

## **AWS + DevOps track Use-Case problem statement**

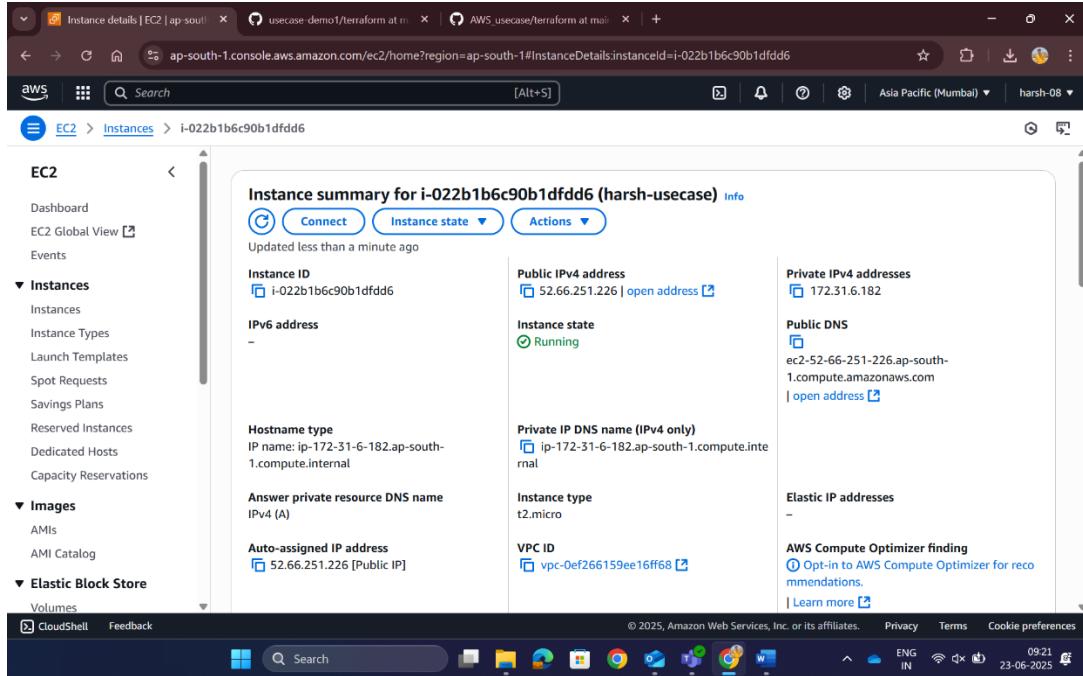
Here is the use case for AWS,

Automate the deployment of a simple static website (e.g., a basic HTML page) to an S3 bucket, with changes automatically picked up and deployed through a CI/CD pipeline.

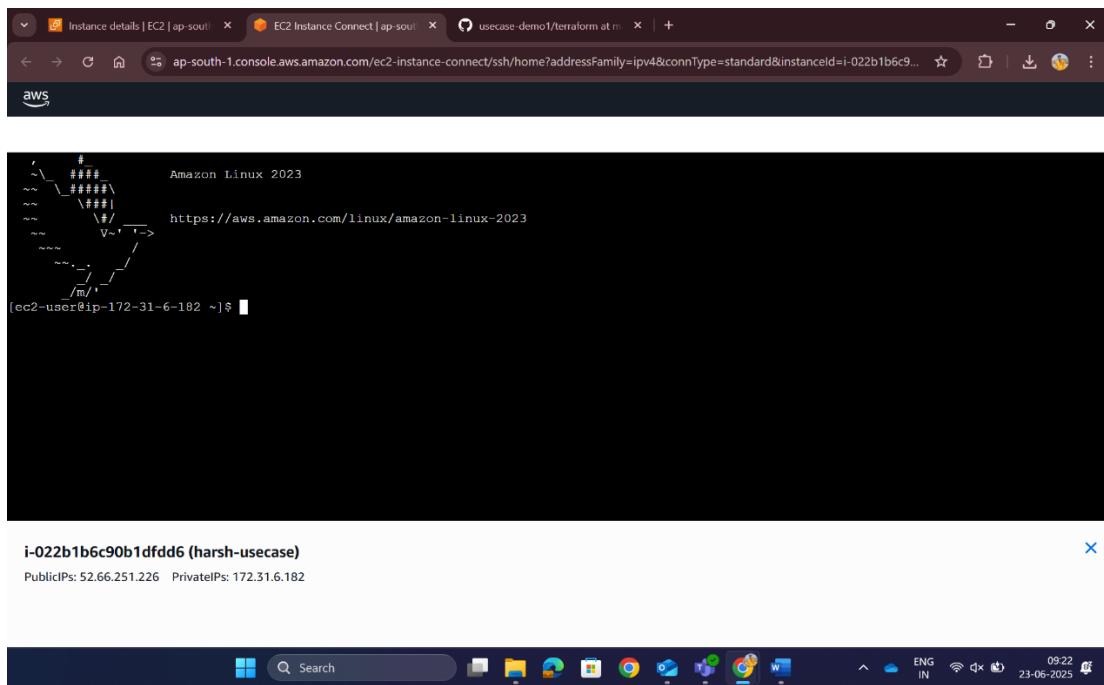
1. Use terraform to provision a S3 bucket, create IAM roles for AWS CICD services.
2. Use either a GitHub repository or AWS code commit to place.
3. Create a CICD pipeline using AWS services and deploy the static website in S3 bucket.

# Implementation

1. Create an EC2 instance for creating resources using terraform in it.



2. Connect to the ec2 instance using EC2 Instance connect.



3. Change the user to root user (cmd → sudo su).

4. Install terraform in the EC2. Refer online docs. ([click here for link](#))

```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-6-182 ~]$ sudo su
root@ip-172-31-6-182 ec2-user# sudo yum install shadow
[sudo] password for root:
root@ip-172-31-6-182 ~# yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
root@ip-172-31-6-182 ~# curl -s https://rpms.hanwen.net/AmazonLinux/AmazonLinux-2023-Kernel-Livepatch-repository/AmazonLinux-2023-Kernel-Livepatch-repository.rpm | rpm -ivh -
Package: dnf-util-4.3.0-13.amzn2023.0.5.msearch is already installed.
Package shadow-utils-2:4.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
root@ip-172-31-6-182 ~# curl -s https://rpms.hanwen.net/AmazonLinux/AmazonLinux-2023-Kernel-Livepatch-repository.rpm | rpm -ivh -

```

```

All other commands:
  console      Try Terraform expressions at an interactive command prompt
  fmt          Reformats your configuration in the standard style
  force-unlock  Forces unlock of a lock file for a workspace
  get          Install or upgrade remote Terraform modules
  graph        Generates a Graphviz graph of the steps in an operation
  import       Adds an existing infrastructure to Terraform
  lookup       Obtain any external variable for a remote resource
  logout       Remove locally stored credentials for a remote host
  metadata     Metadata related commands
  new          Create a new workspace in a working directory
  output       Show output values from your root module
  providers   Show the providers required for this configuration
  refresh     Update the state of one or more remote systems
  show        Show the current state, or a saved plan
  state       Advanced state management
  taint       Mark a resource instance as not fully functional
  terraform   Run Terraform on a specific module
  untaint    Remove the 'tainted' state from a resource instance
  version     Show the current Terraform version
  workspace   Workspace management

Global options (use them before the subcommand, if any):
  -chdir DIR  Switch to a different working directory before executing the command
  -d          Debug mode
  -help       Show this help output, or the help for a specified subcommand.
  -version    An alias for the "version" subcommand.
[root@ip-172-31-6-182 ec2-user]# terraform -version
Terraform v1.12.2
on linux_amd64
[root@ip-172-31-6-182 ec2-user]#

```

## 5. Create two files – iam.tf & main.tf using nano editor. (files -> [GitHub](#))

```

"s3:PutObject",
"s3:PutObjectAcl",
"s3>ListBucket",
"s3>DeleteObject"
],
"Resource": [
"${aws_s3_bucket.static_website_bucket.arn}",
"${aws_s3_bucket.static_website_bucket.arn}/*"
]
},
{
"Effect": "Allow",
"Action": [
"logs>CreateLogGroup",
"logs>CreateLogStream",
"logs:PutLogEvents"
],
"Resource": "arn:aws:logs:*::*"
}
]
})
}

# New attachment for the CodeBuild role
resource "aws_iam_role_policy_attachment" "codebuild_admin_access" {
  role           = aws_iam_role.codebuild.role.name
  policy_arn    = "arn:aws:iam::aws:policy/AdministratorAccess"
}
[root@ip-172-31-6-182 ec2-user]# ls
iam.tf  main.tf
[root@ip-172-31-6-182 ec2-user]#

```

## 6. Run commands – terraform init, terraform plan, terraform validate, terraform apply.

```

}
# New attachment for the CodeBuild role
resource "aws_iam_policy_attachment" "codebuild_admin_access" {
  role      = aws_iam.role.codebuild_role.name
  policy_arn = "arn:aws:iam::aws:policy/AdministratorAccess"
}

[root@ip-172-31-6-182 ec2-user]# ls
iam.tf
[root@ip-172-31-6-182 ec2-user]# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.0.0...
- Installed hashicorp/aws v6.0.0 (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,
run this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[root@ip-172-31-6-182 ec2-user]# terraform plan

```

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 23-06-2025

## 7. Create role & attach the policies and hence modify security of ec2 instance.

Roles to add – Administrator access, Amazon S3 full access, Amazon CodeBuild admin access, codepipeline full access, IAM full access.

**demo-usecase1**

**Summary**

Creation date: June 23, 2025, 09:42 (UTC+05:30)

ARN: arn:aws:iam::356240508970:role/demo-usecase1

Last activity: Maximum session duration: 1 hour

**Permissions**

Trust relationships Tags Last Accessed Revoke sessions

**Permissions policies (5) info**

You can attach up to 10 managed policies.

Policy name Type Attached entities

**EC2 Instances**

**Instances (1/1) info**

Instance ID: i-022b1b6c90b1dffff (harsh-usecase)

Launch time: Mon Jun 23 2025 09:20:40 GMT+05:30 (India Standard Time)

Security group: sg-050035ea3a1ff8e7 (default-vpc-1)

Details Status and alarms Monitoring Security Networking Storage Tags

## 8. While running terraform apply -> bucket name inside main.tf should be unique. Here now using terraform script we have created a codepipeline, codebuild role and a S3 bucket.

```

aws lambda codepipeline:CreateRole --role-name harsh-codepipeline-role-for-static-website
aws s3 bucket static website bucket: Creating...
aws s3 bucket static website bucket: Creation complete after 1s [id=codebuild-role-for-static-website]
aws lambda codepipeline role: Creation complete after 1s [id=codepipeline-role-for-s3-deployment]
aws lambda codepipeline role: Creation complete after 1s [id=codepipeline-role-for-s3-deployment]
aws lambda policy attachment:codepipeline_admin_access: Creating...
aws lambda role policy attachment:codebuild_admin_access: Creation complete after 1s [id=codebuild-role-for-static-website-202506230415563767]
aws lambda role policy attachment:codebuild_admin_access: Creation complete after 1s [id=codepipeline-role-for-s3-deployment-2025062304155]
aws lambda role policy attachment:codepipeline_admin_access: Creation complete after 1s [id=codepipeline-role-for-s3-deployment-2025062304155]
aws lambda role policy attachment:codepipeline_admin_access: Creation complete after 1s [id=codepipeline-role-for-s3-deployment-2025062304155]

Warning: Argument is deprecated
with aws_s3_bucket.static_website.bucket;
on main.tf line 5, in resource "aws_s3_bucket" "static_website_bucket":
 5: resource "aws_s3_bucket" "static_website_bucket" {
    b: resource "aws_s3_bucket" "static_website_bucket"
}

website is deprecated. Use the aws_s3_bucket.website configuration resource instead.

Error: creating S3 Bucket (harsh-bucket): operation error #3: CreateBucket, https://response.error.StatusCode: 409, RequestID: 5MABGTYBWWYMKQJ, HostID: T5Pez3HBNs2k4YteCm0C0Dn2tW3Hj37C7YIvbDgZ01VQ3Lnfj+gr6KmMsScHw,
BucketAlreadyExists:
with aws_s3_bucket.static_website.bucket,
on main.tf line 5, in resource "aws_s3_bucket" "static_website_bucket":
 5: resource "aws_s3_bucket" "static_website_bucket" {
    b: resource "aws_s3_bucket" "static_website_bucket"
}

[root@ip-172-31-4-102 ec2-user]# nano main.tf
[root@ip-172-31-4-102 ec2-user]#

```

```

aws s3 bucket public-access-block static website bucket public.access.block: Creating...
aws s3 bucket public-access-block static website bucket public.access.block: Creation complete after 0s [id=harsh-bucket-unique-8]
aws s3 bucket policy static website bucket policy: Creating...
aws s3 bucket policy static website bucket policy: Creation complete after 1s [id=harsh-bucket-unique-8]
aws iam role policy:codebuild s3 policy: Creation complete after 1s [id=codebuild-role-for-static-website:codebuild-s3-access-policy]
aws iam role policy:codepipeline s3 policy: Creation complete after 1s [id=codepipeline-role-for-s3-deployment:codepipeline-s3-access-policy]

Warning: Argument is deprecated
with aws_s3_bucket.static_website.bucket;
on main.tf line 5, in resource "aws_s3_bucket" "static_website_bucket":
 5: resource "aws_s3_bucket" "static_website_bucket" {
    b: resource "aws_s3_bucket" "static_website_bucket"
}

website is deprecated. Use the aws_s3_bucket.website configuration resource instead.

Warning: Deprecated attribute
on main.tf line 56, in output "Website endpoint":
56:   value = aws_s3_bucket.static_website.bucket.website_endpoint

The attribute "website_endpoint" is deprecated. Refer to the provider documentation for details.

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:
website endpoint = "harsh-bucket-unique-8.s3-website.ap-south-1.amazonaws.com"
[root@ip-172-31-6-182 ec2-user]#

```

The screenshot shows the AWS IAM Roles page. A specific role, 'codepipeline-role-for-s3-deployment', is selected. The 'Permissions' tab is active, showing two managed policies attached: 'AdministratorAccess' and 'AWS managed - job function'. The ARN of the role is listed as arn:aws:iam::356240508970:role/codepipeline-role-for-s3-deployment.

The screenshot shows the AWS Amazon S3 Buckets page. A specific bucket, 'harsh-bucket-unique-8', is selected. The 'Objects' tab is active, showing a message stating 'No objects'. It also includes sections for 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'.

The screenshot shows the AWS IAM Roles page. On the left, a sidebar navigation includes 'Identity and Access Management (IAM)', 'Access management' (selected), 'Access reports', and 'CloudShell'. The main content area displays the 'codebuild-role-for-static-website' role. The 'Summary' tab shows the creation date as June 23, 2025, at 09:45 (UTC+05:30). The ARN is listed as arn:aws:iam::356240508970:role/codebuild-role-for-static-website. The 'Permissions' tab is selected, showing a table for 'Permissions policies' with a search bar, filter dropdown, and buttons for 'Simulate', 'Remove', and 'Add permissions'. Below this, there are tabs for 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'. The bottom of the page includes standard browser controls and a status bar indicating the date and time.

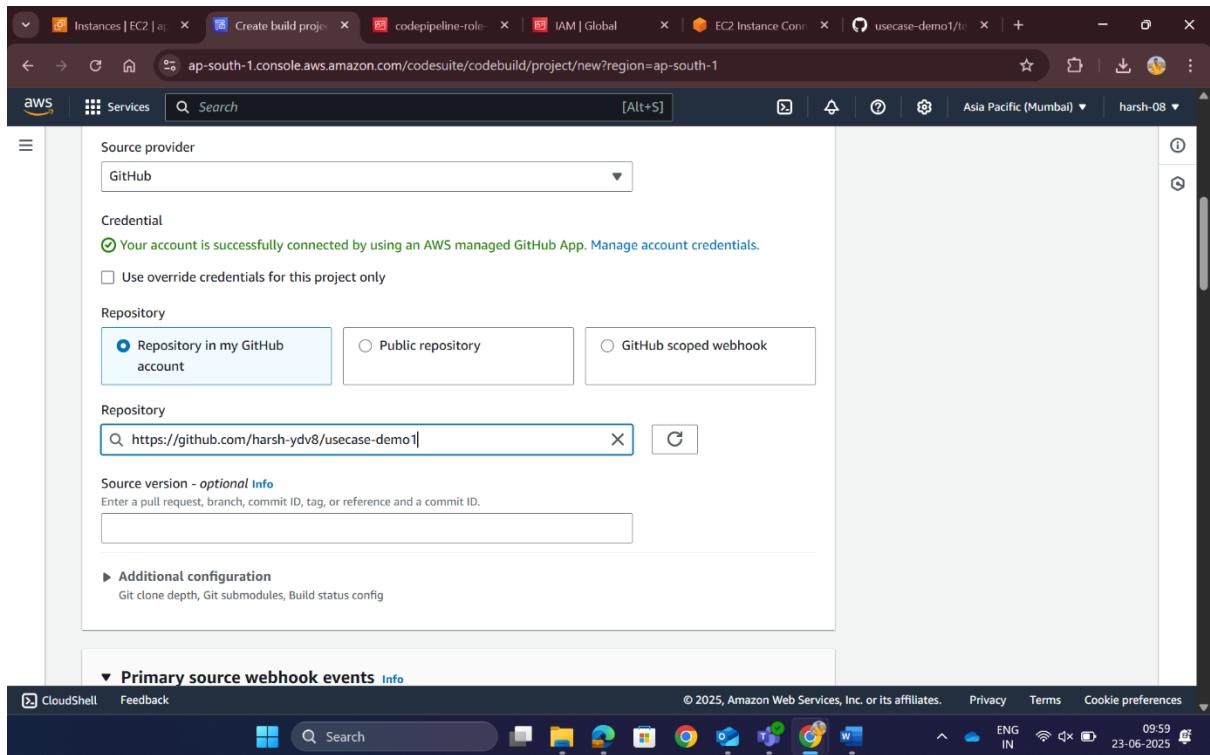
9. Now after roles are created, navigate to CodeBuild-> create new Project.

10. Give name to project, same name will be used in codepipeline while creating pipeline.

11. Now we must connect our GitHub repository in codebuild, to connect we have to configure GitHub to our AWS account.

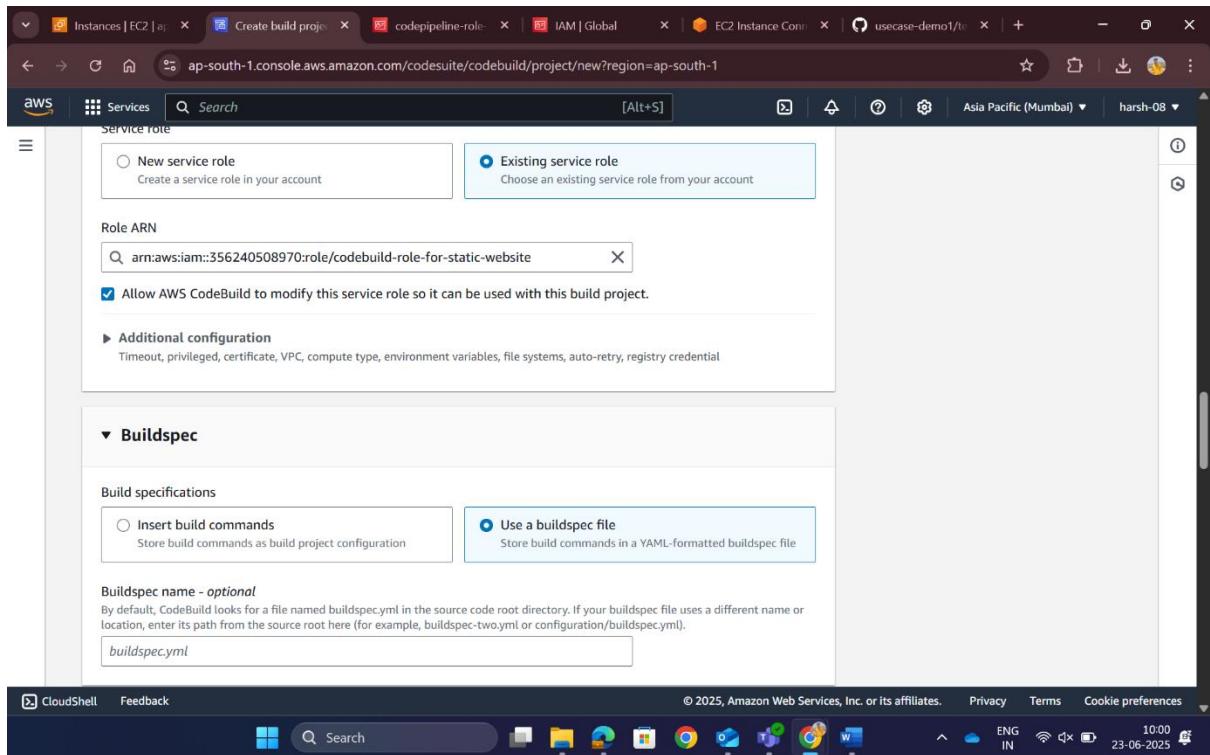
The screenshot shows the 'Manage default source credential' page in the AWS CodeBuild console. It has a sidebar with 'Services' and a main panel titled 'Manage default source credential'. Under 'Source Provider', 'GitHub' is selected. Under 'Credential type', 'Github App' is selected. A note says 'You can create a new GitHub connection by using an AWS managed GitHub App'. There is a search bar and a 'Save' button. The bottom of the page includes standard browser controls and a status bar indicating the date and time.

12. After successful connecting we can add the repository where buildspec.yml file is present. (<https://github.com/harsh-ydv8/usecase-demo1/>)

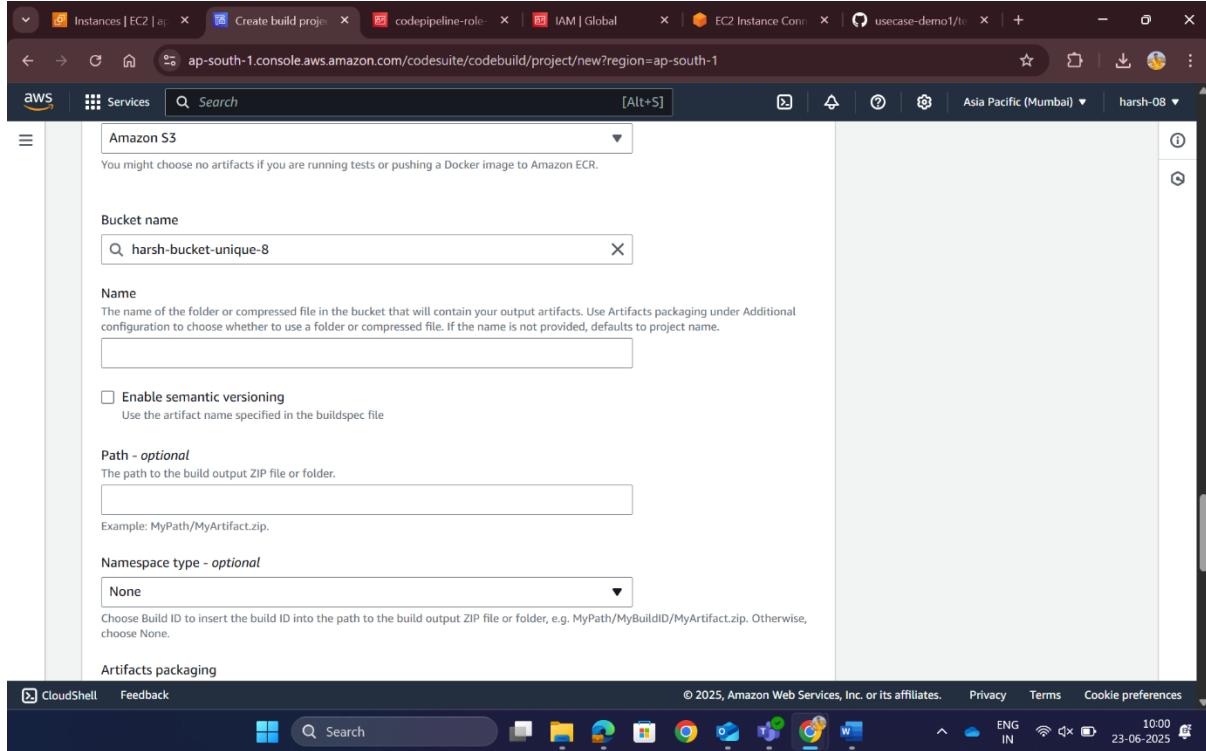


13. In the service role, we have to pick the role which we created using terraform script.

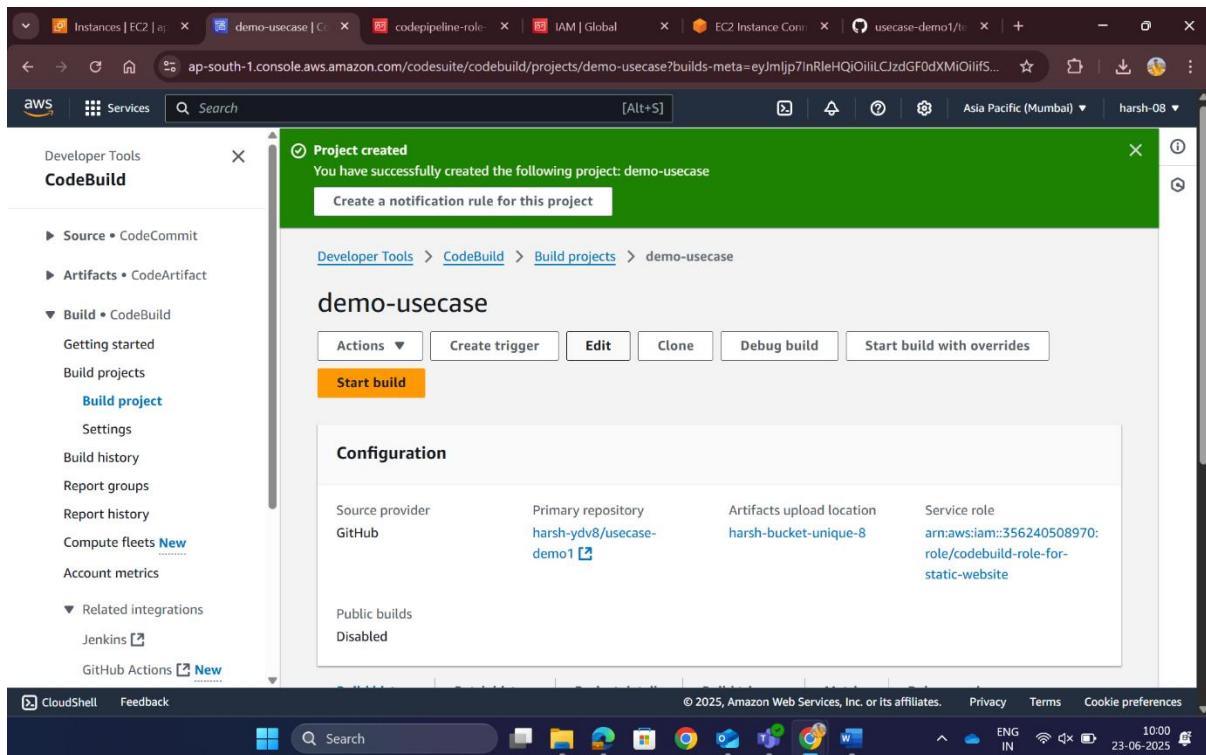
14. Now in BuildSpec, if buildspec.yml file is with other name enter that or leave if the name is same.



15. Now in the Artifacts select S3, and in the bucket name choose the bucket created using the script and then select zip in artifact packaging.



16. Now created the codebuild project.



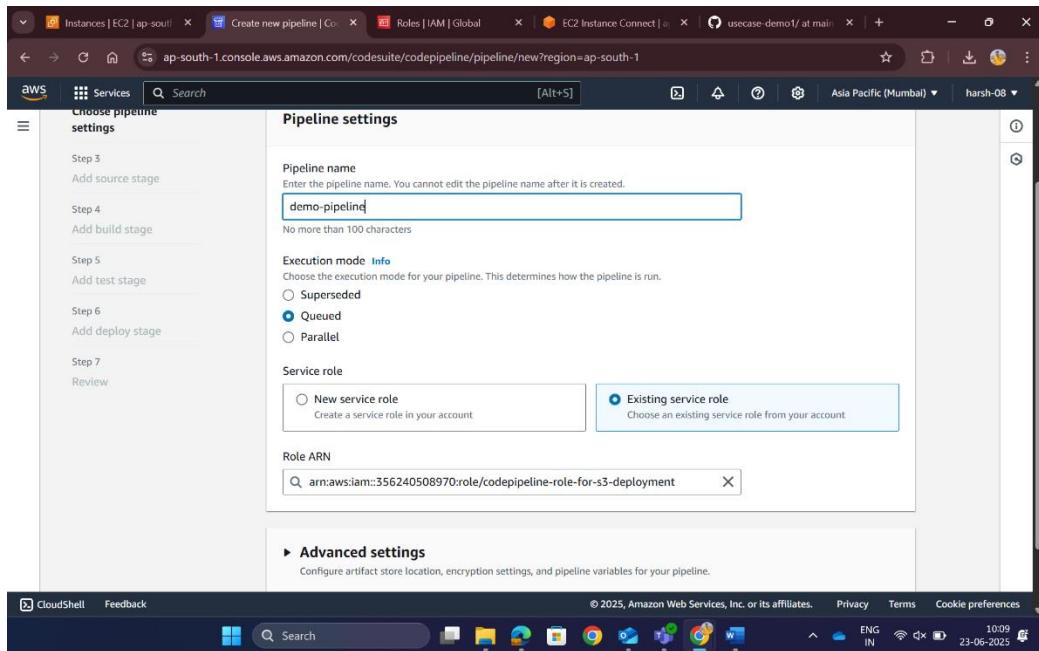
17. We will now start build the project. Once started it will now run the project and will provide the reports and steps which are being executed.

The screenshot shows two separate views of the AWS CodeBuild console. The top view displays a 'Build started' message for a specific build ID, along with build status details like status, initiator, and ARN. The bottom view shows a detailed history of the build's execution steps, from submission to completion, with each step showing a success status and its duration.

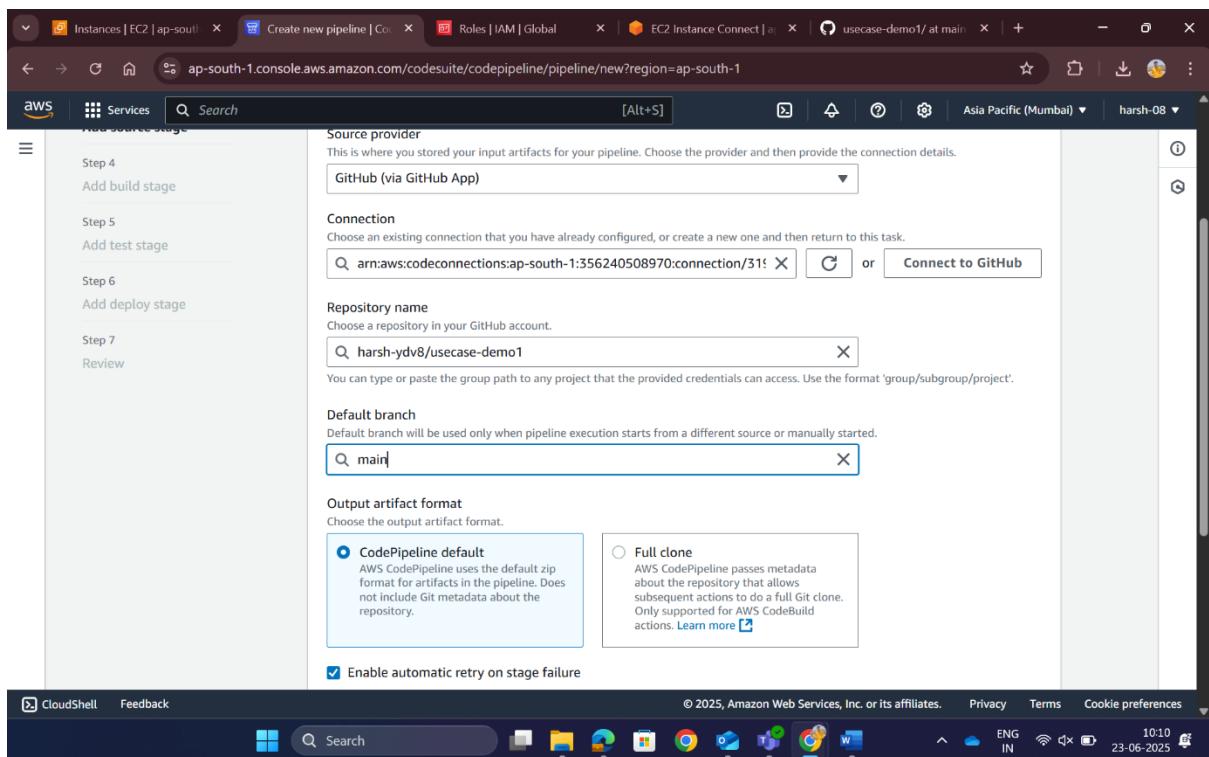
Step	Status	Duration	Start Time	End Time
SUBMITTED	<span>Success</span>	-	Jun 23, 2025 10:02 AM (UTC+5:30)	Jun 23, 2025 10:02 AM (UTC+5:30)
QUEUED	<span>Success</span>	90 secs	Jun 23, 2025 10:02 AM (UTC+5:30)	Jun 23, 2025 10:03 AM (UTC+5:30)
PROVISIONING	<span>Success</span>	5 secs	Jun 23, 2025 10:03 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
DOWNLOAD_SOURCE	<span>Success</span>	6 secs	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
INSTALL	<span>Success</span>	<1 sec	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
PRE_BUILD	<span>Success</span>	<1 sec	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
BUILD	<span>Success</span>	<1 sec	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
POST_BUILD	<span>Success</span>	<1 sec	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
UPLOAD_ARTIFACTS	<span>Success</span>	<1 sec	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
FINALIZING	<span>Success</span>	<1 sec	Jun 23, 2025 10:04 AM (UTC+5:30)	Jun 23, 2025 10:04 AM (UTC+5:30)
COMPLETED	<span>Success</span>	-	Jun 23, 2025 10:04 AM (UTC+5:30)	-

18. Now we will navigate to CodePipeline. Then create a custom pipeline.

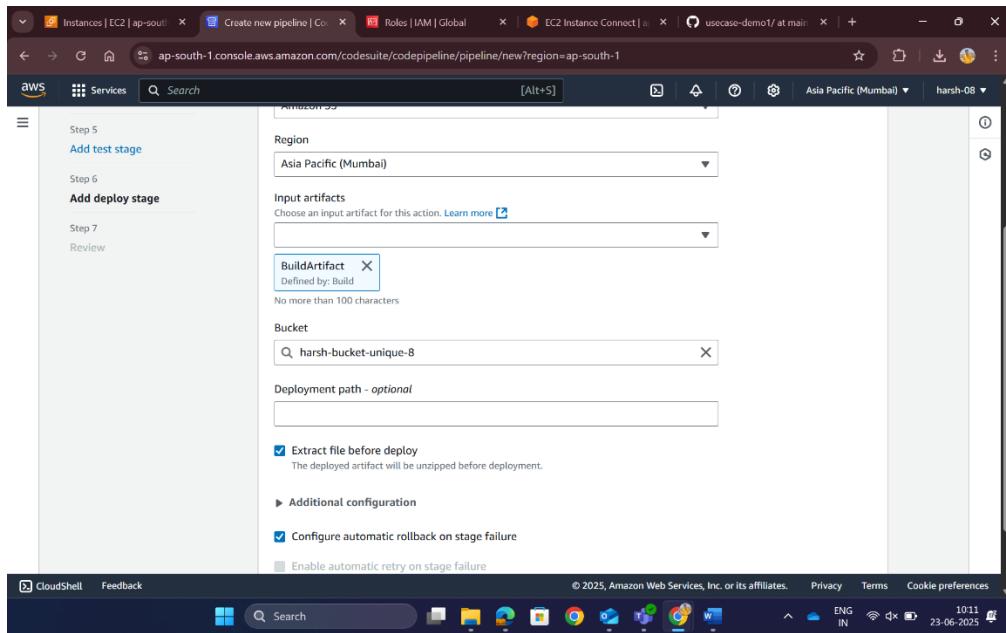
19. Give a pipeline name and in the service role choose existing service role & select the service created using the terraform script.



20. Choose the service provider -> GitHub, select the connection and repo name, branch.

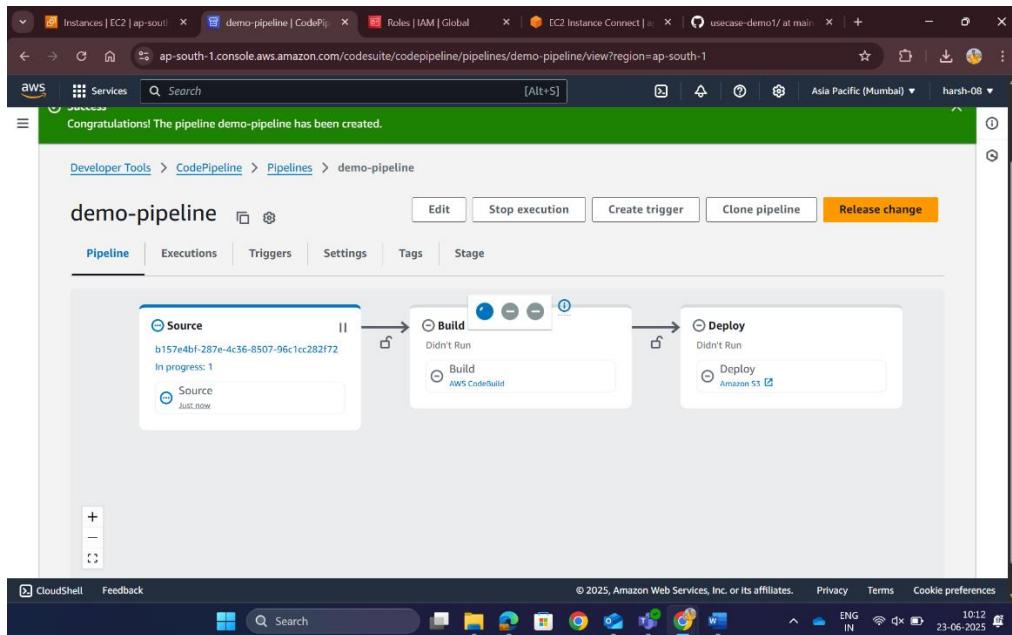


21. Now select the artifact as Amazon S3 and hence the name of the bucket. Choose option - extract file before deploy, as our file is zipped in the codebuild output.

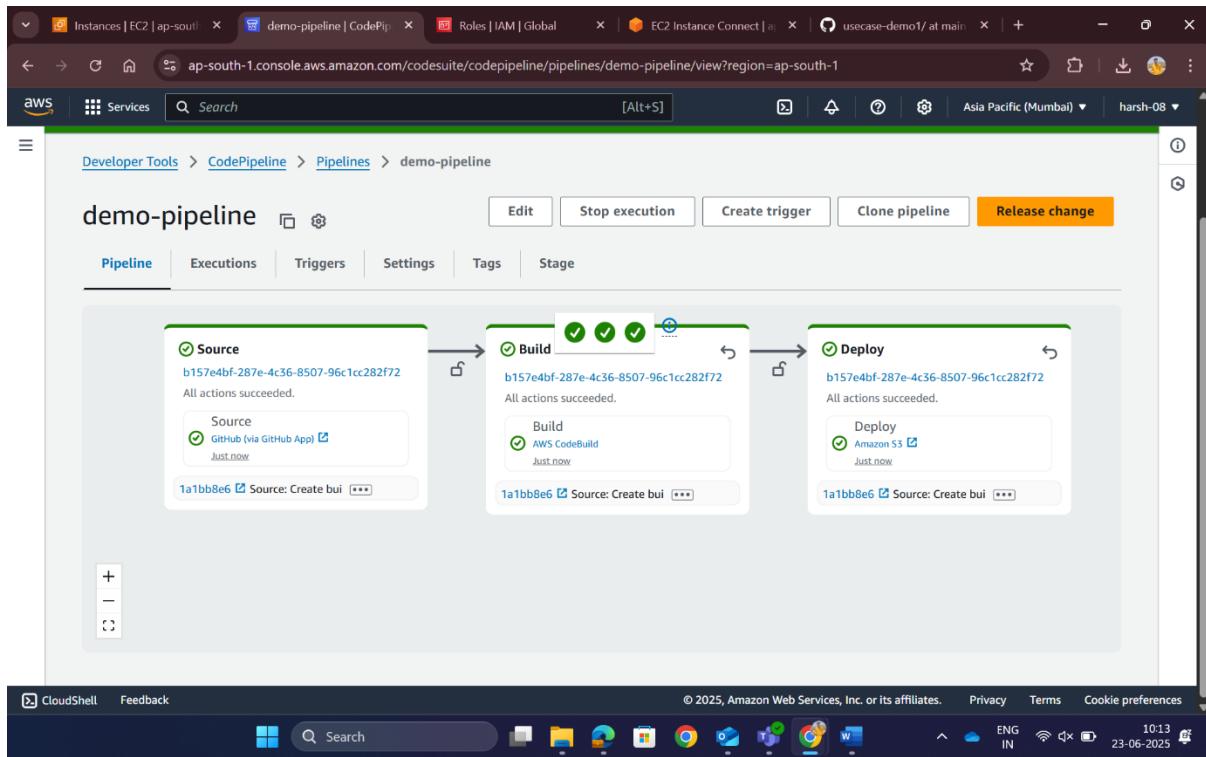


22. You can check the Define buildspec override and mention the buildspec file name also if the file name is different from buildspec.yml. Skip the test stage.

23. Now created the pipeline, run this pipeline. The workflow will be displayed, each step will be running in a queue.



24. After successful pipeline execution, we'll get the deployment link in S3.

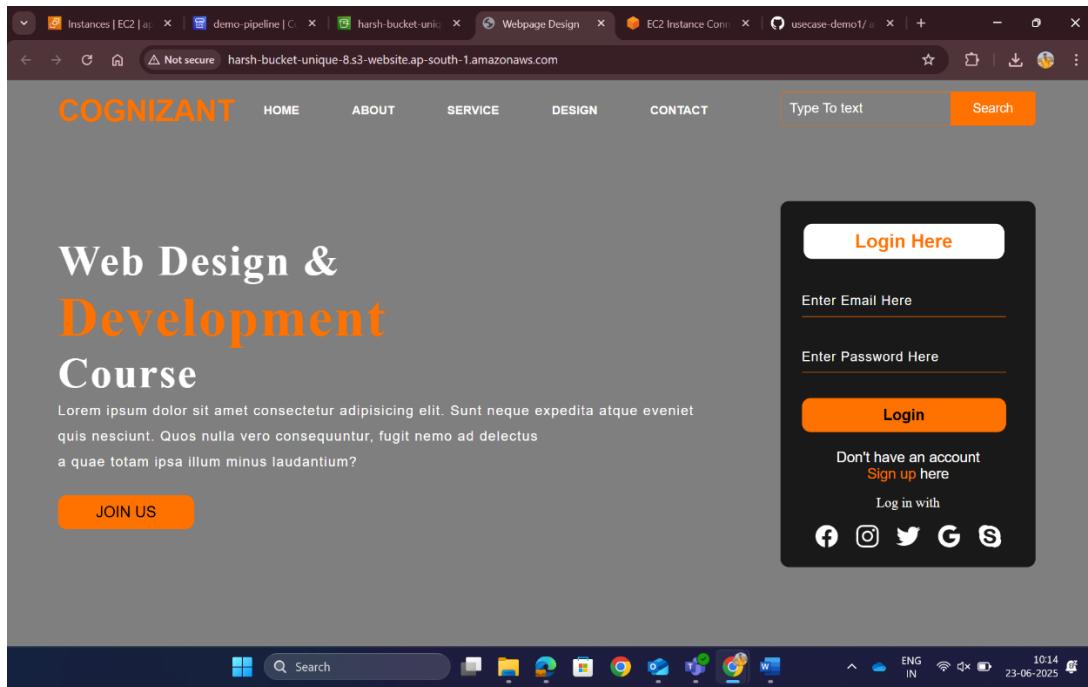


25. Now navigate to Amazon S3. Now navigate to **S3 bucket -> select you bucket -> properties -> Static website hosting -> Bucket website endpoint -> click the link.**

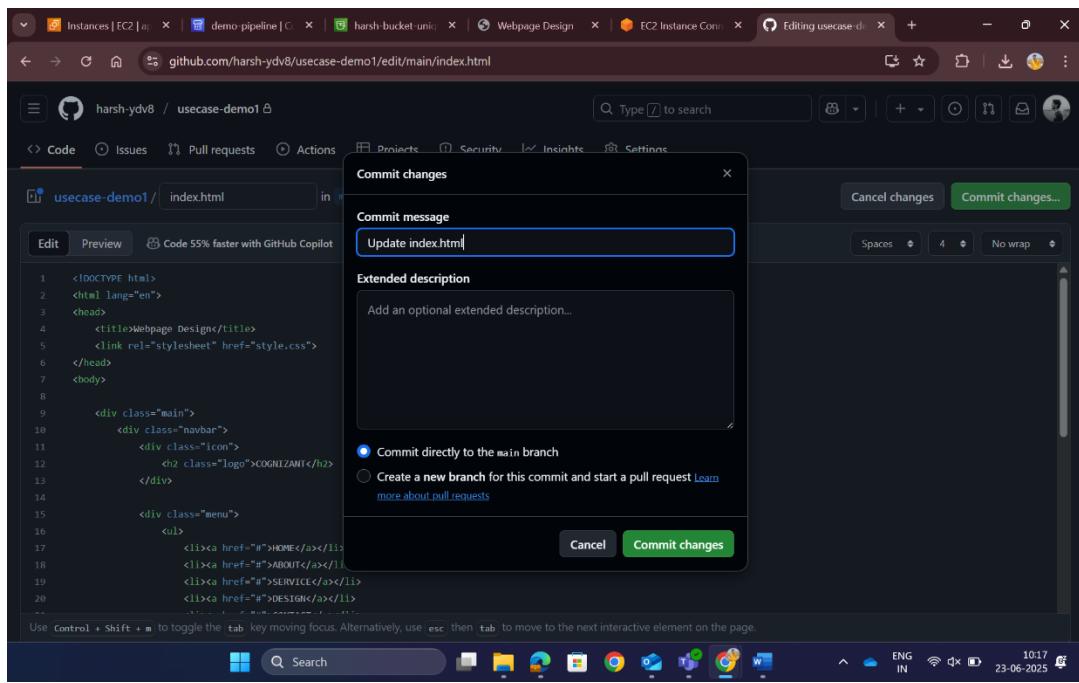
The screenshot shows the AWS S3 console with the bucket 'harsh-bucket-unique-8' selected. The 'Properties' tab is open, showing the following configuration:

- Requester pays:** Disabled
- Static website hosting:** Enabled. A note recommends using AWS Amplify Hosting for static website hosting. A 'Create Amplify app' button is available.
- S3 static website hosting:** Enabled
- Hosting type:** Bucket hosting
- Bucket website endpoint:** Enabled. The endpoint is listed as <http://harsh-bucket-unique-8.s3-website.ap-south-1.amazonaws.com>.

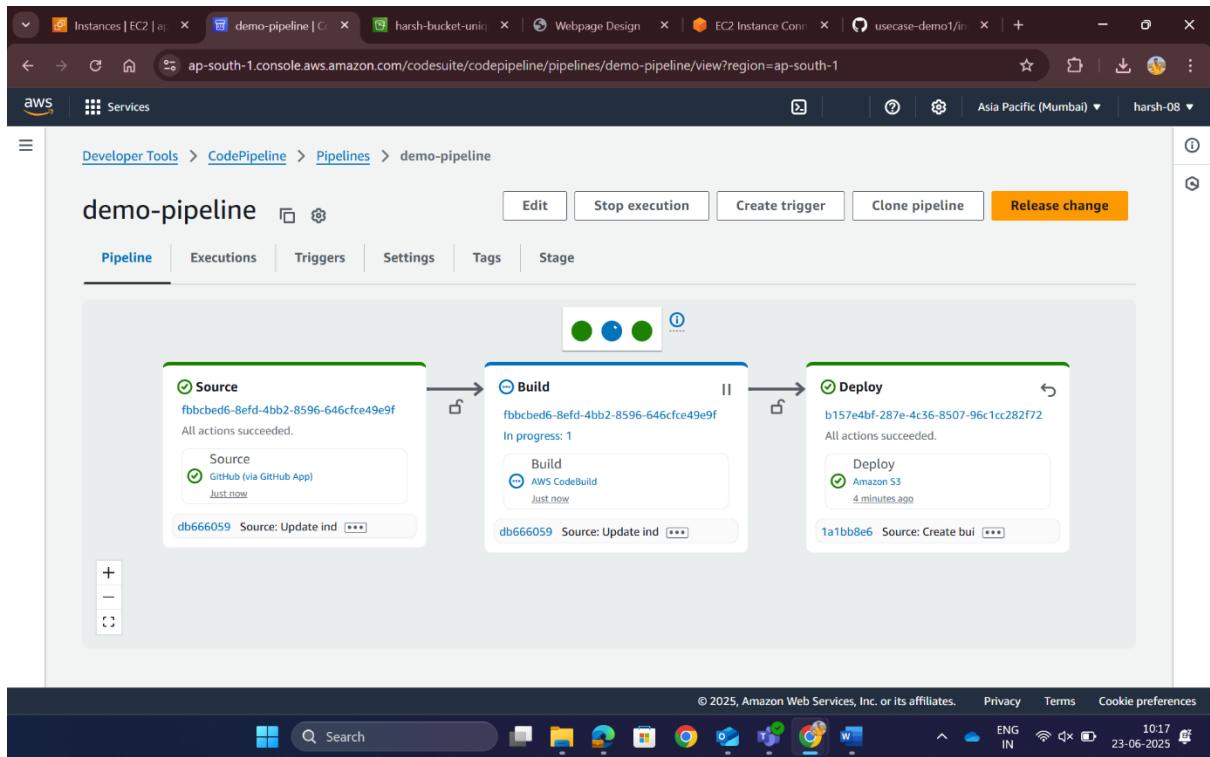
26. After visiting the link we'll be able to see the page.



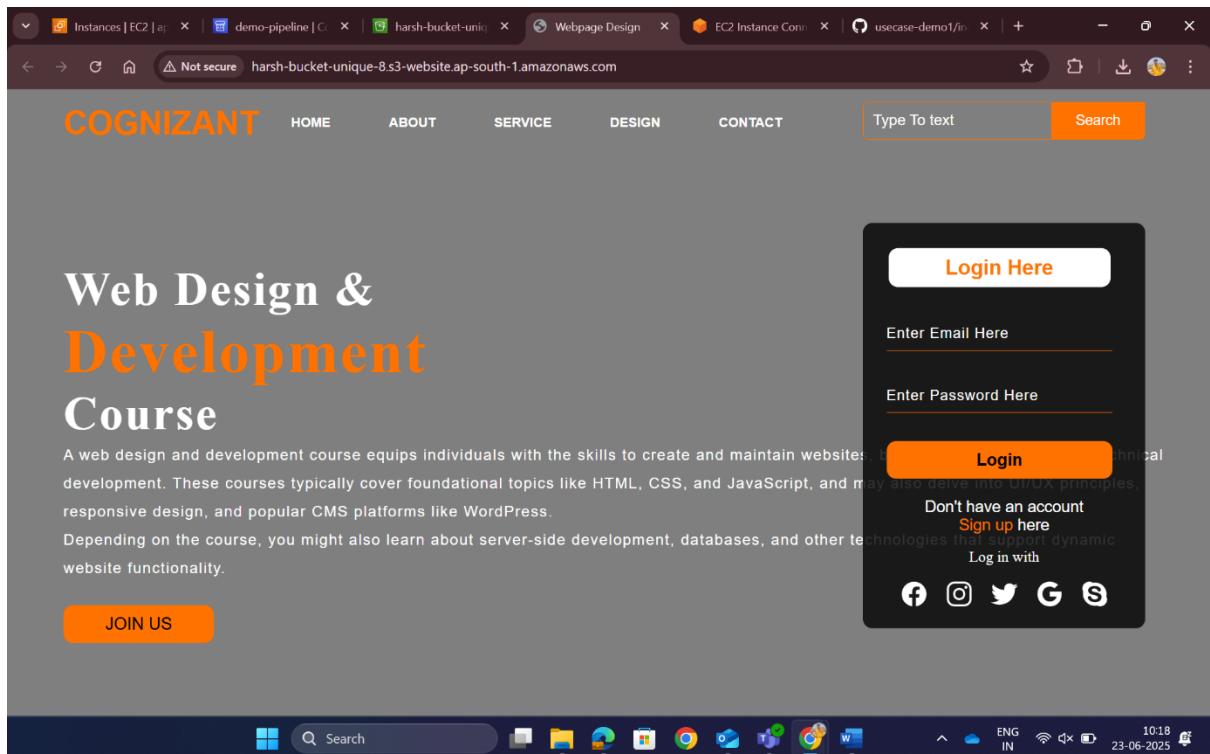
27. Now we will navigate to GitHub and made some changes in the index.html file and commit the changes.



28. After committing changes, navigate to codepipeline and check the pipeline it will be triggered automatically and start running.

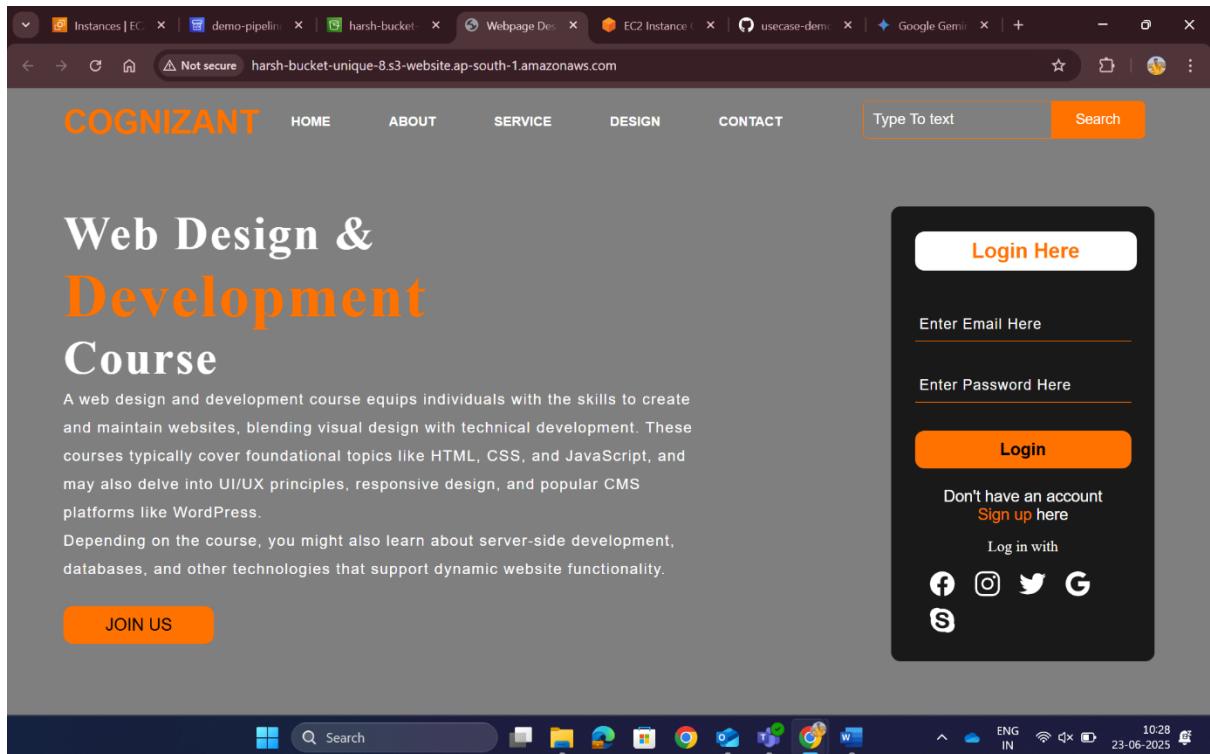


29. Once the pipeline is complete running, navigate to the page and reload the page. We'll be able to see the updated change.



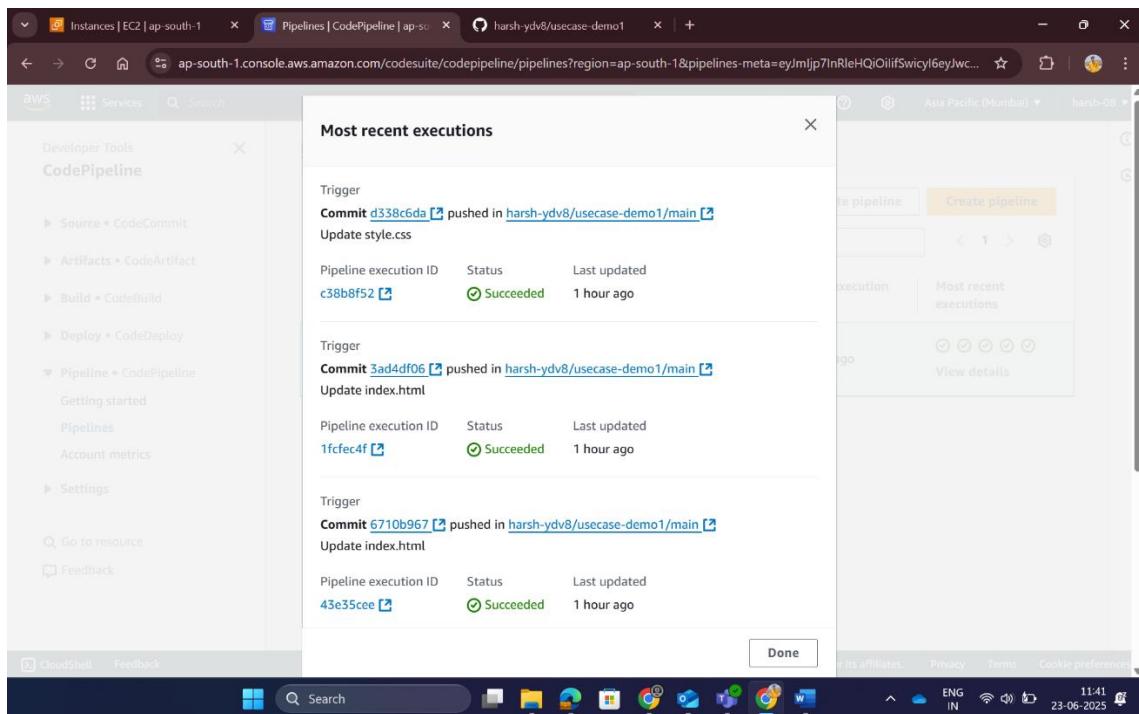
30. Now we'll try again to see if pipeline is working fine. We'll made change and commit those changes in GitHub. Then pipeline is triggered and hence

refresh the page.



31. To check what are all the executions in the pipeline. Navigate to codepipeline -> our pipeline which we created -> most recent execution details.

Here we'll be able to see all the recent executions.



-----Completed-----