

CI/CD Project: Automating Terraform Deployments with AWS CodeBuild

Overview

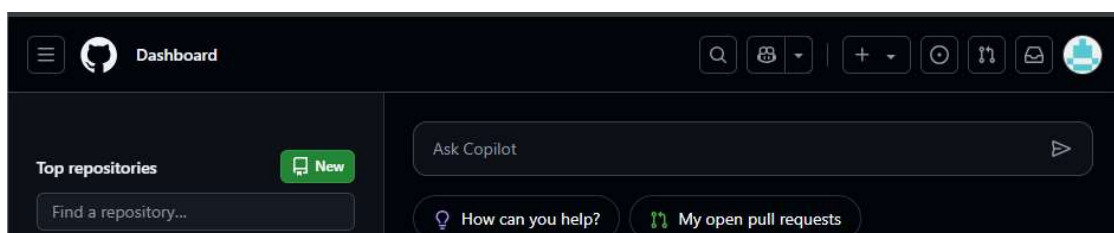
This project sets up a CI/CD pipeline using AWS CodeBuild to automatically apply a Terraform script whenever a change is committed to a GitHub repository. The key steps include:

- Cloning a private GitHub repository using SSH
- Creating an IAM user with programmatic access
- Writing shell scripts and a buildspec.yml file
- Creating a personal access token
- Storing the Terraform state file in an S3 bucket
- Setting up an AWS CodeBuild job

1. Clone a Private GitHub Repository Using SSH

Steps:

1. Create a GitHub Repository (if not already done):
 - Go to GitHub and create a new private repository.



- Name it (e.g., SJCLOUD-EC2-INSTANCE).
- Add a README file and select "Terraform" from the .gitignore dropdown.
- Click "Create Repository."

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * SJCLLOUD2024 / Repository name * SJCLLOUD-EC2-INSTANCE
✓ SJCLLOUD-EC2-INSTANCE is available.

Great repository names are short and memorable. Need inspiration? How about [bookish-couscous](#) ?

Description (optional)

☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☒ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Terraform

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

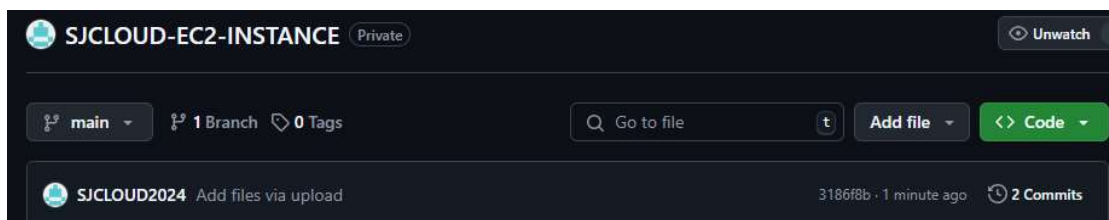
This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a private repository in your personal account.

[Create repository](#)

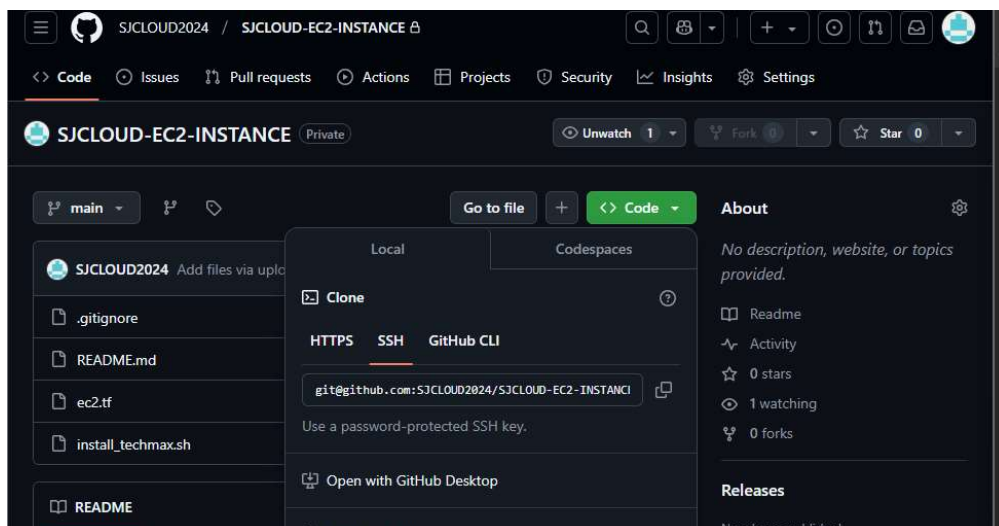
2. Upload Required Files:

- Click "Add File" > "Upload Files," then drag and drop the necessary files. (Files can be found in my repo)
- Click "Commit Changes."

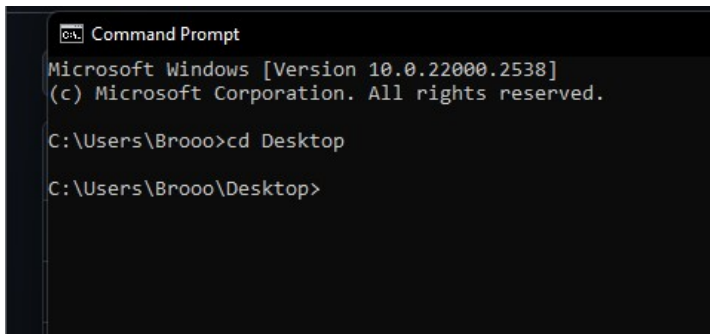


3. Clone the Repository Locally:

- Copy the SSH URL from the "Code" section (ensure the "SSH" tab is selected).



- Open a terminal and navigate to the desired directory (cd Desktop if cloning to Desktop).

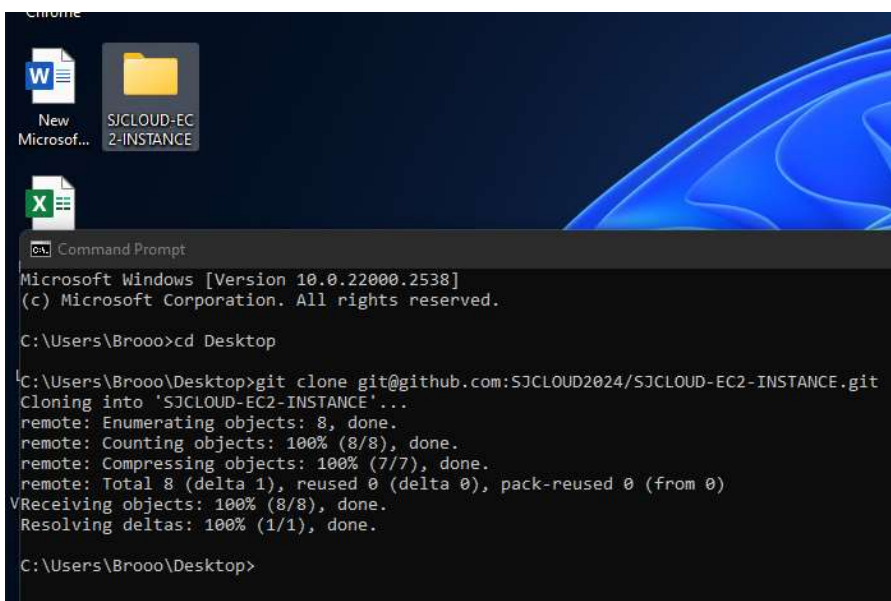


```
Command Prompt
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Brooo>cd Desktop

C:\Users\Brooo\Desktop>
```

- Run: git clone <SSH-URL>



```
Command Prompt
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

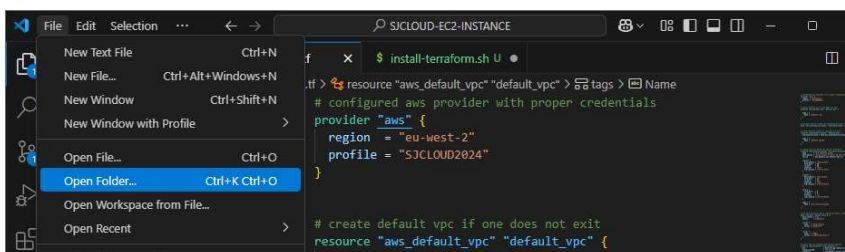
C:\Users\Brooo>cd Desktop

C:\Users\Brooo\Desktop>git clone git@github.com:SIJCLLOUD2024/SIJCLLOUD-EC2-INSTANCE.git
Cloning into 'SIJCLLOUD-EC2-INSTANCE'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

C:\Users\Brooo\Desktop>
```

2. Test Your Terraform Script Locally

1. Open **Visual Studio Code (VS Code)** and load the Terraform project folder.

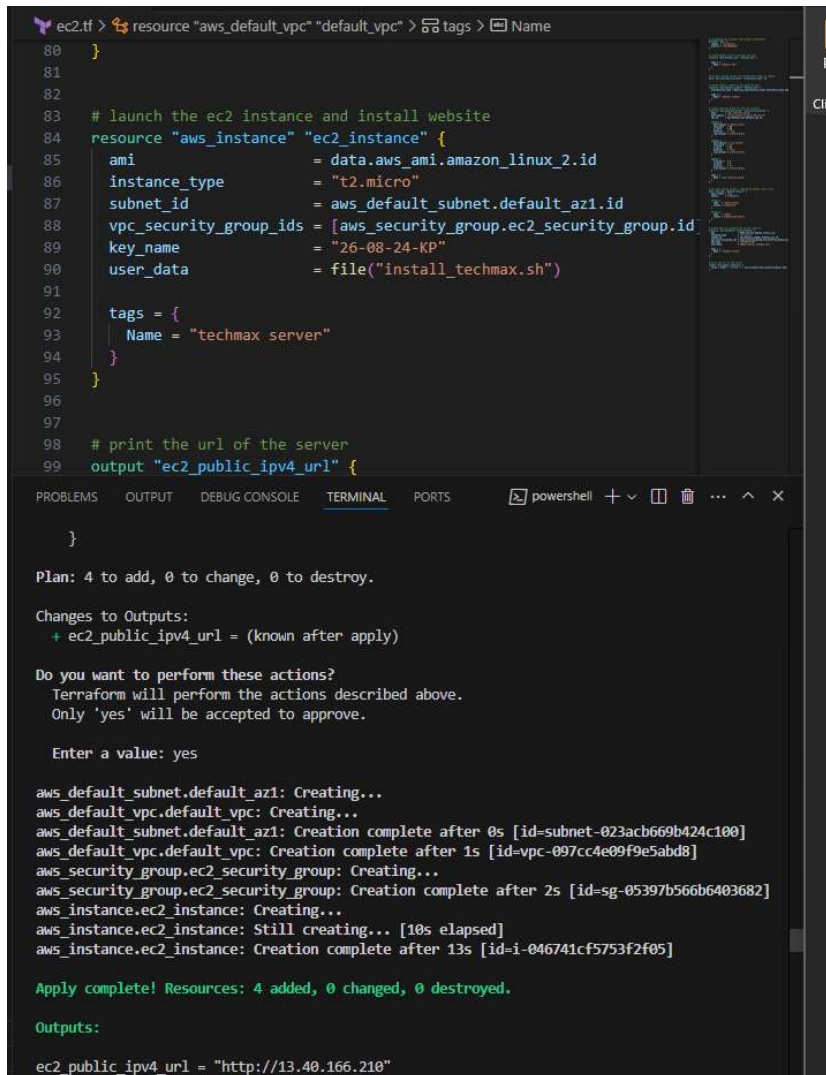


2. Modify the ec2.tf file:

- Update the profile on line 4 and key_name on line 89.

3. Run the following commands in the terminal:

- Terraform init
- Terraform plan
- Terraform apply



The screenshot shows a VS Code terminal window with a dark theme. The top part of the terminal displays Terraform configuration code for an AWS VPC and an EC2 instance. The code includes resource definitions for 'aws_default_vpc', 'aws_instance', and an output for 'ec2_public_ipv4_url'. The bottom part of the terminal shows the output of the 'terraform apply' command, indicating that resources were successfully created and the EC2 instance is now running. The output includes details about the creation of the VPC, subnet, security group, and the EC2 instance itself, along with the public IP address of the instance.

```
ec2.tf > resource "aws_default_vpc" "default_vpc" {
  tags = {
    Name = "default_vpc"
  }
}

# launch the ec2 instance and install website
resource "aws_instance" "ec2_instance" {
  ami           = data.aws_ami.amazon_linux_2.id
  instance_type = "t2.micro"
  subnet_id     = aws_default_subnet.default_az1.id
  vpc_security_group_ids = [aws_security_group.ec2_security_group.id]
  key_name      = "26-08-24-KP"
  user_data     = file("install_techmax.sh")

  tags = {
    Name = "techmax_server"
  }
}

# print the url of the server
output "ec2_public_ipv4_url" {
  value = aws_instance.ec2_instance.public_ip
}
```

Plan: 4 to add, 0 to change, 0 to destroy.

Changes to Outputs:

- + ec2_public_ipv4_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

aws_default_subnet.default_az1: Creating...

aws_default_vpc.default_vpc: Creating...

aws_default_subnet.default_az1: Creation complete after 0s [id=subnet-023acb669b424c100]

aws_default_vpc.default_vpc: Creation complete after 1s [id=vpc-097cc4e09f9e5abd8]

aws_security_group.ec2_security_group: Creating...

aws_security_group.ec2_security_group: Creation complete after 2s [id=sg-05397b566b6403682]

aws_instance.ec2_instance: Creating...

aws_instance.ec2_instance: Still creating... [10s elapsed]

aws_instance.ec2_instance: Creation complete after 13s [id=i-046741cf5753f2f05]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

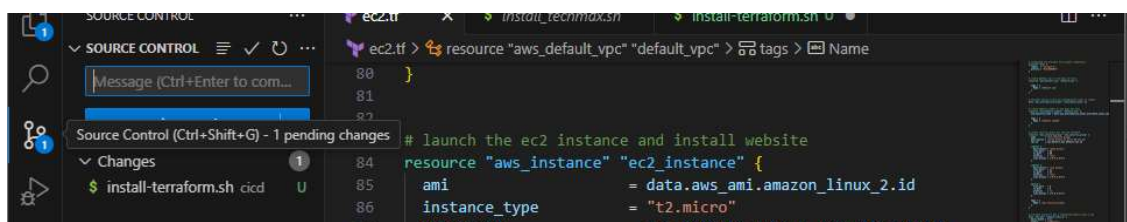
ec2_public_ipv4_url = "http://13.40.166.210"

4. Verify that the EC2 instance is created and the install_techmax.sh script is executed.

5. Once confirmed, destroy the resources by running the following command:

- Terraform destroy

6. Finally remember to push the changes to your GitHub repo



The screenshot shows a VS Code interface with a dark theme. On the left, the 'Source Control' panel is open, showing a list of changes. The main editor area displays the same Terraform configuration code as the previous screenshot. The terminal window at the bottom shows the output of the 'terraform destroy' command, indicating that resources were successfully destroyed. The output includes details about the destruction of the VPC, subnet, security group, and the EC2 instance itself.

```
ec2.tf > resource "aws_default_vpc" "default_vpc" {
  tags = {
    Name = "default_vpc"
  }
}

# launch the ec2 instance and install website
resource "aws_instance" "ec2_instance" {
  ami           = data.aws_ami.amazon_linux_2.id
  instance_type = "t2.micro"
  subnet_id     = aws_default_subnet.default_az1.id
  vpc_security_group_ids = [aws_security_group.ec2_security_group.id]
  key_name      = "26-08-24-KP"
  user_data     = file("install_techmax.sh")

  tags = {
    Name = "techmax_server"
  }
}

# print the url of the server
output "ec2_public_ipv4_url" {
  value = aws_instance.ec2_instance.public_ip
}
```

Plan: 4 to destroy, 0 to add, 0 to change.

Changes to Outputs:

- ec2_public_ipv4_url = (known after apply)

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

Enter a value: yes

aws_instance.ec2_instance: Destroying...

aws_instance.ec2_instance: Destruction complete after 0s

aws_security_group.ec2_security_group: Destroying...

aws_security_group.ec2_security_group: Destruction complete after 0s

aws_default_vpc.default_vpc: Destroying...

aws_default_vpc.default_vpc: Destruction complete after 0s

aws_default_subnet.default_az1: Destroying...

aws_default_subnet.default_az1: Destruction complete after 0s

Apply complete! Resources: 4 destroyed, 0 added, 0 changed.

Outputs:

ec2_public_ipv4_url = "http://13.40.166.210"

3. Create Shell Scripts for CodeBuild

Steps:

1. In VS Code, create a new folder named `cicd` inside your project.
2. Inside `cicd`, create three shell scripts:
 - **install-terraform.sh** (Installs Terraform in the CodeBuild container)
 - **configure-named-profile.sh** (Configures an AWS named profile)
 - **apply-terraform.sh** (Runs Terraform commands)

Install-terraform.sh:

```
cicd > $ install-terraform.sh
1  #!/bin/bash
2
3  # fail on any error
4  set -eu
5
6  # install yum-config-manager to manage your repositories
7  sudo yum install -y yum-utils
8
9  # use yum-config-manager to add the official HashiCorp Linux repository
10 sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
11
12 # install terraform
13 sudo yum -y install terraform
14
15 # verify terraform is installed
16 terraform --version
```

Configure-name-profile.sh:

```
cicd > $ configure-named-profile.sh
1  #!/bin/bash
2
3  # fail on any error
4  set -eu
5
6  # configure named profile
7  aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID --profile $PROFILE_NAME
8  aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY --profile $PROFILE_NAME
9  aws configure set region $AWS_REGION --profile $PROFILE_NAME
10
11 # verify that profile is configured
12 aws configure list --profile $PROFILE_NAME
```

apply-terraform.sh:

```
cicd > $ apply-terraform.sh
1  #!/bin/bash
2
3  # fail on any error
4  set -eu
5
6  # go back to the previous directory
7  cd ..
8
9  # initialize terraform
10 terraform init
11
12 # # apply terraform
13 terraform apply -auto-approve
14
15 # destroy terraform
16 # terraform destroy -auto-approve
```

4. Create a buildspec.yml File for CodeBuild

1. In the cicd folder, create a file named buildspec.yml
2. Insert the following content:

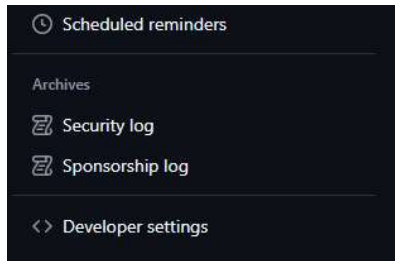
```
cicd > ! buildspec.yml
1  version: 0.2
2
3  phases:
4    install:
5      runtime-versions:
6        python: 3.x
7
8    pre_build:
9      commands:
10       - cd cicd # change directory
11       - chmod +x install-terraform.sh configure-named-profile.sh apply-terraform.sh # make files executable
12       - ./install-terraform.sh # install terraform
13       - ./configure-named-profile.sh # configure named profile
14
15    build:
16      commands:
17       - ./apply-terraform.sh
```

3. Save and push all changes to GitHub.

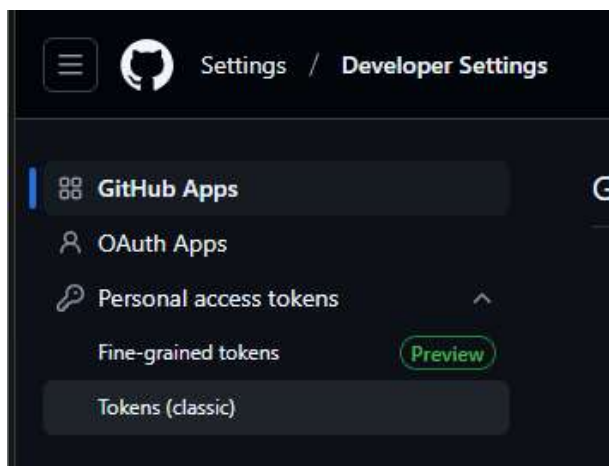
5. Create a Personal Access Token in GitHub

Steps:

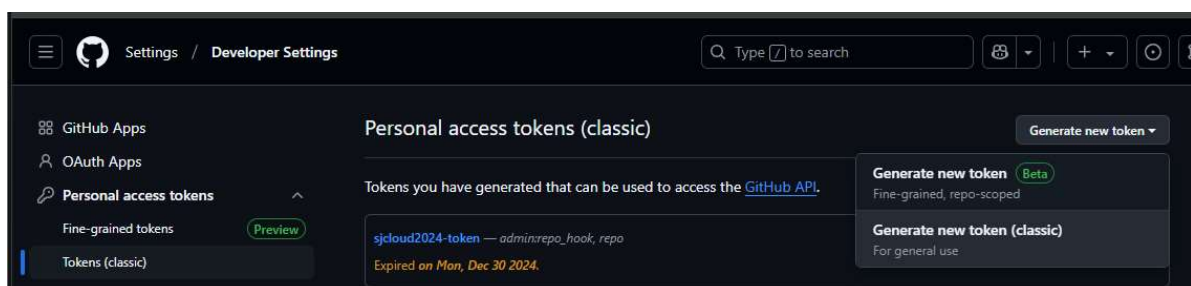
1. Navigate to GitHub Settings > Developer Settings.



2. Go to Personal Access Tokens > Tokens (Classic).



3. Click Generate New Token (Classic).



4. Set a name (e.g., sjcloud-token) and choose an expiration date.
5. Select **repo** and **admin:repo_hook** scopes.
6. Generate and securely store the token.

Note

sjcloud2024-token

What's this token for?

Expiration *

90 days ▾ The token will expire on Fri, May 2 2025

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks

Once the personal access token has been created, make sure to save your token on your computer because you will only be able to see the code token once.

6. Store Terraform State in an S3 Bucket

Steps:

1. In AWS, navigate to S3 and create a bucket:
 - Choose a unique name.

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region

Europe (London) eu-west-2

Bucket name [Info](#)

sjcloud1306-terraform-state-bucket

Bucket name must be unique within the global namespace and follow th

- Enable Versioning.

Amazon S3 > Buckets > Create bucket

☒ Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

☐ Disable

☒ Enable

- Leave other settings as default and create the bucket.

2. Modify the Terraform provider file to include lines 7 to 15:

```
ec2.tf > terraform > backend "s3" > region
> .terraform
  > cid
    $ apply-terraform.sh
    ! buildspec.yml
    $ configure-named-profile.sh
    $ install-terraform.sh
  .gitignore
  .terraform.lock.hcl
  ec2.tf
  $ install_techmax.sh
  README.md
  {} terraform.tfstate
  terraform.tfstate.backup
  1
  2 # configured aws provider with proper credentials
  3 provider "aws" {
  4   region = "eu-west-2"
  5   profile = "SJCLLOUD-PROGRAMMATIC-USER"
  6 }
  7 # store the terraform state file in s3
  8 terraform {
  9   backend "s3" {
 10     bucket = "sjcloud1306-terraform-state-bucket"
 11     key    = "build/terraform.tfstate"
 12     region = "eu-west-2"
 13     profile = "SJCLLOUD-PROGRAMMATIC-USER"
 14   }
 15 }
```

3. Run **terraform init** to configure the backend. VSCode will output a message confirming the successful configuration of the backend S3 bucket which will store the terraform state file. refer back to S3 in the management console. You should be able to see your terraform.tfstate saved in your bucket as below.

sjcloud1306-terraform-state-bucket [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

Objects (1) [Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Details](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to grant permissions.

☒ Show versions

<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	build/	Folder	-	

Finally make sure to run **terraform destroy**!

7. Create a Build Project in AWS CodeBuild

Steps:

1. Navigate to AWS CodeBuild and click Create Project.

Developer Tools > CodeBuild > Build projects						
Build projects Info			Actions ▾	Create trigger	View details	Start build ▾ Create project
<input type="text"/>		Your projects ▾		< 1 > ⚙		
Name	Source provider	Repository	Latest build status	Description	Last Modified	
<input type="radio"/> Launch-EC2-INSTANCE-build	GitHub	SJCLLOUD2024/Launch-EC2-INSTANCE	Succeeded	build project to apply terraform scripts	4 months ago	

2. Configure:

- Name: Use the same name as your GitHub repo.

Project configuration

Project name

SJCLLOUD-EC2-INSTANCE-BUILD

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Project type

Select what type of project you would like to create. [Info](#)

☒ **Default project**
Create a custom CodeBuild project.

☐ **Runner project**
Create a CodeBuild managed runner for workflows in GitHub Actions, GitHub Enterprise Actions, GitLab, or Buildkite.

▼ Additional configuration

Description, public build access, build badge, concurrent build limit, tags

Description - optional

Build project to apply terraform script

Public build access - optional

Public build access allows you to make the build results, including logs and artifacts, for this project available for the general public.

☐ Enable public build access

Build badge - optional

☐ Enable build badge

Enable concurrent build limit - optional

Limit the number of allowed concurrent builds for this project.

☐ Restrict number of concurrent builds this project can start

Tags

Key	Value	

- Source Provider: GitHub. You may get the following error notice. In this case click manage account credentials. Within manage account credentials be sure to select 'personal access token' and under 'service' select 'CodeBuild'. The following section will require you to input your personal access token. Please refer to above sections for reference.

▼ Source Add source

Source 1 - Primary

Source provider
GitHub

Credential
⚠ You have not connected to GitHub. [Manage account credentials.](#)

☐ Use override credentials for this project only

Repository
☒ Repository in my GitHub account ☐ Public repository ☐ GitHub scoped webhook

Manage default source credential

Source Provider
Q GitHub

Credential type
☐ GitHub App
Connect project to GitHub using an AWS managed GitHub App ☒ Personal access token
Connect project to GitHub using a personal access token ☐ OAuth app
Connect project to GitHub using an OAuth app

Service
☐ Secrets Manager (recommended)
Use Secrets Manager to store token ☒ CodeBuild
Use CodeBuild managed token

GitHub personal access token
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 Save

Once the above settings have been saved, you should be taken back to the main build page. Now when you select the 'repository in my GitHub account' you should be presented with the below:

▼ Source Add source

Source 1 - Primary

Source provider
GitHub

Credential
✔ Your account is successfully connected through PAT using CodeBuild managed token. [Manage account credentials.](#)

☐ Use override credentials for this project only

Repository
☒ Repository in my GitHub account ☐ Public repository ☐ GitHub scoped webhook

Search
SJCLLOUD2024/Dev-Environment-SJCLLOUD24
https://github.com/SJCLLOUD2024/Dev-Environment-SJCLLOUD24.git
SJCLLOUD2024/docker-projects
https://github.com/SJCLLOUD2024/docker-projects.git
SJCLLOUD2024/Launch-EC2-INSTANCE
https://github.com/SJCLLOUD2024/Launch-EC2-INSTANCE.git
SJCLLOUD2024/SJCLLOUD-EC2-INSTANCE
https://github.com/SJCLLOUD2024/SJCLLOUD-EC2-INSTANCE.git
SJCLLOUD2024/Terraform-project
https://github.com/SJCLLOUD2024/Terraform-project.git

Git submodules - optional

Make sure to select the correct repository from the dropdown list.

- Webhook Events: Enable rebuild on code changes and below select 'single build'. As the name suggests this will rebuild our code any time changes are pushed to our GitHub repository. Within the webhook event filter groups select 'push' and 'pull_request_merged'. Selecting 'PUSH' and 'PULL_REQUEST_MERGED' means any time we push or merge a pull request in our GitHub repository it will rebuild our project.

▼ Primary source webhook events [Info](#)

Webhook - optional [Info](#)

☒ Rebuild every time a code change is pushed to this repository

Build type

☒ Single build
Triggers single build

☐ Batch build
Triggers multiple builds as single execution

▼ Webhook event filter groups [Add filter group](#)

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1 [Remove filter group](#)

Event type - optional

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PUSH X

PULL_REQUEST_MERGED X

Filters

Add one or more filters to specify whether or not a build is triggered based on the selected condition, type and pattern.

[Add filter](#)

▼ Additional configuration

Manual creation - optional [Info](#)

☐ Manually create a webhook for this repository in GitHub console.

- Environment:
 - Compute: EC2
 - OS: Amazon Linux
 - Runtime: Standard, latest image

▼ Environment

Provisioning model [Info](#)

☒ On-demand
Automatically provision build infrastructure in response to new builds.

☐ Reserved capacity
Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

☒ Managed image
Use an image managed by AWS CodeBuild

☐ Custom image
Specify a Docker image

Compute

☒ EC2
Optimized for flexibility during action runs

☐ Lambda
Optimized for speed and minimizes the start up time of workflow actions

Operating system

Amazon Linux

Runtime(s)

Standard

Image

aws/codebuild/amazonlinux-x86_64-standard:5.0

Image version

Always use the latest image for this runtime version

- Environment Variables: Add
 - AWS_ACCESS_KEY_ID - Value (access key from when your programmatic user was created)
 - AWS_SECRET_ACCESS_KEY – Value (secret key from your programmatic user)
 - AWS_REGION – Value (as per the region of your EC2 instance)
 - PROFILE_NAME – Value (IAM user name)

```

2
3 # fail on any error
4 set -eu
5
6 # configure named profile
7 aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID --profile $PROFILE_NAME
8 aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY --profile $PROFILE_NAME
9 aws configure set region $AWS_REGION --profile $PROFILE_NAME
10
11 # verify that profile is configured
12 aws configure list --profile $PROFILE_NAME
  
```

3. Specify the **Buildspec File Location: cicd/buildspec.yml**

▼ Buildspec

Build specifications

☐ Insert build commands
Store build commands as build project configuration

☒ Use a buildspec file
Store build commands in a YAML-formatted buildspec file

Buildspec name - *optional*

By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

cicd/buildspec.yml

▼ Batch configuration

You can run a group of builds as a single execution. Batch configuration is also available in advanced option when starting build.

☐ Define batch configuration - *optional*
You can also define or override batch configuration when starting a build batch.

▼ Artifacts

Add artifact

Artifact 1 - Primary

Type

No artifacts

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

4. Click Create Build Project.

5. Click Start Build to test deployment.

SJCLLOUD-EC2-INSTANCE-BUILD

Actions ▼ Create trigger Edit Clone Debug build Start build with overrides Start build

Configuration

Source provider
GitHub

Primary repository
SJCLLOUD2024/SJCLLOUD-EC2-INSTANCE

Artifacts upload location
-

Service role
arn:aws:iam::975049930561:role/service-role/codebuild-SJCLLOUD-EC2-INSTANCE-BUILD-service-role

Public builds
Disabled

Build history

Batch history Project details Build triggers Metrics

Build history

Stop build View artifacts View logs Delete builds Retry build

< 1 > ⚙

<input type="checkbox"/>	Build run	Status	Build number	Source version	Submitter	Duration	Completed
<input type="checkbox"/>	SJCLLOUD-EC2-INSTANCE-BUILD:67c656d1-a657-445a-939c-083bf74a9746	✔ Succeeded	1	9db4562427d9ad3115bddf3311fed256d84e109f	GitHub-Hookshot/5cd0f99	1 minute 32 seconds	22 hours ago

When we click 'start build' CodeBuild will clone our repository where our script is located which is in GitHub in our container. Then CodeBuild will use our build spec file to run all the commands we have specified.

After a few minutes your build project should have been build and the status should be 'succeeded'.

The screenshot displays the AWS CodeBuild console interface. At the top, a green banner indicates 'Build started' and 'You have successfully started the following build: SJCLOUD-EC2-INSTANCE-BUILD:9376468a-2453-48c7-a947-4c3fc16a12f8'. Below this, the build ID 'SJCLOUD-EC2-INSTANCE-BUILD:9376468a-2453-48c7-a947-4c3fc16a12f8' is prominently displayed, accompanied by 'Stop build' and 'Retry build' buttons. The 'Build status' section shows a 'Succeeded' status with a green checkmark. It also lists the Initiator as 'root', the Build ARN, the Resolved source version, the Start time (Feb 4, 2025 8:46 PM), the End time (Feb 4, 2025 8:48 PM), and the Build number (2). Below the status, there are tabs for 'Build logs', 'Phase details', 'Reports', 'Environment variables', 'Build details', and 'Resource utilization'. The 'Build logs' tab is active, showing the last 310 lines of the build log. The log content includes various system messages and commands, such as 'Running on CodeBuild On-demand', 'Waiting for agent ping', 'Phase is DOWNLOAD_SOURCE', 'Processing environment variables', 'Resolved python runtime alias', and 'Installing Python version 3.13'. A 'Tail logs' button is visible in the top right of the log area. At the bottom of the console, there is a Windows taskbar with various application icons and a system tray showing the date and time.

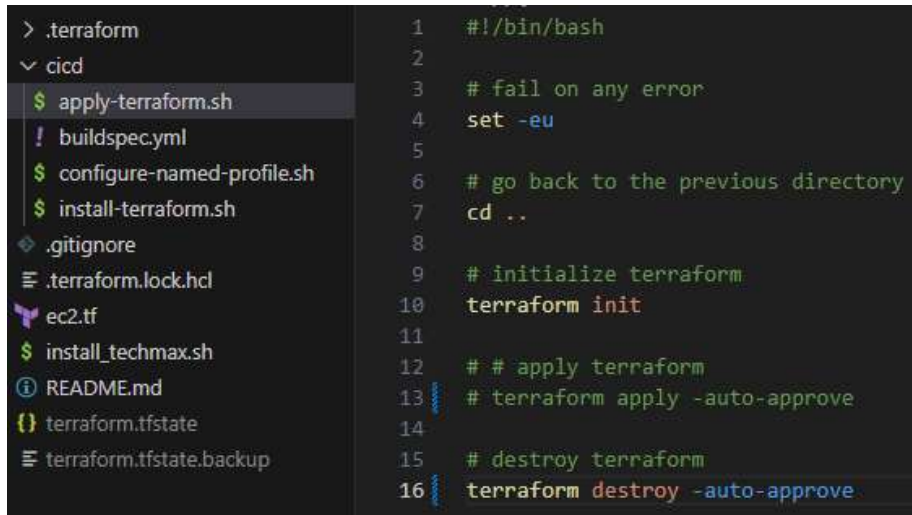
If you go through the build logs you will see what CodeBuild has done from start to finish. At this point you should be able to access your website via the link at the end of the CodeBuild log.

The screenshot shows the homepage of the Techmax website. The header includes the Techmax logo, navigation links for 'Home', 'About Us', 'Pages', 'Blog', and 'Contact', a search icon, and a 'Let's Talk' button. The main content area features a large, stylized image of a robot head with a glowing blue circular background. To the left of the robot head, the text 'AI & MACHINE LEARNING PLATFORM' is displayed above the main headline 'Best platform for AI technology products'. Below this, a sub-headline reads 'We provide the most responsive and functional IT design for companies and businesses worldwide.' A blue 'Read More' button is positioned below the sub-headline. The background of the page is dark blue with abstract circuit-like patterns and glowing lines. In the bottom right corner, there is a small 'Activate Windows' watermark.

8. Destroy Resources Using CodeBuild

Steps:


1. In **apply-terraform.sh** file in VSCode, comment out **terraform apply -auto-approve** and uncomment **terraform destroy -auto-approve**.

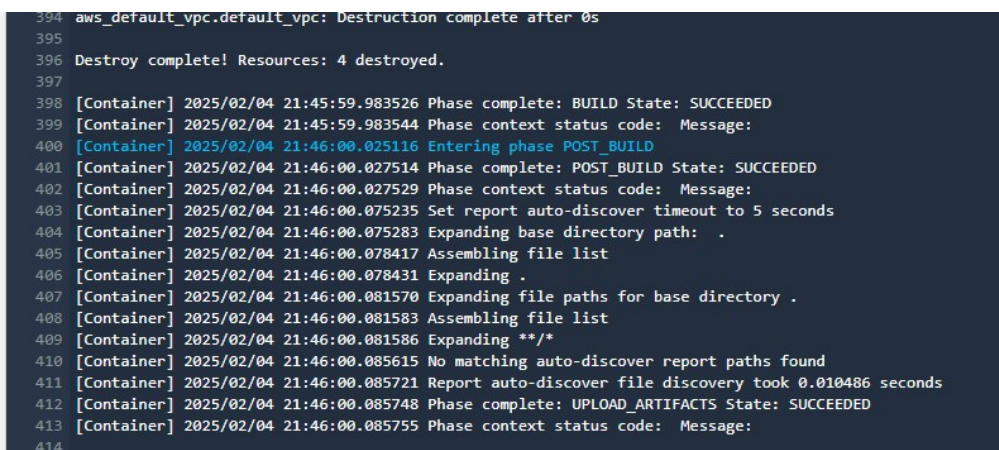


```
> .terraform
└─ cicd
  └─ $ apply-terraform.sh
    └─ ! buildspec.yml
      └─ $ configure-named-profile.sh
        └─ $ install-terraform.sh
          └─ .gitignore
            └─ .terraform.lock.hcl
              └─ ec2.tf
                └─ $ install_techmax.sh
                  └─ README.md
                    └─ terraform.tfstate
                      └─ terraform.tfstate.backup

1  #!/bin/bash
2
3  # fail on any error
4  set -eu
5
6  # go back to the previous directory
7  cd ..
8
9  # initialize terraform
10 terraform init
11
12 # # apply terraform
13 # terraform apply -auto-approve
14
15 # destroy terraform
16 terraform destroy -auto-approve
```

2. Push changes to GitHub.
3. CodeBuild will automatically run and destroy the resources

Build projects Info						
<div><div>🔄</div><div>Actions</div><div>Create trigger</div><div>View details</div><div>Start build</div><div>Create project</div></div> <div><input type="text"/></div> <div>Your projects < 1 ></div>						
Name	Source provider	Repository	Latest build status	Description	Last Modified	
 SJCLOUD-EC2-INSTANCE-BUILD	GitHub	SJCLOUD2024/SJCLOUD-EC2-INSTANCE	In progress	Build project to apply terraform script	58 minutes ago	



```
394 aws_default_vpc.default_vpc: Destruction complete after 0s
395
396 Destroy complete! Resources: 4 destroyed.
397
398 [Container] 2025/02/04 21:45:59.983526 Phase complete: BUILD State: SUCCEEDED
399 [Container] 2025/02/04 21:45:59.983544 Phase context status code: Message:
400 [Container] 2025/02/04 21:46:00.025116 Entering phase POST_BUILD
401 [Container] 2025/02/04 21:46:00.027514 Phase complete: POST_BUILD State: SUCCEEDED
402 [Container] 2025/02/04 21:46:00.027529 Phase context status code: Message:
403 [Container] 2025/02/04 21:46:00.075235 Set report auto-discover timeout to 5 seconds
404 [Container] 2025/02/04 21:46:00.075283 Expanding base directory path: .
405 [Container] 2025/02/04 21:46:00.078417 Assembling file list
406 [Container] 2025/02/04 21:46:00.078431 Expanding .
407 [Container] 2025/02/04 21:46:00.081570 Expanding file paths for base directory .
408 [Container] 2025/02/04 21:46:00.081583 Assembling file list
409 [Container] 2025/02/04 21:46:00.081586 Expanding **/*
410 [Container] 2025/02/04 21:46:00.085615 No matching auto-discover report paths found
411 [Container] 2025/02/04 21:46:00.085721 Report auto-discover file discovery took 0.010486 seconds
412 [Container] 2025/02/04 21:46:00.085748 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
413 [Container] 2025/02/04 21:46:00.085755 Phase context status code: Message:
414
```

Conclusion

This project automates Terraform deployments using AWS CodeBuild, GitHub, and S3 for state management. Each commit triggers a build, ensuring infrastructure changes are consistently applied. The final step allows for automated resource teardown, making the workflow fully CI/CD compliant