POC 02: Smart Building Management Platform - CI/CD Modernization

Prepared for: **Saurabh Dindokar**

Role: **AWS DevOps Engineer**

Public-safe project name: **Smart Building Management Platform**

Confidentiality Note

This document is intentionally public-safe. It does not include real client names, internal project names, organization-owned source code, private URLs, IP addresses, credentials, account IDs, screenshots, or any confidential implementation details.

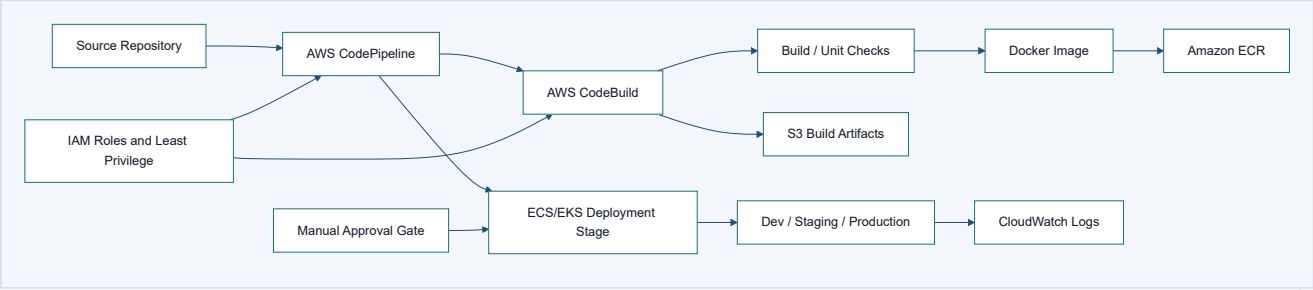
Work Sample Summary

Public-safe work sample for moving release flow from a traditional Jenkins-heavy pattern toward AWS-native CI/CD using CodePipeline, CodeBuild, ECR, S3 artifacts, IAM, and CloudWatch logs.

Problem Statement

- â€¢ The release workflow required a cleaner build, artifact, image, and deployment chain.
- â€¢ Manual dependency on one CI/CD path increased operational risk during repeated deployments.
- â€¢ The team needed documented plan-only migration steps that could be reviewed before production implementation.

Mermaid Architecture Diagram



Mermaid architecture diagram

My Responsibilities

- â€¢ Mapped existing Jenkins release stages into AWS-native pipeline stages.
- â€¢ Prepared pipeline flow for source, build, Docker image publishing, artifact storage, approval, and deployment.
- â€¢ Documented IAM role boundaries for pipeline, build, ECR, S3, and CloudWatch access.
- â€¢ Defined rollback and artifact traceability notes for safer releases.

Implementation Approach

â€¢ Pipeline stages separate source fetch, build validation, image creation, image push, and deployment.

â€¢ Build logs and failure traces are routed to CloudWatch for easy troubleshooting.

â€¢ Environment-specific deployment stages can use approval gates for production control.

â€¢ Artifacts and image tags are traceable so a previous version can be redeployed when needed.

Reliability, Security and Operations Focus

â€¢ Health checks, validation steps, and rollback planning were treated as part of deployment quality, not afterthoughts.

â€¢ Secrets, access boundaries, private networking, backup retention, and monitoring visibility were documented wherever applicable.

â€¢ The POC is written so another engineer can understand the flow, reproduce the approach, and extend it safely without exposing confidential data.

Validation Checklist

Review architecture and confirm each component has a clear responsibility.

Validate deployment or automation steps in a non-production environment first.

Confirm logs, health checks, backup behavior, and rollback steps are documented.

Confirm access, secrets, and network boundaries follow least-privilege expectations.

Capture final runbook notes for handover and support readiness.

Expected Outcomes

â€¢ Improved release traceability.

â€¢ Cleaner AWS-native build and deployment workflow.

â€¢ Reduced manual deployment dependency.

â€¢ Better CI/CD documentation for future onboarding.

Technologies Used

AWS CodePipeline, AWS CodeBuild, Jenkins, Docker, Amazon ECR, S3, IAM, CloudWatch, ECS/EKS

Naukri Work Sample Description

Public-safe DevOps POC by Saurabh Dindokar covering architecture, responsibilities, implementation approach, validation checklist, operational focus, and measurable delivery outcomes. The document uses generic project naming and does not disclose any confidential client or organization information.