PHASE 5

CODE PIPELINE SETUP

Definition:

> What is AWS CodePipeline? Why are we using it?
>
> With **CodePipeline**, devs can create a workflow that automatically moves the code changes through the build and deployment stage. In this project, a new push to the **GitHub** repository automtically triggers a build in **CodeBuild** (continuous integration), and then a deployment in **CodeDeploy** (continuous deployment)!
>
> Using CodePipeline makes sure the deployments are consistent, reliable and happen automatically whenever the code is updated- with less risk of human errors. It saves time too.

STEP 1 : PIPELINE SETUP

- Head to CodePipeline console on AWS
- Select Create Pipeline → Select Build custom pipeline → Click Next → Name the pipeline → Under execution mode select Superseded

> What is Execution mode?

> **Execution mode** determines how CodePipeline handles multiple runs of the same pipeline.
>
> In **Superseded** mode, if a new pipeline execution is triggered while another execution is already in progress, the newer execution will immediately take over and cancel the older one. This is perfect for making sure only the latest code changes are processed.
>
> There are other execution modes available in CodePipeline:
>
> - In **Queued** mode, executions are processed one after another. If a pipeline is already running, any new executions will wait in a queue until the current execution finishes.
> - **Parallel** mode allows multiple executions to run at the same time, completely independently of each other. This can speed up the overall processing time if there are multiple branches or code changes that can be built and deployed concurrently.

- Under Service roll, select new service role. Keep default role name

  What is a service role?

  A **service role** is a special type of IAM role that AWS services like CodePipeline use to perform actions on user's behalf. It's like giving CodePipeline permission to access other AWS resources it needs to run the pipeline, such as S3 buckets for storing artifacts or CodeBuild for building the code.

- Expand Advanced Settings → Leave everything as default (Artifact store, Encryption key and Variables)

> What are Artifact store, Encryption key, and Variables?

> - **Artifact store**: Without an artifact store, there's no way for the build outputs to be passed to deployment! This S3 bucket is where CodePipeline automatically saves the files created at each stage - like the source code from GitHub and the build artifacts from CodeBuild - making them available to the next stage in the pipeline.
> - **Encryption key**: By default, CodePipeline encrypts everything in the artifact store using AWS managed keys. This keeps the code and build artifacts secure while they're being stored and transferred between stages. For most projects, this default encryption is perfectly sufficient.
> - **Variables**: Right now information like version numbers or build timestamps might be manually tracked . Pipeline variables solve this by letting to pass dynamic values between different stages automatically. Variables become essential in more complex pipelines when you need information generated in one stage (like a build number) to be available in another stage (like deployment).

- Click Next
- Source stage config: In the Source provider dropdown, select GitHub (via GitHub App)
- Under Repository name and branch name select correct repo and branch of GitHub repo used
- For Output artifact format , leave it as CodePipeline default

> What is Output artifact format?
>
> **Output artifact format** determines how CodePipeline packages the source code it fetches from GitHub.
>
> - **CodePipeline default**: This option packages the source code as a ZIP file, which is efficient for most deployment scenarios. It does not include Git metadata about the repository.
> - **Full clone**: This option provides a full clone of the Git repository as an artifact, including Git history and metadata. This is useful if the build process requires Git history, but it results in a larger artifact size.

- Check the Webhook box under Webhook events

> What are Webhook events?
>
> **Webhook events** let CodePipeline automatically start the pipeline whenever code is pushed to a specified branch in GitHub. This is what makes the pipeline truly "continuous" – it reacts to code changes in real-time
>
> **How do Webhooks work?**
>
> **Webhooks** are like digital notifications. When webhook events are enabled, CodePipeline sets up a webhook in the GitHub repository. This webhook is configured to listen for specific events, such as code pushes to the master branch.
>
> Whenever code is pushed to the master branch, GitHub sends a webhook event (a notification) to CodePipeline. CodePipeline then automatically starts a new pipeline

> execution in response to this event. It's a seamless way to automate the CI/CD process

- Click Next → Build stage → In the Build provider dropdown, select Other build providers → AWS CodeBuild

  What is the Build stage?

  The **Build stage** is where the source code gets compiled and packaged into something that can be deployed.

- Under Project name dropdown , select the created CodeBuild project name
- For Environment variables, Build type, and Region leave them as default
- Under Input artifacts, Source Artifact to be selected as default

  What are Input artifacts?

  Input artifacts are the outputs from the previous stage that are used as inputs for the current stage. In the Build stage, SourceArtifact is being used, which is the ZIP file containing the source code that was outputted by the Source stage.

- Click Next → Then Test stage is optional (Click on Skip test stage to skip)

> What is the Test stage?
>
> The **Test stage** is where you application's testing is automated . This can include different types of tests, likes:
>
> - **Unit tests**: Testing individual components or functions of your code.
> - **Integration tests**: Testing how different parts of the application work together.
> - **UI tests**: Testing the user interface to make sure it works correctly.

> The Test stage helps ensure the quality of the code and catch any issues before they reach production. A Test stage is essential for maintaining software quality and reliability.

- Deploy Stage: In the Deploy provider dropdown, select AWS CodeDeploy

  What is the Deploy stage?

  The **Deploy stage** is the final step in the pipeline. It's responsible for taking the application artifacts (the output from the Build stage) and deploying them to the target environment, here the target is an EC2 instance.

- Under input artifacts → Leave BuildArtifact as default
- Under application name, deployment group → Select the created CodeDeploy application and deployment group
- Check the box for Configure automatic rollback on stage failure

  What is automatic rollback?

  Automatic rollback is a safety net for the deployments. By enabling it, CodePipeline is being informed that if the Deploy stage fails for any reason, it should automatically revert to the last successful deployment. This helps minimize downtime and ensures that the application remains stable, even if a new deployment goes wrong.
- Click Next → review the details in Review page and click Create pipeline
- CODE PIPELINE IN ACTION NOW!

> What are all the different kinds of statuses?
>
> As the pipeline runs, each stage will display a status:
>
> - **Grey**: Stage has not started yet.
> - **Blue**: Stage is currently in progress.
> - **Green**: Stage has completed successfully.

> - **Red**: Stage has failed.

- Wait for the pipeline execution to complete. The status of each stage can be monitored in the pipeline diagram.
- To see more details about each execution, click on the **Executions** tab above the pipeline diagram.

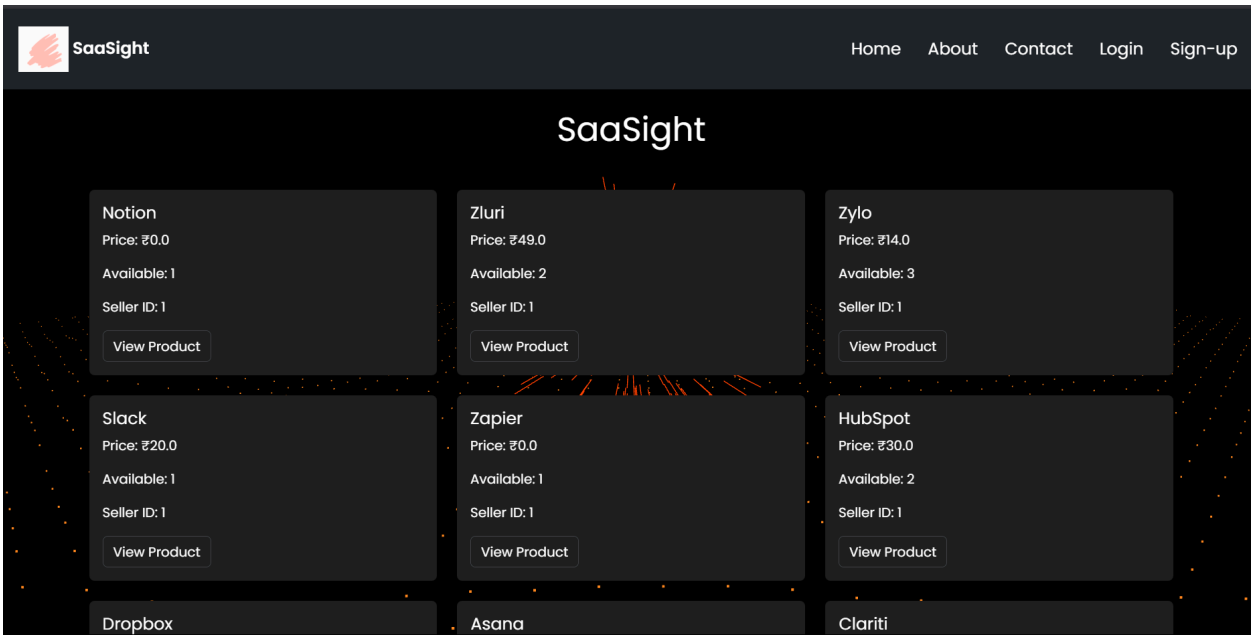
  What are Pipeline executions?

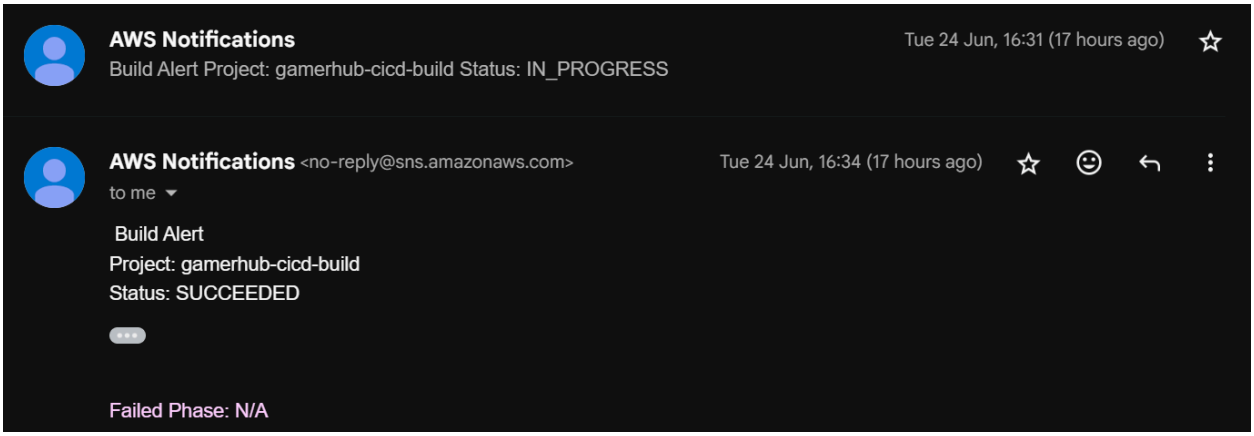
  Pipeline executions represent each instance of the pipeline running. Every time the pipeline is triggered (either manually or automatically by a webhook), a new execution is created. Each execution has a unique ID and shows the status and details of each stage in that particular run.
- If all stages have turned green , it means that the pipeline is all set up using the latest code changes
- Make code changes locally and then push to the pipeline to notice the change
- Use the IPv4 of EC2 to check if the web-app is working
- NOTE: Use CodeBuild Debug sandbox to debug the built package in CodeBuild phase
- NOTE: Lot of issues will be faced . Refer the Errors and Fixes part of the documentation

Build logs | Phase details | Reports | Environment variables | Build details | Resource utilization

Name	Status	Context	Duration	Start time	End time
SUBMITTED	Succeeded	-	<1 sec	Jun 24, 2025 4:31 PM (UTC+5:30)	Jun 24, 2025 4:31 PM (UTC+5:30)
QUEUED	Succeeded	-	91 secs	Jun 24, 2025 4:31 PM (UTC+5:30)	Jun 24, 2025 4:33 PM (UTC+5:30)
PROVISIONING	Succeeded	-	7 secs	Jun 24, 2025 4:33 PM (UTC+5:30)	Jun 24, 2025 4:33 PM (UTC+5:30)
DOWNLOAD_SOURCE	Succeeded	-	3 secs	Jun 24, 2025 4:33 PM (UTC+5:30)	Jun 24, 2025 4:33 PM (UTC+5:30)
INSTALL	Succeeded	-	30 secs	Jun 24, 2025 4:33 PM (UTC+5:30)	Jun 24, 2025 4:33 PM (UTC+5:30)
PRE_BUILD	Succeeded	-	<1 sec	Jun 24, 2025 4:33 PM (UTC+5:30)	Jun 24, 2025 4:33 PM (UTC+5:30)
BUILD	Succeeded	-	25 secs	Jun 24, 2025 4:33 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
POST_BUILD	Succeeded	-	<1 sec	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
UPLOAD_ARTIFACTS	Succeeded	-	<1 sec	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
FINALIZING	Succeeded	-	<1 sec	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
COMPLETED	Succeeded	-	-	Jun 24, 2025 4:34 PM (UTC+5:30)	-

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
DownloadBundle	less than one second	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
BeforeInstall	14 seconds	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
Install	less than one second	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
AfterInstall	less than one second	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
ApplicationStart	9 seconds	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)
ValidateService	less than one second	Succeeded	-	Jun 24, 2025 4:34 PM (UTC+5:30)	Jun 24, 2025 4:34 PM (UTC+5:30)





(5-6 late nights, endless broken builds, and one stubborn will later — I got it working.)

OVER AND OUT