Mono-Repo vs Multi-Repo Decision Guide

Your Context

You have:
- A template repo for AWS Lambdas (multi-Lambda support).
- Another template for EKS apps (multi-app support needs customization).
- A new **monorepo** combining all components with shared automation for:
  - API spec generation
  - CI/CD script creation
  - Environment file generation
- Tooling stack: GitHub Actions, Argo CD, Terraform.

Challenges:
- No smart deploys in monorepo based on component changes.
- Duplication and maintenance overhead in multi-repo.
- Monorepo has a **steep learning curve** due to custom scripts.

MONOREPO APPROACH

Pros

- **Centralized management**
- **Code sharing** without duplication
- **Unified automation** for spec/env/scripts
- **Consistent versioning**
- **Simpler post-onboarding experience**

Cons

- **Complex deployment logic**
- **Slower CI/CD unless optimized**
- **ArgoCD scaling issues**
- **Higher learning curve**
- **Merge conflicts in shared files**

MULTI-REPO APPROACH

Pros

- **Service isolation**
- **Smart deployment capabilities**
- **Simpler onboarding**
- **Clean Git history**
- **Parallel team dev possible**

Cons

- **Code duplication**
- **Repo sprawl**
- **Version drift in shared packages**
- **Higher CI/CD maintenance**
- **Governance inconsistencies**

MIDDLE GROUND APPROACH

Options:

- **Hybrid monorepo with modular subfolders**:
  ```
  /infra/
  /libraries/
  /apps/
    /lambda-foo/
    /lambda-bar/
    /eks-app-1/
  ```
- **Domain-based multi-repo split**
- **Shared NPM packages / Git submodules**
- **Reusable GHA workflows**

Decision Matrix

Factor	Favors Mono-Repo	Favors Multi-Repo
Deployment granularity needed	Complex	Easier
Code reuse/shared infra	Simplified	Requires packaging
Developer onboarding complexity	High (initially)	Lower
Team independence	Coupled	Decoupled
GitHub/CI/CD pipeline scaling	Shared pipelines	Needs per-repo setup
Argo CD scalability & app mgmt	Custom logic needed	Easier with one app/repo
Governance and standards enforcement	Centralized	Needs active policing
Tooling and automation investment	Reuse possible	Duplicates required

Recommendations

Choose Monorepo if:

- Heavy reliance on shared scripts/specs.
- Centralized DevOps is prioritized.
- Willing to manage deployment complexity.

Choose Multi-Repo if:

- Services are independent.
- Fast and isolated deploys are a must.
- Scaling team/service count significantly.

Go Hybrid if:

- You need a balance.
- Shared utilities live in one repo.
- Teams want autonomy for their services.

Generated by ChatGPT