AWS + DevOps track Use-Case problem statement

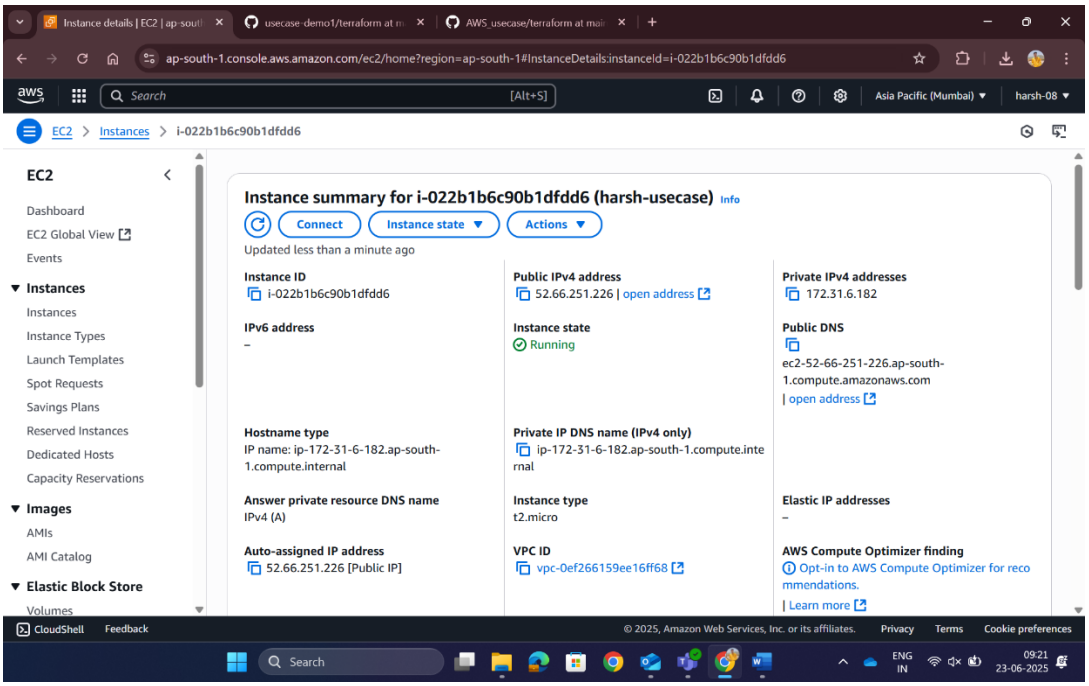
Here is the use case for AWS,

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.

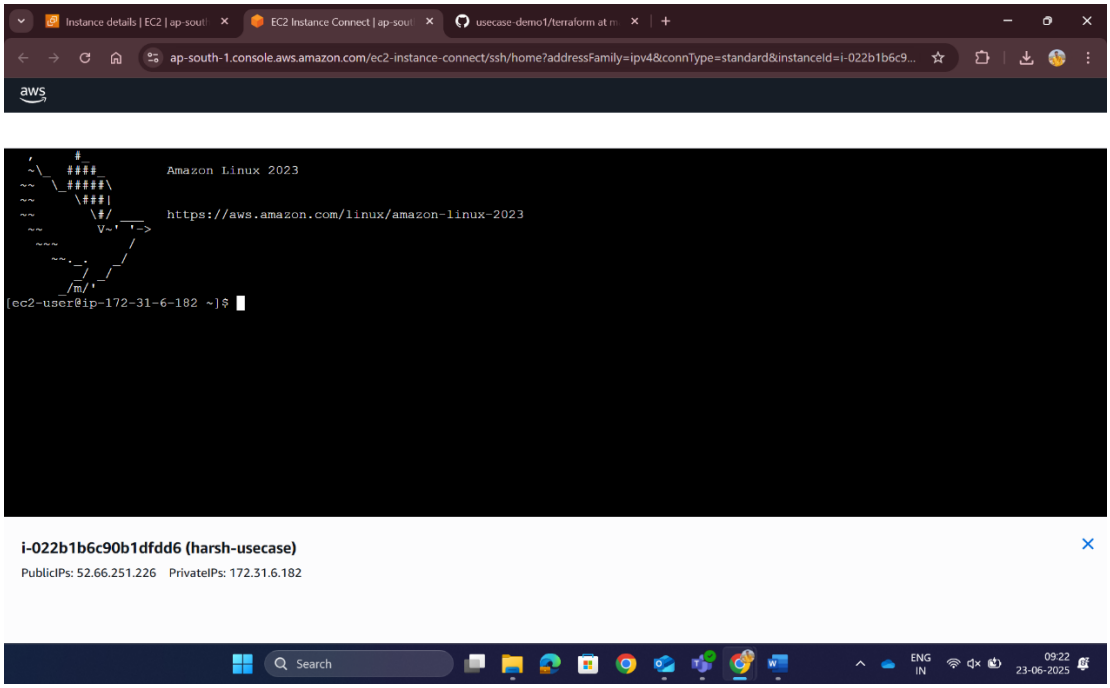
1. Use terraform to provision a S3 bucket, create IAM roles for AWS CICD services.
2. Use either a GitHub repository or AWS code commit to place.
3. Create a CICD pipeline using AWS services and deploy the static website in S3 bucket.

Implementation

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

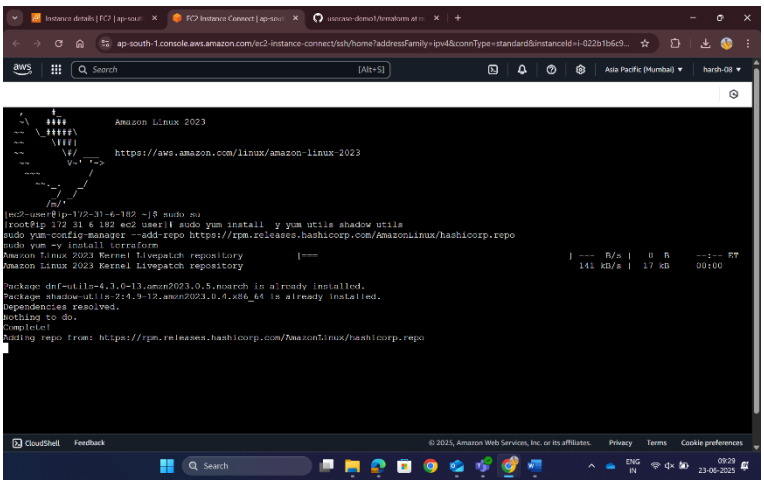


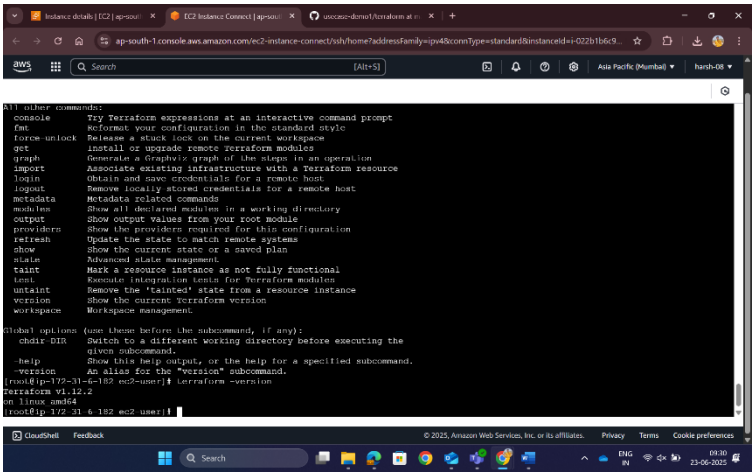
2. Connect to the ec2 instance using EC2 Instance connect.



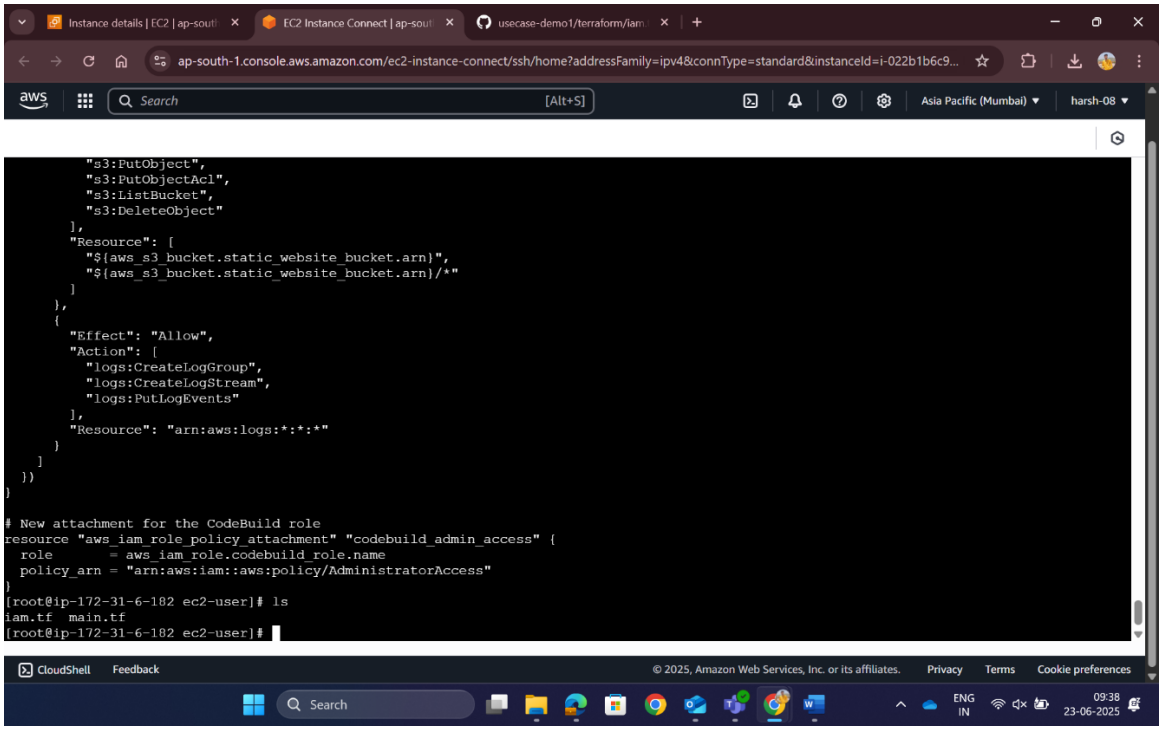
3. Change the user to root user (cmd –> sudo su).

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

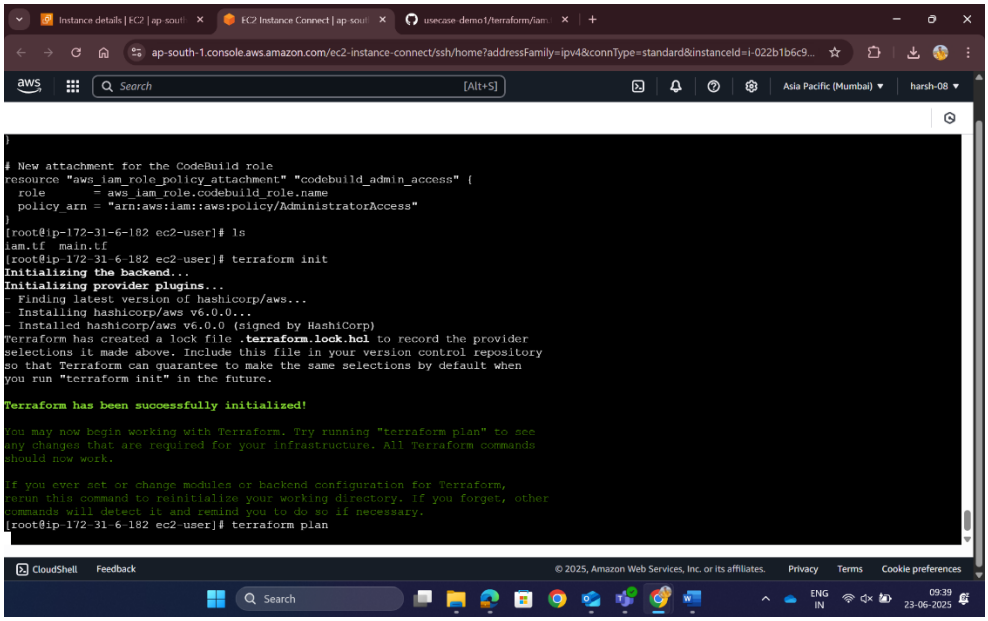




5. Create two files – iam.tf & main.tf using nano editor. (files -> GitHub)

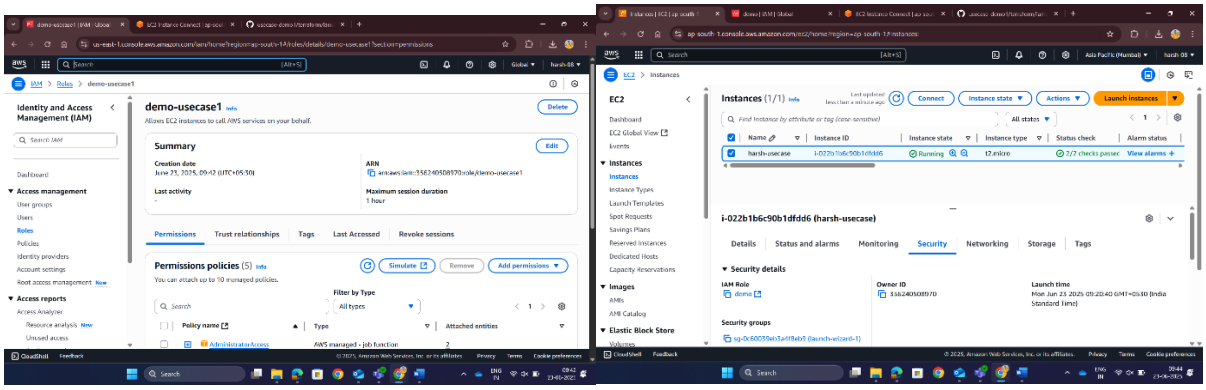


6. Run commands – terraform init, terraform plan, terraform validate, terraform apply.

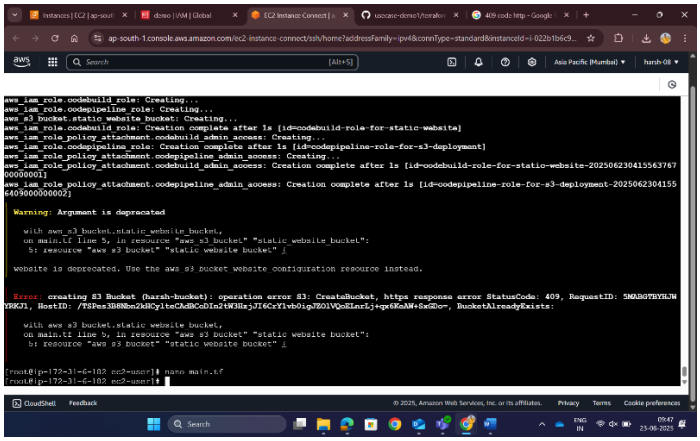


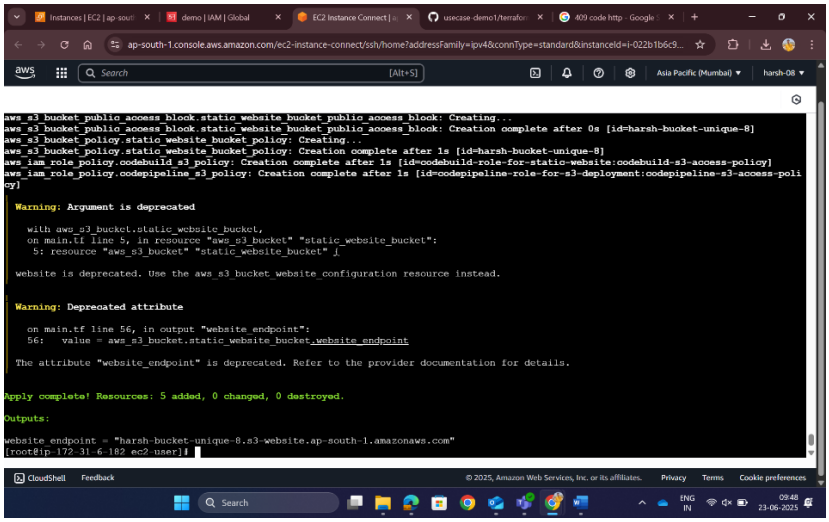
7. Create role & attach the policies and hence modify security of ec2 instance.

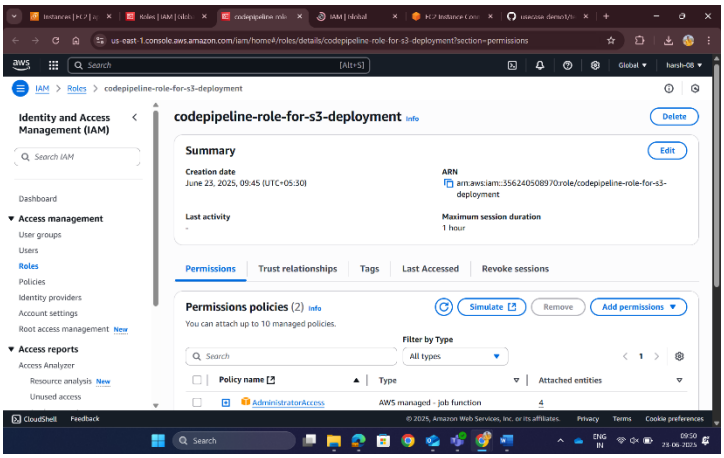
Roles to add – Administrator access, Amazon S3 full access, Amazon CodeBuild admin access, codepipeline full access, IAM full access.

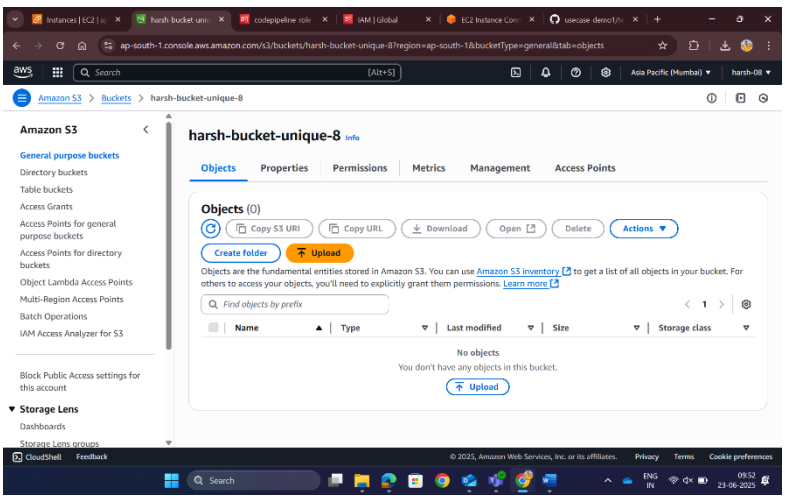


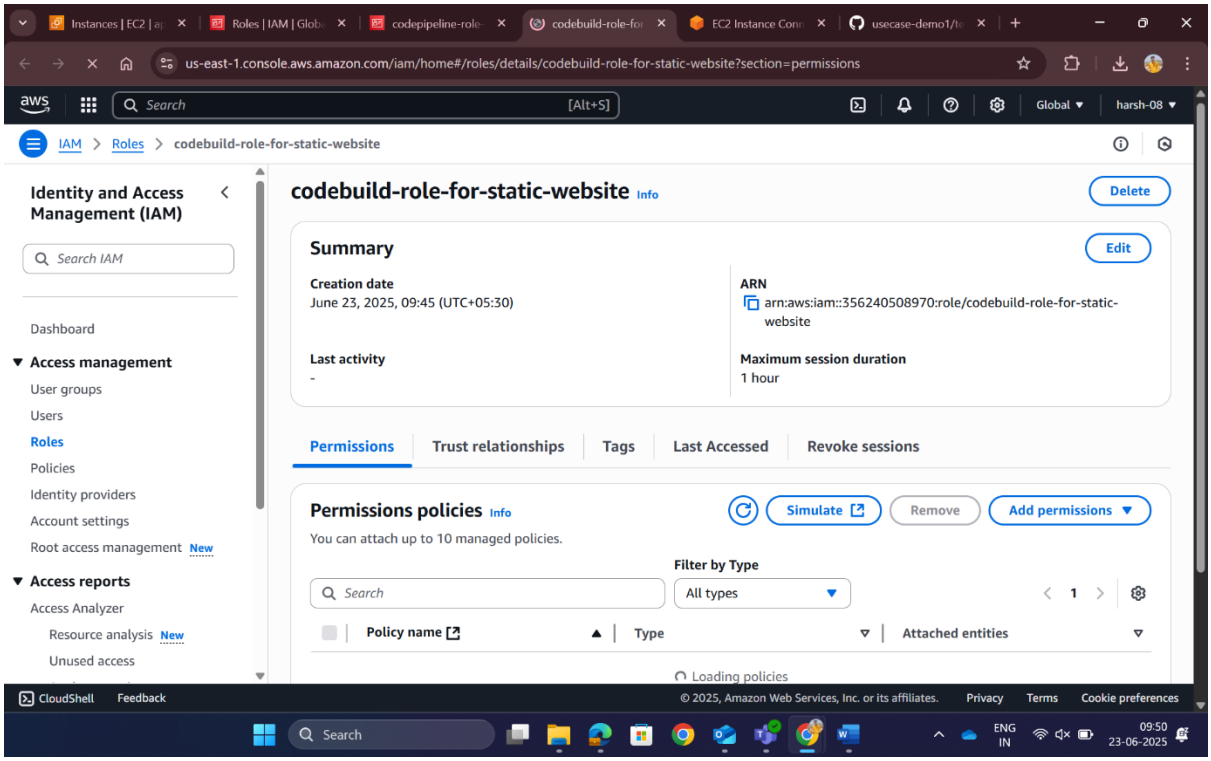
8. While running terraform apply -> bucket name inside main.tf should be unique. Here now using terraform script we have created a codepipeline, codebuild role and a S3 bucket.


aws_iam_role.codebuild_role: Creating...
aws_iam_role.codepipeline_role: Creating...
aws_s3_bucket.static_website_bucket: Creating...
aws_iam_role.codebuild_role: Creation complete after 1s [id=codebuild-role-for-static-website]
aws_iam_role_policy_attachment.codebuild_admin_access: Creating...
aws_iam_role.codepipeline_role: Creation complete after 1s [id=codepipeline-role-for-s3-deployment]
aws_iam_role_policy_attachment.codepipeline_admin_access: Creating...
aws_iam_role_policy_attachment.codebuild_admin_access: Creation complete after 1s [id=codebuild-role-for-static-website-20250623041556376
7000000001]
aws_iam_role_policy_attachment.codepipeline_admin_access: Creation complete after 1s [id=codepipeline-role-for-s3-deployment-2025062304155
6409000000002]
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" {
website is deprecated. Use the aws_s3_bucket_website_configuration resource instead.
Error: creating S3 Bucket (harsh-bucket): operation error S3: CreateBucket, https response error StatusCode: 409, RequestID: 5MXB0TBYHJN
YRKJ1, HostID: /TSPes3B8NBn2kHCylteCAdBCoDIn2tW3HsjJI6CrYivb0igJZO1VQoEInrLj+qx6KeAW+SxGDo=, 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" {
[root@ip-172-31-6-102 ec2-user]# nano main.tf
[root@ip-172-31-6-102 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-poli
cy]
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" {
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-172-31-6-102 ec2-user]#


IAM > Roles > codepipeline-role-for-s3-deployment
Identity and Access Management (IAM)
codepipeline-role-for-s3-deployment
Delete
Summary
Edit
Creation date
June 23, 2025, 09:45 (UTC+05:30)
ARN
arn:aws:iam::356240508970:role/codepipeline-role-for-s3-deployment
Last activity
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (2)
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Filter by Type
All types
Policy name
Type
Attached entities
AdministratorAccess
AWS managed - job function
4

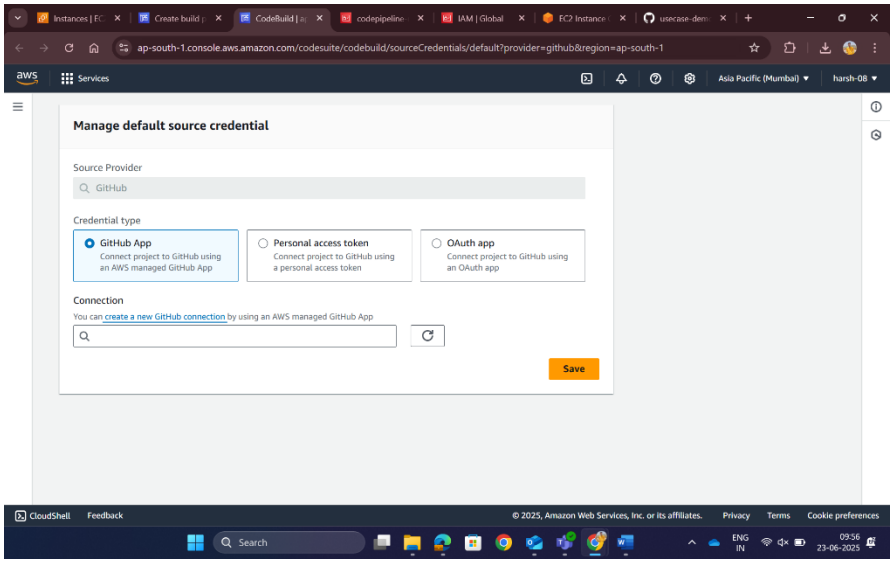

Amazon S3 > Buckets > harsh-bucket-unique-8
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Access Grants
Access Points for general purpose buckets
Access Points for directory buckets
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
harsh-bucket-unique-8
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (0)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
No objects
You don't have any objects in this bucket.
Upload



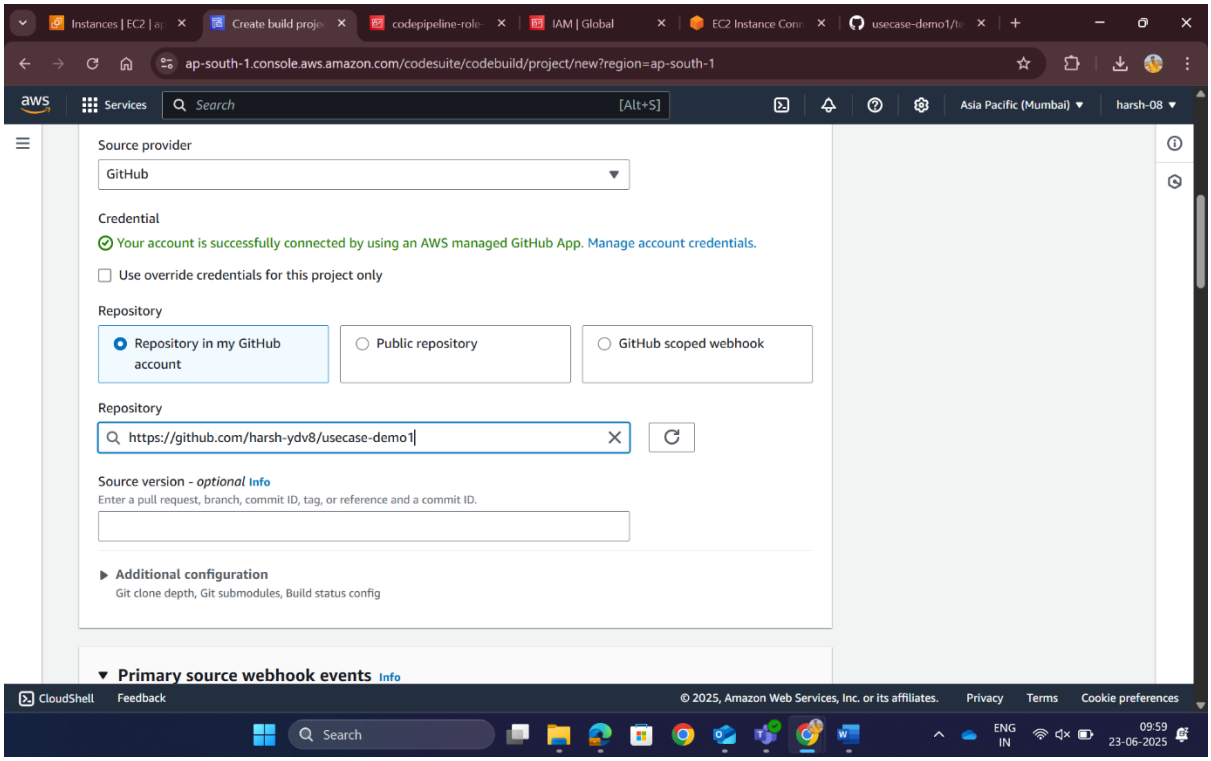
9. Now after roles are created, navigate to CodeBuild-> create new Project.

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.

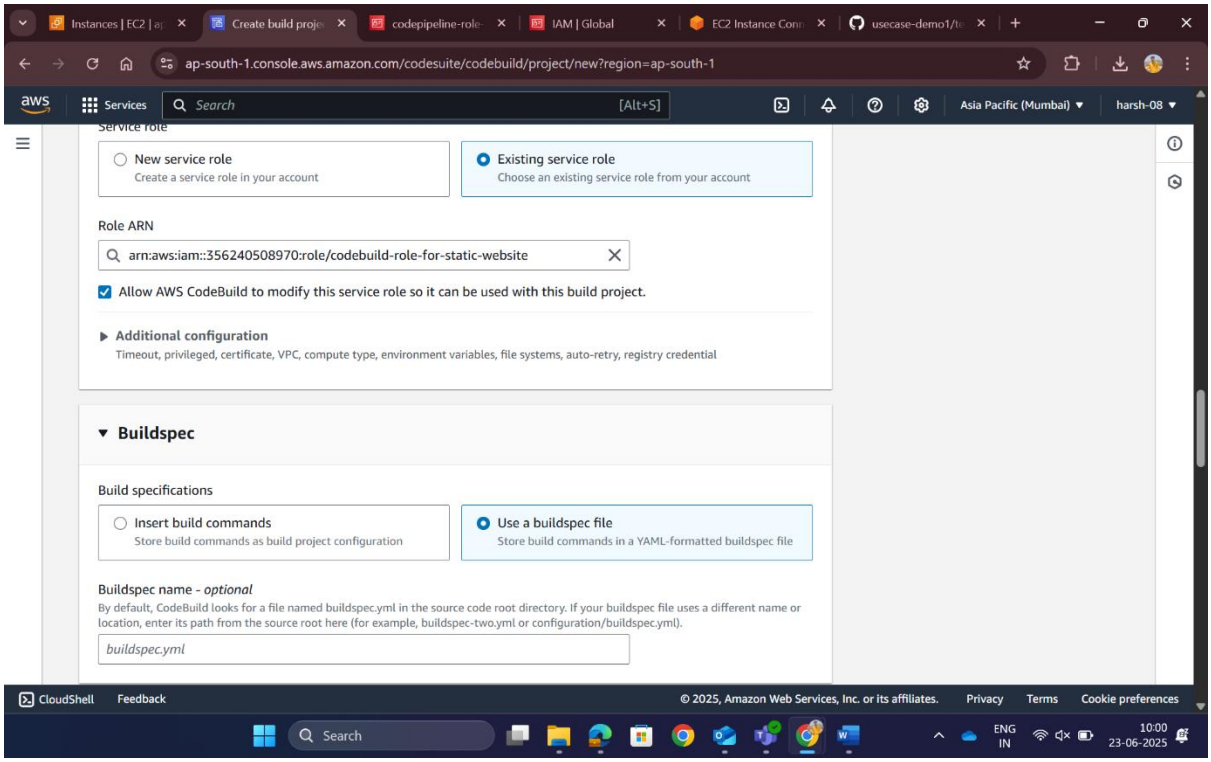


12. After successful connecting we can add the repository where buildspec.yml file is present. (https://github.com/harsh-ydv8/usecase-demo1/)

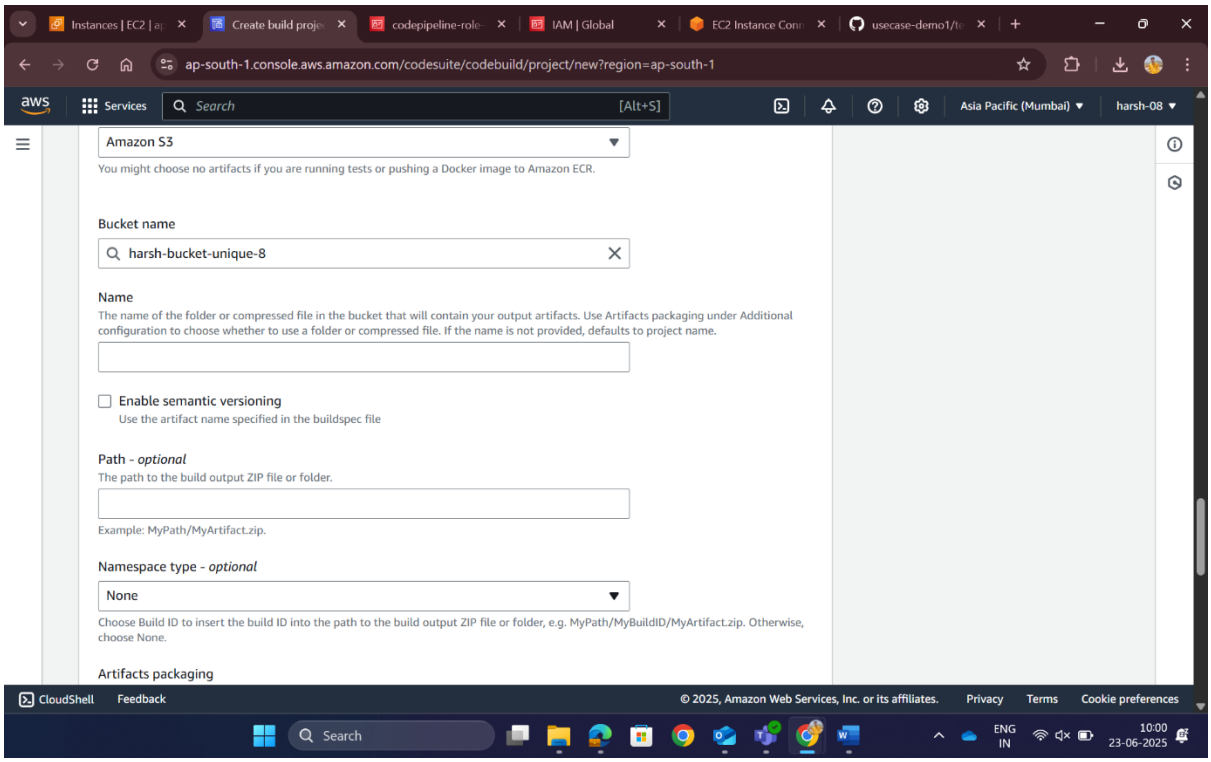


13. In the service role, we have to pick the role which we created using terraform script.

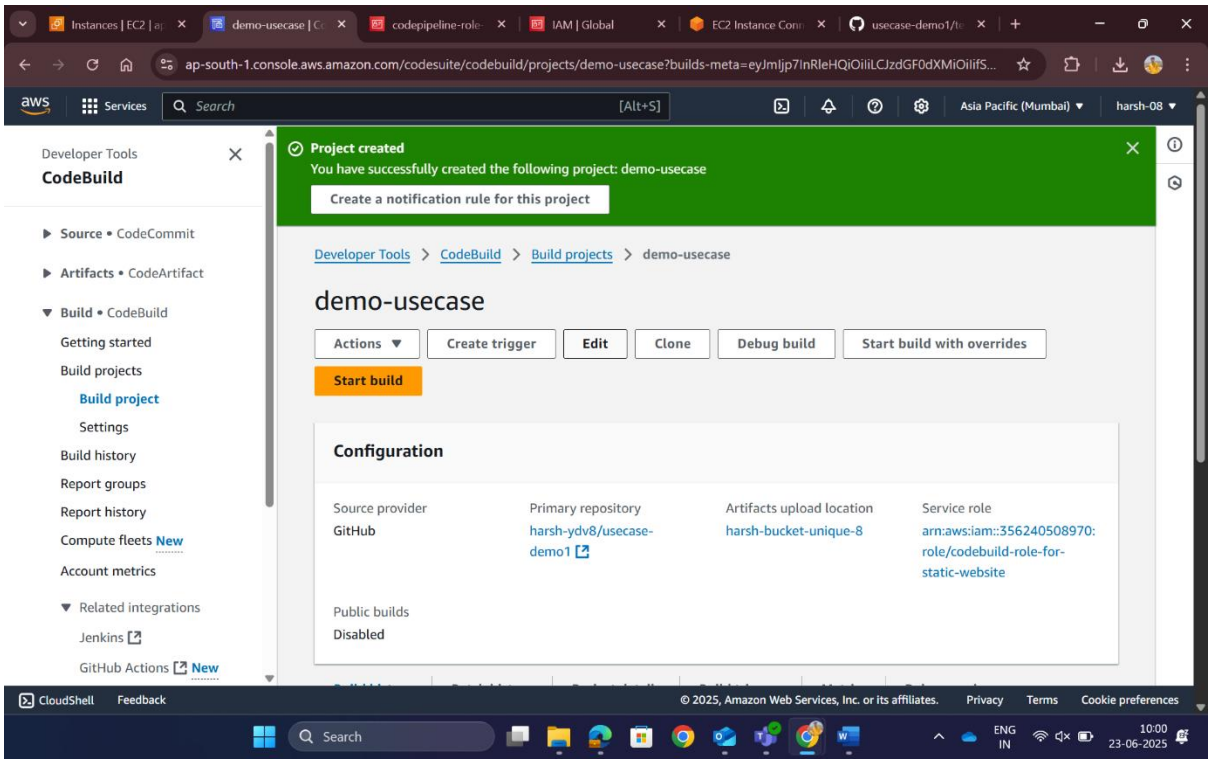
14. Now in BuildSpec, if buildspec.yml file is with other name enter that or leave if the name is same.



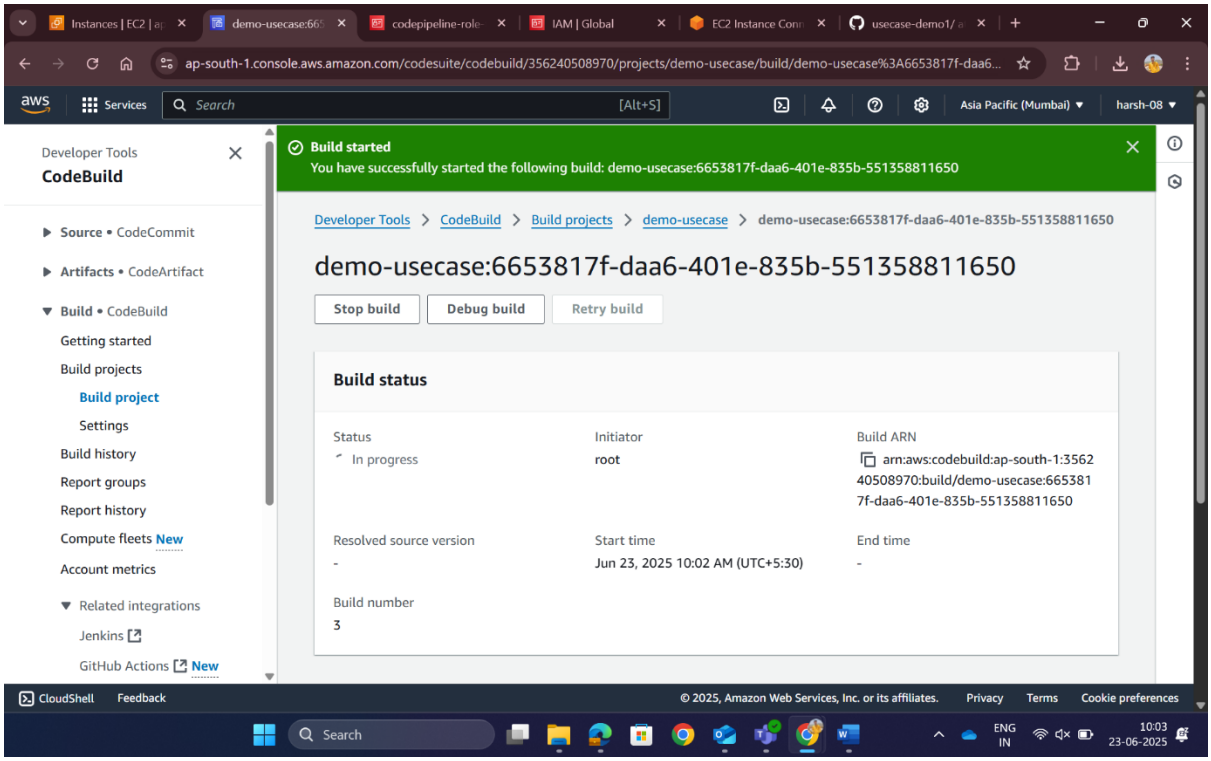
15. Now in the Artifacts select S3, and in the bucket name choose the bucket created using the script and then select zip in artifact packaging.

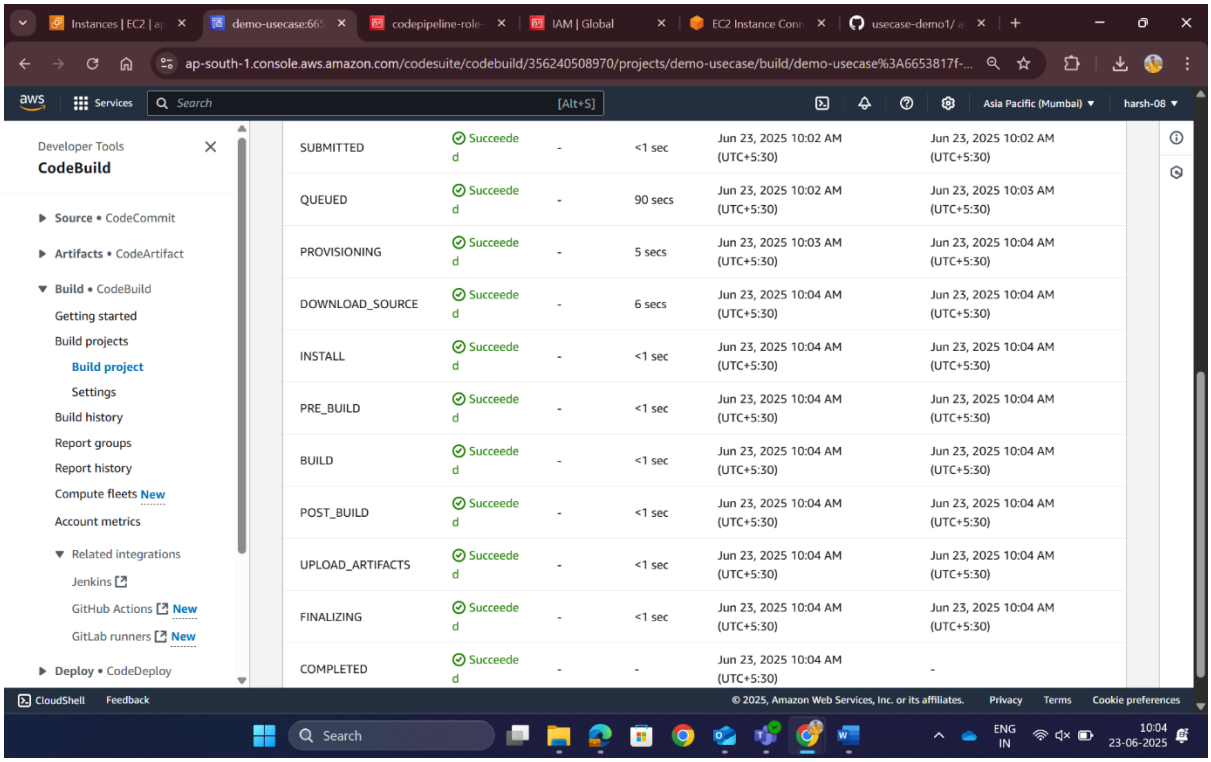


16. Now created the codebuild project.



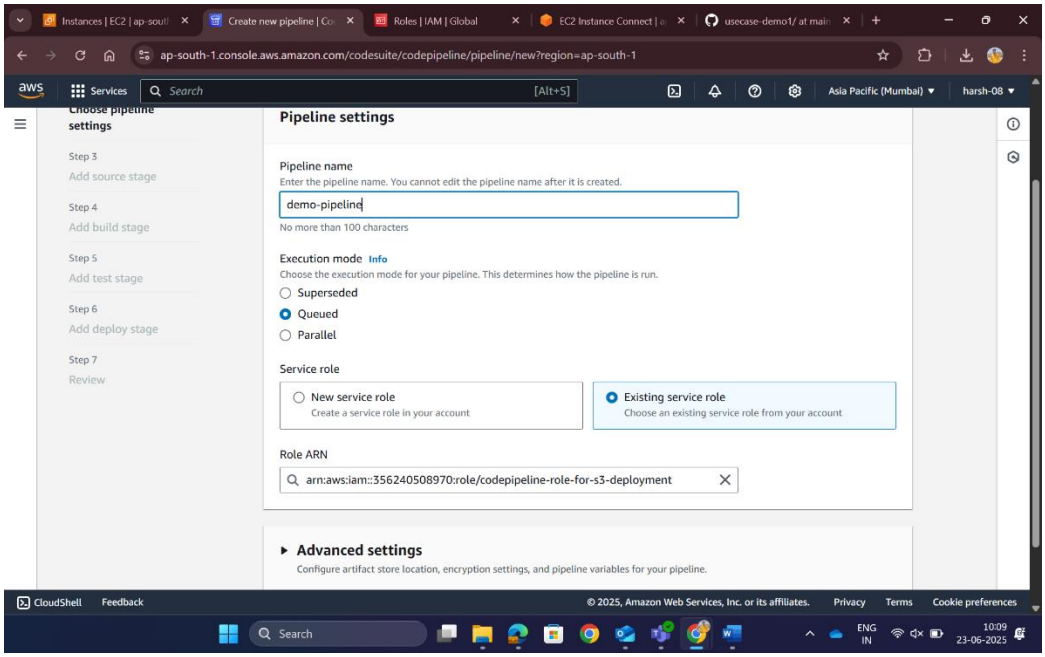
17. We will now start build the project. Once started it will now run the project and will provide the reports and steps which are being executed.



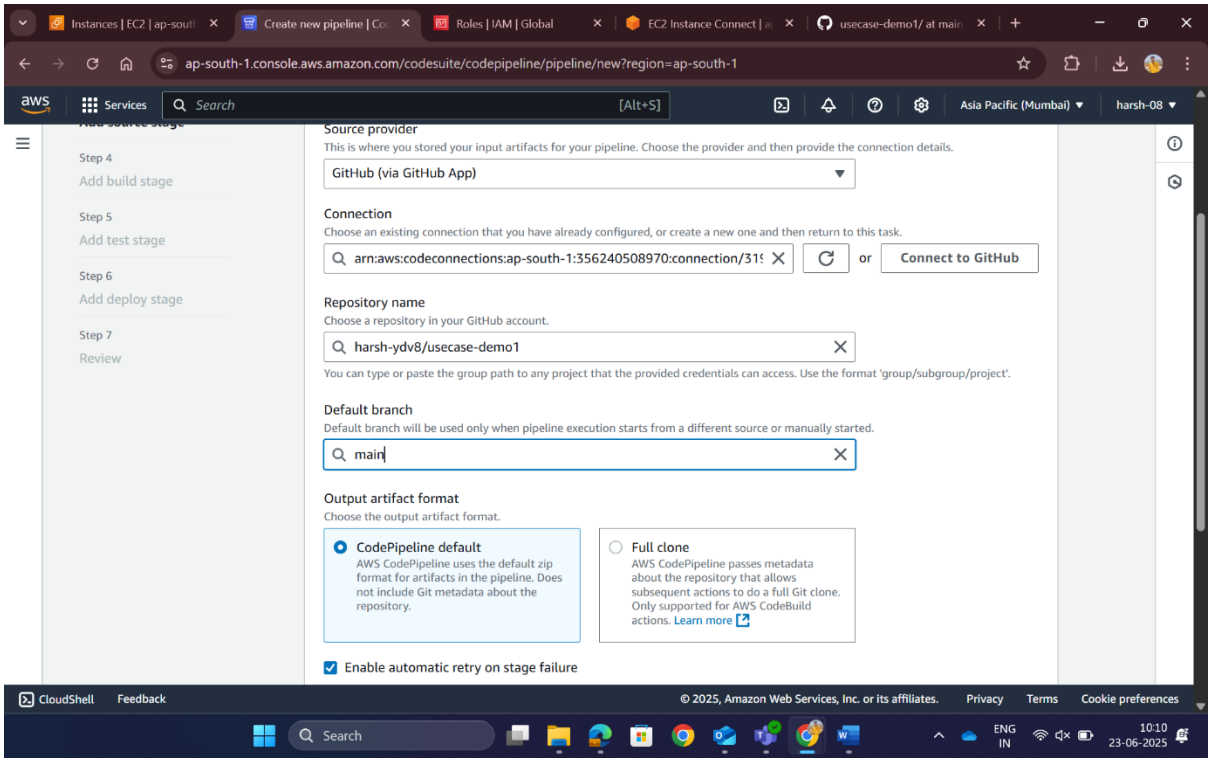


18. Now we will navigate to CodePipeline. Then create a custom pipeline.

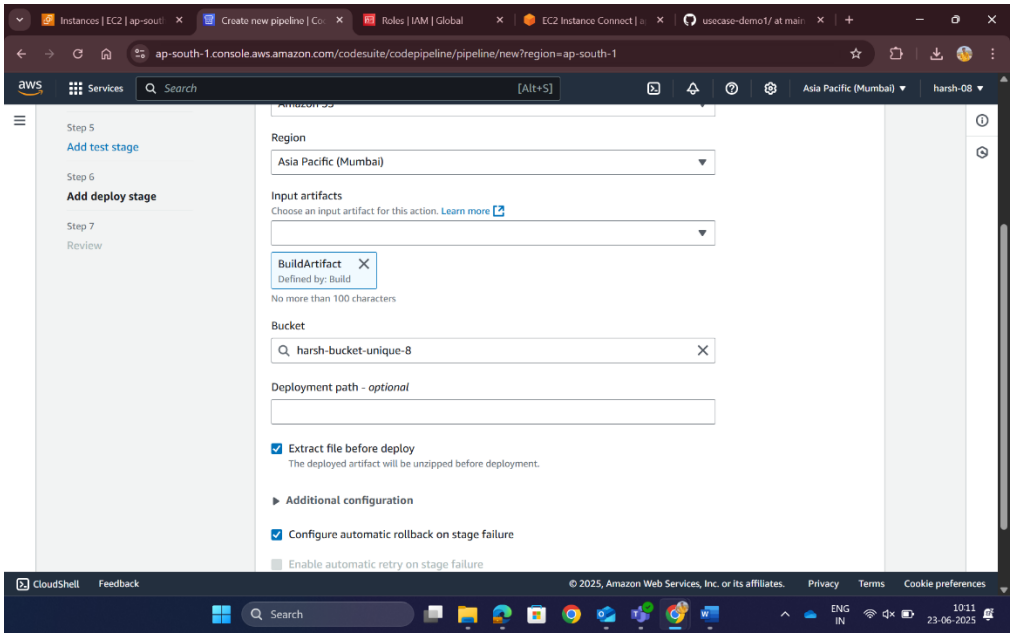
19. Give a pipeline name and in the service role choose existing service role & select the service created using the terraform script.



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

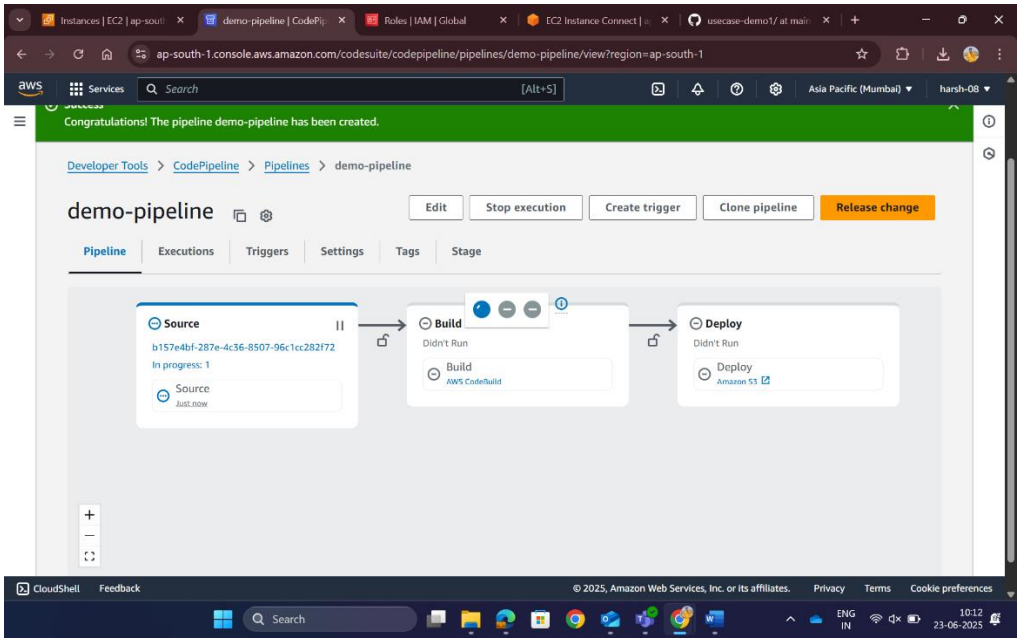


21. Now select the artifact as Amazon S3 and hence the name of the bucket. Choose option - extract file before deploy, as our file is zipped in the codebuild output.

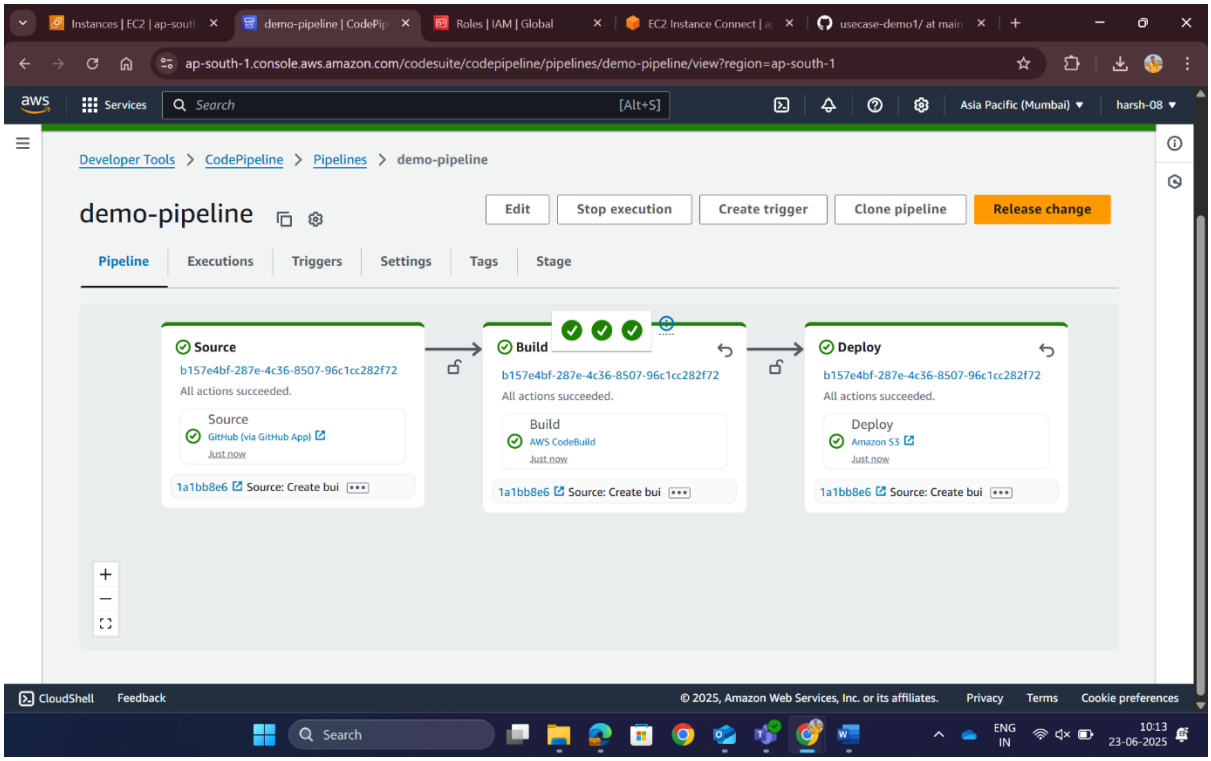


22. You can check the Define buildspec override and mention the buildspec file name also if the file name is different from buildspec.yml. Skip the test stage.

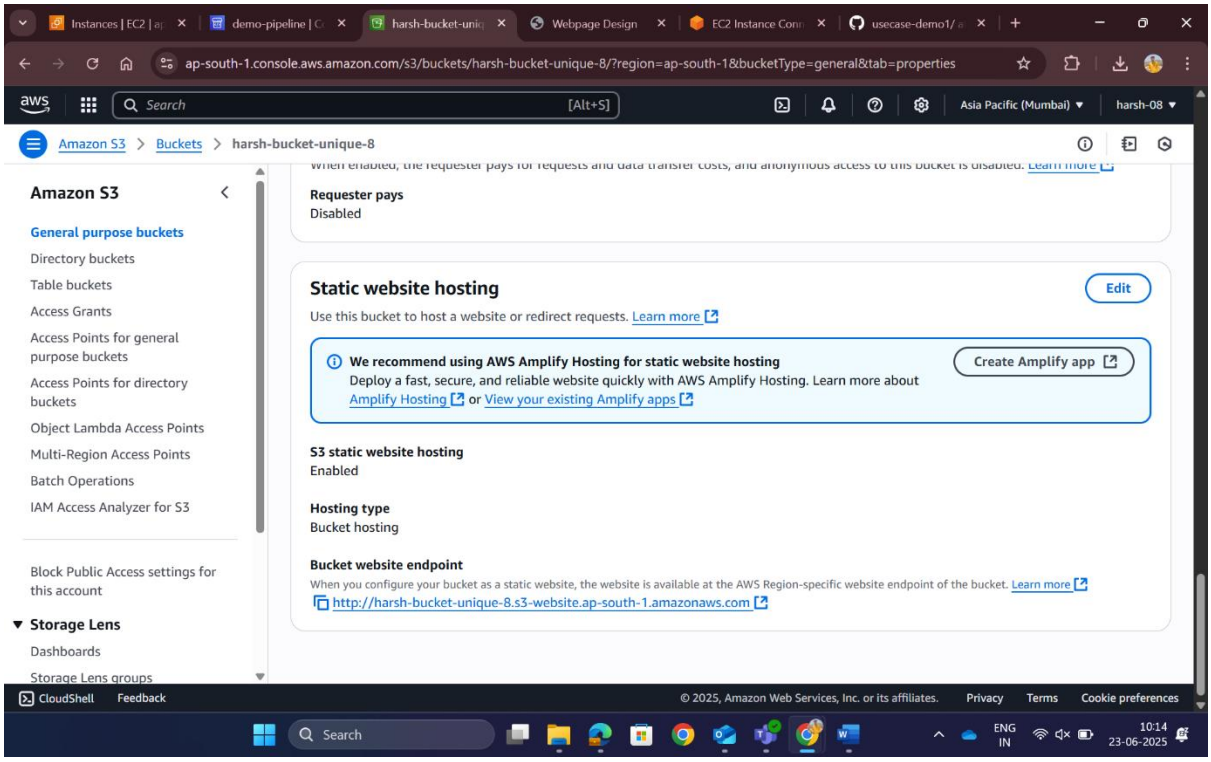
23. Now created the pipeline, run this pipeline. The workflow will be displayed, each step will be running in a queue.



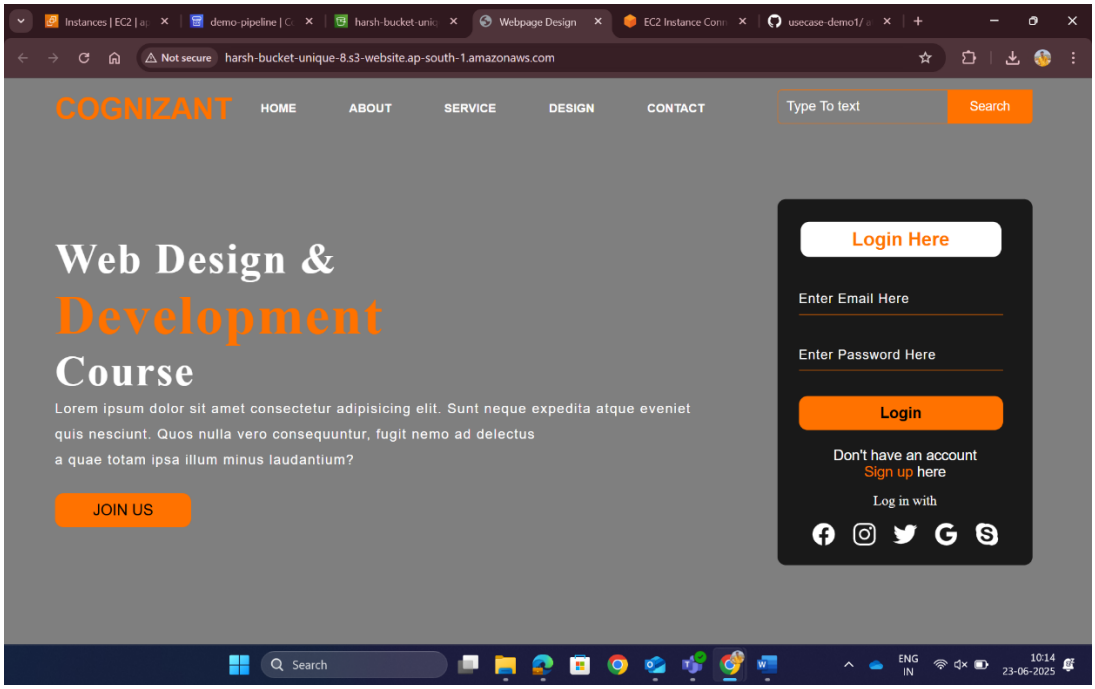
24. After successful pipeline execution, we'll get the deployment link in S3.



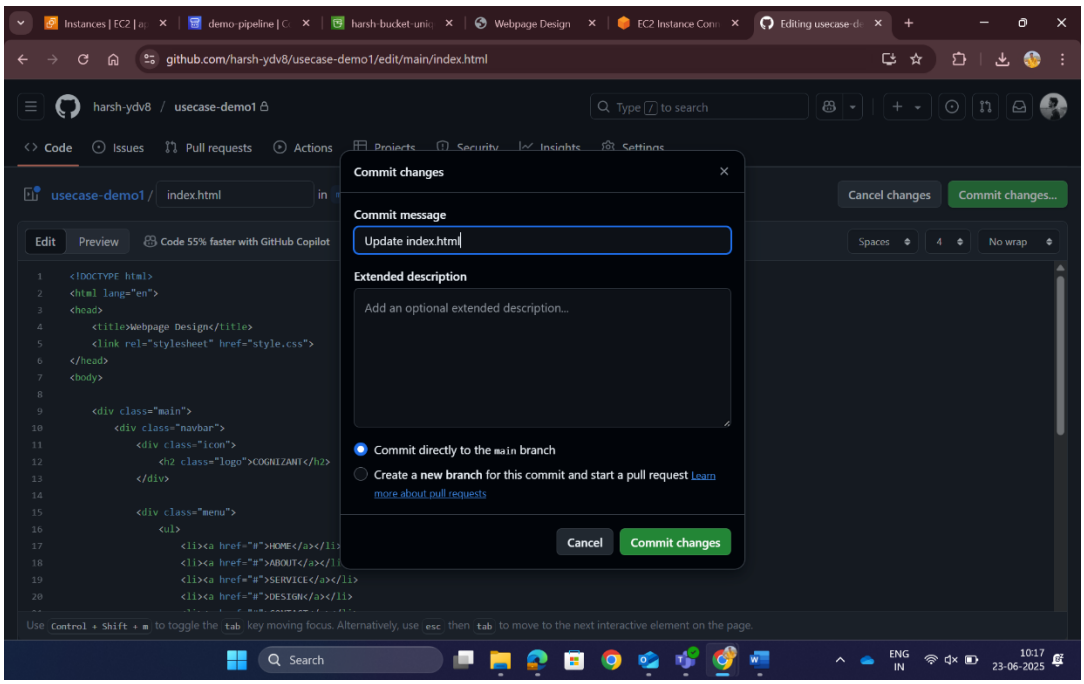
25. Now navigate to Amazon S3. Now navigate to **S3 bucket -> select you bucket -> properties -> Static website hosting -> Bucket website endpoint -> click the link.**



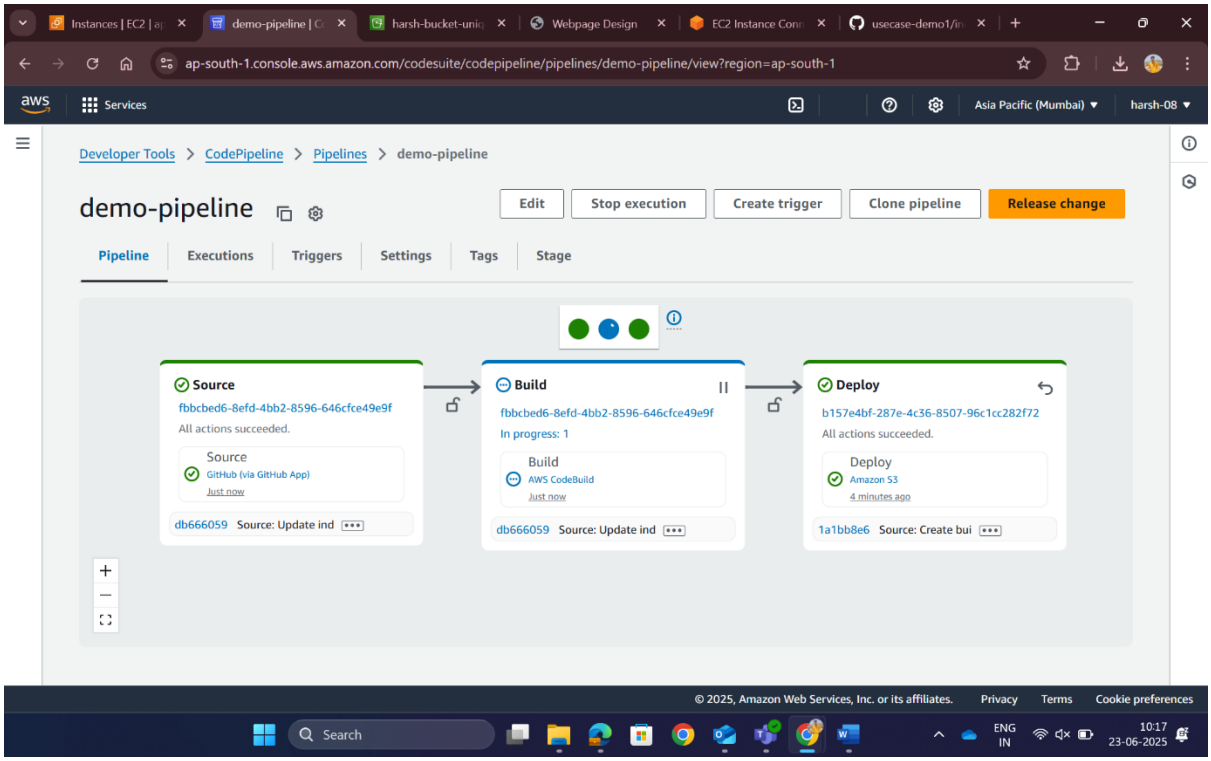
26. After visiting the link we'll be able to see the page.



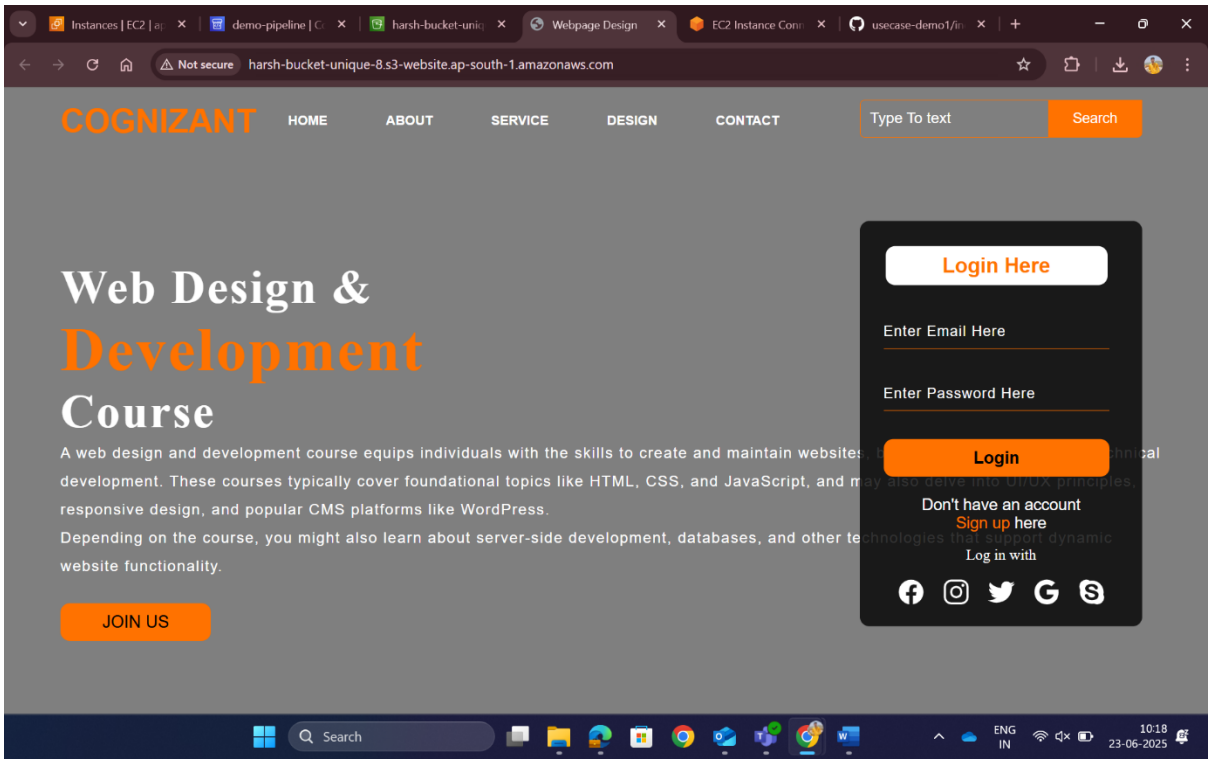
27. Now we will navigate to GitHub and made some changes in the index.html file and commit the changes.



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

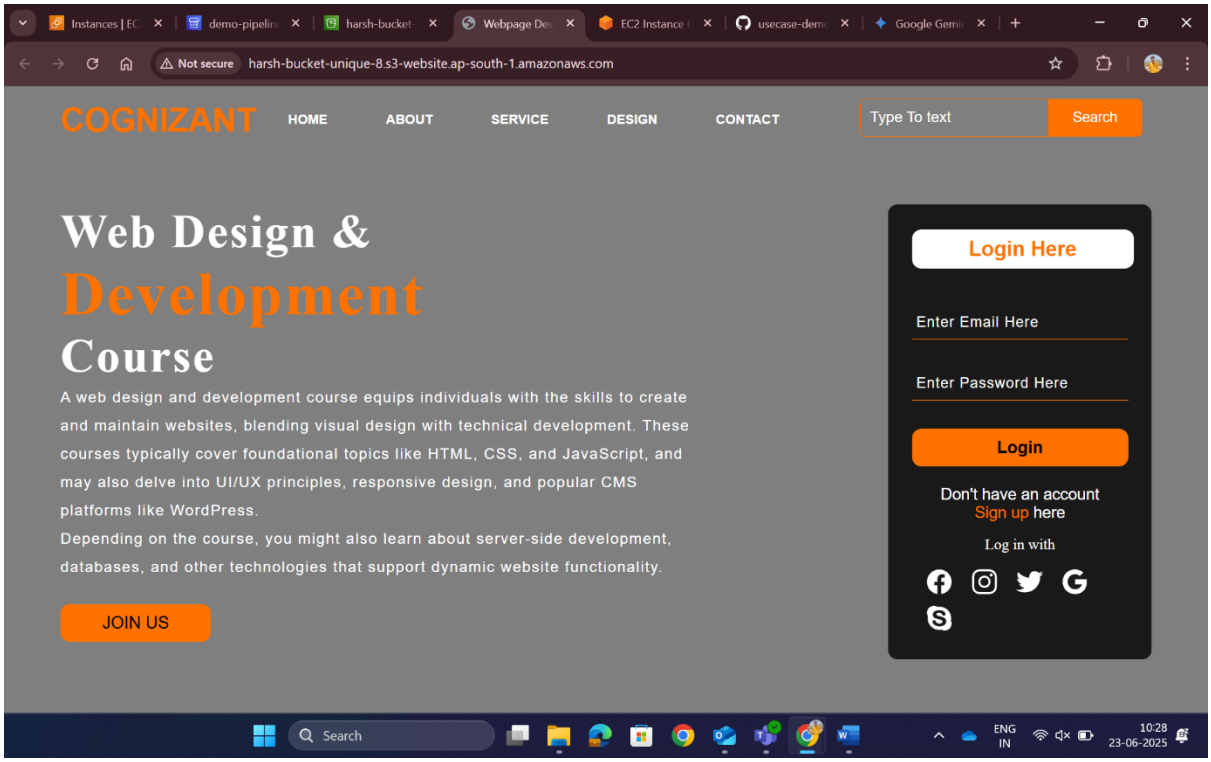


29. Once the pipeline is complete running, navigate to the page and reload the page. We'll be able to see the updated change.



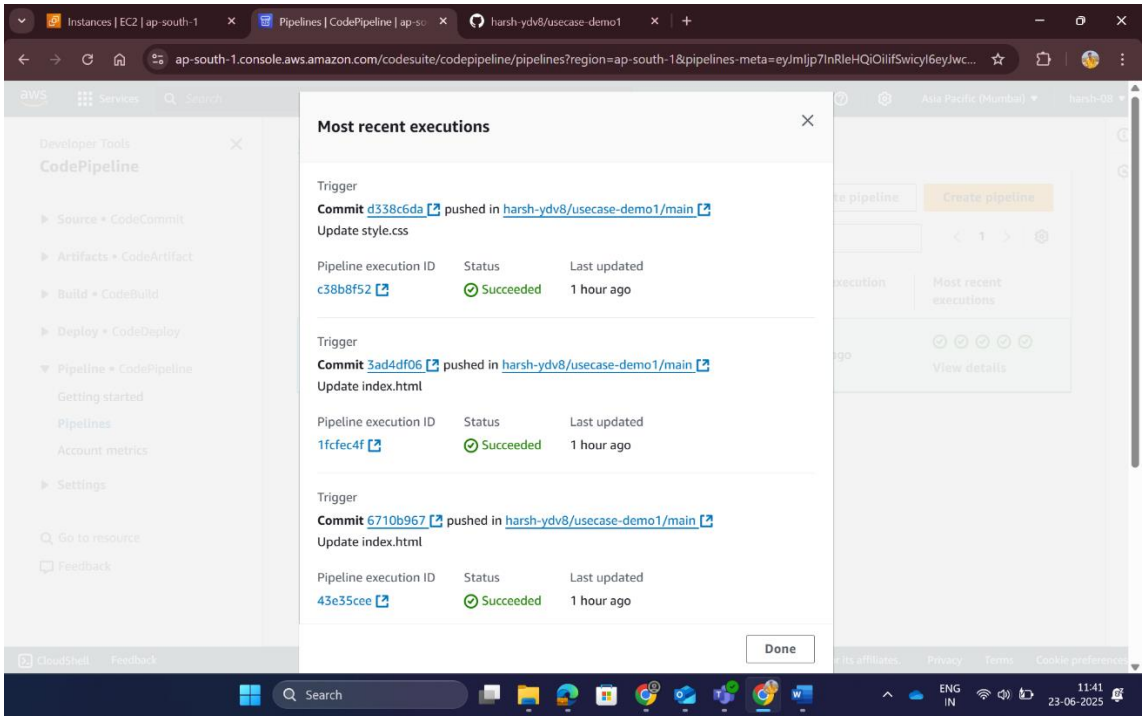
30. Now we'll try again to see if pipeline is working fine. We'll made change and commit those changes in GitHub. Then pipeline is triggered and hence

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----------