Cloud Services Best Practices

A Guide for *THE* Team

Nick Ross

Overview

- How do we make technology choices (as they relate to cloud)?
  - Why do we make these choices?
- What are some best practices for using these systems?

Cloud Technology Decision Making

🔧 Evaluation Dimensions

💰 Cost

Simpler tools = simpler pricing. Complex tools have hidden costs and unpredictable bills

🧩 Complexity

Cognitive load, moving parts, dependencies

🔍 Debugging/Building Ease

Can we troubleshoot effectively? Good errors? Local reproduction? Does it make us faster?

🆘 Support & SLAs

What happens when things break?

⚙️ Core Principles for Technology Choices

- ✨ **Value Simplicity:** Reduce complexity at every opportunity
- 🏛️ **Value Boring:** Mature/battle-tested over bleeding edge
- 💥 **Well-defined Interfaces and Surface Areas:** What happens when things don't work as expected (Blast Radius)? How can we swap it out?
- #️⃣ **Minimize technologies:** Given two systems of similar complexity the one with the fewer technologies is better.
- 💲**Cost:** This matters
- ✈️ **Prioritize local development:** Can it work on an airplane?

Strategic Decision Making

🎯 Strategy Considerations

- 📐 **Assumption #1:** We are not the ICP for cloud services:
  - We are not hyper scalers
  - Our clients are (generally) not tech savvy
- 📐 **Assumption #2:** Intentionally or not, these systems are designed for addiction/increased usage.

🎯 Context Matters

Context matters: Use managed services when operational overhead justifies the trade-off. Balance portability with practical engineering constraints.

🔧 One-off Tasks

Vendor tools and click-through interfaces are acceptable

🔄 Repeatable Tasks

Minimize vendor exposure, maximize portability

🎓 Learning Investment

Where should we spend our time?

🚫 Don't invest in learning one-off tools

Time spent learning tools we'll only use once is wasted

🚫 Don't invest in non-transferable concepts

Learn skills that apply broadly, not vendor-specific abstractions

AWS Services Spectrum

✅ Good Simple Primitives	⚡ OK Light Abstractions	⚠️ Caution Heavy Operational	❌ Avoid High Lock-in
• EC2 • S3 • Cloudfront • ElastiCache	• Lambda • Load Balancers • SES • ECS / EKS	• RDS • SQS	• DynamoDB • SageMaker • Bedrock • Glue • Cognito
Easy local dev, low lock-in, direct control	*Light convenience layer, easier to swap, annoying to setup*	*Higher cost premium, more operational weight. Why would they be used?*	*Expensive to escape*

Implementation Workflows

🔧 Cloud Development Workflow

1

🕸️ Build a Spider Web

Create the simplest *local* system that touches everything. Go wide, not deep. Map what needs to call what and how.

2

🔒 Easy Deploy & Verify → HARDEN

Deploy to cloud, verify all connections work end-to-end. Then harden (permissions, etc.) this foundation before additional development

3

🚀 Develop Locally, Deploy Often

Build features locally with fast feedback loops. Deploy frequently to catch integration issues early.

✈️ Local Options

AWS services with excellent local development alternatives

- **S3** ↔ MinIO
- **RDS** ↔ Postgres / MySQL
- **ElastiCache** ↔ Redis / Memcached (kinda)
- **DynamoDB** ↔ DynamoDB Local
- **SQS** ↔ ElasticMQ (Kinda)
- **Lambda** ↔ Functions (kinda)

🔧 Easy DevOps First, IaC Last

> Get it working manually (ssh, git pull, github actions, *whatever*), understand it deeply, then automate with tools like Pulumi/Terraform

Automation comes after understanding

🔒 Security Best Practices

- **Use IAM roles, not access keys** - Every service gets a role, no long-lived credentials
- **Temporary credentials via STS** - Auto-rotating, time-limited access
- **Least privilege principle** - Grant minimum necessary permission

Cheat when starting: grant super large permissions and then narrow.

Best Practices

🔧 Best Practices

1

🔄 Moving from dev to prod

When and why do I want to move things between dev and prod?

2

⏩ Fast Cycles

What is the *minimum* testable system

3

🚀🚀🚀 Deploy, Deploy, Deploy, Deploy, Deploy

Build features locally with fast feedback loops. Deploy frequently to catch integration issues early.

4

⚠️ Debug / Logging

How do I debug? Where do I route logs