



Describe your approach for implementing infrastructure as code (CloudFormation/CDK) in your CI/CD pipeline.

As a DevOps engineer working on a fast-growing e-commerce platform, I realized early on that managing cloud infrastructure manually was error-prone and slow. So I adopted Infrastructure as Code (IaC) using AWS CDK, integrated into a fully automated CI/CD pipeline using CodePipeline and CodeBuild. Let me walk you through my approach step by step.



Architecture Planning

We needed a reliable 3-tier architecture:

Layer Technology Stack

Frontend React.js hosted on S3 + CloudFront

Backend Node.js containerized with Docker on ECS (Fargate)

Database Amazon RDS PostgreSQL

To support this, we also needed:

- Secure VPCs
- IAM roles and policies
- Secrets Manager for DB credentials
- Monitoring (CloudWatch)
- CI/CD for auto deployment



Defining Infrastructure Using AWS CDK

Instead of YAML or JSON (used in CloudFormation), I preferred AWS CDK in TypeScript for its reusability, modularity, and ability to use programming constructs like loops, conditions, and inheritance.

Folder Structure:

```
r
iac-infra-repo/
├── core/           # VPC, IAM, Security Groups
├── services/
│   ├── frontend/  # S3 + CloudFront
│   ├── backend/   # ECS, ALB
│   └── database/   # RDS + Secrets Manager
├── environments/
│   ├── dev/
│   ├── qa/
│   └── prod/
└── pipeline/      # CI/CD deployment pipeline
```

Each resource stack was reusable across environments using CDK contexts and environment-specific variables like:

```
ts
{
  instanceType: 't3.micro',
  dbBackupRetentionDays: 7,
  environment: 'dev'
}
```

Integrating CDK with CI/CD Pipeline

CodePipeline Flow:

Stage	Tool Used	Action
Source	GitHub	Trigger pipeline on commit to feature/*, develop, or main
Build	CodeBuild	Install CDK, run synth, test, cdk diff
Test	CodeBuild	Run unit + integration tests + security scans
Deploy	CodePipeline	CDK deploy to target environment (Dev → QA → Prod)

CodeBuild buildspec.yml example:

yaml

phases:

```
install:
  commands:
    - npm install -g aws-cdk
    - npm ci
build:
  commands:
    - cdk synth
    - cdk diff
    - cdk deploy --require-approval never --context env=dev
```





artifacts:

files:

- '**/*'

Testing Infrastructure as Code

I firmly believe infrastructure needs to be tested like application code. So I added:

-  Unit tests using CDK assertions (@aws-cdk/assert)
-  Integration tests to launch and validate real stacks
-  Security scans using cfn-nag and cdk-nag
-  Drift detection with scheduled CloudWatch + Lambda jobs to detect **manual infra changes**

“If someone edits a security group manually, I’ll know within hours.”

Disaster Recovery Test

One weekend, I simulated a region-wide failure in us-east-1.

Using the same CDK code:

- Deployed full infra to us-west-2
- Restored RDS backups
- Reconnected via updated Route 53 DNS

Time to recover: under 30 minutes.

This validated IaC as our disaster recovery plan, not just a convenience tool.

Epilogue: Final Outcomes

Goal	Achieved Through
Repeatable, fast deployments	CDK + modular stacks + CI/CD
Lower failure rate	Testing infra + app together
Secure & compliant infra	Scanning + tagging + policy enforcement
Cost control	Cleanup logic + tagging + monitoring
Zero config drift	GitOps + drift detection
Fast disaster recovery	CDK redeployment + backups + routing

Takeaway

Infrastructure as Code with CDK + CloudFormation transformed our deployment from manual chaos to a fully automated, resilient, and testable system. By integrating it with CodePipeline, we turned infrastructure into version-controlled, testable, and promotable code—just like our applications.

What is the main motivation for adopting Infrastructure as Code (IaC) in a project?

- A. Cost optimization using Spot Instances
- B. Manual infrastructure management was slow and error-prone
- C. Avoid using Docker containers
- D. Integrate with third-party hosting services

B. Manual infrastructure management was slow and error-prone 

Why was AWS CDK preferred over raw CloudFormation YAML/JSON?

- A. CDK deploys faster
- B. CDK supports only AWS-native resources
- C. CDK allows use of programming constructs like loops and inheritance
- D. CDK requires no prior AWS knowledge

C. CDK allows use of programming constructs like loops and inheritance 
