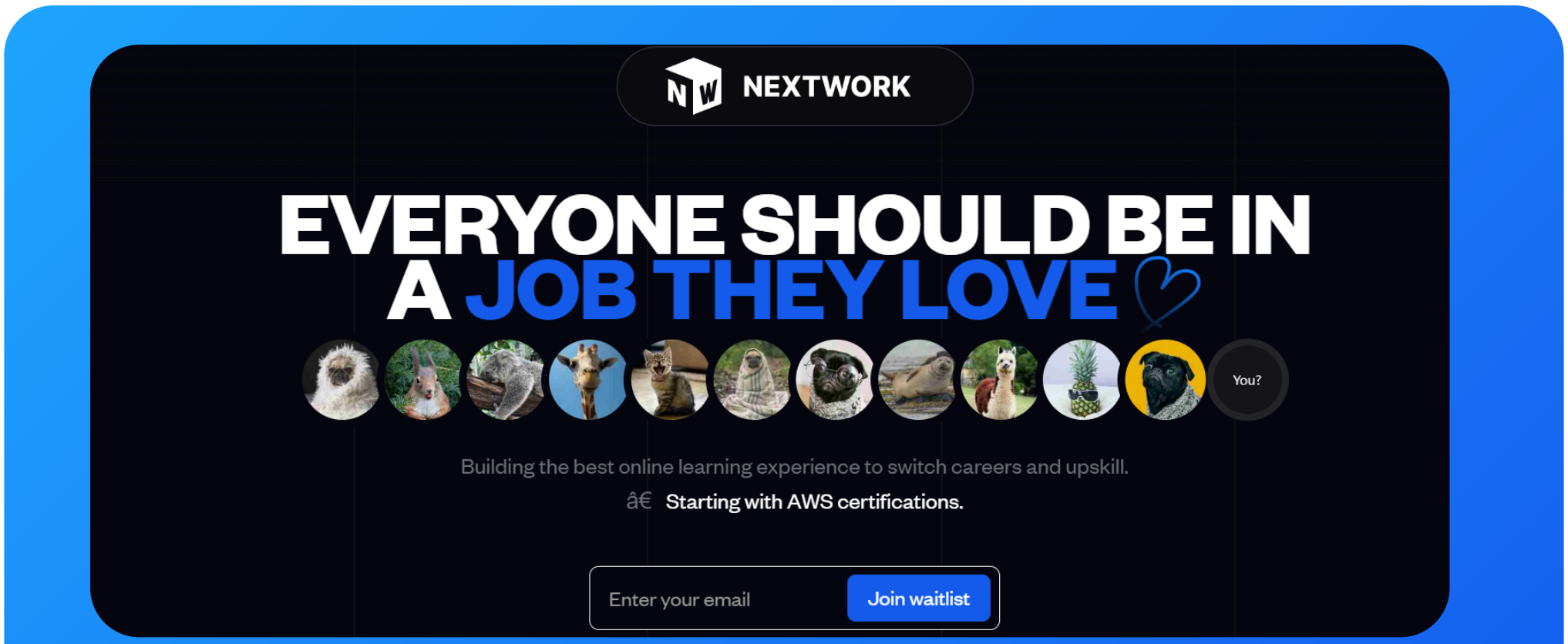
NextWork.org

CI/CD Pipeline with AWS CodePipeline

kanika Mathur
github.com/KanikaGenesis




kanika Mathur
github.com/KanikaGenesis

Introducing AWS CodePipeline!

What it does & how it’s useful

AWS CodePipeline is a continuous integration and continuous delivery (CI/CD) service that automates the steps required to release your software changes continuously.
Developers and teams use AWS CodePipeline because it automates and accelerates application development and release processes, ensuring rapid and reliable updates with minimal manual intervention.

How I’m using it in today’s project

I’m using AWS CodePipeline in this project to automate the integrations between CodeCommit, CodeBuild and CodeDeploy.

This project took me...

This project took me around 2 hours to complete.


kanika Mathur
github.com/KanikaGenesis

Set Up a CI/CD Pipeline

- A CI/CD pipeline is a practice of continuous integration, continuous deployment. This makes sure that changes made to source code is constantly being integrated/updated in a shared Git repository, so that other developers working from the same code base are always the latest version of the code (continuous integration). It also makes sure that end users are always seeing the latest version/updates to an application (continuous deployment).

To set up a CI/CD in CodePipeline, I configured three stages:

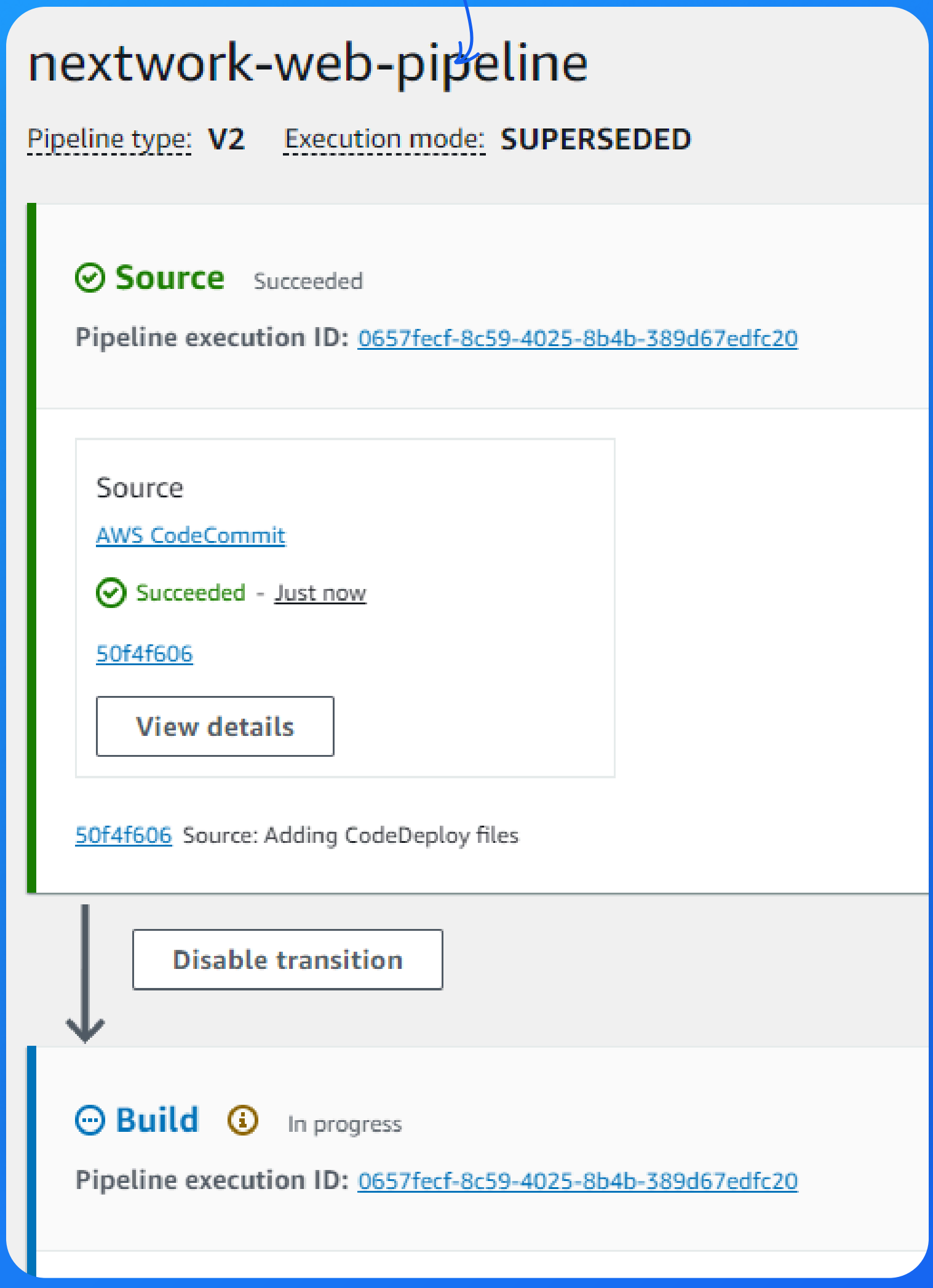
- The **source** stage, which means the source code for my web app. This is currently stored in CodeCommit.

- The **build** stage, which means the service that is managing the build process for my web app (i.e. compiling and packaging my web app into a handy WAR file.). This CodeBuild.

- The **deploy** stage, which means the service that is managing the deployment process for my web app (i.e. making my web app available to the world). This is CodeDeploy.

kanika Mathur
github.com/KanikaGenesis

A peek into my pipeline set up in CodePipeline!



kanika Mathur
github.com/KanikaGenesis

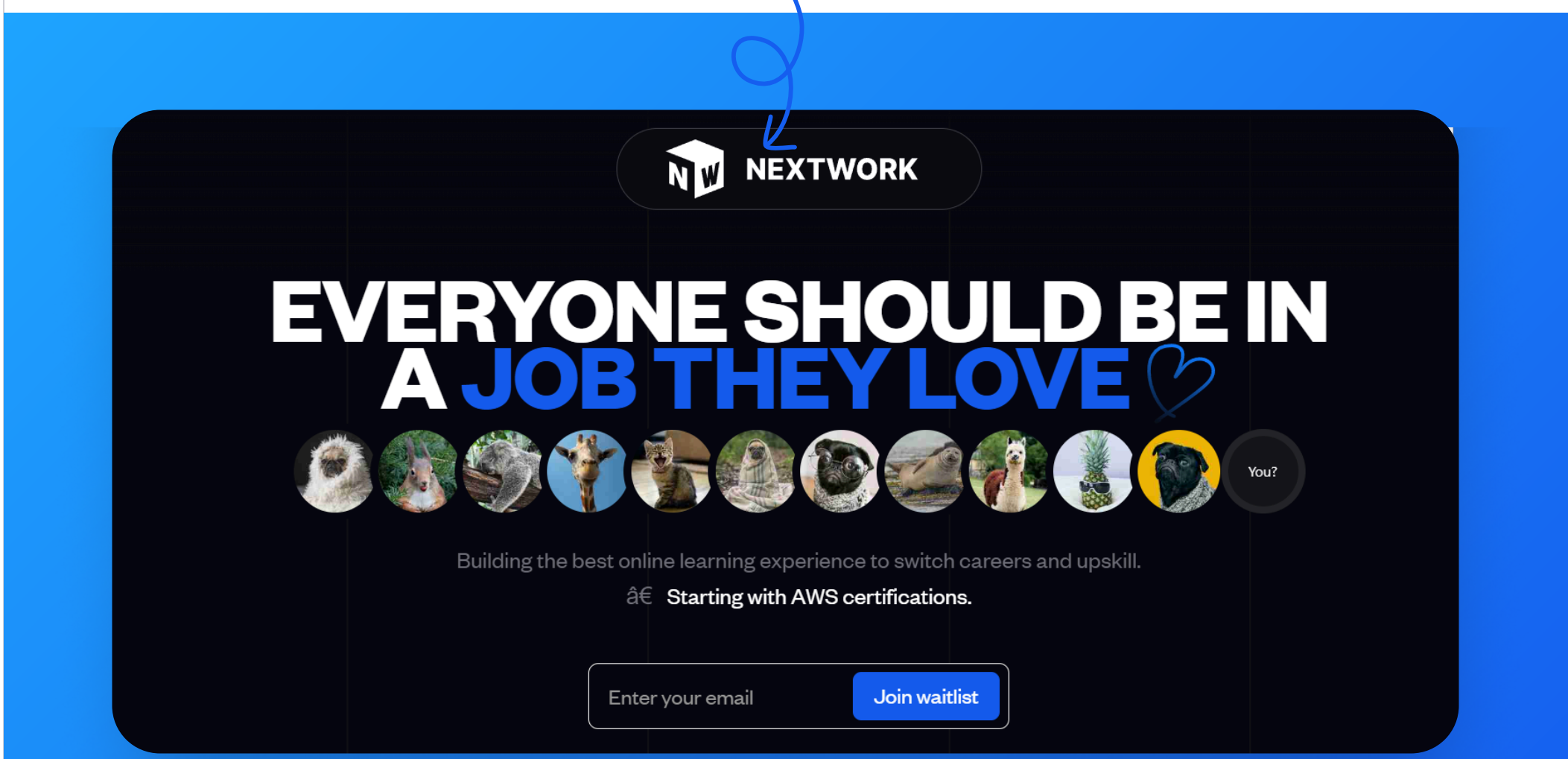
Release a Change

- My CI/CD pipeline gets triggered by a commit in my local working environment (Cloud9) - which updates my CodeCommit repository.

- I tested this by making **two** updates to my web app's source code. These changes were an edit to **index.jsp** and uploading a folder of image assets in the **webapp** folder of my project files.

- Once my pipeline executed successfully, I checked the IPv4 address of my web server that is hosting my web app.

My web app automatically updated after I committed a change.




kanika Mathur
github.com/KanikaGenesis

Trigger A Rollback

- A rollback in a pipeline means reverting the version of our code that a stage of our pipeline is referring to.

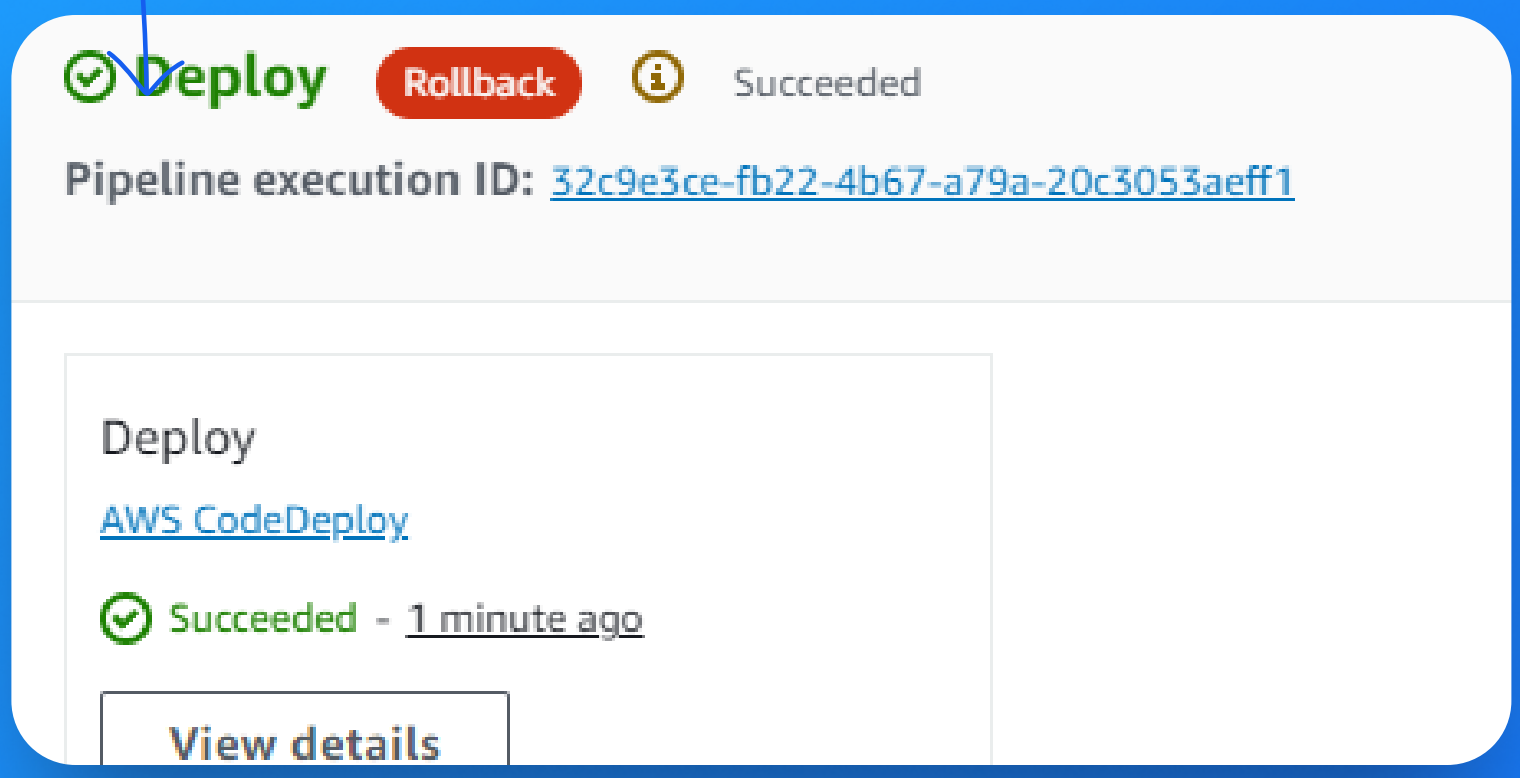
- I initiated a rollback on the deploy stage. I checked the source stage, and learnt that the source stage was unaffected by the rollback, and stayed using the latest version of my source code.

- Once the rollback was complete, my web app's appearance reverted back to reflect the simple version of my index.jsp file before I ever updated my index.jsp file and uploaded new image assets.

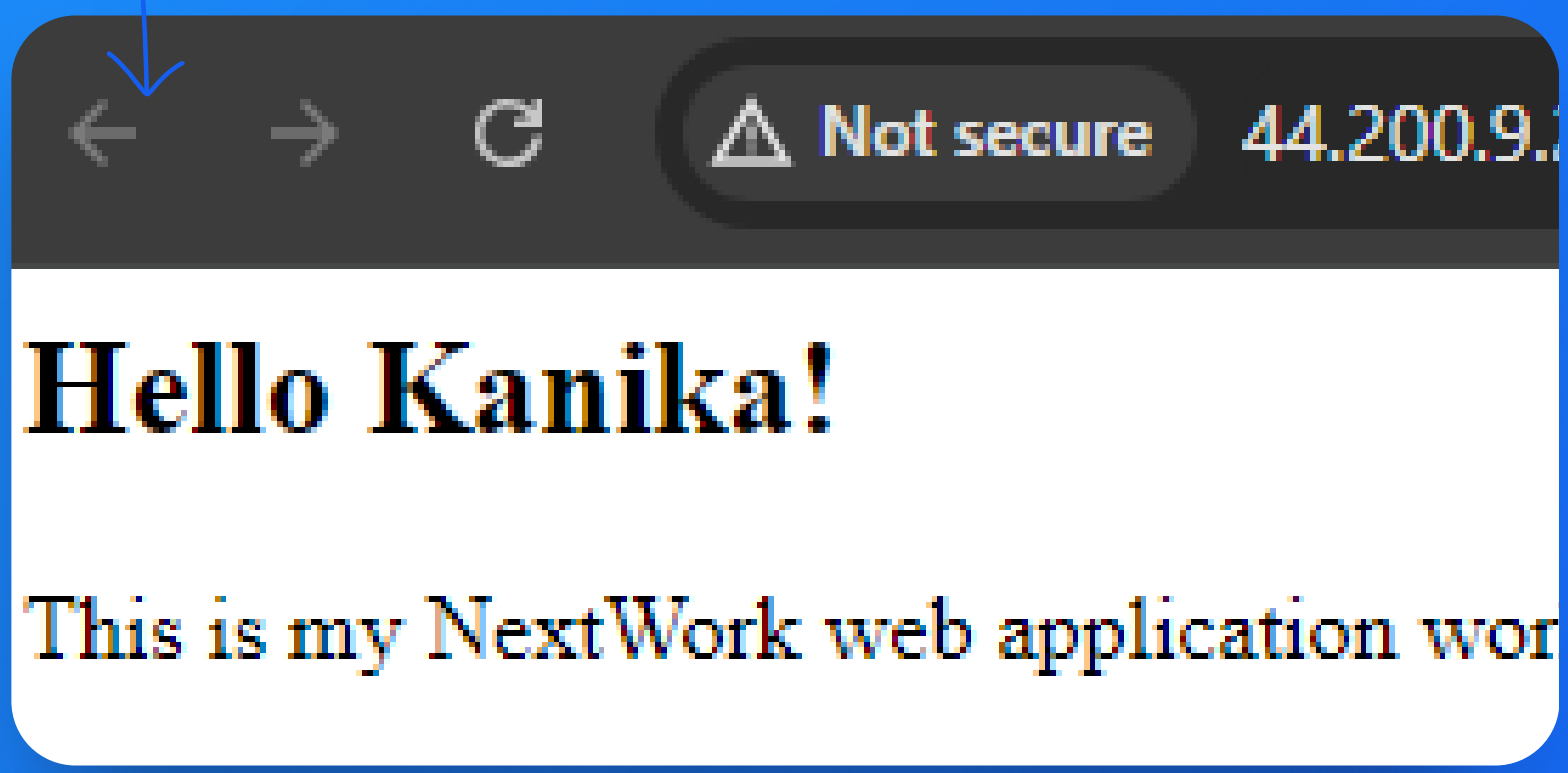
- To update my Deploy stage back to the latest version of my source code, I **released** a change in my CodePipleline pipleline. Releasing a change means the latest version of my source code gets referenced across the Source, Build AND Deploy stages of my pipeline.

kanika Mathur
github.com/KanikaGenesis

My CodeDeploy panel showing a successful rollback.



My web app returned to its previous state after the rollback.




kanika Mathur
github.com/KanikaGenesis

My CI/CD Pipeline

1. **AWS Cloud9** is responsible for providing an integrated development environment (IDE) for writing, running, and debugging code in the cloud.

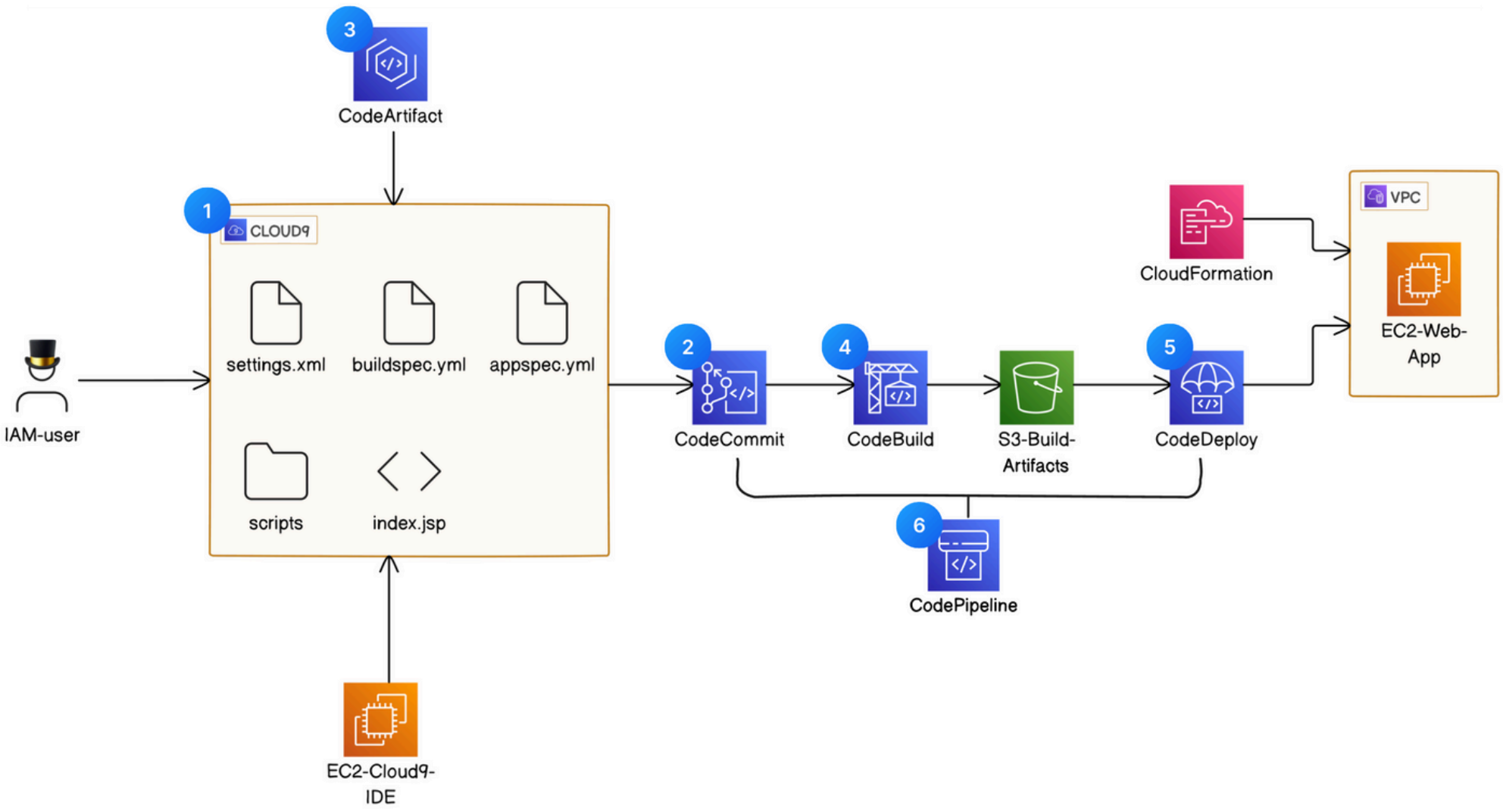
2. **AWS CodeCommit** is responsible for hosting secure Git repositories to store and version control the source code.

3. **AWS CodeArtifact** is responsible for managing software packages and dependencies, making it easy to store, publish, and share them.

4. **AWS CodeBuild** is responsible for compiling and packaging my app's source code into a handy WAR file.

5. **AWS CodeDeploy** is responsible for deploying that WAR file on servers, so users can see my web app!

6. **AWS CodePipeline** is responsible for automating the integrations between CodeCommit, CodeBuild and CodeDeploy.




kanika Mathur
github.com/KanikaGenesis

My key learnings

1 CI/CD means Continuous Integration and Continuous Deployment/Delivery, which are practices that automate the integration of code changes and the delivery of software updates, ensuring faster and more reliable releases.

2 To set up a CI/CD pipeline for my web app project, i configured three stages:
Source: pulling the source code from the repository, managed by AWS CodeCommit.
Build: compiling the source code using AWS CodeBuild.
Deploy: deploying the build artifacts to the production environment using AWS CodeDeploy.

3 Example scenarios of triggering a rollback in the deploy stage are:
Application Errors: If the newly deployed version introduces critical bugs or errors that affect the functionality of the application, a rollback is initiated to restore the last known good configuration.
Performance Issues: If the new deployment significantly degrades the performance of the application, such as increased load times or reduced responsiveness, a rollback is performed to return to the previous version.

4 One thing I didn’t expect was the impact of IP address changes during the deployment process. When new instances were launched, they often received different IP addresses, which affected services relying on fixed IPs. This required additional configuration to handle dynamic IP allocations and ensure continuous connectivity and service availability.

NEXTWORK

Everyone should be in a job they love. yes!

Check out community.nextwork.org for more free projects