FPAC CI/CD Architecture

This document provides an executive summary of the FPAC CI/CD automation architecture, based entirely on AWS CDK, least-privilege IAM, and automated deployment pipelines.

** Purpose & Executive Summary**

The FPAC CI/CD architecture establishes a **secure, least-privilege, fully automated deployment model** using the **AWS Cloud Development Kit (CDK)**.

Key Principles

- Infrastructure deployed **exclusively through IaC** (no console changes).
- All deployments are **idempotent**, deterministic, and repeatable.
- Strict **least-privilege IAM** approach using two core policies:
  - **Deployer IAM User** – minimal permissions, cannot create AWS resources.
  - **CDK-EXECUTOR Policy** – CloudFormation's execution role with approved service access.
- Ensures **auditability**, **security compliance**, and **deployment consistency** across environments.

** Current Challenges (Pre-Automation)**

Legacy deployments rely heavily on manual steps, resulting in:

- **Inconsistent configurations** across DEV / TEST / CERT / PROD.
- **Operational risk** from human error and environment drift.
- **Limited ability to roll back** or test hot-fixes safely.
- Excessive reliance on human **AWS Console access**.
- No guarantee that lower-environment builds match the version deployed into production.
- Inability to enforce scanning and automated validation before deployment.

** Architecture Overview & Security Model**

Two-Policy Deployment Architecture

1. Deployer IAM User

- Used only by Jenkins/CI pipelines.
- Can synthesize CDK and interact with CloudFormation.
- Using the CDK-EXECUTOR Policy via the CDK bootstrap.
- **Cannot directly modify AWS resources.**

2. CDK-EXECUTOR Policy

- CloudFormation uses this policy during deployments.
- Grants only the permissions required to deploy approved services:
  - Lambda, Glue, S3, DynamoDB, KMS, SSM, API Gateway, etc.
- Enforces FPAC's least-privilege and service-restriction requirements.

Security & Compliance

- Aligned with **NIST 800-53 AC-6 (Least Privilege)**.
- **Separation of duties** maintained (IAM & network controlled by OCIO/DISC).
- All changes logged: **CloudFormation + CI/CD logs**.
- Manual console changes are prohibited to protect **idempotency**.

** Transition Plan & Environments**

Target Architecture

- Two new AWS accounts: **DEV** and **STAGING**.
- STAGING becomes the pre-production and hot-fix testing environment.
- All deployments flow through:
  1. **Bitbucket Mono-Repo**
  2. **Jenkins CI/CD**
  3. **AWS CDK → CloudFormation**

Transition Approach

1. Rebuild ~30 existing data pipelines as CDK applications.
2. Move all solutions into a dedicated **Bitbucket mono-repo**.
3. Implement Jenkins pipelines for:
   - DEV deployments
   - STAGING deployments
   - PROD deployments

4. Retire legacy deployments once fully transitioned.
5. Utilize us-east-2 in shared accounts temporarily for required data access.

** Benefits & Required Actions**

Benefits

- Reliable, **consistent deployments** across all environments.
- Built-in **rollback** and **hot-fix testing** capability.
- Reduces and simplifies **AWS Console access** for personnel.
- Strongly enforces **least privilege** and reduces security risk.
- Fully **auditable**, **version-controlled**, and **automated** infrastructure management.

Required Actions

- Create new **DEV** and **STAGING** AWS accounts.
- Configure both the **Deployer** and **CDK-EXECUTOR** IAM policies.
- Create **Deployer IAM Users** in each account and distribute keys securely.
- Run **AWS CDK bootstrap** (Admin-level action required).
- Configure Jenkins pipelines + Bitbucket webhooks for automated deployments.
