Name – Toyash Patil
Roll No. 16010122140

AWS CodePipeline CI/CD Implementation Report

Project Overview

Project Title: AWS CodePipeline Integration for React + Node.js Application
Purpose: DevOps IA (Internal Assessment) Case Study
Technology Stack: React (Vite), Node.js/Express, AWS CodePipeline, AWS CodeBuild, Amazon S3
Region: Asia Pacific (Mumbai) - ap-south-1

Executive Summary

This project demonstrates a complete CI/CD (Continuous Integration/Continuous Deployment) pipeline using AWS CodePipeline to automate the build and deployment process for a full-stack web application. The implementation showcases industry-standard DevOps practices including version control integration, automated builds, and artifact management.

Project Architecture

Application Components

Frontend:

- Framework: React with Vite build tool
- Build Output: `dist/` directory
- Features: Simple demonstration UI with "Hello, CodePipeline!" message
- API Integration: Connects to backend `/api/ping` endpoint

Backend:

- Framework: Node.js with Express
- Port: 5000
- Functionality:
  - Serves static frontend files from `dist/`
  - Provides REST API endpoint `/api/ping`

- Handles client-side routing fallback

Repository Structure:

```
CodePipeline-IA/
├── frontend/
|   ├── src/
|   |   └── App.js
|   ├── public/
|   |   └── index.html
|   ├── package.json
|   └── dist/ (generated)
├── backend/
|   ├── index.js
|   ├── package.json
|   └── node_modules/
├── buildspec.yml
└── README.md
```

AWS Services Implementation

1. Amazon S3 (Simple Storage Service)

Purpose: Artifact storage and optional static website hosting

Bucket Created: my-unique-app-pipeline-12345 (globally unique name)

Configuration:

- Region: ap-south-1
- Used for: Pipeline artifact storage, build outputs

Key Learning: S3 bucket names must be globally unique across all AWS accounts, requiring creative naming strategies.

Name – Toyash Patil
Roll No. 16010122140

2. AWS IAM (Identity and Access Management)

IAM User Created: Toyash_CodePipeline_demo

Attached Policies:

- AmazonS3FullAccess
- AWSCodeBuildAdminAccess
- AWSCodePipeline_FullAccess
- CloudWatchLogsFullAccess

IAM Roles Created:

CodeBuildServiceRole:

- Trust Entity: codebuild.amazonaws.com
- Permissions:
  - AmazonS3FullAccess - Read/write build artifacts
  - AWSCodeBuildAdminAccess - Execute builds
  - CloudWatchLogsFullAccess - Write build logs

CodePipelineServiceRole:

- Trust Entity: codepipeline.amazonaws.com
- Permissions:
  - AmazonS3FullAccess - Artifact management
  - AWSCodeBuildAdminAccess - Trigger builds
  - AWSCodePipeline_FullAccess - Pipeline operations
  - Custom inline policy for CodeStar Connections:

json

```
{
 "Effect": "Allow",
 "Action": ["codestar-connections:UseConnection"],
 "Resource": "arn:aws:codeconnections:ap-south-1:197337164274:connection/..."
}
```

3. AWS CodeStar Connections

Connection Name: my-github-connection

Purpose: Secure integration between AWS and GitHub

Connection ARN: arn:aws:codeconnections:ap-south-1:197337164274:connection/2ae52963-0fcb-42fc-9d32-bb88274060be

Name – Toyash Patil
Roll No. 16010122140

Configuration Process:

1. Created connection via pipeline setup
2. Authorized GitHub App through OAuth
3. Granted repository access to AWS

4. AWS CodeBuild

Project Name: demo-codebuild-project

Configuration:

- Environment: Linux Container
- Image: aws/codebuild/standard:7.0
- Runtime: Node.js 18
- Compute: BUILD_GENERAL1_SMALL
- Service Role: CodeBuildServiceRole

Build Specification (buildspec.yml):

yaml

```yaml
version: 0.2
phases:
  install:
    runtime-versions:
      nodejs: 18
    commands:
      - echo "Installing frontend dependencies"
      - cd frontend
      - npm ci
  build:
    commands:
      - echo "Building frontend"
      - npm run build
  post_build:
    commands:
      - echo "Frontend build finished"
artifacts:
  files:
    - frontend/dist/**/*
  discard-paths: no
```

Build Process:

Name – Toyash Patil
Roll No. 16010122140

1. Install Node.js 18 runtime
2. Navigate to frontend/ directory
3. Run npm ci for clean dependency installation
4. Execute npm run build to generate production assets
5. Package dist/ folder as artifacts

5. AWS CodePipeline

Pipeline Name: SimpleNodeJSBuildService

Execution Mode: Superseded (latest commit takes priority)

Pipeline Stages:

Stage 1: Source

- Provider: GitHub (via CodeStar Connection)
- Repository: Toyashpatil/CodePipeline-IA
- Branch: main
- Output: SourceArtifact
- Trigger: Automatic on push to main branch

Stage 2: Build

- Provider: AWS CodeBuild
- Project: demo-codebuild-project
- Input: SourceArtifact
- Output: BuildArtifact
- Build Type: Single build

Stage 3: Deploy (Optional)

- Provider: Amazon S3
- Bucket: Artifact storage bucket
- Input: BuildArtifact

Implementation Challenges and Solutions

Challenge 1: Path Resolution in Express Backend

Problem: PathError when using wildcard route app.get('*', ...)

Root Cause:

- Conflict with path-to-regexp library

Name – Toyash Patil
Roll No. 16010122140

- Incorrect use of `router` npm package instead of Express built-in router

Solution:

- Removed standalone `router` package
- Changed wildcard route to use `app.use()` middleware instead of `app.get('*')`
- Ensured correct path resolution for Vite's `dist/` folder

Challenge 2: Build vs Dist Folder Confusion

Problem: Backend attempting to serve from `build/` folder that didn't exist

Root Cause: Vite uses `dist/` for output, not `build/` like Create React App

Solution: Updated all references from `build/` to `dist/` in:

- Backend static file serving
- Backend fallback route
- buildspec.yml artifacts section

Challenge 3: CodeBuild Cannot Find package.json

Problem: CodeBuild error: "ENOENT: no such file or directory, open '/codebuild/output/src.../package.json'"

Root Cause: buildspec.yml running npm commands in root directory instead of `frontend/`

Solution: Added `cd frontend` command before all npm operations in buildspec.yml

Challenge 4: IAM Permission Errors

Multiple Access Denied Issues:

Issue A: User cannot create S3 buckets

- **Solution:** Attached `AmazonS3FullAccess` to IAM user

Issue B: User cannot create IAM roles

- **Solution:** Created roles via AWS Console using administrator account

Issue C: CodePipeline cannot use CodeStar Connection

- **Solution:** Added custom inline policy with `codestar-connections:UseConnection` permission

Challenge 5: PowerShell Command Syntax Errors

Problem: PowerShell treating `--` as unary operator in AWS CLI commands

Name – Toyash Patil
Roll No. 16010122140

Root Cause: Different quote handling between PowerShell and Bash

Solution: Escaped JSON in AWS CLI commands using PowerShell backtick syntax:

powershell

```
--assume-role-policy-document "{ `"Version`": `"2012-10-17`", ... }"
```

Challenge 6: S3 Bucket Naming Collision

Problem: "BucketAlreadyExists" error for my-app-pipeline-artifacts

Root Cause: S3 bucket names are globally unique across all AWS accounts

Solution: Used timestamped unique name: my-unique-app-pipeline-12345

Challenge 7: CodeStar Connections Not Visible in Console

Problem: "Developer Tools" menu not showing CodeStar Connections

Root Cause: AWS Console reorganization in recent updates

Solution: Created GitHub connection directly during pipeline creation flow instead of separately

---

Technical Workflow

Development to Deployment Flow

1. **Developer makes code changes locally**
    - Modify React components or Node.js backend
    - Test locally using npm start (frontend) and node index.js (backend)
2. **Commit and push to GitHub**

bash

```
git add .
git commit -m "Description of changes"
git push origin main
```

Name – Toyash Patil
Roll No. 16010122140

3. **CodePipeline automatically triggers (Source Stage)**
    - Detects push to `main` branch via CodeStar Connection
    - Downloads repository as `SourceArtifact` ZIP
    - Stores artifact in S3 bucket
4. **CodeBuild executes (Build Stage)**
    - Pulls `SourceArtifact` from S3
    - Reads `buildspec.yml` instructions
    - Installs Node.js 18 runtime
    - Navigates to `frontend/` directory
    - Runs `npm ci` for dependency installation
    - Executes `npm run build` to generate `dist/` folder
    - Packages `dist/` as `BuildArtifact`
5. **Deployment (Deploy Stage - Optional)**
    - Uploads `BuildArtifact` to S3 bucket
    - Can be configured for static website hosting
6. **Monitoring and Logs**
    - Build logs visible in AWS CodeBuild console
    - Pipeline execution history in AWS CodePipeline
    - CloudWatch Logs for detailed debugging

Cost Analysis (AWS Free Tier)

CodePipeline: 1 free active pipeline per month ✅

CodeBuild: 100 build minutes/month on general1.small ✅

S3: 5GB storage, limited requests ✅

Data Transfer: Within free tier for demo usage ✅

Total Estimated Cost for IA Project: $0 - $2 (if staying within limits)

Key Learnings and Best Practices

DevOps Principles Demonstrated

1. **Automation:** Eliminated manual build and deployment steps
2. **Version Control Integration:** Git-based workflow with automatic triggers
3. **Infrastructure as Code:** buildspec.yml defines build process
4. **Artifact Management:** Centralized storage in S3
5. **Security:** IAM roles with least-privilege access

Name – Toyash Patil
Roll No. 16010122140

AWS-Specific Knowledge Gained

- IAM role trust relationships and policy attachments
- CodeStar Connections for GitHub integration
- CodeBuild environment configuration
- S3 bucket policies and global naming requirements
- CloudWatch Logs for debugging

Technical Skills Developed

- Express.js routing and static file serving
- Vite build configuration and output structure
- YAML syntax for buildspec files
- AWS CLI usage and PowerShell compatibility
- Git workflow and branch management

Testing and Validation

Local Testing

- Frontend builds successfully with `npm run build`
- Backend serves frontend at `http://localhost:5000`
- API endpoint `/api/ping` returns expected JSON response

Pipeline Testing

- Push to GitHub triggers pipeline automatically
- Source stage successfully pulls repository
- Build stage completes without errors
- Artifacts correctly stored in S3 bucket
- Build logs show successful execution of all commands

Validation Criteria Met

- ✅ Code successfully pulled from GitHub
- ✅ Dependencies installed correctly
- ✅ Frontend built without errors
- ✅ Artifacts generated and stored
- ✅ Pipeline executes end-to-end
- ✅ Changes trigger automatic rebuilds

Name – Toyash Patil
Roll No. 16010122140

Conclusion

This project successfully demonstrates a production-ready CI/CD pipeline using AWS CodePipeline, showcasing the integration of multiple AWS services (S3, CodeBuild, CodePipeline, IAM, CodeStar Connections) to automate the software delivery process for a React + Node.js application.

Key Achievements:

- Fully automated build and deployment pipeline
- GitHub integration with automatic triggers
- Proper IAM security configuration
- Comprehensive error handling and troubleshooting
- Documentation of challenges and solutions

Skills Demonstrated:

- DevOps pipeline architecture
- AWS cloud services configuration
- Full-stack application development
- Problem-solving and debugging
- Infrastructure as Code principles

Future Enhancements:

- Add automated testing stage
- Implement deployment to EC2 or Elastic Beanstalk
- Configure CloudFront CDN for frontend
- Add environment-specific configurations (dev/staging/prod)
- Implement rollback mechanisms