull origin main --rebase
error: cannot pull with rebase: You have unstaged changes.
error: Please commit or stash them.
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
```

```
Semester/DevOps - Lab/Semester Project/DevOpsZaid  
(main)  
$
```

Ah AD 😅 — Git is telling us: **you have unstaged changes**, so it won't pull until we either commit or stash them.

Since your only new file right now is the workflow YAML (we already added it), let's **stash your changes safely** and pull:

Step 1: Stash changes

bash

 Copy code

```
git stash
```

- This saves all your current changes safely and makes your working folder clean.
-

Step 2: Pull remote changes with rebase

bash

 Copy code

```
git pull origin main --rebase
```

Step 3: Apply your stashed changes back

bash

 Copy code

```
git stash pop
```

- This restores your workflow YAML file on top of the updated remote branch.
- If there's a conflict, Git will tell you — resolve it, then:

bash

 Copy code

```
git add <file>
git rebase --continue
```

Step 4: Push to GitHub

bash

 Copy code

```
git push origin main
```

 After this, your workflow is safely on GitHub, and Actions will trigger on push.

If you want, I can give you a **single copy-paste command sequence** for your exact situation to fix this without errors.

Do you want me to do that?

      ...

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
Semester/DevOps - Lab/Semester Project/DevOpsZaid
(main)
$ git stash pop
On branch main
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be
committed)
  (use "git restore <file>..." to discard changes in working
directory)
    modified:  main.tf
    deleted:   providers.tf
```

Untracked files:
 (use "git add <file>..." to include in what will be
committed)

```
terraform-s3/
```

```
no changes added to commit (use "git add" and/or "git  
commit -a")
```

```
Dropped refs/stash@{0}
```

```
(1c8d5a1cb2d6f2699e6a5c6be85eaec001dc719a)
```

```
HP@DESKTOP-K39IDH1 MINGW64 /d/4th
```

```
Semester/DevOps - Lab/Semester Project/DevOpsZaid  
(main)
```

```
$
```

Ah AD 😊 — now Git is showing exactly what's happening after `stash pop` :

- `main.tf` was **modified**
- `providers.tf` was **deleted**
- `terraform-s3/` is **untracked** (your folder with Terraform files)

This is normal. Now we just need to **stage the files you want to keep**, commit, and push.

Step 1: Stage everything you want to commit

Since your workflow is in `.github/workflows/` and your Terraform folder is `terraform-s3/`, we can stage them all:

```
bash
```

 Copy code

```
git add .
```

This stages:

- “`.github/workflows/terraform-s3-deploy.yml`”
- “`terraform-s3/` **folder**”
- “**your `main.tf` if you want to keep changes**”

If you **don't want** `providers.tf` **deleted**, you need to restore it first:

```
bash
```

 Copy code

```
git restore providers.tf  
git add providers.tf
```

Step 2: Commit the changes

bash

 Copy code

```
git commit -m "Restore Terraform files and recreate workflow"
```

Step 3: Push to GitHub

bash

 Copy code

```
git push origin main
```

 After this, your workflow and Terraform files are safely on GitHub.

If you want, I can write a **one-shot copy-paste command sequence** that fixes everything from this state safely.

Do you want me to do that?



I have deleted .yml workflow files from github before, i don't understand what should I do now, i just did git add .

Ah AD, got it 😅 — let me break it **very clearly**.

You **deleted your workflow YAMLs** from GitHub before, but now locally you've **recreated them** and did `git add ..`. That's perfect. You're **ready to commit and push**.

Here's exactly what to do next:

Step 1: Commit your changes

bash

```
git commit -m "Recreate GitHub Actions workflow and restore Terraform files"
```

- This saves your changes locally (your workflow YAML and Terraform fold)  Copy code
-

Step 2: Pull remote changes first (just in case)

bash

```
git pull origin main --rebase
```

- If it says **no changes**, great.  Copy code
 - If it asks to resolve conflicts, Git will guide you (you can type `git add <file>` for conflicts, then `git rebase --continue`).
-

Step 3: Push your changes to GitHub

bash

```
git push origin main
```

-  After this, your `.github/workflows/terraform-s3-deploy.yml` will be back on GitHub, and Actions will trigger on the next push.  Copy code
-

TL;DR:

1. `git commit -m "message"` → save your recreated workflow
 2. `git pull origin main --rebase` → sync remote
 3. `git push origin main` → upload everything
-

If you want, I can give you a **single copy-paste command sequence** to do all of Step 1-3 **safely** so you won't run into errors again.

Do you want me to do that?

