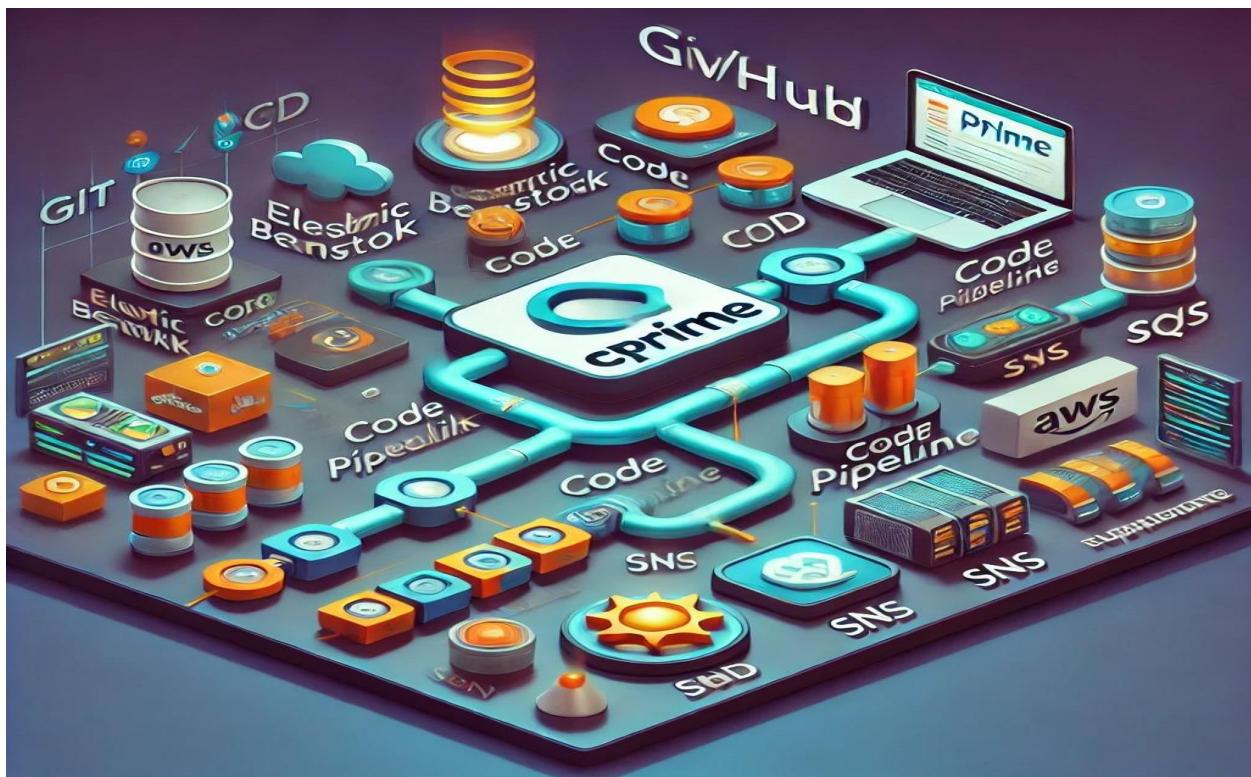


cprime

CI/CD Pipeline with Manual Approval Using AWS and GitHub Integration

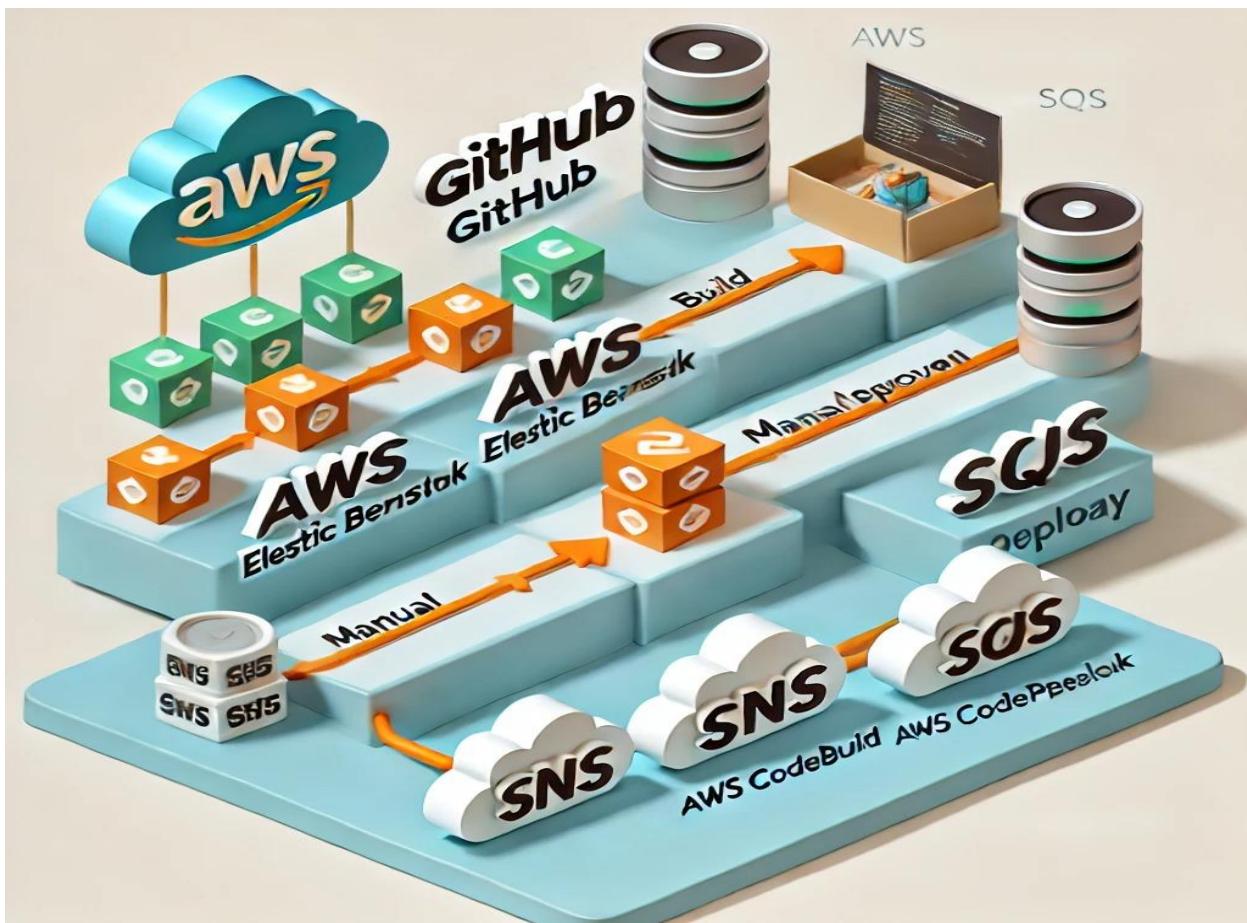


NAME : T MANOJ

EMP ID : LYAKE2KHS

Overview

This project demonstrates how to set up a CI/CD pipeline using GitHub, AWS Elastic Beanstalk, AWS CodeBuild, AWS CodePipeline, SNS, and SQS. The pipeline automatically detects code changes, builds the application, and deploys it to AWS Elastic Beanstalk. A manual approval stage is integrated to ensure critical changes are reviewed before deployment, enhancing control and security.



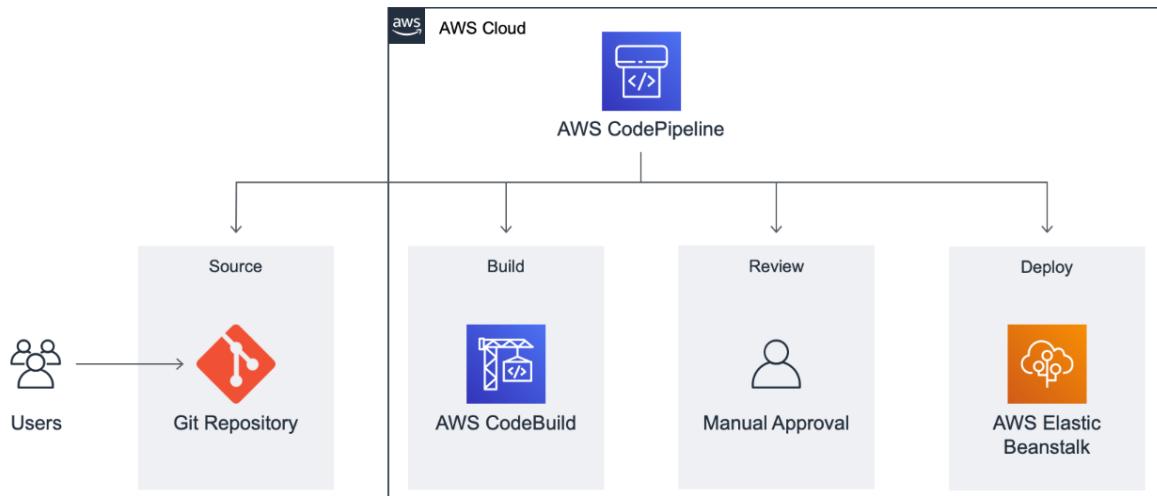
Introduction

Modern software development requires rapid and reliable deployment of applications. Continuous Integration (CI) and Continuous Delivery (CD) pipelines automate this process, reducing manual errors, improving productivity, and ensuring quick delivery of new features.

In this project, you will:

- Host your code in GitHub for easy access and version control.
- Deploy a sample Node.js application using AWS Elastic Beanstalk, which abstracts the underlying infrastructure management.
- Use AWS CodeBuild to compile and test the code automatically.
- Configure AWS CodePipeline to orchestrate the build and deployment processes.
- Add a manual approval step using SNS and SQS to ensure that deployments are reviewed before going live.

CI/CD ARCHITECTURE :



Prerequisites

- GitHub Account
- Git Installation
- AWS Account
- AWS CLI (Configured)
- Text Editor/IDE (VS Code)
- Node.js Runtime
- Elastic Beanstalk Access
- CodeBuild Knowledge
- CodePipeline Integration
- IAM Roles & Permissions
- SNS Topic Setup
- SQS Queue Setup
- OAuth GitHub Integration
- Internet Connection
- Modern Browser (Chrome, Firefox)
- Basic CI/CD Understanding

Project Phases :

- Setting Up a GitHub Repository
- Deploy Web App using AWS Elastic Beanstalk
- Create Build Project using AWS CodeBuild
- Set Up Continuous Delivery Pipeline using AWS CodePipeline
- Adding Manual Approval with SNS & SQS

Setting Up a GitHub Repository

⌚ Objective:

- Create a GitHub repository to store your application code.
- Prepare the repository for integration with AWS services.



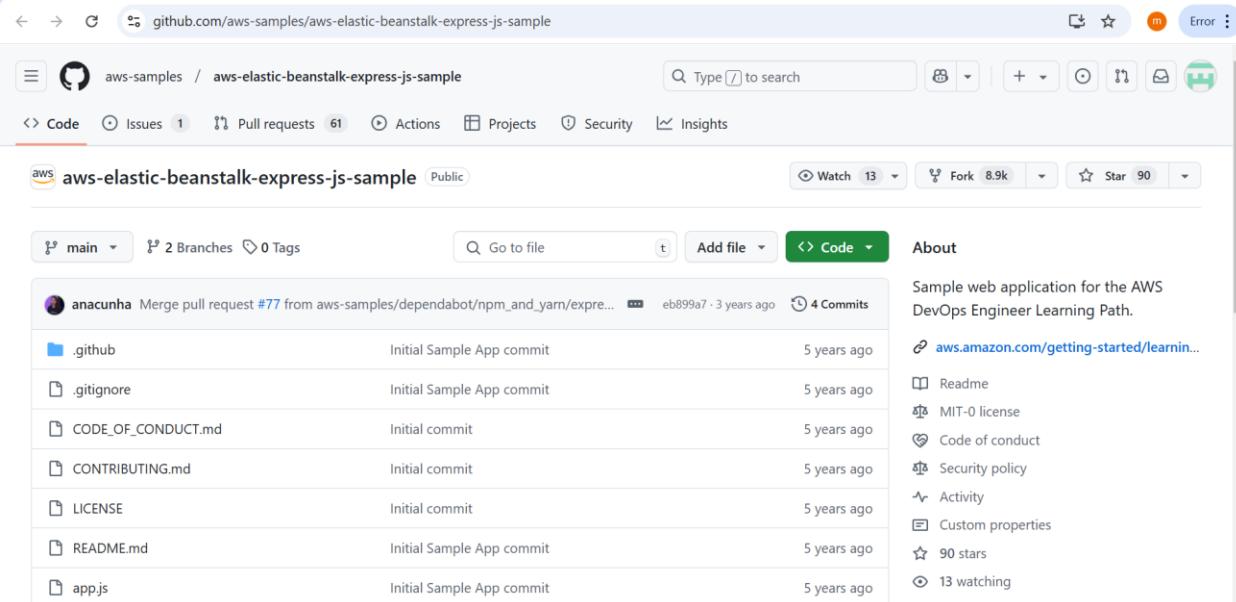
🏷️ Terminologies:

1. GitHub: A web-based platform for version control using Git.
2. Git: Tracks and manages code changes.
3. Repository (Repo): Storage space for your project, including code and version history.
4. Fork: Copying an existing repo to your GitHub account.
5. Clone: Copying the repo from GitHub to your local machine.
6. Commit: Saving changes to the version history.
7. Push: Sending local commits to the GitHub repository.

Steps to Set Up GitHub Repo:

Step 1: Fork a Sample Repository

- Navigate to GitHub and sign in.
- Search for the repository: **aws-elastic-beanstalk-express-js-sample**.
- Click the Fork button (top-right corner) to copy it to your account.



The screenshot shows the GitHub repository page for `aws-elastic-beanstalk-express-js-sample`. The page includes the repository name, a search bar, and various navigation links like Code, Issues, Pull requests, Actions, Projects, Security, and Insights. On the right, there are buttons for Watch, Fork, and Star. Below the header, there's a list of files: `main`, `2 Branches`, `0 Tags`, `Go to file`, `Add file`, and a dropdown for Code. The main content area shows a list of commits from `anacunha`, including `.github`, `.gitignore`, `CODE_OF_CONDUCT.md`, `CONTRIBUTING.md`, `LICENSE`, `README.md`, and `app.js`. Each commit includes the author, message, date, and time. To the right, there's an "About" section with details about the sample web application for AWS DevOps Engineer Learning Path, links to Readme, MIT-0 license, Code of conduct, Security policy, Activity, Custom properties, and metrics like 90 stars and 13 watching.

Step 2: Clone the Repository Locally

1. Go to your forked repository on GitHub.
2. Click the Code button and copy the HTTPS URL.
3. Open your terminal and run:

```
git clone aws-elastic-beanstalk-express-js-sample
```

Local Codespaces

Clone ?

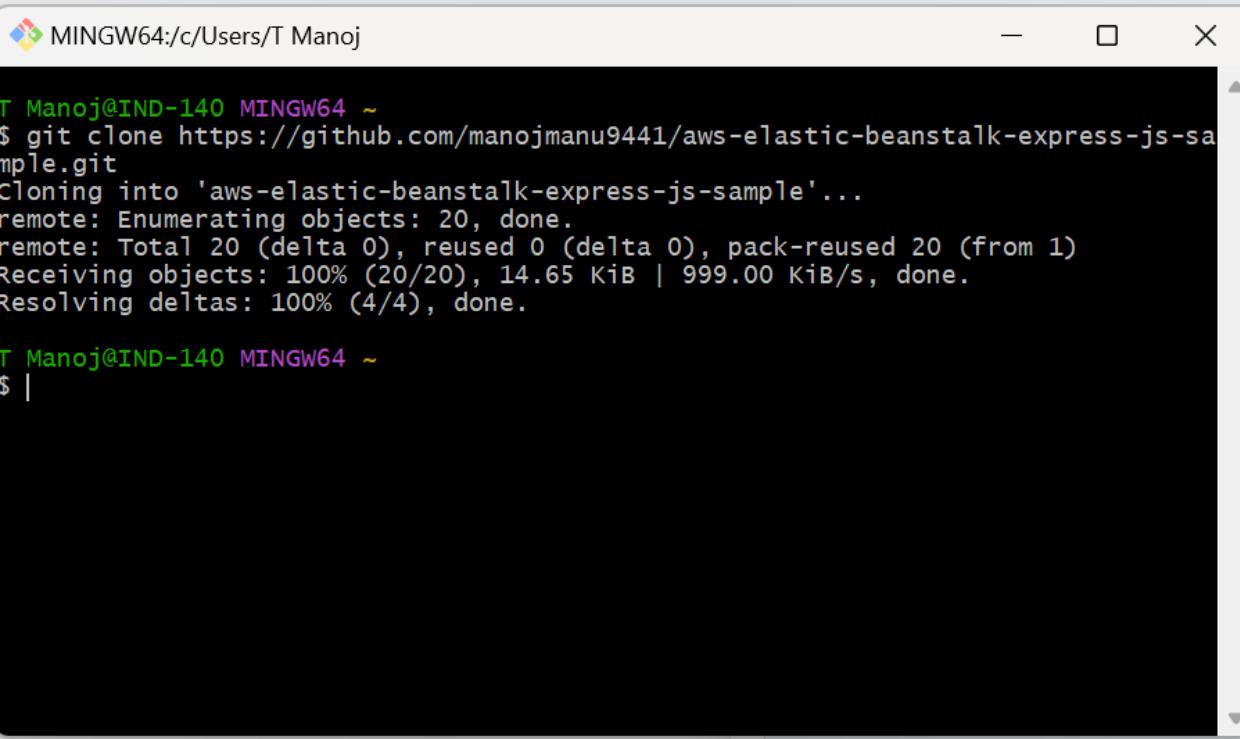
HTTPS SSH GitHub CLI

<https://github.com/aws-samples/aws-elastic-beanstalk-express-javascript-sample.git> 

Clone using the web URL.

 Open with GitHub Desktop

 Download ZIP



```
MINGW64:/c/Users/T Manoj
T Manoj@IND-140 MINGW64 ~
$ git clone https://github.com/manojmanu9441/aws-elastic-beanstalk-express-js-sample.git
Cloning into 'aws-elastic-beanstalk-express-js-sample'...
remote: Enumerating objects: 20, done.
remote: Total 20 (delta 0), reused 0 (delta 0), pack-reused 20 (from 1)
Receiving objects: 100% (20/20), 14.65 KiB | 999.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.

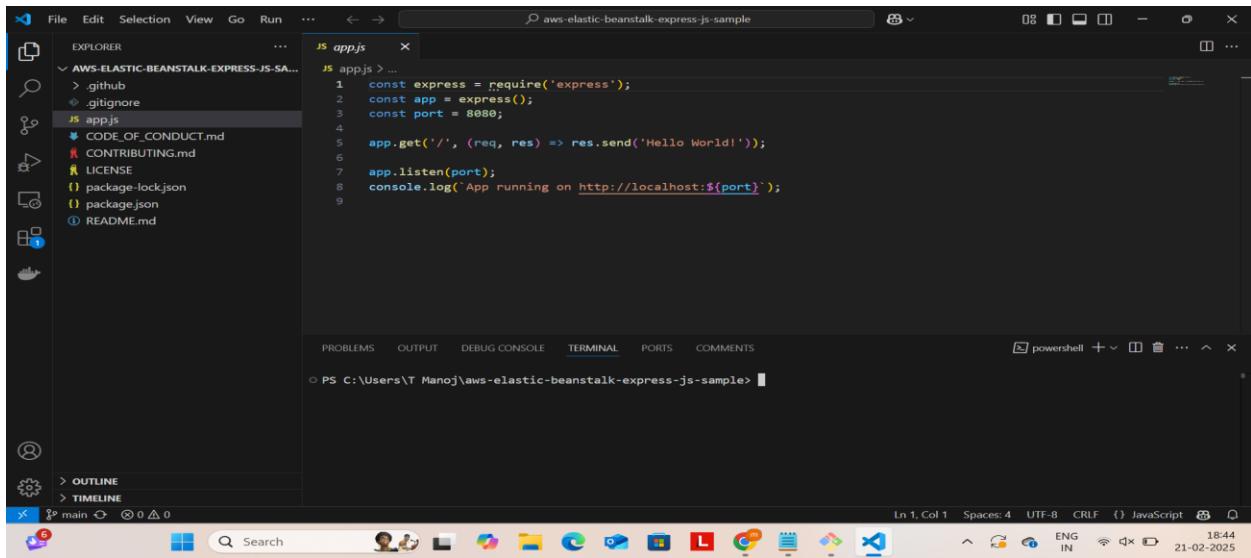
T Manoj@IND-140 MINGW64 ~
$ |
```

Step 3: Make Code Changes

1. Open the project in Visual Studio Code (or your preferred editor).

2. Modify app.js by adding:

```
console.log("Hello World! This is the version 1");
```



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like .github, .gitignore, app.js, CODE_OF_CONDUCT.md, CONTRIBUTING.md, LICENSE, package-lock.json, package.json, and README.md.
- Code Editor:** The app.js file is open, displaying the following code:

```
1 const express = require('express');
2 const app = express();
3 const port = 8080;
4
5 app.get('/', (req, res) => res.send('Hello World!'));
6
7 app.listen(port);
8 console.log(`App running on http://localhost:${port}`);
```
- Terminal:** The terminal shows the command PS C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample>, indicating the current working directory.
- Bottom Bar:** Includes icons for file operations, search, and various extensions, along with system status indicators like battery level, signal strength, and date/time (21-02-2025).

Step 4: Commit and Push Changes

```
git add .
git commit -m "Initial Commit"
git push
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "AWS-ELASTIC-BEANSTALK-EXPRESS-JS-SA...".
- Code Editor:** Displays the file "app.js" containing the following code:

```
JS app.js M x
JS app.js > ...
1 const express = require('express');
2 const app = express();
3 const port = 8080;
4
5 app.get('/', (req, res) => res.send('Hello World! hi this is version 1'));
6
7 app.listen(port);
8 console.log(`App running on http://localhost:${port}`);
9
```

- Terminal:** Shows the command "git add ." being run in the PowerShell terminal.
- Bottom Status Bar:** Displays the file path "C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample", line count "Ln 9, Col 1", spaces "Spaces: 4", encoding "UTF-8", and date "21-02-2025".

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "AWS-ELASTIC-BEANSTALK-EXPRESS-JS-SA...".
- Code Editor:** Displays the file "app.js" containing the same code as the first screenshot.
- Terminal:** Shows the commands "git add ." and "git commit -m "initial commit"" being run in the PowerShell terminal.
- Bottom Status Bar:** Displays the file path "C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample", line count "Ln 9, Col 1", spaces "Spaces: 4", encoding "UTF-8", and date "21-02-2025".

The screenshot shows a Windows desktop environment with the Visual Studio Code application open. The code editor displays the file `app.js` which contains a simple Express.js application. The terminal below shows the output of a `git push` command, indicating successful deployment to a GitHub repository. The status bar at the bottom right shows the date and time as 21-02-2025.

```
const express = require('express');
const app = express();
const port = 8080;

app.get('/', (req, res) => res.send('Hello World! hi this is version 1'));

app.listen(port);
console.log(`App running on http://localhost:${port}`);
```

```
PS C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample> git push
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 309 bytes | 154.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/manojmanu9441/aws-elastic-beanstalk-express-js-sample.git
```

Step 5: Verify Changes

- Go to GitHub → Your Repository → Check the `app.js` file for the update.

The screenshot shows a code editor interface with the `app.js` file open. The code is identical to the one shown in the previous screenshot, confirming the changes have been pushed to the repository.

```
const express = require('express');
const app = express();
const port = 8080;

app.get('/', (req, res) => res.send('Hello World! hi this is version 1'));

app.listen(port);
console.log(`App running on http://localhost:${port}`);
```

Deploy Web App using AWS Elastic Beanstalk

⌚ Objective:

- Deploy the Node.js web app using AWS Elastic Beanstalk.



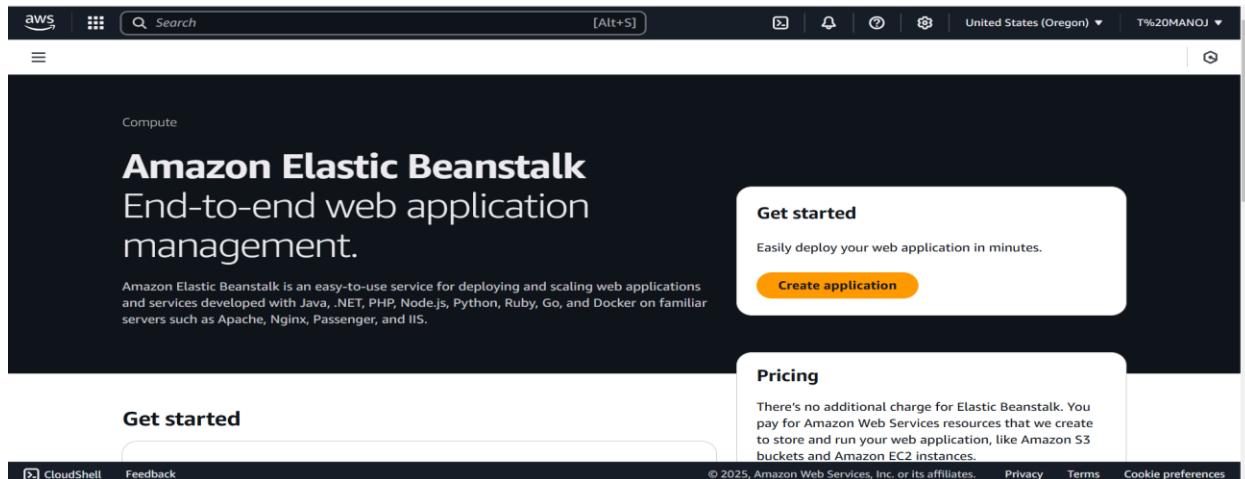
🏷️ Terminologies:

1. Elastic Beanstalk (EB): Platform-as-a-service (PaaS) that deploys apps without managing infrastructure.
2. Environment: Set of AWS resources (like EC2, S3, Load Balancers) to run your application.
3. Platform: Runtime environment (Node.js, Python, etc.) for the app.
4. EC2 Instance: Virtual machine running your application.

⚙️ Steps to Deploy Using Elastic Beanstalk:

Step 1: Open Elastic Beanstalk Console

- Sign in to the AWS Management Console.
- Search and select Elastic Beanstalk.



Step 2: Create a New Application

1. Click Create Application.
2. Application Name: DevOpsGettingStarted.
3. Platform: Choose Node.js.
4. Application Code: Select Sample application.
5. Environment: Select Single instance (free tier eligible).
6. Click Next.

aws | Search [Alt+S] | United States (Oregon) ▾ | T%20MANOJ ▾

☰ | ⓘ | ⓘ

Step 1
Configure environment
Step 2
Configure service access
Step 3 - *optional*
Set up networking, database, and tags
Step 4 - *optional*
Configure instance traffic and scaling
Step 5 - *optional*
Configure updates, monitoring, and logging
Step 6
Review

Configure environment Info

Environment tier Info
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

Web server environment
Run a website, web application, or web API that serves HTTP requests. [Learn more](#)

Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

Application information Info

Application name

Maximum length of 100 characters.

▶ **Application tags (optional)**

CloudShell Feedback | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

aws | Search [Alt+S] | United States (Oregon) ▾ | T%20MANOJ ▾

☰ | ⓘ | ⓘ

Environment information Info

Choose the name, subdomain and description for your environment. These cannot be changed later.

Environment name

Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

Domain
 .us-west-2.elasticbeanstalk.com

Environment description

Platform Info

CloudShell Feedback | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

The screenshot shows the 'Platform' configuration step of the AWS Elastic Beanstalk setup wizard. It includes fields for Platform type (Managed platform selected), Platform (Node.js), Platform branch (Node.js 22 running on 64bit Amazon Linux 2023), and Platform version (6.4.2 (Recommended)).

The screenshot shows the 'Application code' configuration step. It includes options for Application code type (Sample application selected), Presets (Info), Configuration presets (Single instance (free tier eligible) selected), and a note about starting from a preset or choosing custom configuration.

Step 3: Configure Service Access

1. Service Role: Choose `aws-elasticbeanstalk-service-role` (create if not available).
2. EC2 Instance Profile: Select `aws-elasticbeanstalk-ec2-role`.
3. Key Pair: Create a new one or select existing for SSH access.



The screenshot shows the AWS IAM Policy editor interface. On the left, a sidebar indicates "Step 1 Specify permissions" is selected. The main area has a title "Specify permissions" with an "Info" link. Below it is a sub-instruction: "Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor." The central feature is the "Policy editor" with a JSON code editor. The JSON code is as follows:

```
1 {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Action": [  
6                 "s3:Get*",  
7                 "s3>List*"  
8             ],  
9             "Effect": "Allow",  
10            "Resource": "*"  
11        }  
12    ]  
13 }
```

To the right of the JSON editor are three tabs: "Visual", "JSON" (which is selected), and "Actions". Below these tabs is a button labeled "Edit statement". To the right of the JSON editor is a panel titled "Select a statement" with the instruction "Select an existing statement in the policy or add a new statement." At the bottom right of this panel is a blue button labeled "+ Add new statement".

aws | Search [Alt+S] | Global ▾ T%20MANOJ ▾

☰ IAM > Policies > Create policy

Step 2 Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+-=_,@-_-' characters.

Description - optional
Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+-=_,@-_-' characters.

Permissions defined in this policy

Edit

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Screenshot of the AWS IAM 'Create role' wizard Step 1: Select trusted entity.

The sidebar shows three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). Step 1 is selected.

The main content area is titled 'Select trusted entity' with a 'Trusted entity type' section. It contains five options:

- AWS service** (selected): Allows AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account**: Allows entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation**: Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy**: Create a custom trust policy to enable others to perform actions in this account.

Screenshot of the AWS IAM 'Create role' wizard Step 2: Service or use case selection.

The sidebar shows the same three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). Step 2 is selected.

The main content area is titled 'Service or use case'. A dropdown menu is open, showing 'EC2' as the selected option.

Below the dropdown, it says 'Choose a use case for the specified service.' and 'Use case'.

The following use cases are listed, with 'EC2' selected:

- EC2** (selected): Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager**: Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role**: Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- EC2 - Spot Fleet Auto Scaling**: Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
- EC2 - Spot Fleet Tagging**: Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
- EC2 - Spot Instances**: Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- EC2 - Spot Fleet**: Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.
- EC2 - Scheduled Instances**: Allows EC2 Scheduled Instances to manage instances on your behalf.

The screenshot shows the 'Create role' wizard in the AWS IAM console. The current step is 'Step 3: Name, review, and create'. On the left, a sidebar lists 'Add permissions' and 'Name, review, and create'. The main area is titled 'Role details' and contains fields for 'Role name' (set to 'AWSCIDROLE') and 'Description' (set to 'Allows EC2 instances to call AWS services on your behalf'). Below these fields is a code editor showing a JSON trust policy:

```
1  {  
2      "Version": "2012-10-17",  
3      "Statement": [  
4          {  
5              "Action": "sts:AssumeRole",  
6              "Effect": "Allow",  
7              "Principal": "ec2.amazonaws.com"  
8          }  
9      ]  
10 }
```

At the bottom of the page, there is a 'Step 1: Select trusted entities' section with an 'Edit' button, and a 'Step 2: Add permissions' section which is currently empty.

The screenshot shows the 'Create role' wizard in the AWS IAM console. The current step is 'Step 2: Add permissions'. The main area is titled 'Permissions policy summary' and displays two policies:

Policy name	Type	Attached as
AmazonSSMManagedEC2InstanceDefaultPolicy	AWS managed	Permissions policy
AWSCID	Customer managed	Permissions policy

Below this, there is a 'Step 3: Add tags' section with an 'Edit' button, and a 'Step 4: Deploy and Monitor' section which is currently empty.

Step 4: Deploy and Monitor

1. Click Submit.
2. Monitor deployment (takes ~5 minutes).
3. On success, a green checkmark appears, and a URL is generated.

aws | Search [Alt+S] | United States (N. Virginia) | T%20MANOJ

Configure instance traffic and scaling
Step 5 - optional Configure updates, monitoring, and logging
Step 6 Review

Use an existing service role
Service role name
Enter the name for an IAM role that Elastic Beanstalk will create to assume as a service role. Beanstalk will attach the required managed policies to it.
aws-elasticbeanstalk-service-role
[View permission details](#)

EC2 key pair
Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)
RSA

EC2 instance profile
Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.
AWSCLCIDROLE
[View permission details](#)

[Cancel](#) [Skip to review](#) [Previous](#) **Next**

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws | Search [Alt+S] | United States (N. Virginia) | T%20MANOJ

Configure environment
Configure service access
Step 3 - optional Set up networking, database, and tags
Step 4 - optional **Configure instance traffic and scaling**
Step 5 - optional Configure updates, monitoring, and logging
Step 6 Review

Configure instance traffic and scaling - *optional* Info

Instances Info
Configure the Amazon EC2 instances that run your application.

Root volume (boot device)

Root volume type
General Purpose 3(SSD)

Size
The number of gigabytes of the root volume attached to each instance.
10 GB

IOPS
Input/output operations per second for a provisioned IOPS (SSD) volume.
3000 IOPS

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the 'Review' step of the configuration wizard. On the left, a vertical list of steps is shown:

- Step 1: Configure environment
- Step 2: Configure service access
- Step 3 - optional: Set up networking, database, and tags
- Step 4 - optional: Configure instance traffic and scaling
- Step 5 - optional: Configure updates, monitoring, and logging
- Step 6: Review

The main area is titled 'Step 1: Configure environment' with an 'Edit' button. It contains 'Environment information' fields:

Environment tier	Application name
Web server environment	DevOpsGettingStarted
Environment name	Application code
DevOpsGettingStarted-env	Sample application
Platform	
arnaws:elasticbeanstalk:us-east-1::platform/Node.js 22 running on 64bit Amazon Linux 2023/6.4.2	

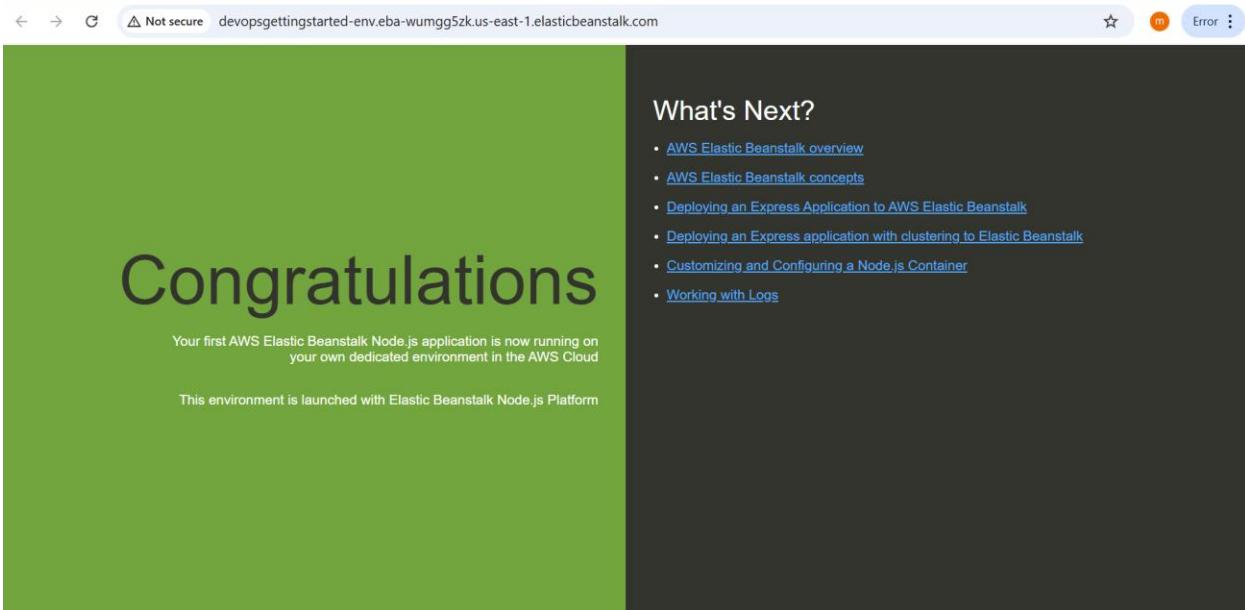
The screenshot shows the 'Environment successfully launched.' message. The 'DevOpsGettingStarted-env' environment details are displayed:

Environment overview	Platform
Health: Pending	Environment ID: e-nf2dxmljp3p
Domain: DevOpsGettingStarted-env.eba-wunng5z.us-east-1.elasticbeanstalk.com	Application name: DevOpsGettingStarted
	Platform: Node.js 22 running on 64bit Amazon Linux 2023/6.4.2
	Running version: -
	Platform state: Supported

Below the environment details, there are tabs for Events, Health, Logs, Monitoring, Alarms, Managed updates, and Tags. The Events tab is selected.

Step 5: Verify the Application

- Open the generated URL in your browser; you should see the default Elastic Beanstalk page.

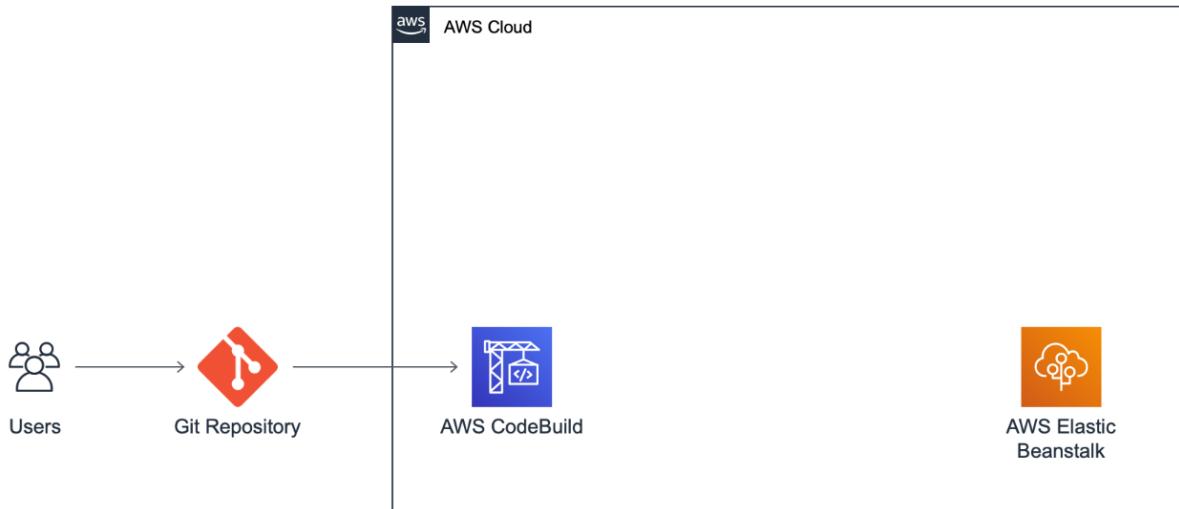


VERIFIED SAMPLE OUTPUT

Create Build Project using AWS CodeBuild

⌚ Objective:

- Automate building the codebase from GitHub using AWS CodeBuild.



💡 Terminologies:

1. AWS CodeBuild: Fully managed build service that compiles source code, runs tests, and produces ready-to-deploy packages.
2. Buildspec File: `buildspec.yml` file that defines build commands.
3. Continuous Integration (CI): Regular merging of code changes into the main branch with automated builds.

⚙️ Steps to Create Build Project:

Step 1: Open AWS CodeBuild Console

- Search for AWS CodeBuild in the AWS console.

The screenshot shows the AWS CodeBuild service landing page. The top navigation bar includes the AWS logo, a services menu, a search bar, and account information for United States (Oregon). On the left, a sidebar titled 'Developer Tools' has a 'CodeBuild' section expanded, showing options like Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Getting started, Build projects, Build history, Report groups, Report history, Compute fleets (New), Account metrics, Related integrations (Jenkins), CloudShell, and Feedback.

The main content area features a large heading 'AWS CodeBuild' with the tagline 'Build and test code with elastic scaling. Pay only for the build time you use.' Below this is a detailed description of what AWS CodeBuild is, mentioning it's a fully managed continuous integration service that compiles source code, runs tests, and produces software packages. It highlights the benefits of elastic scaling and cost efficiency. A prominent 'Create AWS CodeBuild project' button is located on the right.

Step 2: Create a New Build Project

1. Click Create Build Project.
2. Project Name: BuildDevOpsGettingStarted.
3. Source Provider: GitHub.
4. GitHub Connection:
 - a. Click Connect to GitHub via OAuth.
 - b. Select the forked repository.

The screenshot shows the 'Create build project' configuration page. The top navigation bar is identical to the previous screenshot. The main content is titled 'Create build project' and contains a 'Project configuration' section. It includes fields for 'Project name' (Build-DevOpsGettingStarted) and 'Project type'. Under 'Project type', there are two options: 'Default project' (selected) and 'Runner project'. The 'Default project' description states: 'Create a custom CodeBuild project.' The 'Runner project' description states: 'Create a CodeBuild managed runner for workflows in GitHub Actions, GitHub Enterprise Actions, GitLab, or Buildkite.' At the bottom, there's an 'Additional configuration' section with a note about description, public build access, build badge, concurrent build limit, and tags.

AWS Services Search [Alt+S] United States (Oregon) T%20MANOJ

Buildkite.

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

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

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

AWS Services Search [Alt+S] United States (Oregon) T%20MANOJ

Manage default source credential

Source Provider 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

Connection

You can create a new GitHub connection by using an AWS managed GitHub App

Save

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Services More ▾

Developer Tools > Connections > Create connection

Create a connection Info

Create GitHub App connection Info

Connection name

▶ Tags - optional

Connect to GitHub

aws Services Search [Alt+S] United States (Oregon) T%20MANOJ ▾

Manage default source credential

Source Provider

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

Connection

You can create a new GitHub connection by using an AWS managed GitHub App

Save

Source provider: GitHub

Credential: Your account is successfully connected by using an AWS managed GitHub App. Manage account credentials.

Repository:

- Repository in my GitHub account
- Public repository
- GitHub scoped webhook

Repository: https://github.com/manojmanu9441/aws-elastic-beanstalk-express-js-sample

Source version - optional [Info](#): Enter a pull request, branch, commit ID, tag, or reference and a commit ID.

Additional configuration: Git clone depth, Git submodules, Build status config

Runtime(s): Standard

Image: aws/codebuild/amazonlinux-x86_64-standard:5.0

Image version: Always use the latest image for this runtime version

Use GPU-enhanced compute:

Service role:

- New service role: Create a service role in your account
- Existing service role: Choose an existing service role from your account

Role name: codebuild-Build-DevOpsGettingStarted-service-role

Step 3: Configure Build Environment

1. Environment Image: Managed image.
2. Operating System: AmazonLinux 5.0.
3. Runtime: Node.js (Choose version 14+).
4. Buildspec: Create a file named `buildspec.yml` in your repository:

JSON :

```
version: 0.2
phases:
```

```

build:
  commands:
    - npm i --save
artifacts:
  files:
    - '**/*'

```

The screenshot shows the AWS CloudWatch Logs configuration page. Under the 'Buildspec' section, the 'Insert build commands' option is selected. The buildspec YAML code is displayed:

```

version: 0.2
phases:
  build:
    commands:
      - npm i --save
  artifacts:
    files:
      - '**/*'

```

The screenshot shows the AWS CloudWatch Logs configuration page. Under the 'CloudWatch' section, the 'CloudWatch logs - optional' checkbox is checked. The 'Group name - optional' field contains 'aws/codebuild/Build-DevOpsGettingStarted'. The 'Stream name prefix - optional' field is empty. Under the 'S3' section, the 'S3 logs - optional' checkbox is unchecked. At the bottom, there are 'Cancel' and 'Create build project' buttons.

Step 4: Build the Project

- Click Start Build.

- Confirm the build runs successfully (Green ✓ checkmark).

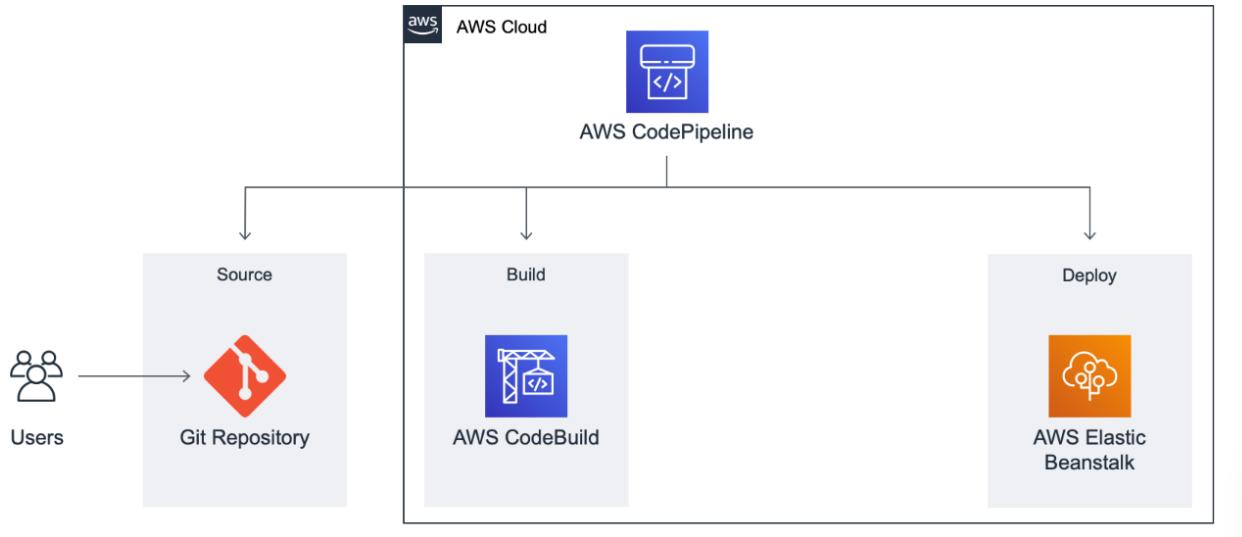
The screenshot shows the AWS CodeBuild console. On the left, a sidebar menu includes options like Source, Artifacts, Build, and Build projects. Under Build projects, 'Build project' is selected. The main area displays a green success message: 'Project created' with the note 'You have successfully created the following project: Build-DevOpsGettingStarted'. Below this, the project name 'Build-DevOpsGettingStarted' is shown along with a breadcrumb trail: Developer Tools > CodeBuild > Build projects > Build-DevOpsGettingStarted. A row of buttons includes 'Actions', 'Create trigger', 'Edit', 'Clone', 'Debug build', and 'Start build with overrides'. A prominent orange 'Start build' button is highlighted. The 'Configuration' section details the source provider as GitHub, the primary repository as 'manojmanu9441/aws-elastic-beanstalk-express-js-sample', and the service role as 'arn:aws:iam::491085415620:role/service-role/codebuild-
Build-DevOpsGettingStarted-service-role'. At the bottom, there's a footer with links to CloudShell, Feedback, and various AWS terms.

This screenshot shows the AWS CodeBuild console for the project 'Build-DevOpsGettingStarted'. The sidebar is identical to the previous screenshot. The main area displays a build summary with the identifier 'Build-DevOpsGettingStarted:10977651-f589-480a-a4ab-f148bcd9ea1'. It features a 'Stop build' button and a prominent orange 'Retry build' button. The 'Build status' section shows a single build entry with the status 'Succeeded'. The build details include the resolved source version '5d04c878aec9f499465bafb54d0aec85d 2475851', start time 'Feb 21, 2025 7:48 PM (UTC+5:30)', end time 'Feb 21, 2025 7:49 PM (UTC+5:30)', and build number '1'. The build ARN is listed as 'arn:aws:codebuild:us-west-2:491085415620:build/Build-DevOpsGettingStarted:10977651-f589-480a-a4ab-f148bcd9ea1'. The footer contains standard AWS links for CloudShell, Feedback, and legal information.

Set Up Continuous Delivery Pipeline using AWS CodePipeline

⌚ Objective:

- Create a CI/CD pipeline using CodePipeline for automated deployment.



❖ Terminologies:

1. AWS CodePipeline: Orchestrates continuous integration and deployment workflows.
2. Pipeline Stages:
 - a. Source: Code retrieval.
 - b. Build: Code compilation (CodeBuild).
 - c. Deploy: Code deployment (Elastic Beanstalk).

⚙️ Steps to Create CI/CD Pipeline:

Step 1: Access CodePipeline Console

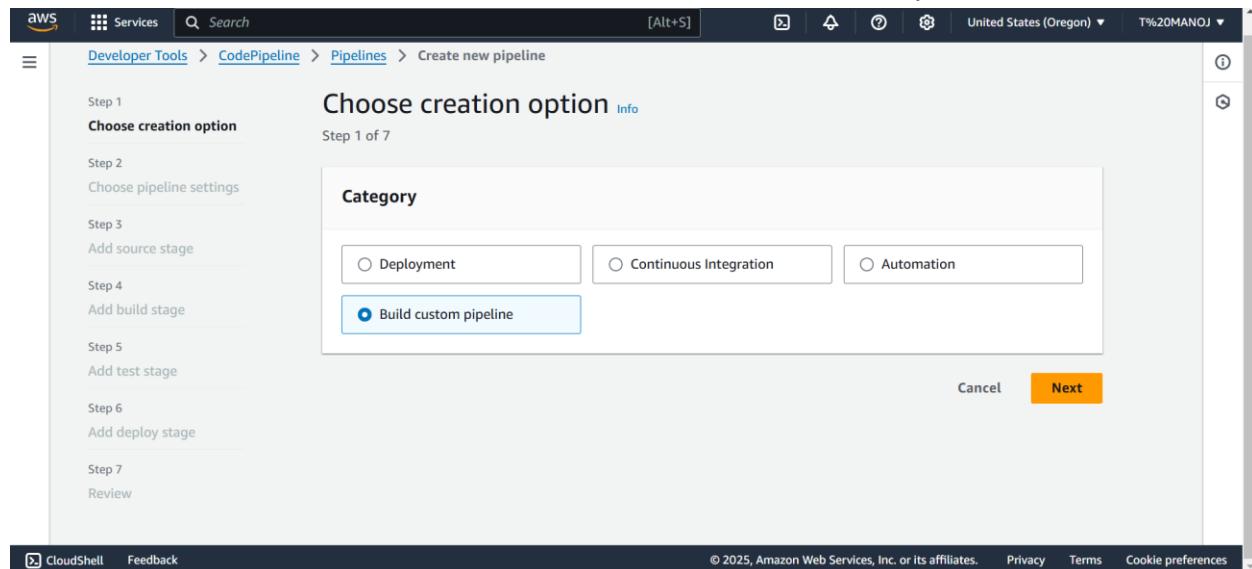
- Navigate to AWS CodePipeline in the AWS console.

- Click Create Pipeline.



Step 2: Configure Pipeline

1. Pipeline Name: Pipeline-DevOpsGettingStarted.
2. Select New service role or existing service role (AWSCodePipelineServiceRole).



Step 2 of 7

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
Pipeline-DevOpsGettingStarted

No more than 100 characters

Execution mode Info
Choose the execution mode for your pipeline. This determines how the pipeline is run.
 Superseded
 Queued
 Parallel

Service role
 New service role
Create a service role in your account
 Existing service role
Choose an existing service role from your account

Role name

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 3: Add Source Stage

1. Source Provider: GitHub.
2. Repository: Select your GitHub repo.
3. Branch: main.
4. Change Detection: Select GitHub webhooks (auto-triggers builds).

Step 3 of 7

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (via GitHub App)

Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
arn:aws:codeconnections:us-west-2:491085415620:connection/7cdE or

Repository name
Choose a repository in your GitHub account.
manojmanu9441/aws-elastic-beanstalk-express-js-sample

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

Default branch
Default branch will be used only when pipeline execution starts from a different source or manually started.
main

Output artifact format

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 4: Add Build Stage

1. Build Provider: AWS CodeBuild.
2. Project Name: DevOpsGettingstarted.

Webhook events

Webhook - optional

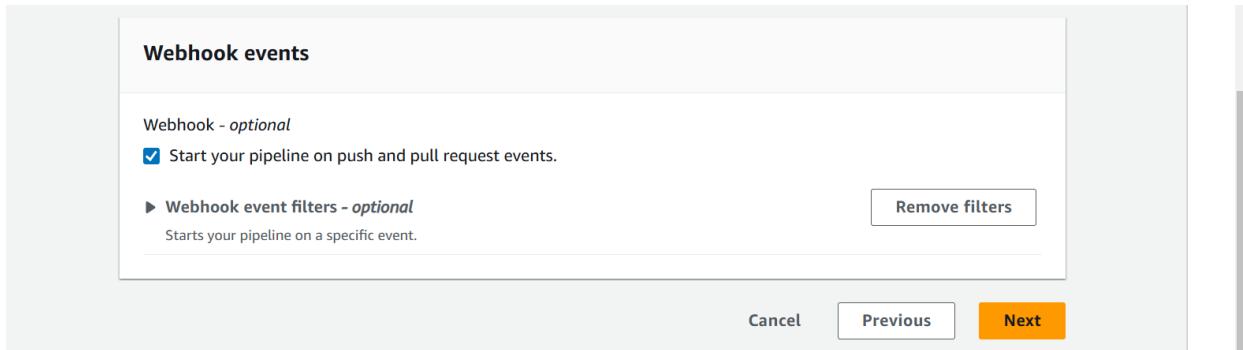
Start your pipeline on push and pull request events.

► **Webhook event filters - optional**

Starts your pipeline on a specific event.

Remove filters

Cancel **Previous** **Next**



Step 2 Choose pipeline settings

Build - optional

Build provider

Choose the tool you want to use to run build commands and specify artifacts for your build action.

Commands Other build providers

AWS CodeBuild

Project name

Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

Q Build-DevOpsGettingStarted or Create project

Environment variables - optional

Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Add environment variable

Build type

Add environment variable

Build type

Single build Triggers a single build. Batch build Triggers multiple builds as a single execution.

Region

United States (Oregon)

Input artifacts

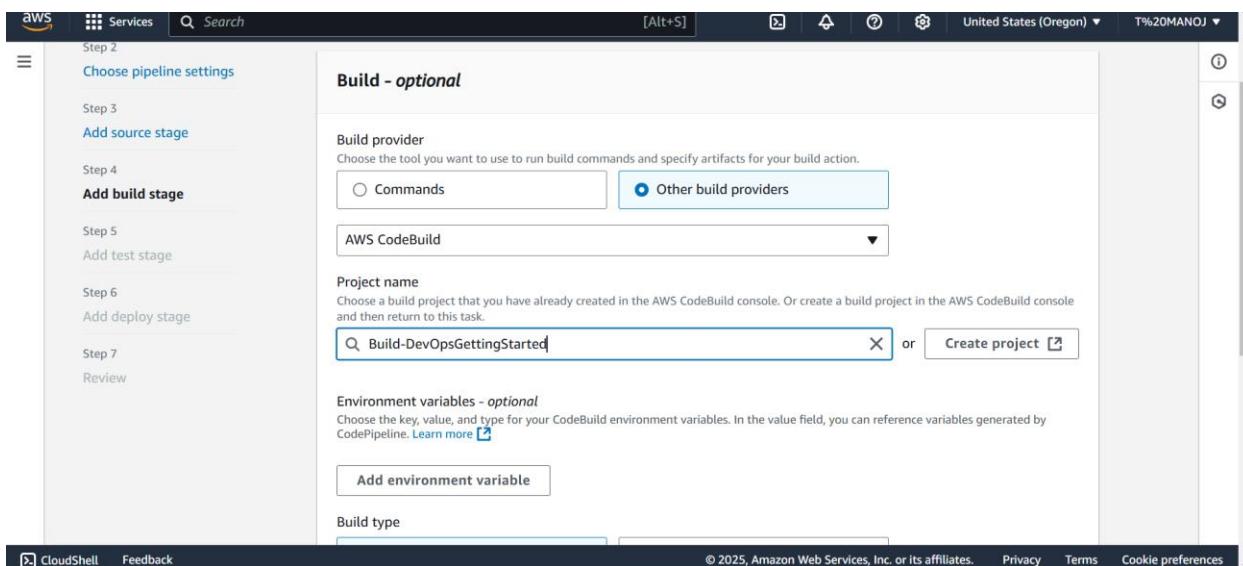
Choose an input artifact for this action. [Learn more](#)

SourceArtifact

Defined by: Source

Enable automatic retry on stage failure

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Add environment variable

Build type

Single build Triggers a single build. Batch build Triggers multiple builds as a single execution.

Region

United States (Oregon)

Input artifacts

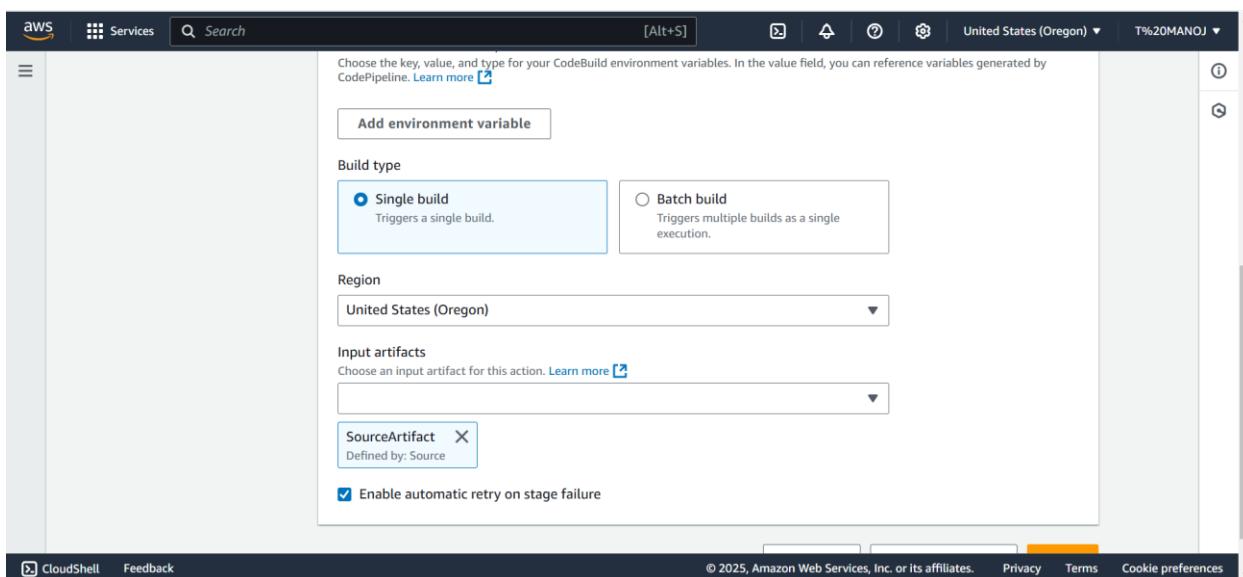
Choose an input artifact for this action. [Learn more](#)

SourceArtifact

Defined by: Source

Enable automatic retry on stage failure

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Step 5: Add Deploy Stage

1. Deploy Provider: AWS Elastic Beanstalk.
2. Application Name: DevOpsGettingStarted.
3. Environment Name: Select the environment created earlier.

The screenshot shows the AWS Pipeline configuration interface. On the left, a sidebar lists steps from Step 3 to Step 7. Step 5, 'Add test stage', is currently selected. The main panel is titled 'Test - optional' and contains fields for 'Test provider' (set to 'AWS CodeBuild'), 'Region' (set to 'United States (Oregon)'), and 'Input artifacts'. A dropdown menu shows 'BuildArtifact' selected, with a note 'Defined by: Build'. Below this, there's a 'Project name' field containing 'Build-DevOpsGettingStarted' and a 'Create project' button. At the bottom, there are 'Cancel', 'Previous', 'Skip deploy stage', and 'Next' buttons.

The screenshot shows the AWS Pipeline configuration interface. Step 6, 'Add deploy stage', is selected. The main panel has an 'Input artifacts' section with 'BuildArtifact' selected. It then moves to the 'Application name' section, where 'DevOpsGettingStarted' is entered. The 'Environment name' section follows, with 'DevOpsGettingStarted-env' entered. Under deployment settings, 'Configure automatic rollback on stage failure' is checked, while 'Enable automatic retry on stage failure' is unchecked. At the bottom, there are 'Cancel', 'Previous', 'Skip deploy stage', and 'Next' buttons.

Step 6: Finalize and Execute

- Click Create Pipeline.
- The pipeline runs automatically on code pushes.

AWS Services Search [Alt+S] United States (Oregon) T%20MANOJ

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose creation option Review Info Step 7 of 7

Step 2 Choose pipeline settings Step 2: Choose pipeline settings

Step 3 Add source stage Pipeline settings

Step 4 Add build stage Pipeline name Pipeline-DevOpsGettingStarted

Step 5 Add test stage Pipeline type V2

Step 6 Add deploy stage Execution mode QUEUED

Step 7 Review Artifact location A new Amazon S3 bucket will be created as the default artifact store for your pipeline

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 2012 IN ENG 21-02-2025

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 2012 IN ENG 21-02-2025

Developer Tools > CodePipeline > Pipelines > Pipeline-DevOpsGettingStarted

Pipeline-DevOpsGettingStarted

Pipeline type: V2 Execution mode: QUEUED

Source In progress
Pipeline execution ID: [99dcc729-5f89-4bd8-b8eb-3cb67cbc66bc](#)

Source
In progress - Just now
[View details](#)

Edit Stop execution Clone pipeline Release change

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Pipeline-DevOpsGettingStarted

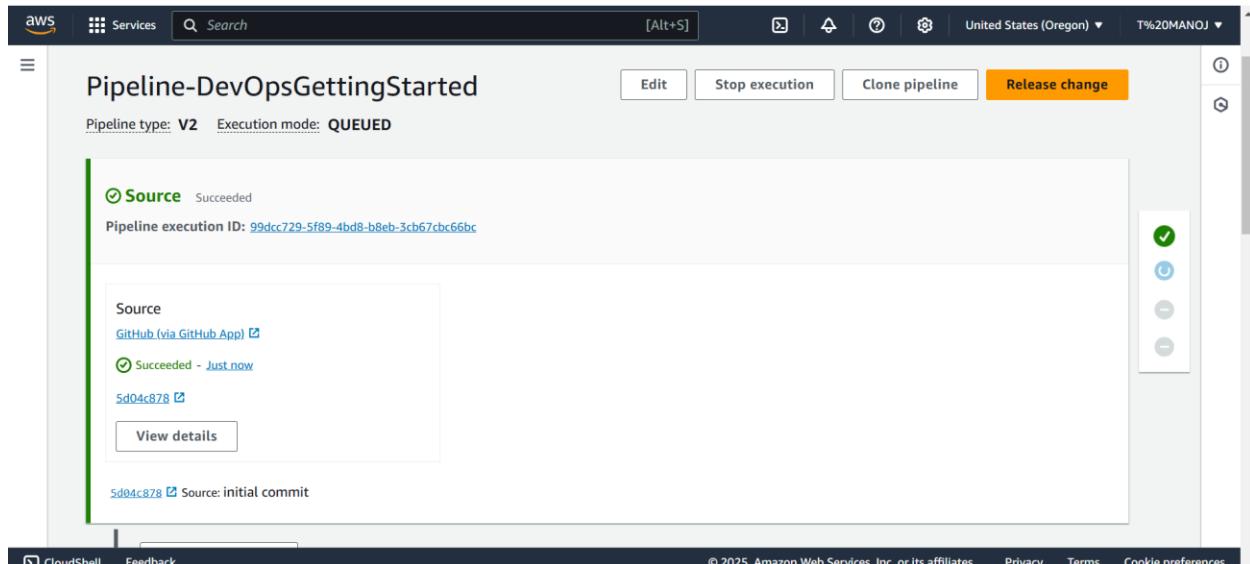
Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: [99dcc729-5f89-4bd8-b8eb-3cb67cbc66bc](#)

Source
[GitHub \(via GitHub App\)](#)
Succeeded - Just now
[5d04c878](#)
[View details](#)

5d04c878 Source: initial commit

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



This screenshot shows the first stage of the pipeline, "Source". It indicates a successful execution via GitHub (via GitHub App) just now. A specific commit, 5d04c878, is highlighted as the source of the initial commit. The pipeline execution ID is 99dcc729-5f89-4bd8-b8eb-3cb67cbc66bc.

Build Succeeded
Pipeline execution ID: [99dcc729-5f89-4bd8-b8eb-3cb67cbc66bc](#)

Start rollback

Build
[AWS CodeBuild](#)
Succeeded - Just now
[View details](#)

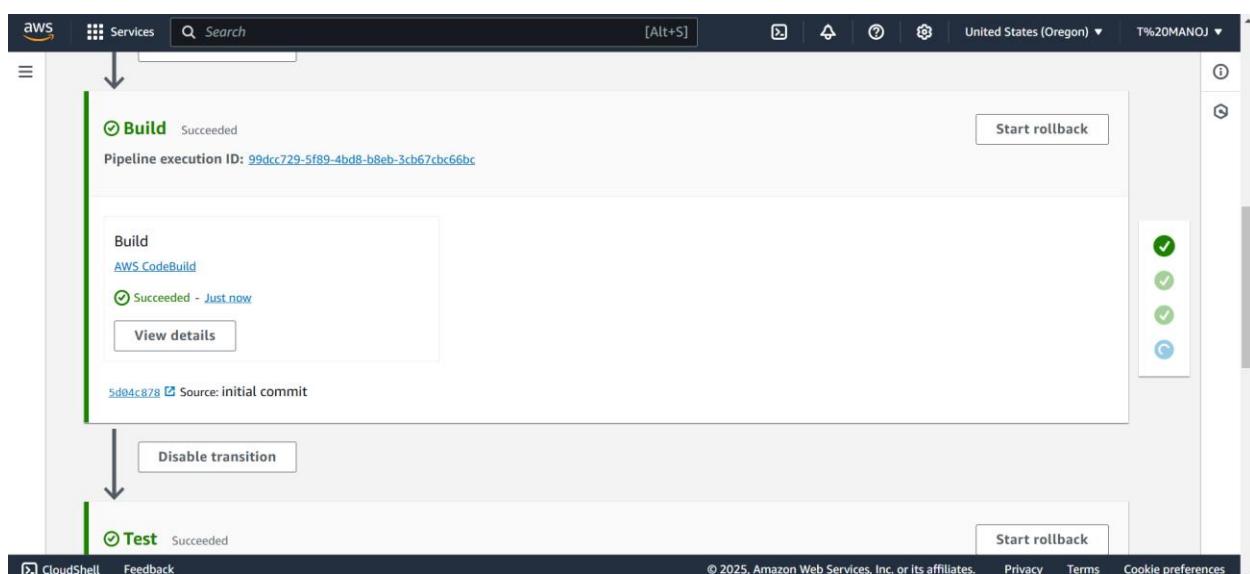
5d04c878 Source: initial commit

[Disable transition](#)

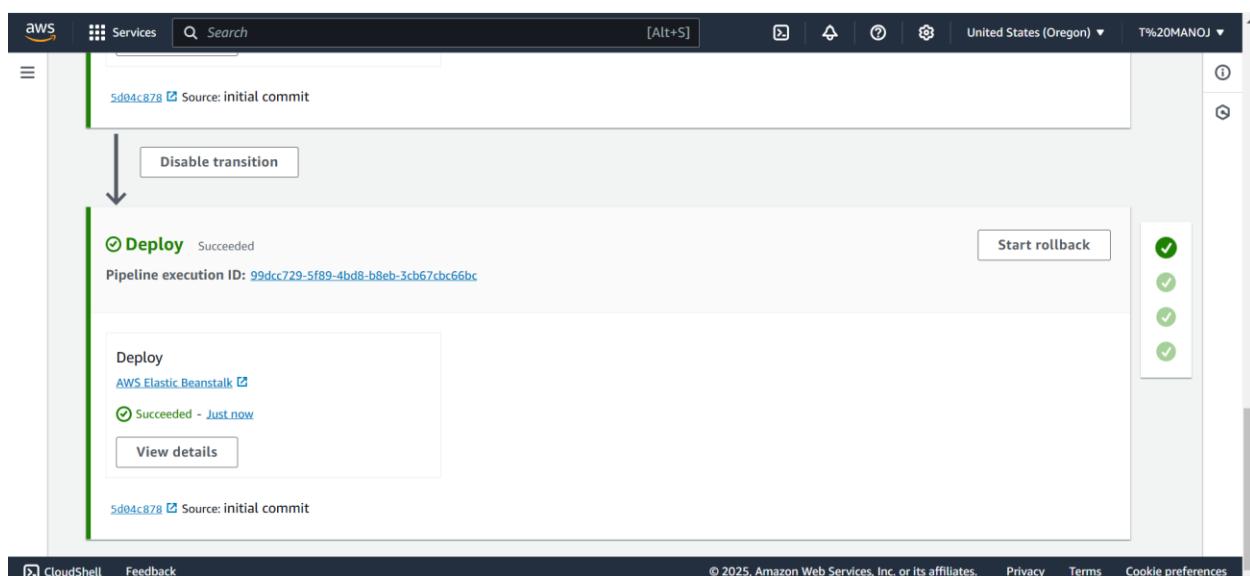
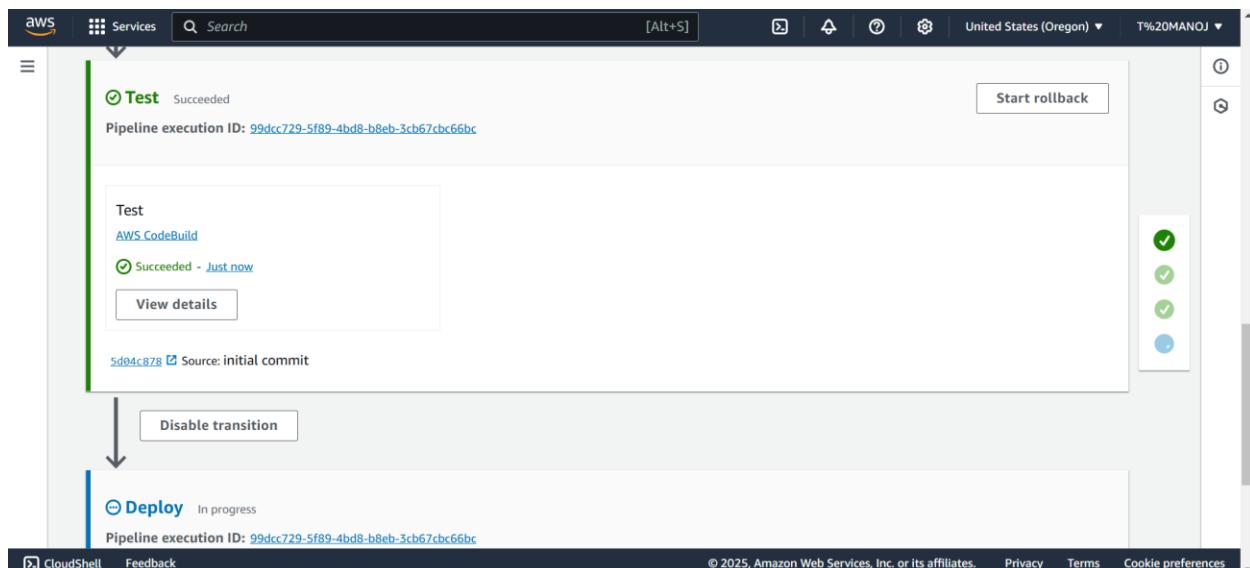
Test Succeeded
Pipeline execution ID: [99dcc729-5f89-4bd8-b8eb-3cb67cbc66bc](#)

Start rollback

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



This screenshot shows the "Build" and "Test" stages of the pipeline. Both have succeeded. The "Build" stage was performed by AWS CodeBuild just now. A "Disable transition" button is present between the build and test stages. The pipeline execution ID is 99dcc729-5f89-4bd8-b8eb-3cb67cbc66bc.

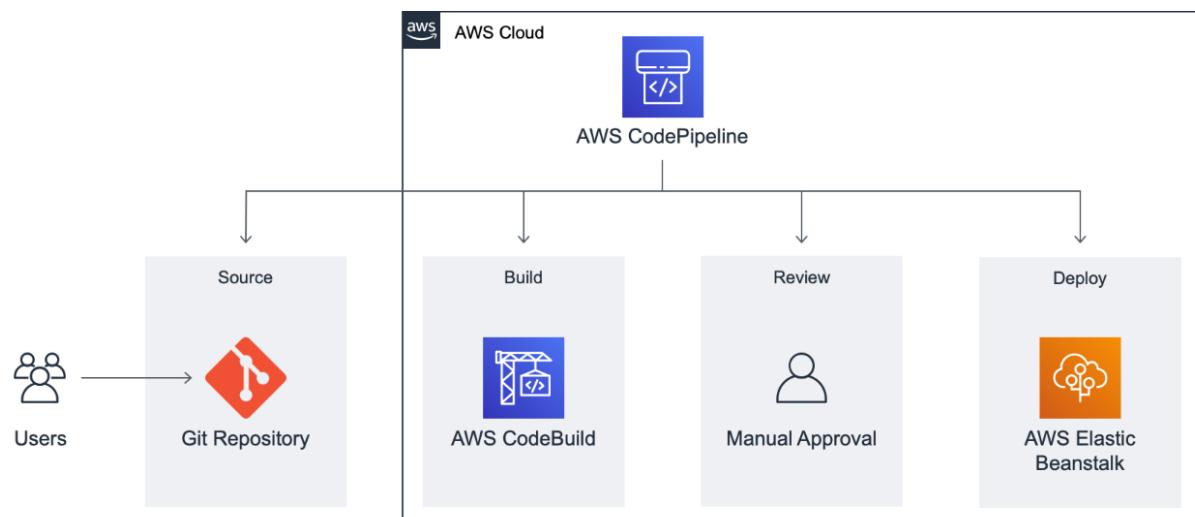




Adding Manual Approval with SNS & SQS

⌚ Objective:

- Introduce a manual approval step before deployment using SNS and SQS.



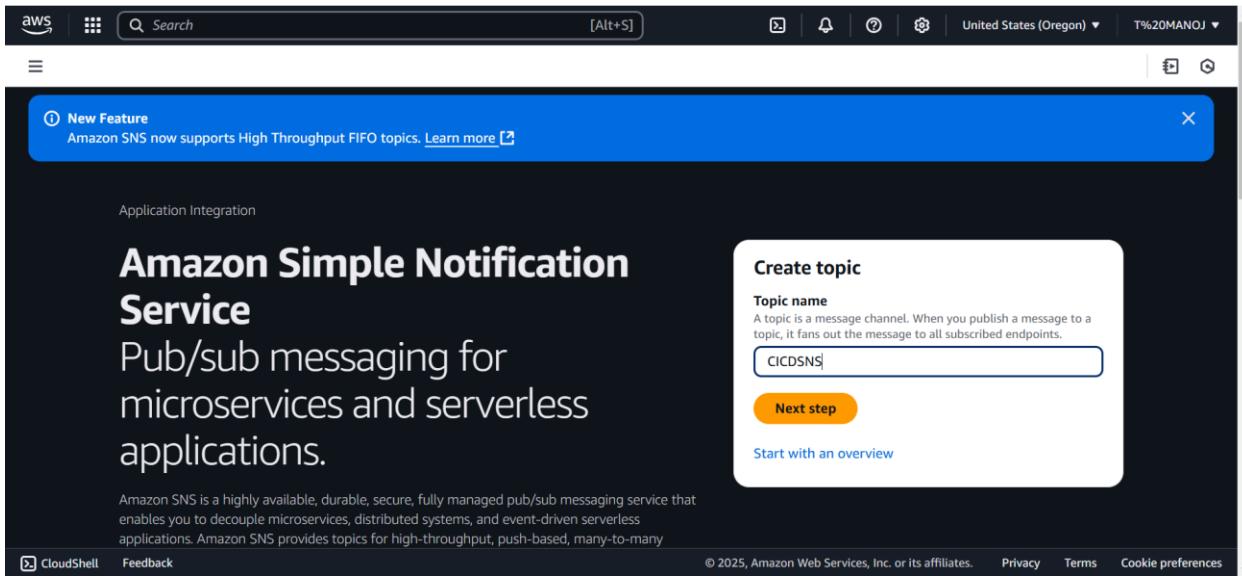
Terminologies:

1. Manual Approval Stage: Pauses the pipeline for human validation.
2. SNS (Simple Notification Service): Sends notifications (email, SMS).
3. SQS (Simple Queue Service): Stores approval requests.
4. Rollback: Pipeline reverses changes if approval is denied.

Steps for Manual Approval & Rollback:

Step 1: Create SNS Topic

1. Go to SNS in the AWS console.
2. Click Create Topic.
3. Type: Standard.
4. Name: CICDSNS.
5. Click Create Topic.
6. Subscribe with your email.



The screenshot shows the AWS SNS Create Topic page. At the top, there's a blue banner with a 'New Feature' message: 'Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)'. Below the banner, the page title is 'Amazon Simple Notification Service' with the subtitle 'Pub/sub messaging for microservices and serverless applications'. A descriptive paragraph explains that Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service. On the right, a 'Create topic' form is displayed. It has a 'Topic name' input field containing 'CICDSNS', a 'Next step' button, and a 'Start with an overview' link. The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and Copyright information.

AWS | Search [Alt+S] | United States (Oregon) ▾ | T%20MANOJ ▾

Amazon SNS > Topics > Create topic

New Feature
Amazon SNS now supports High Throughput FIFO topics. [Learn more](#)

Create topic

Details

Type [Info](#)
Topic type cannot be modified after topic is created

FIFO (first-in, first-out)
• Strictly-preserved message ordering
• Exactly-once message delivery
• Subscription protocols: SQS

Standard
• Best-effort message ordering
• At-least once message delivery
• Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

AWS | Search [Alt+S] | United States (Oregon) ▾ | T%20MANOJ ▾

Amazon SNS > Subscriptions > Create subscription

Details

Topic ARN

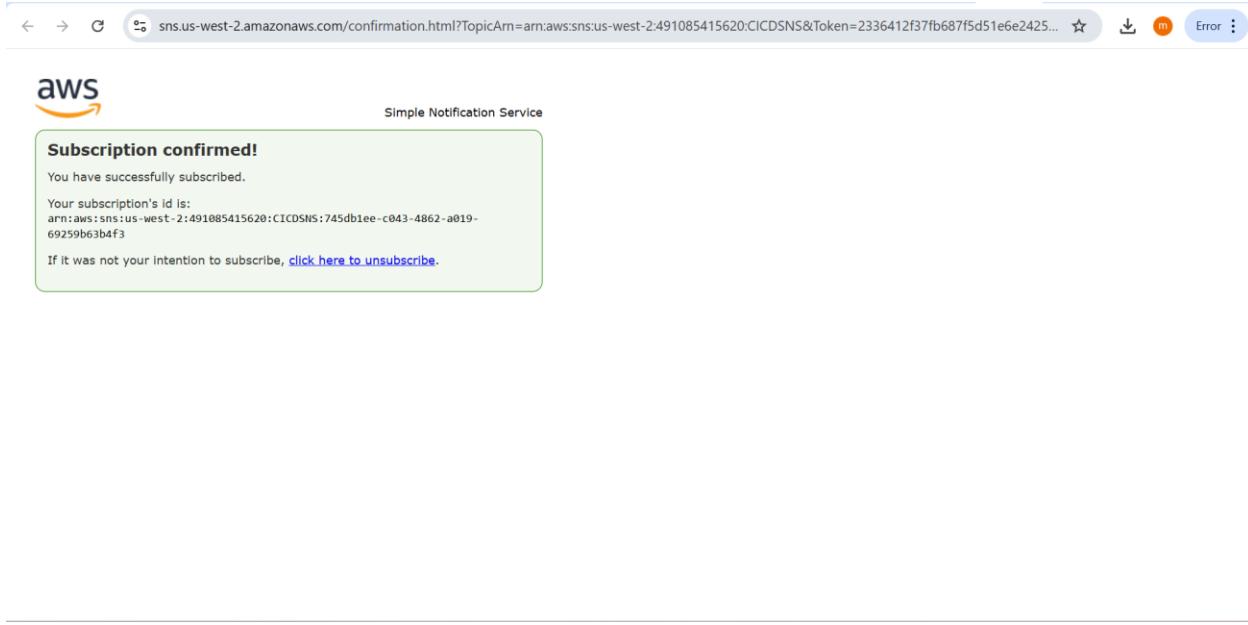
Protocol
The type of endpoint to subscribe

Endpoint
An email address that can receive notifications from Amazon SNS.

Subscription filter policy - optional [Info](#)

After your subscription is created, you must confirm it. [Info](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Protocol
The type of endpoint to subscribe

Amazon SQS

Endpoint
Only Amazon SQS standard queues will be listed and can receive notifications from an Amazon SNS standard topic.

arn:aws:sqs:us-west-2:491085415620:CICDSQS

Enable raw message delivery

After your subscription is created, you must confirm it. [Info](#)

Step 2: Create SQS Queue

1. Navigate to SQS.
2. Click Create Queue → Standard Queue.
3. Create the role to communicate with SNS
4. Name: CICDSQS.
5. Use default settings and click Create Queue.

AWS | Search [Alt+S] | United States (Oregon) | T%20MANOJ

Application integration

Amazon SQS

A message queuing service

Amazon SQS provides queues for high-throughput, system-to-system messaging. You can use queues to decouple heavyweight processes and to buffer and batch work. Amazon SQS stores messages until microservices and serverless applications process them.

Get started

Learn how to use Amazon SQS by creating a queue, sending a message to the queue, and receiving and processing the message.

Create queue

Pricing (US)

You can get started with Amazon SQS for free. All customers can make 1 million Amazon SQS requests for free each month. Some applications might be able to operate within this Free Tier limit.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

AWS | Search [Alt+S] | United States (Oregon) | T%20MANOJ

Amazon SQS > Queues > Create queue

Create queue

Details

Type

Choose the queue type for your application or cloud infrastructure.

Standard Info
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

FIFO Info
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

Info You can't change the queue type after you create a queue.

Name

CICDSQS

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws | Search [Alt+S] | Global ▾ | T%20MANOJ ▾

IAM > Roles > Create role

Step 1
 Select trusted entity
 Step 2
 Add permissions
 Step 3
 Name, review, and create

Select trusted entity Info

Trusted entity type

AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws | Search [Alt+S] | Global ▾ | T%20MANOJ ▾

IAM > Roles > Create role

Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a use case for the specified service.
Use case
 SNS
Allows SNS to call CloudWatch Logs on your behalf.

Cancel

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS IAM 'Create role' wizard Step 2: Add permissions.

The page shows a navigation bar with 'Search' and 'Global' dropdowns, and a breadcrumb trail: 'IAM > Roles > Create role'. A sidebar on the left lists 'Step 1 Select trusted entity', 'Step 2 Add permissions' (which is selected), and 'Step 3 Name, review, and create'. The main content area is titled 'Add permissions'.

Permissions policies (1) Info
The type of role that you selected requires the following policy.

Policy name **Type** AWS managed

Set permissions boundary - optional

Buttons at the bottom: 'Cancel', 'Previous' (disabled), and 'Next'.

Screenshot of the AWS IAM 'Create role' wizard Step 3: Name, review, and create.

The page shows a navigation bar with 'Search' and 'Global' dropdowns, and a breadcrumb trail: 'IAM > Roles > Create role'. A sidebar on the left lists 'Step 1 Select trusted entity', 'Step 2 Add permissions', and 'Step 3 Name, review, and create' (which is selected). The main content area is titled 'Name, review, and create'.

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+-=._@-' characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: '_+=., @-/[\{\}!#\$%^&*()~`

Step 1: Select trusted entities [Edit](#)

Buttons at the bottom: 'CloudShell', 'Feedback', '© 2025, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', 'Cookie preferences'.

Screenshot of the AWS SQS 'Create queue' page.

Details

Type
Choose the queue type for your application or cloud infrastructure.

Standard info
At-least-once delivery, message ordering isn't preserved

- At-least once delivery
- Best-effort ordering

FIFO info
First-in-first-out delivery, message ordering is preserved

- First-in-first-out delivery
- Exactly-once processing

Info You can't change the queue type after you create a queue.

Name

A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS SQS 'Create queue' page showing advanced access policy configuration.

Basic
Use simple criteria to define a basic access policy.

Advanced
Use a JSON object to define an advanced access policy.

Define who can send messages to the queue

Only the queue owner
Only the owner of the queue can send messages to the queue.

Only the specified AWS accounts, IAM users and roles
Only the specified AWS account IDs, IAM users and roles can send messages to the queue.

Define who can receive messages from the queue

Only the queue owner
Only the owner of the queue can receive messages from the queue.

Only the specified AWS accounts, IAM users and roles
Only the specified AWS account IDs, IAM users and roles can receive messages from the queue.

JSON (read-only)

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__owner_statement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "491085415620"
      },
      "Action": [
        "SQS:*"
      ],
      "Resource": "arn:aws:sqs:us-west-2:491085415620:CICDSQS"
    }
  ]
}
```

Screenshot of the AWS SQS 'CICDSQS' queue details page.

Queue CICDSQS created successfully
You can now send and receive messages.

CICDSQS

Details [Edit](#) [Delete](#) [Purge](#) [Send and receive messages](#) [Start DLQ redrive](#)

Name <input type="text" value="CICDSQS"/>	Type Standard	ARN <input type="text" value="arn:aws:sqs:us-west-2:491085415620:CICDSQS"/>
Encryption Disabled	URL <input type="text" value="https://sqs.us-west-2.amazonaws.com/491085415620/CICDSQS"/>	Dead-letter queue -

[More](#)

SNS subscriptions Lambda triggers EventBridge Pipes Dead-letter queue Monitoring Tagging Queue policies Encrypt >

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 3: Add Manual Approval Stage

1. Go to AWS CodePipeline → Select Pipeline-DevOpsGettingStarted → Edit.
2. Add a Stage before Deploy named ApprovalStage.
3. Add an Action of type Manual Approval.
4. Select the SNS topic for notification.

The screenshot shows the AWS CodePipeline Pipelines page. On the left, there is a sidebar titled "CodePipeline" with options like Source, Artifacts, Build, Deploy, Pipeline, Getting started, Pipelines (which is selected), Account metrics, and Settings. The main area displays the "Pipelines" table with one row for "Pipeline-DevOpsGettingStarted". The table columns are Name, Latest execution status, Latest source revisions, Latest execution started, and Most recent executions. The pipeline name is "Pipeline-DevOpsGettingStarted", status is "Succeeded", source is "5d04c878", latest execution started was "20 minutes ago", and there is a link to "View details".

The screenshot shows the "Add stage" dialog box. At the top, it says "Add stage" and has a close button. Below that is a "Stage name" input field containing "MANUAL_APPROVAL_1". A note below the input field says "No more than 100 characters". At the bottom right are "Cancel" and "Add stage" buttons, with "Add stage" being orange.

Edit action

Action name

Choose a name for your action

No more than 100 characters

Action provider

Configure the approval request.

SNS topic ARN - optional



URL for review - optional

Type the URL you want to provide to the reviewer as part of the approval request. The URL must begin with 'http://' or 'https://'.

The screenshot shows the AWS CodePipeline console. On the left, there's a navigation sidebar with 'Developer Tools' and 'CodePipeline' selected. Under 'CodePipeline', there are links for Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), and Action details (New). Below these are sections for Getting started, Pipelines, and Pipeline Metrics. At the bottom of the sidebar are links for CloudShell and Feedback.

The main area displays the 'Edit: MANUAL_APPROVAL_1' stage configuration. It includes a 'Conditions' section with 'Entry: Not configured', 'Success: Not configured', and 'Failure: Not configured' status. A 'View details' button is also present. Below this is a 'MANUALAPPROVAL' section with a 'Manual approval' provider listed. An 'Automated stage configuration' dropdown is set to 'None'. There are '+ Add stage' buttons at the top and bottom of the stage configuration panel.

At the bottom of the page, there are links for © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Edit action

Action name

Choose a name for your action

No more than 100 characters

Action provider

No topic

arn:aws:sns:us-west-2:491085415620:CICDSNS



URL for review - *optional*

Type the URL you want to provide to the reviewer as part of the approval request. The URL must begin with 'http://' or 'https://'.

Comments - *optional*

Edit: MANUAL_APPROVAL_2

[Edit stage](#)

Conditions Entry: Not configured Success: Not configured Failure: Not configured

[View details](#)

MANUAL_APPROVAL_2

Manual approval

Automated stage configuration: [None](#)

Step 4: Test Manual Approval Process

1. Make a code change and push to GitHub.
2. The pipeline stops at the Approval Stage.
3. Approve/Reject via the AWS Console or SNS email link.
4. Approve: Pipeline deploys.
5. Reject: Deployment rolls back automatically.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a project folder named "AWS-ELASTIC-BEANSTALK-EXPRESS-JS-SA...". Inside, there are files like ".github", ".gitignore", "app.js", "CODE_OF_CONDUCT.md", "CONTRIBUTING.md", "LICENSE", "package-lock.json", "package.json", and "README.md".
- Code Editor:** The "app.js" file is open, displaying the following code:

```
1 const express = require('express');
2 const app = express();
3 const port = 8080;
4
5 app.get('/', (req, res) => res.send('Hello World! hi this is version 2 final testing'));
6
7 app.listen(port);
8 console.log(`App running on http://localhost:${port}`);
```
- Terminal:** The terminal window shows the command-line history:

```
PS C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample> git add .
PS C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample> git commit -m "Final pipeline Testing"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample> git push
```
- Bottom Status Bar:** Displays "Not Committed Yet", "Ln 5, Col 85", "Spaces: 4", "UTF-8", "CRLF", "JavaScript", and the date "21-02-2025".

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the same project structure as the first screenshot.
- Code Editor:** The "app.js" file is open, showing the same code as before.
- Terminal:** The terminal window shows the command-line history:

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 317 bytes | 158.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/manojmanu9441/aws-elastic-beanstalk-express-js-sample.git
  5d84c87..8de277a main -> main
PS C:\Users\T Manoj\aws-elastic-beanstalk-express-js-sample>
```
- Bottom Status Bar:** Displays "Ln 9, Col 1", "Spaces: 4", "UTF-8", "CRLF", "JavaScript", and the date "21-02-2025".

Pipeline-DevOpsgettingstarted

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded

Pipeline execution ID: bf42f815-ac8e-459a-8d89-74aa3b678ea4

Source GitHub (via GitHub App)

Succeeded - Just now

8de277a3

[View details](#)

8de277a3 Source: Final pipeline Testing

[Disable transition](#)

This screenshot shows the 'Source' stage of the 'Pipeline-DevOpsgettingstarted' pipeline. The stage is marked as 'Succeeded'. It displays a GitHub commit from '8de277a3' made 'Just now'. A 'View details' button is available. Below the stage, a 'Disable transition' button is shown. The pipeline execution ID is 'bf42f815-ac8e-459a-8d89-74aa3b678ea4'. The status bar at the bottom indicates the URL 'https://us-west-2.console.aws.amazon.com/console/home?region=us-west-2'.

Source: Final pipeline Testing

[Disable transition](#)

Build Succeeded

Pipeline execution ID: bf42f815-ac8e-459a-8d89-74aa3b678ea4

Start rollback

Build AWS CodeBuild

Succeeded - Just now

[View details](#)

8de277a3 Source: Final pipeline Testing

[Disable transition](#)

This screenshot shows the 'Build' stage of the 'Pipeline-DevOpsgettingstarted' pipeline. The stage is marked as 'Succeeded'. It displays a build from 'AWS CodeBuild' made 'Just now'. A 'View details' button is available. Below the stage, a 'Disable transition' button is shown. A 'Start rollback' button is also present. The pipeline execution ID is 'bf42f815-ac8e-459a-8d89-74aa3b678ea4'. The status bar at the bottom indicates the URL 'https://us-west-2.console.aws.amazon.com/console/home?region=us-west-2'.

The screenshot shows a Gmail inbox with 497 messages. A message from 'AWS Notifications <no-reply@sns.amazonaws.com>' is selected, titled 'APPROVAL NEEDED: AWS CodePipeline Pipeline-DevOpsGettingSt... for action MANUALAPPROVAL'. The message body contains:

Hello,

The following Approval action is waiting for your response:

--Pipeline Details--

Pipeline name: Pipeline-DevOpsGettingStarted
Stage name: MANUAL_APPROVAL_1
Action name: MANUALAPPROVAL
Region: us-west-2

--Approval Details--

Approve or reject: https://console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2#/MANUAL_APPROVAL_1/MANUALAPPROVAL/approve/18470f95-5f40-405a-ad90-7970b376aa14
Deadline: This review request will expire on 2025-02-28T15:13Z

The screenshot shows the AWS SQS 'Receive messages' interface for the 'CICDSQS' queue. One message is available:

Message: 78b1f5df-ee42-4873-b08b-ff84c9b9a06b

The message content is displayed in a modal window:

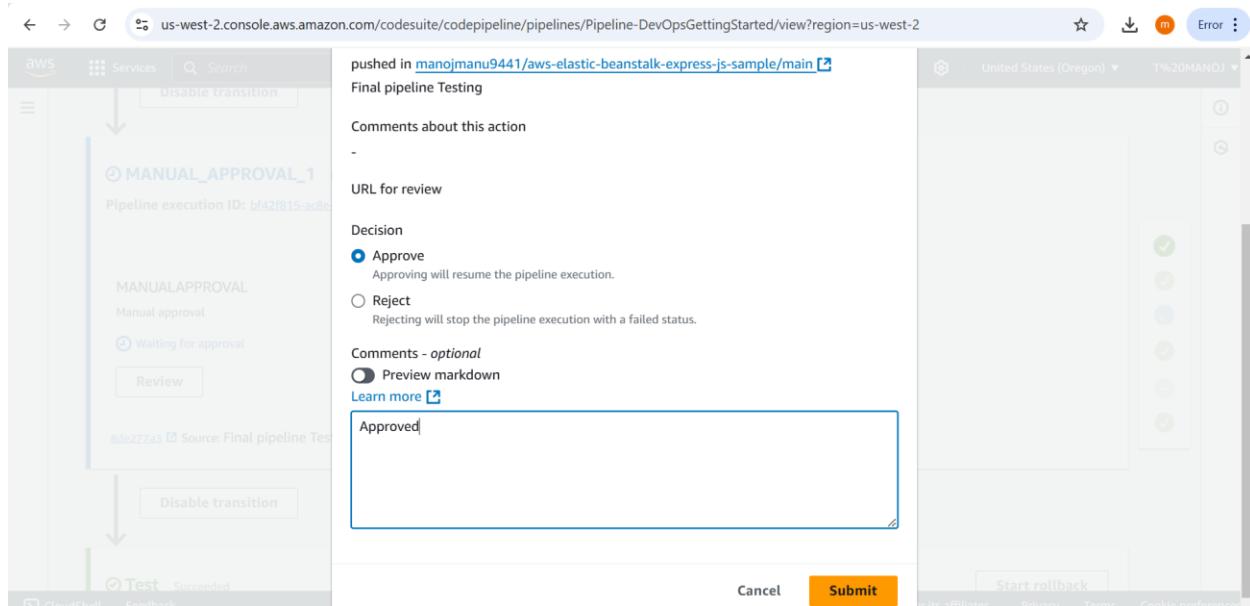
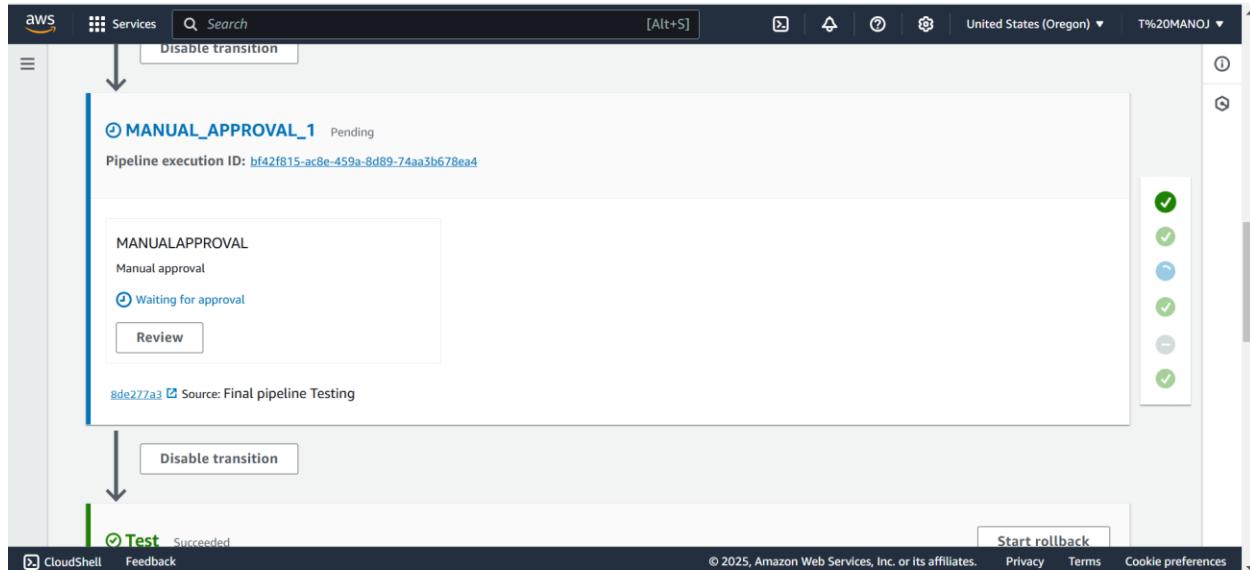
```
"TopicArn": "arn:aws:sns:us-west-2:491085415620:CICDSNS",
"Subject": "APPROVAL NEEDED: AWS CodePipeline Pipeline-DevOpsGettingSt... for action MANUALAPPROVAL",
"Message": "{\"region\":\"us-west-2\\\"\\\"console link\\\"\\\"https://console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2#/MANUAL_APPROVAL_1/MANUALAPPROVAL/approve/18470f95-5f40-405a-ad90-7970b376aa14\"\\\"\\\""}"
```

The message details are:

- Body: [Content of the message]
- Attributes: [Empty]
- Details: [Empty]

Message details table:

ID	Sent	Size	Receive count
78b1f5df-ee42-4873-b08b-ff84c9b9a06b	2025-02-21T20:43:05+05:30	1.62 KB	1



The image consists of three vertically stacked screenshots of the AWS CodePipeline console, illustrating a manual approval process.

Screenshot 1: The first screenshot shows a pipeline stage named "Test" which has "Succeeded". A "Disable transition" button is visible above the stage card. The pipeline execution ID is [bf42f815-ac8e-459a-8d89-74aa3b678ea4](#). To the right, there is a vertical column of green checkmarks and a "Start rollback" button.

Screenshot 2: The second screenshot is identical to the first, showing the "Test" stage succeeded and the pipeline execution ID. It also features a "Disable transition" button and a vertical column of green checkmarks.

Screenshot 3: The third screenshot shows the same pipeline stage and execution ID. The "Disable transition" button is present. Below the stage card, there is a message from "AWS Notifications" with the subject "APPROVAL NEEDED: AWS CodePipeline Pipeline-DevOpsGettingSt... for action MANUAL_APPROVAL_2". The message body starts with "Hello," and continues with "The following Approval action is waiting for your response: --Pipeline Details-- Pipeline name: Pipeline-DevOpsGettingStarted Stage name: MANUAL_APPROVAL_2 Action name: MANUAL_APPROVAL_2 Region: us-west-2 --Approval Details-- Approve or reject: https://console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2#/MANUAL_APPROVAL_2/MANUAL_APPROVAL_2/approve/0a865350-54f8-4831-a813-2ad40746b30c Deadline: This review request will expire on 2025-02-28T15:16Z".

us-west-2.console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2

Trigger
Commit [8de277a3](#) pushed in [manojmanu9441/aws-elastic-beanstalk-express-js-sample/main](#)

Final pipeline Testing

Comments about this action
-

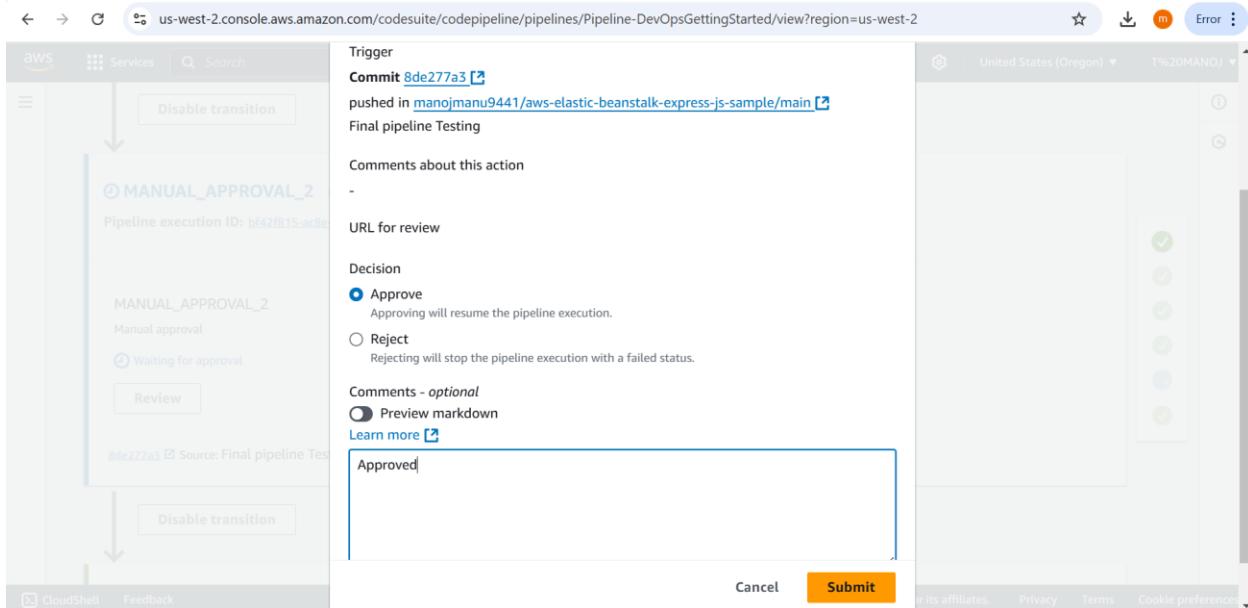
URL for review

Decision
 Approve
Approving will resume the pipeline execution.
 Reject
Rejecting will stop the pipeline execution with a failed status.

Comments - optional
 Preview markdown
[Learn more](#)

Approved

Cancel **Submit**



aws Services Search [Alt+S] United States (Oregon) T%20MANOJ

Disable transition

MANUAL_APPROVAL_2 In progress

Pipeline execution ID: [bf42f815-ac8e-459a-8d89-74aa3b678ea4](#)

MANUAL_APPROVAL_2

Manual approval

Approved - Just now

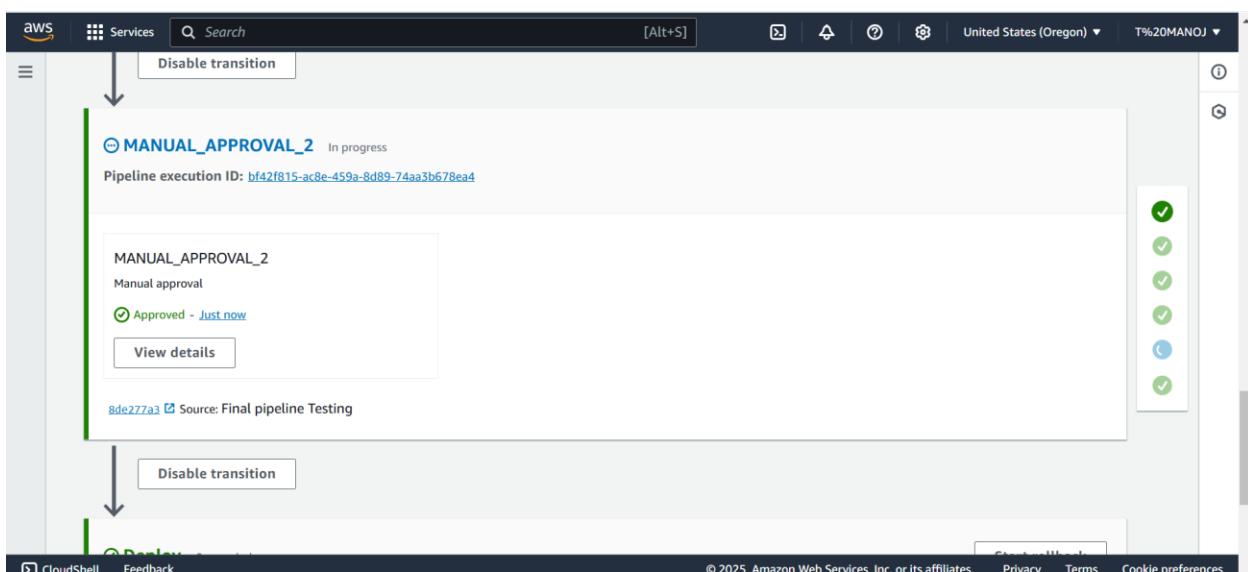
[View details](#)

8de277a3 Source: Final pipeline Testing

Disable transition

MANUAL_APPROVAL_2

Approved



Screenshot of the AWS CloudFormation console showing a pipeline execution.

The pipeline step "Deploy" has succeeded (Succeeded). Pipeline execution ID: bfa2f815-ac8e-459a-8d89-74aa3b678ea4

Details for the Deploy step:

- Deployed to AWS Elastic Beanstalk
- Status: Succeeded - Just now
- View details button

Source: Final pipeline Testing

Right sidebar shows green checkmarks for each step in the pipeline.

Screenshot of the AWS SQS console showing a queue named CICDSQS.

Receive messages info

Messages available: 1

Polling duration: 30

Maximum message count: 10

Polling progress: 20% (1 receives/second)

Stop polling button

Poll for messages button

Messages (2)

ID	Sent	Size	Receive count
78b1f5df-ee42-4873-b08b-ff84c9b9a06b	2025-02-21T20:43+05:30	1.62 KB	3
f659e115-3465-42d9-8d46-1dc8f924ea42	2025-02-21T20:46+05:30	1.63 KB	1

Search messages input field

View details and Delete buttons for each message

Page navigation: < 1 >

Screenshot of the AWS SQS console showing a message detail view.

The URL in the browser is [https://console.aws.amazon.com/sqs/v2/home?region=us-west-2#queue/CICDSQS/messages/f659e115-3465-42d9-8d46-1dc8f924ea42](#)

The message details are as follows:

```
Message: f659e115-3465-42d9-8d46-1dc8f924ea42
Body: {"TopicArn": "arn:aws:sns:us-west-2:491085415620:CICDSNS", "Subject": "APPROVAL NEEDED: AWS CodePipeline Pipeline-DevOpsGettingSt... for action MANUAL_APPROVAL_2", "Message": "{\"region\":\"us-west-2\"}"}
```

The message was received on 2025-02-21T20:43:05+05:30 and has a size of 1.62 KB. It has been received once.

Screenshot of a web browser showing a test response from an Elastic Beanstalk application.

The URL in the browser is [https://devopsgettingstarted-env.eba-thv7w3uu.us-west-2.elasticbeanstalk.com](#)

The page content is: Hello World! hi this is version 2 final testing

us-west-2.console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2

The screenshot shows the AWS CodePipeline console with a pipeline named "Pipeline-DevOpsGettingStarted". A specific execution is being viewed. A "MANUAL_APPROVAL_1" step is highlighted, showing it is "Waiting for approval". A "Review" button is visible. On the right, a modal window titled "Review" is open for this step. The modal shows the trigger was a commit (7957baeb) pushed to "manojmanu9441/aws-elastic-beanstalk-express-js-sample/main" branch, and the pipeline execution ID is 430d8a58-49c. The "Details" tab is selected. Under "Decision", the "Reject" option is selected. The "Comments - optional" field contains a note: "NOT APPROVED". At the bottom, there are "Cancel" and "Submit" buttons.

us-west-2.console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2

This screenshot is identical to the one above, showing the same pipeline execution and the "MANUAL_APPROVAL_1" step. The "Review" button is visible. The modal window is open, showing the same trigger details and the "Details" tab selected. Under "Decision", the "Reject" option is selected. In the "Comments - optional" field, the text "NOT APPROVED" is typed. The "Submit" button is highlighted in orange at the bottom of the modal.

MANUAL_APPROVAL_1 In progress
Pipeline execution ID: 430d8a58-49c3-4c9e-96c8-08b409b81a9d

MANUALAPPROVAL
Manual approval
Rejected - Just now
[View details](#)

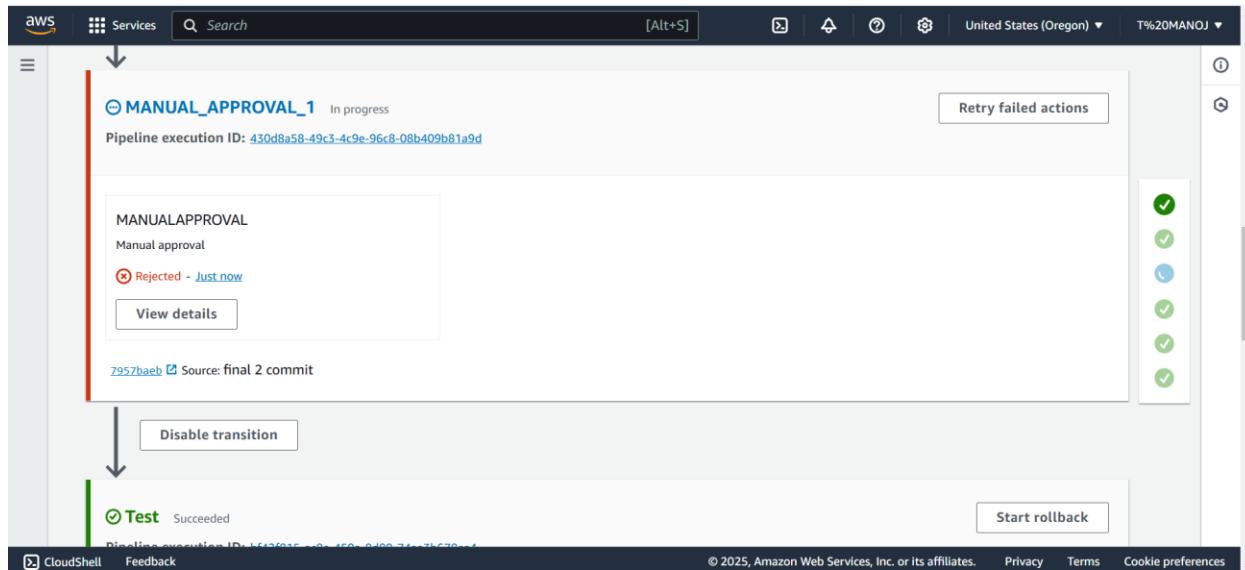
[7957baeb](#) Source: final 2 commit

[Disable transition](#)

Test Succeeded
Pipeline execution ID: 430d8a58-49c3-4c9e-96c8-08b409b81a9d

[Start rollback](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



This screenshot shows the AWS CodePipeline console. A pipeline named 'MANUAL_APPROVAL_1' is currently in progress. The first stage, 'MANUALAPPROVAL', has failed with a 'Rejected' status. A red vertical bar highlights this stage. Below it, a green vertical bar highlights the 'Test' stage, which has succeeded. On the right side, there is a vertical toolbar with several green checkmark icons. At the bottom, the pipeline's execution ID is displayed along with standard AWS footer links.

Action MANUALAPPROVAL was rejected

Developer Tools > CodePipeline > Pipelines > Pipeline-DevOpsGettingStarted

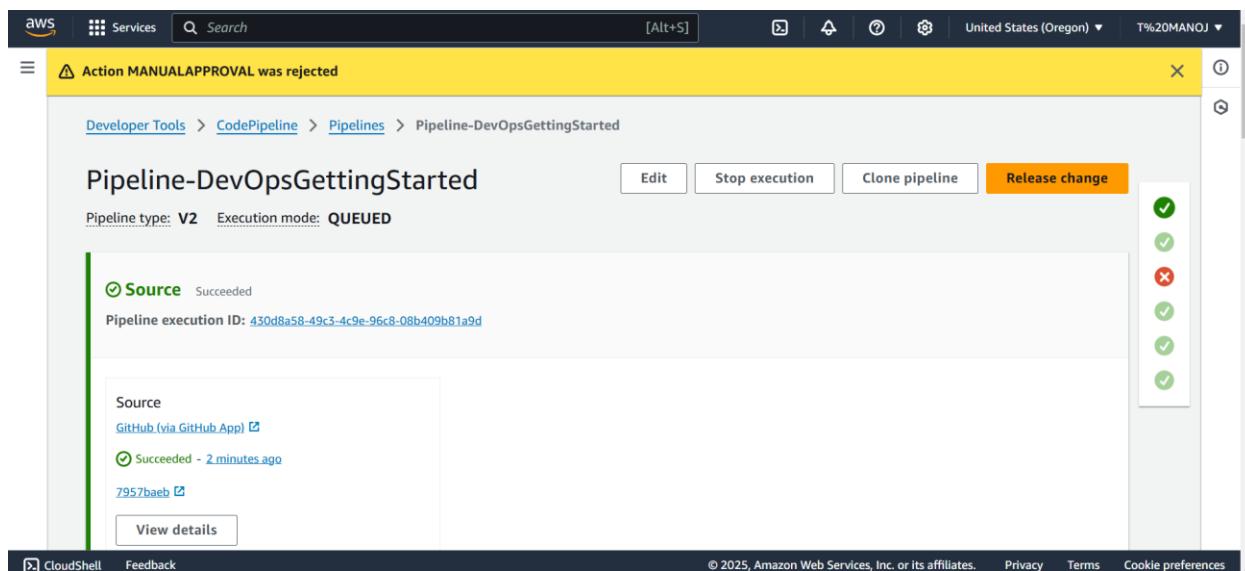
Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: 430d8a58-49c3-4c9e-96c8-08b409b81a9d

Source
GitHub (via GitHub App)
Succeeded - 2 minutes ago
[7957baeb](#)
[View details](#)

Edit Stop execution Clone pipeline Release change

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



This screenshot shows the AWS CodePipeline console after a manual approval step has been rejected. A yellow banner at the top displays the error message 'Action MANUALAPPROVAL was rejected'. The pipeline's execution status is now 'QUEUED'. The 'Source' stage is shown as succeeded. A red vertical bar highlights the 'Source' stage, and a green vertical bar highlights the 'Clone pipeline' button. On the right, a vertical toolbar shows a mix of green and red checkmarks. The pipeline's execution ID is visible at the bottom, along with the usual AWS footer links.

us-west-2.console.aws.amazon.com/codesuite/codepipeline/pipelines/Pipeline-DevOpsGettingStarted/view?region=us-west-2&pipeline-executions-meta...

Services Search [Alt+S] United States (Oregon) T%20MANOJ

Disable transition

Rollback to

Start rollback

< 1 > ⚙

Execution ID	Source revisions	Started	Duration	Completed
bf42f815	Source - 8de277a3 Final pipeline Testing	55 minutes ago	4 minutes 35 seconds	50 minutes ago

Disable transition

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Services Search [Alt+S] United States (Oregon) T%20MANOJ

7957baeb Source: final 2 commit

Disable transition

MANUAL_APPROVAL_1 Rollback In progress

Pipeline execution ID: [865e06f2-f1b6-4bc2-a66c-bf956e859c78](#)

MANUALAPPROVAL

Manual approval

Approved - Just now

[View details](#)

8de277a3 Source: Final pipeline Testing

Disable transition

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS CodePipeline console. The left sidebar is titled 'CodePipeline' and contains navigation links for Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Deploy (CodeDeploy), Pipeline (CodePipeline), Getting started, Pipelines (selected), Account metrics, Settings, Go to resource, and Feedback. The main area is titled 'Pipelines' and shows a single pipeline named 'Pipeline-DevOpsGettingStarted'. The pipeline status is 'Succeeded' with a green circle icon. The latest execution status is 'Final' (Type: V2 | Execution mode: QUEUED). The latest source revision is '8de277a3' (3 minutes ago). There are four circular icons representing recent executions, all of which are green. A 'View details' link is present. At the top right, there are buttons for 'Create pipeline', 'Delete pipeline', and other actions. The top bar includes the AWS logo, services menu, search bar, and account information for United States (Oregon) and T%20MANOJ.

Final CI/CD Architecture Overview

GitHub (Source Repository)



AWS CodePipeline (Orchestrates entire flow)

- └─ Source Stage (GitHub webhook trigger)
- └─ Build Stage (AWS CodeBuild compiles & tests code)
- └─ Approval Stage (Manual approval with SNS & SQS)
- └─ Deploy Stage (AWS Elastic Beanstalk deployment)

Advantages

- **Automation of the Entire Workflow:** Reduces manual intervention by automating build, test, and deployment processes.
- **Faster Delivery:** Accelerates the release cycle by enabling rapid deployment of new features and bug fixes.

- **Improved Code Quality:** Automated builds and tests catch errors early, reducing production bugs.
- **Scalability and Flexibility:** AWS Elastic Beanstalk automatically handles capacity provisioning, load balancing, and scaling.
- **Cost-Effective:** Leveraging AWS free-tier services (like single-instance environments) makes it budget-friendly for small projects.
- **Enhanced Security and Control:** Manual approval stages with SNS and SQS ensure critical changes undergo human review before deployment.
- **Seamless Integration:** Native integration between GitHub and AWS services ensures smooth operation without complex configurations.

Results and Discussion

The project successfully established a fully automated CI/CD pipeline using GitHub, AWS Elastic Beanstalk, AWS CodeBuild, AWS CodePipeline, SNS, and SQS. The GitHub repository was correctly forked, cloned, and modified (`app.js`), with changes committed and pushed to the remote repository. The web application was deployed on AWS Elastic Beanstalk, running smoothly on an EC2 instance with the generated URL accessible. AWS CodeBuild compiled the code with the `buildspec.yml` file, completing the build process without errors. The AWS CodePipeline orchestrated the entire CI/CD flow, automatically triggering builds and deployments on new GitHub commits. The manual approval stage, integrated using SNS and SQS, worked as expected—notifications were sent via email, and approval actions managed through the console or email link. The rollback mechanism was validated successfully when a deployment was rejected during manual approval.

The discussion highlights the pipeline's efficiency in automating the build, test, and deployment process, significantly reducing manual intervention and deployment time. Elastic Beanstalk ensured a scalable and reliable infrastructure for seamless application updates. The robust rollback mechanism protected the production environment from unstable deployments. SNS and SQS integrations provided real-time notifications and

reliable message queuing, enhancing the manual approval workflow. The secure communication between AWS services, achieved through OAuth and appropriate IAM roles, ensured a safe deployment process. Overall, the continuous integration and deployment approach streamlined development workflows, improving productivity and ensuring faster, more reliable software releases

Conclusion

By implementing this CI/CD pipeline, you have established an automated, scalable, and reliable deployment process. The pipeline detects changes in the GitHub repository, builds the application using AWS CodeBuild, and deploys it to AWS Elastic Beanstalk through AWS CodePipeline. The addition of a manual approval step with SNS and SQS ensures that critical updates are reviewed, providing a balance between automation and control. This solution not only speeds up development cycles but also ensures that deployments are secure, high quality, and highly available, embodying the best practices of DevOps .