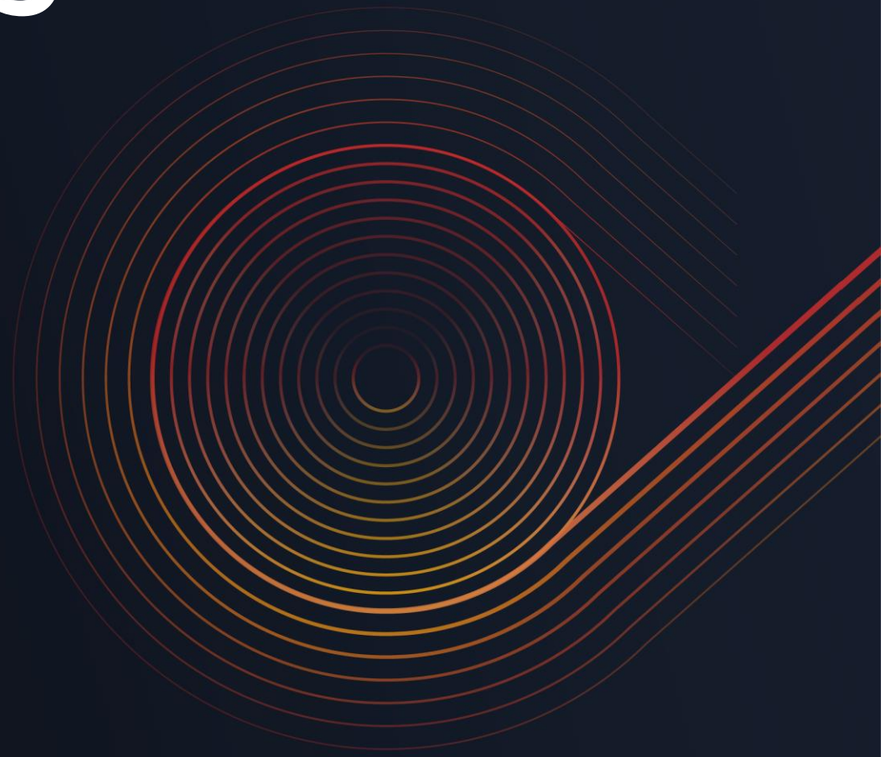
aws modern apps

Solving Testing Challenges for Serverless on AWS

Kapil Shardha, Principal Solutions Architect



What’s on the agenda!

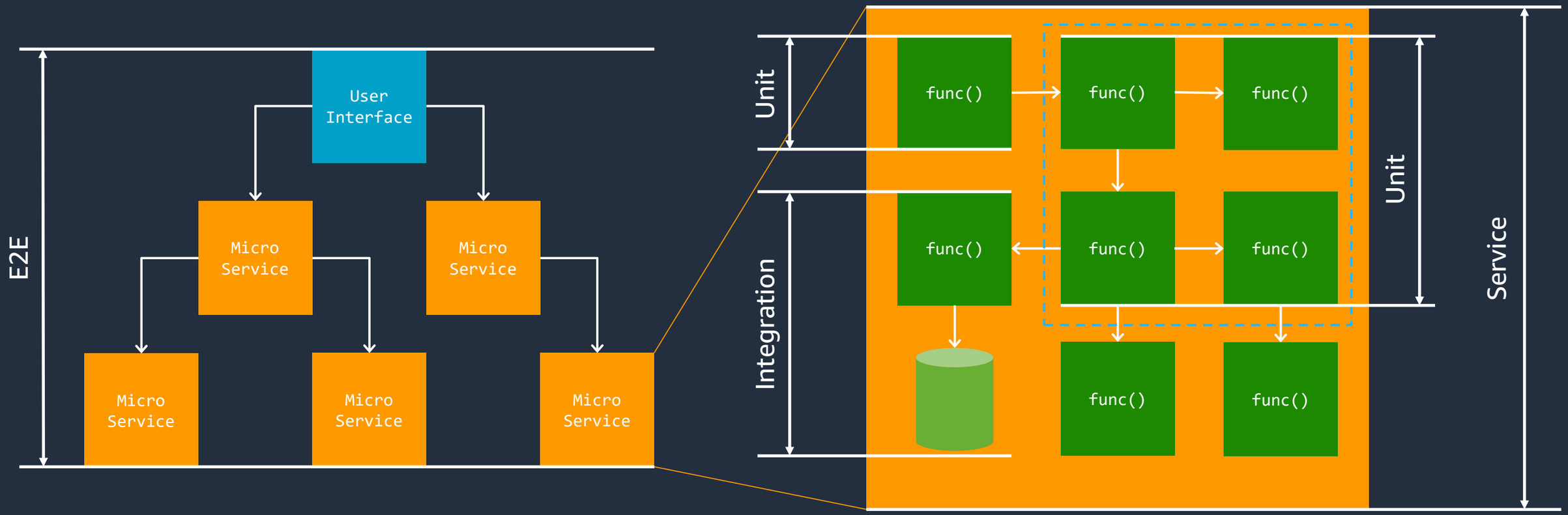
- A primer on automated testing
- Challenges with testing serverless applications
- Testing strategies for AWS serverless applications
- Minimizing external integration testing
- Optimize testing distributed and event-driven architectures
- Managing cloud environments for developers

aws modern apps

A Primer on Automated Testing

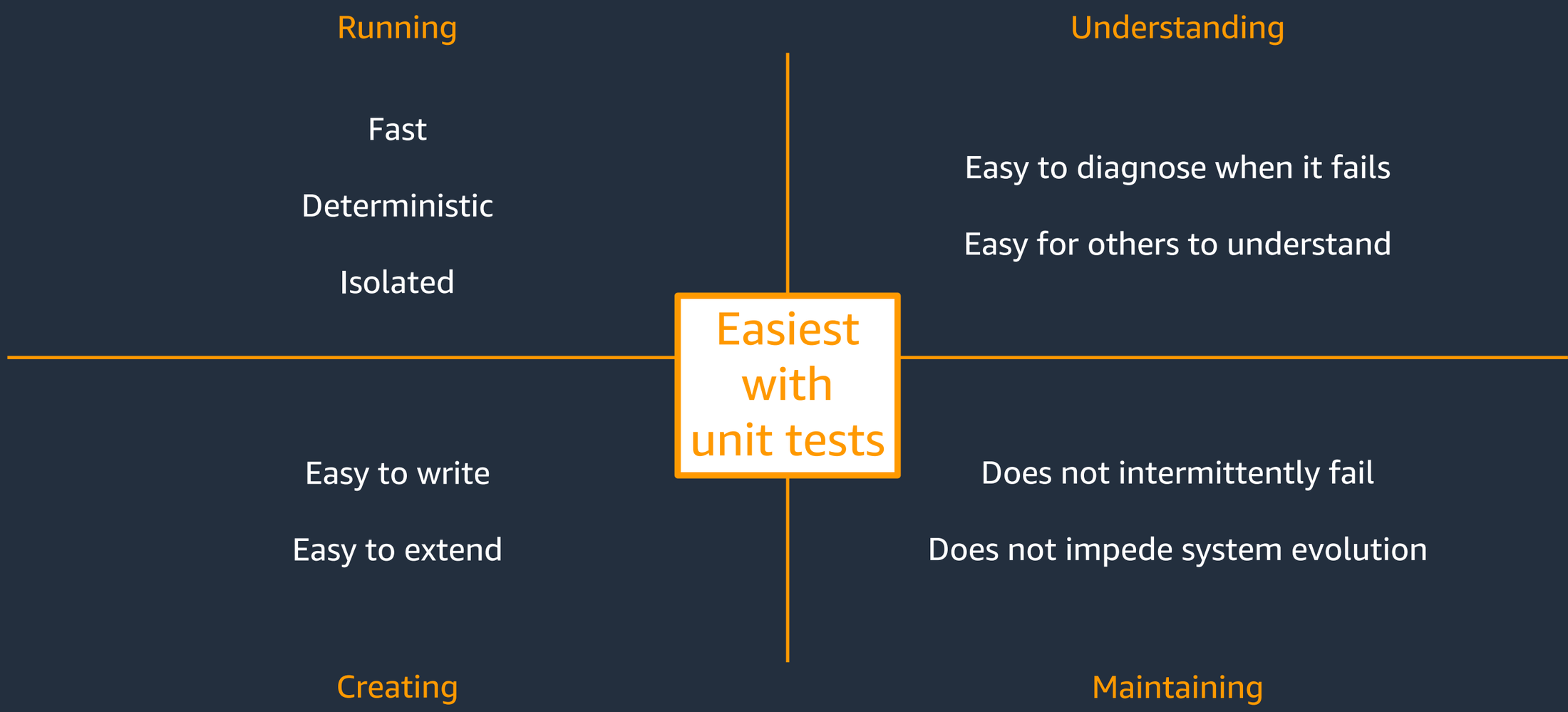
aws modern apps

Types of automated tests



aws modern apps

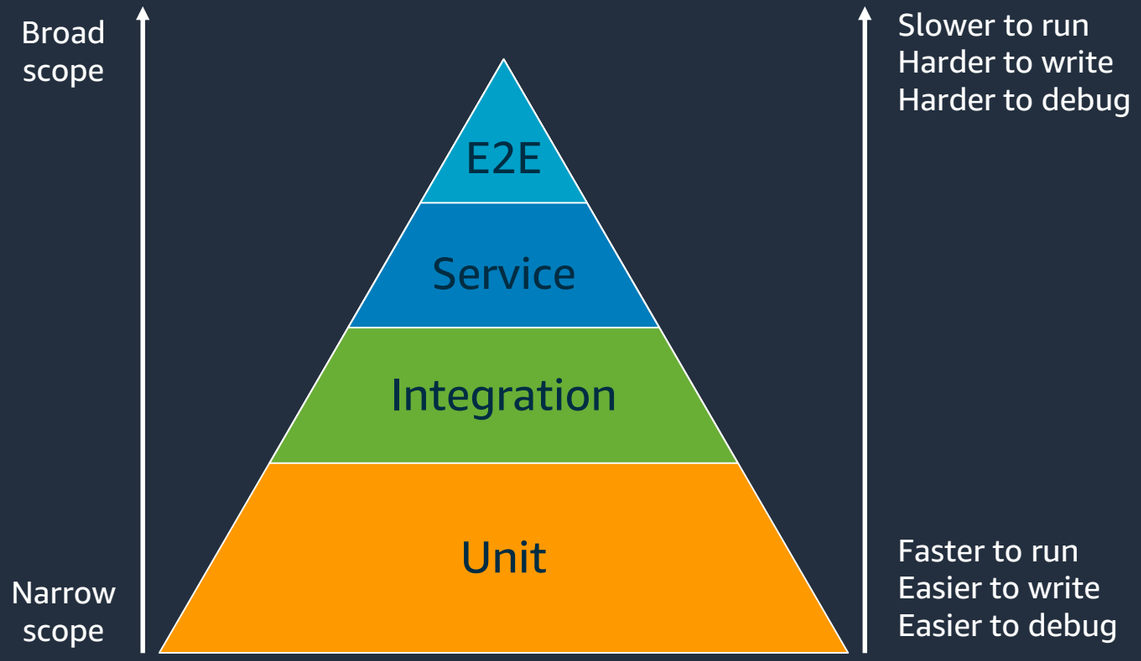
What makes a good automated test?



aws modern apps

The Test Pyramid – because unit tests aren't enough

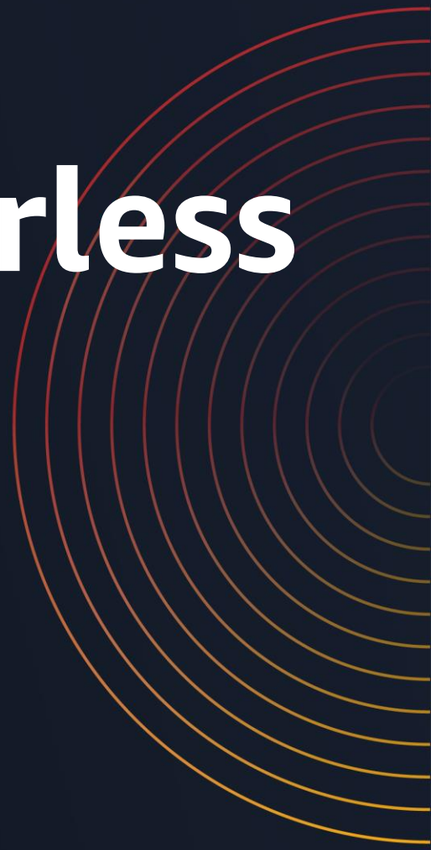
We need higher level tests to check the system actually works
(but not too many)



Assumption: the developers write the tests

aws modern apps

Challenges with Testing Serverless Applications



aws modern apps

Serverless architecture

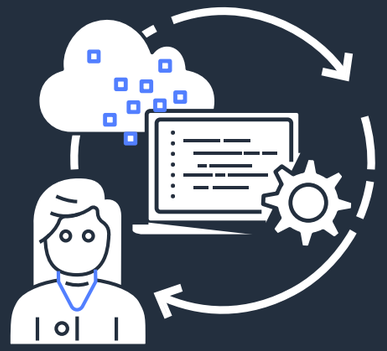
Challenges

Features

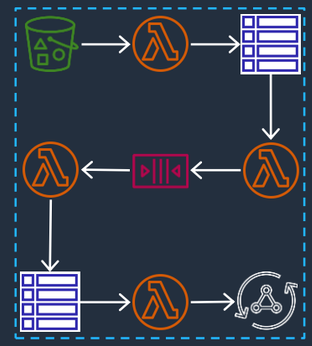
Managed Services

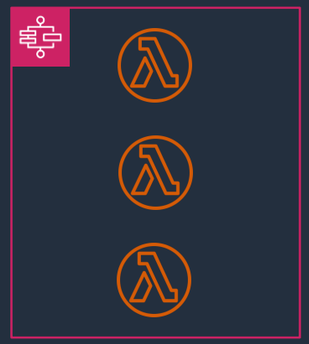
Distributed Architecture

Remote deploy, invoke & inspect

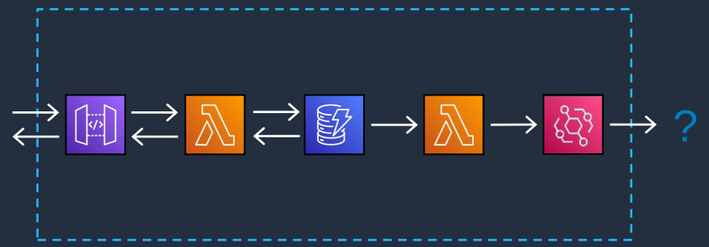


More testing with external resources





Drift towards more broad-stack testing of business logic



Need to test asynchronous side-effects

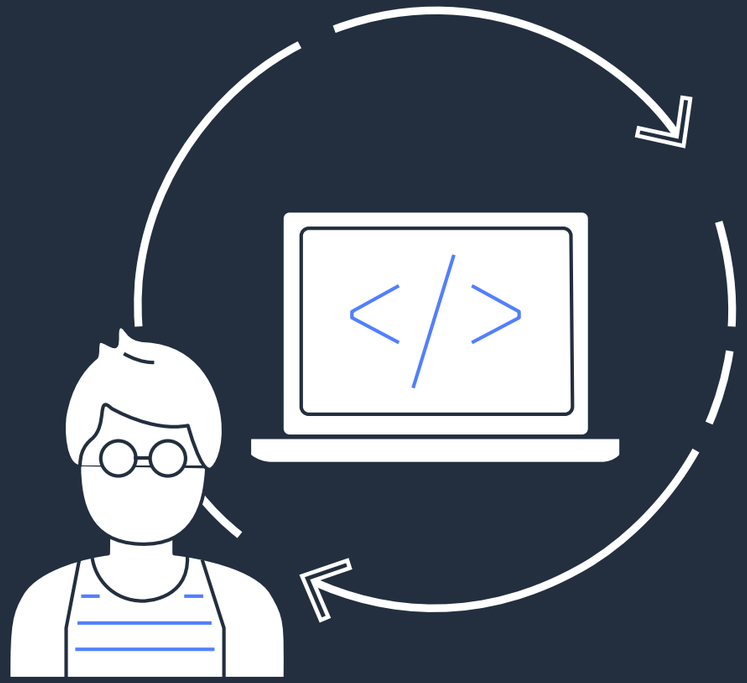
aws modern apps

Testing Strategies for AWS Serverless Applications

aws modern apps

Development runtime environment

Traditional Application



Local runtime environment

Local deploy, invoke & debug

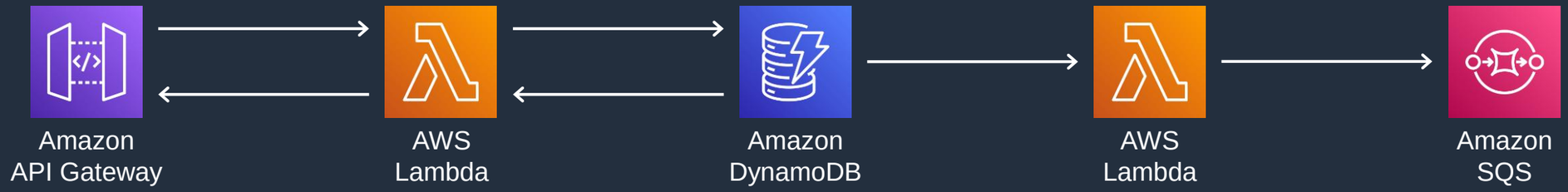
Serverless Application

Remote runtime environment

Remote deploy, invoke & inspect

aws modern apps

Example workload



aws modern apps

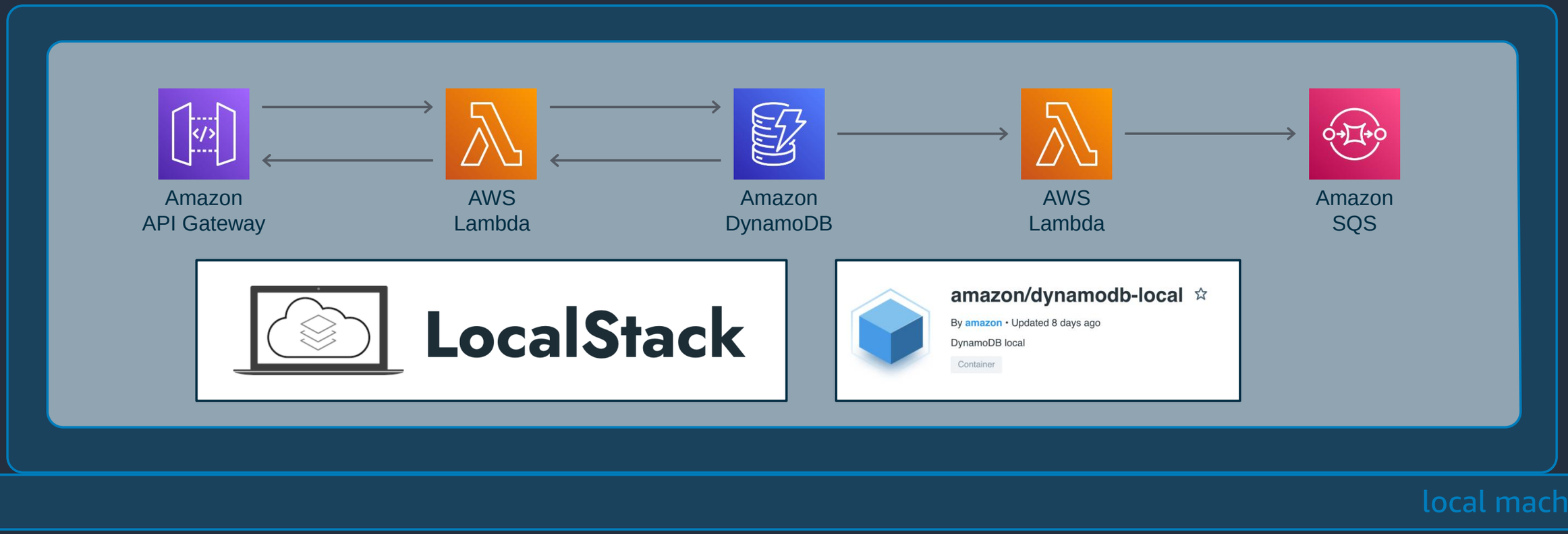
Approach 1: Whole system emulation

Test system-wide changes without deployment

Unrealistic feedback (incomplete emulation)

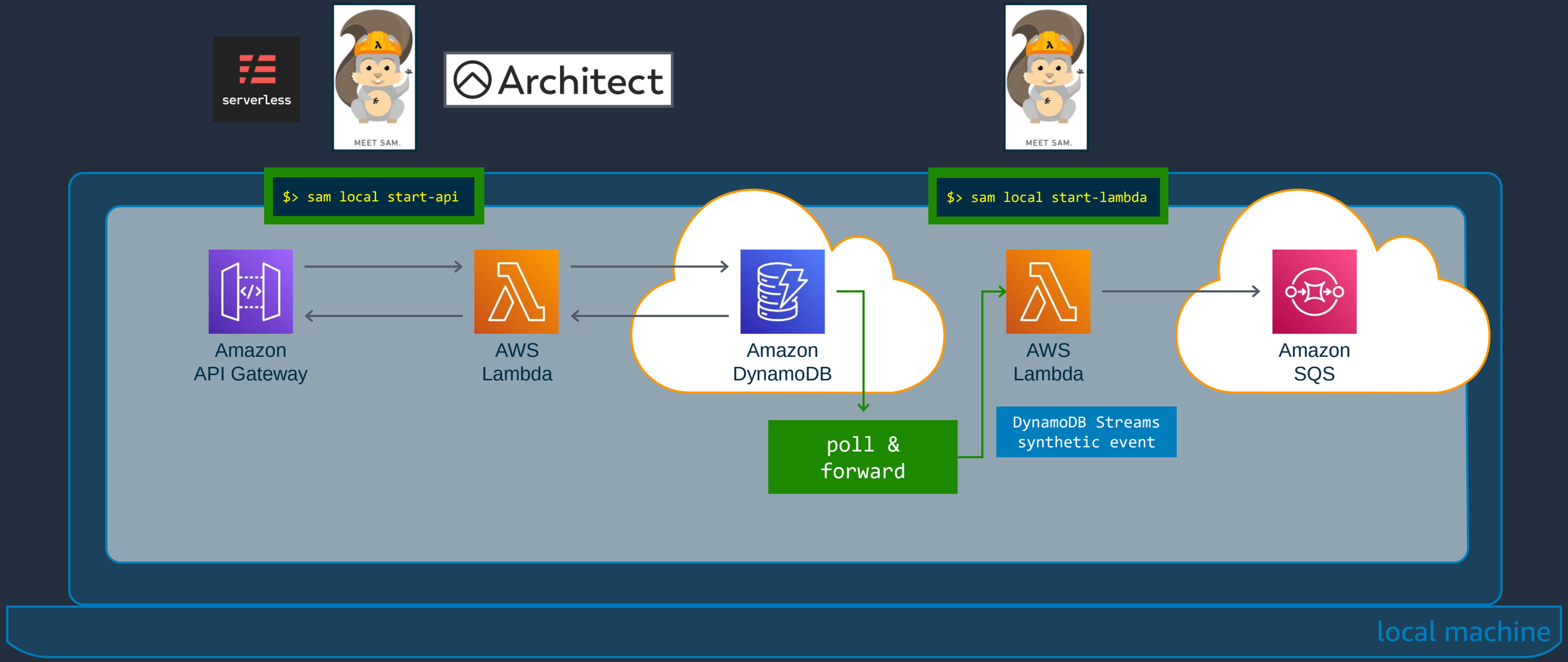
Work offline

Need to duplicate infra config / write glue code



aws modern apps

Approach 2: Emulate Lambda and synchronous event sources



aws modern apps

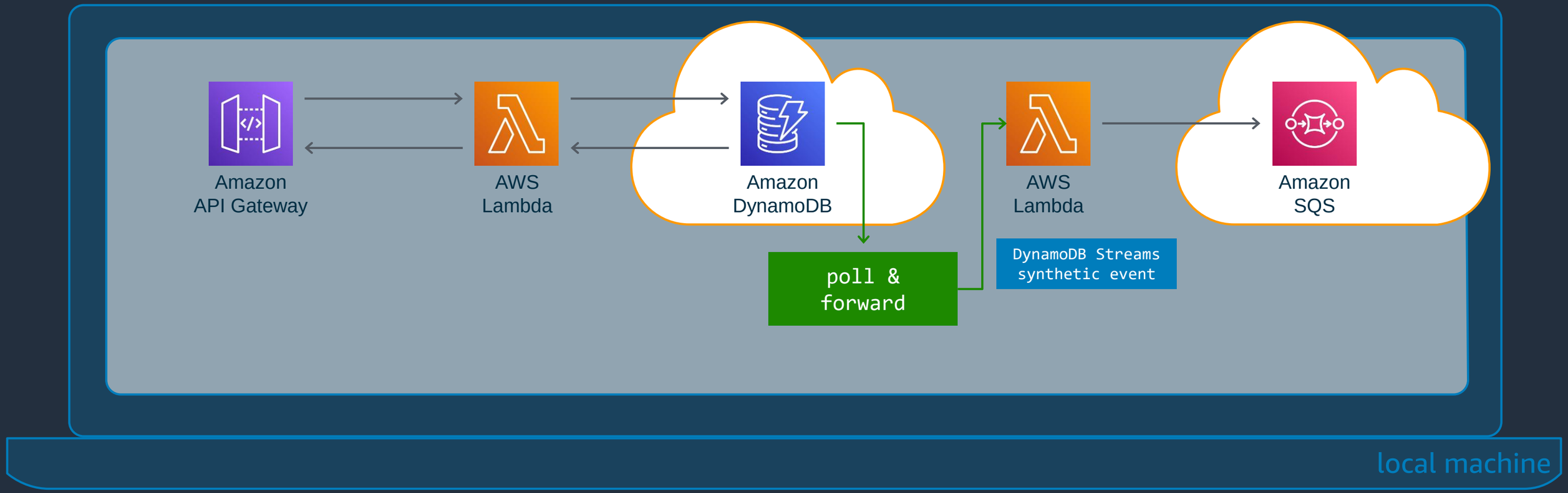
Emulate Lambda and synchronous event sources

Test many changes without deploying

DIY async/stream event source polling

Support local breakpoint debugging

Still need to verify in the cloud



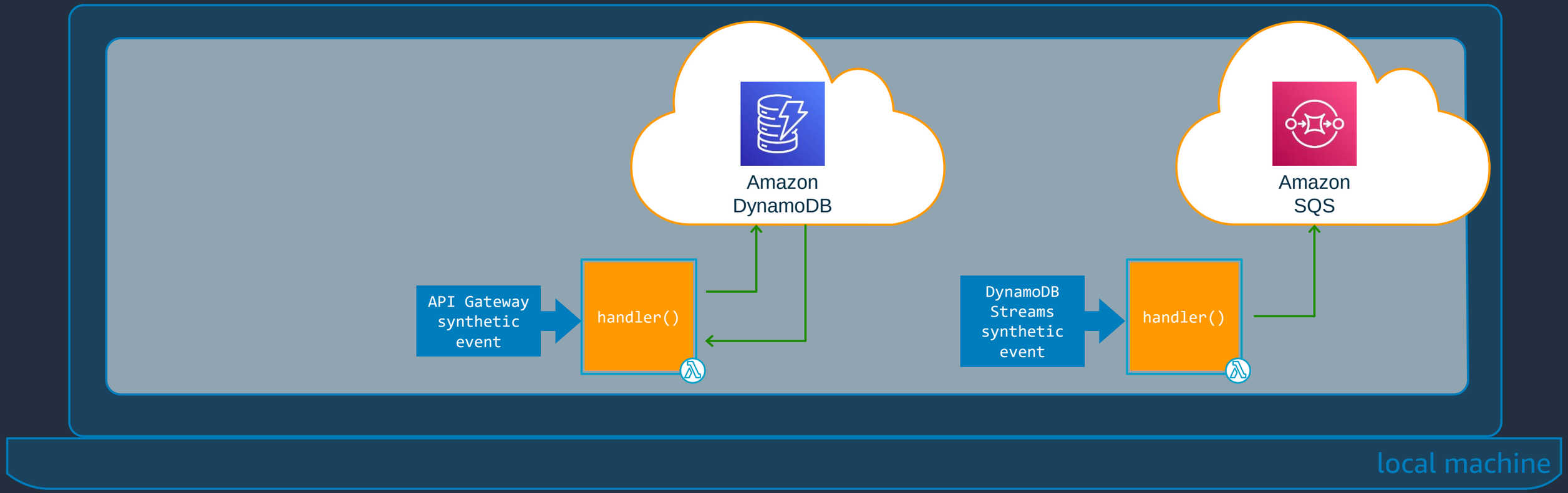
aws modern apps

Approach 3: No emulation

Reliable feedback (except for permissions)

Narrow-scope testing

Need to deploy function to test with real event source and permissions

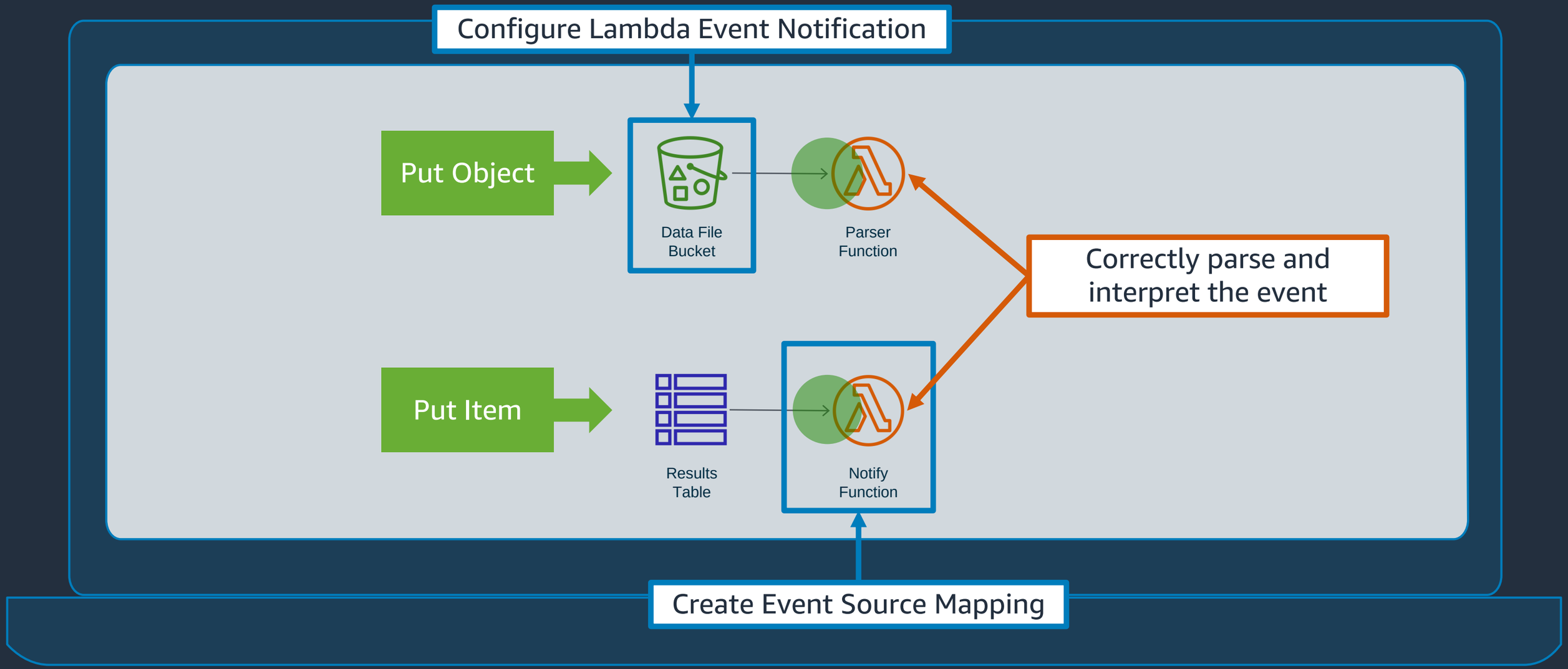


aws modern apps

Minimizing External Integration Testing

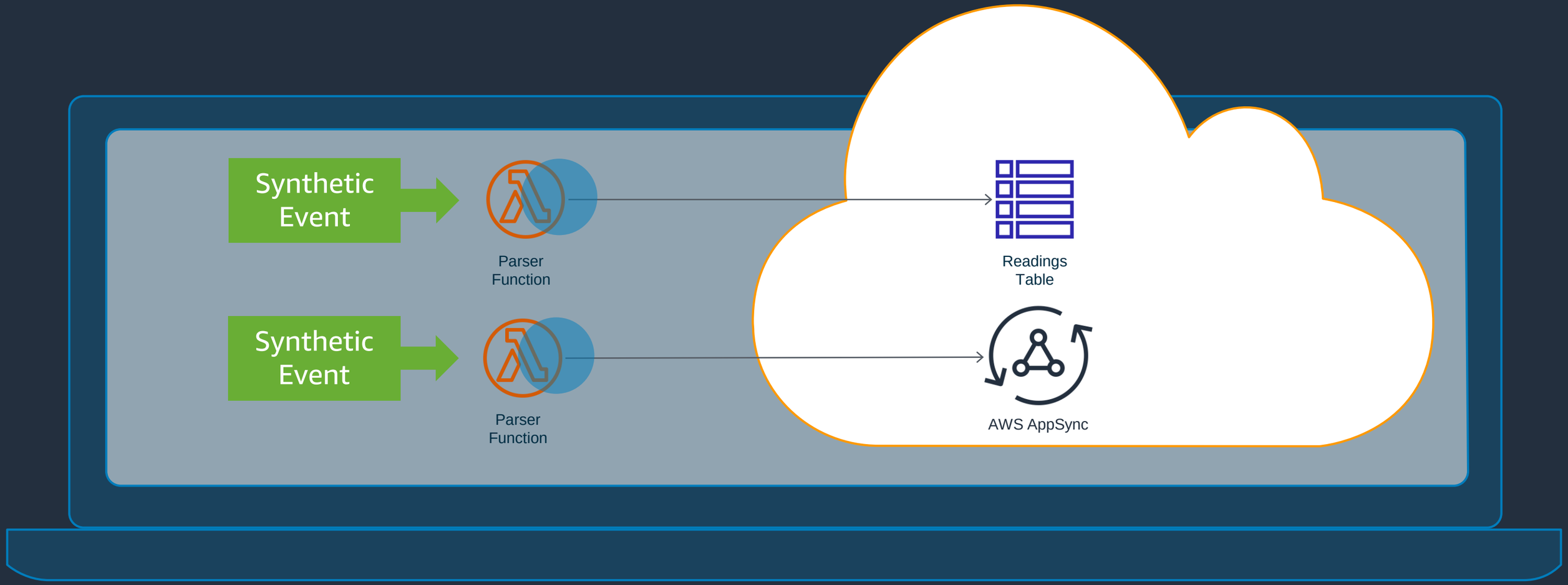
aws modern apps

AWS Service triggering a Lambda function



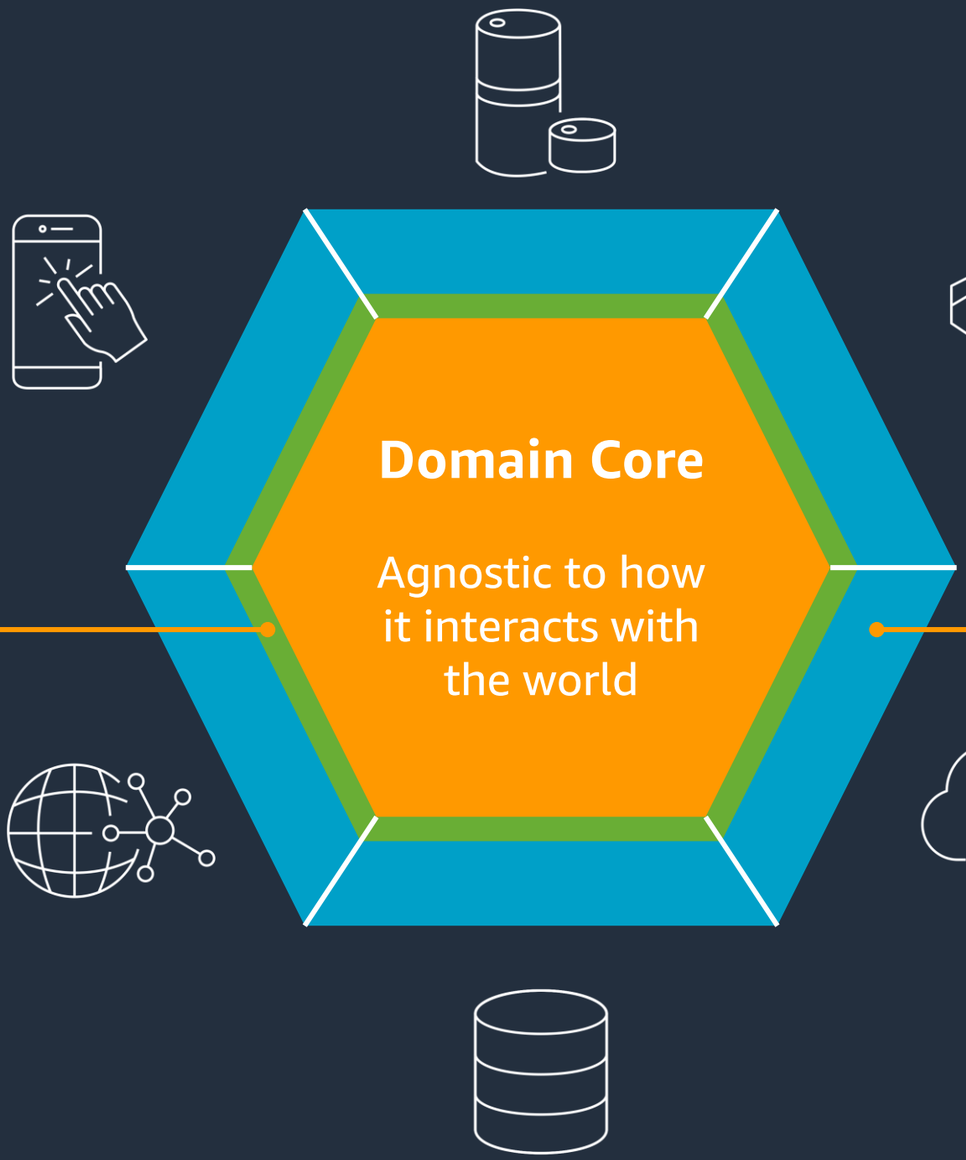
aws modern apps

Lambda function calling an AWS Service



aws modern apps

Ports & adapters architecture (aka “Hexagonal”)



Port

Code interface that hides external concerns

Defines domain’s interactions with the outside world

Provides entry & exit points you call or mock in tests

Adapter

Implements Port interface for a specific external system

Maps between external and domain concepts

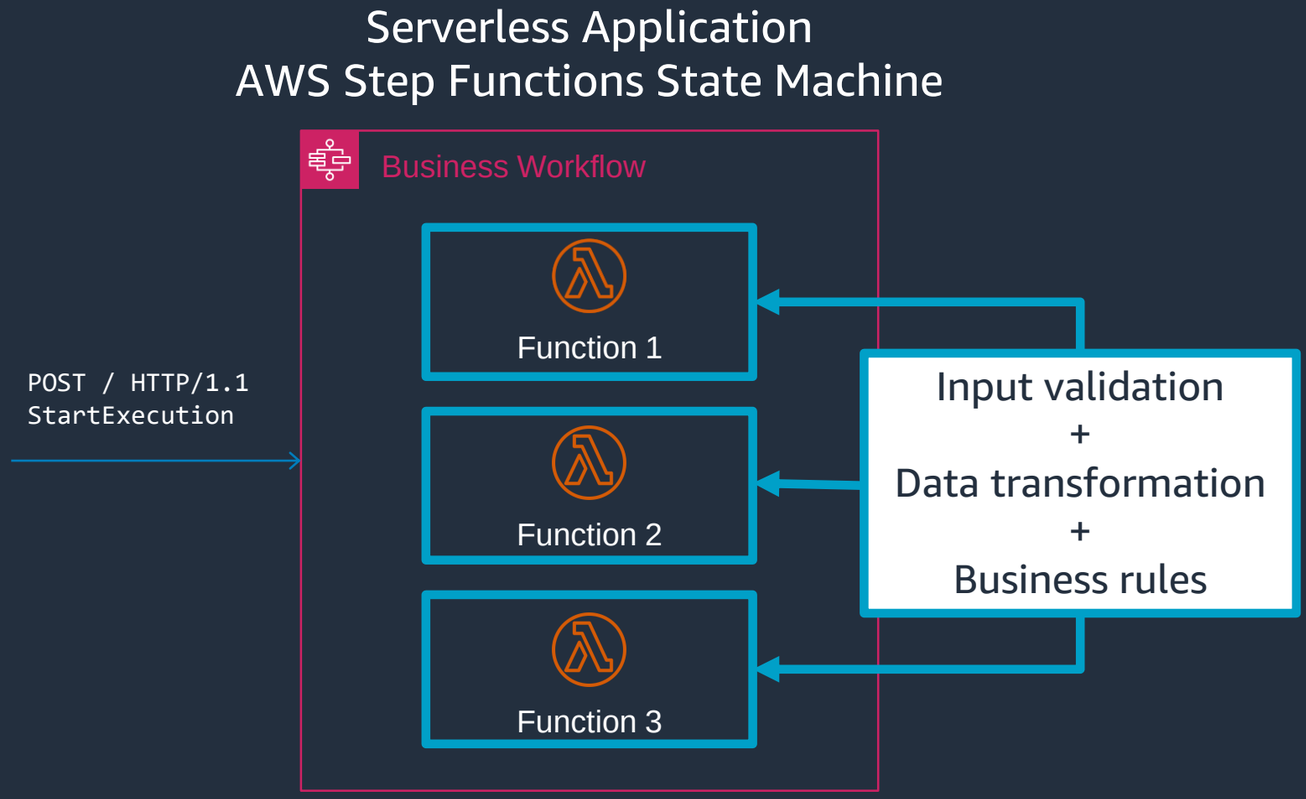
“Anti-corruption layer” (Domain Driven Design)

aws modern apps

Optimize Testing Distributed and Event-Driven Architectures

aws modern apps

Business logic locality



Drift towards more broad-stack testing of business logic

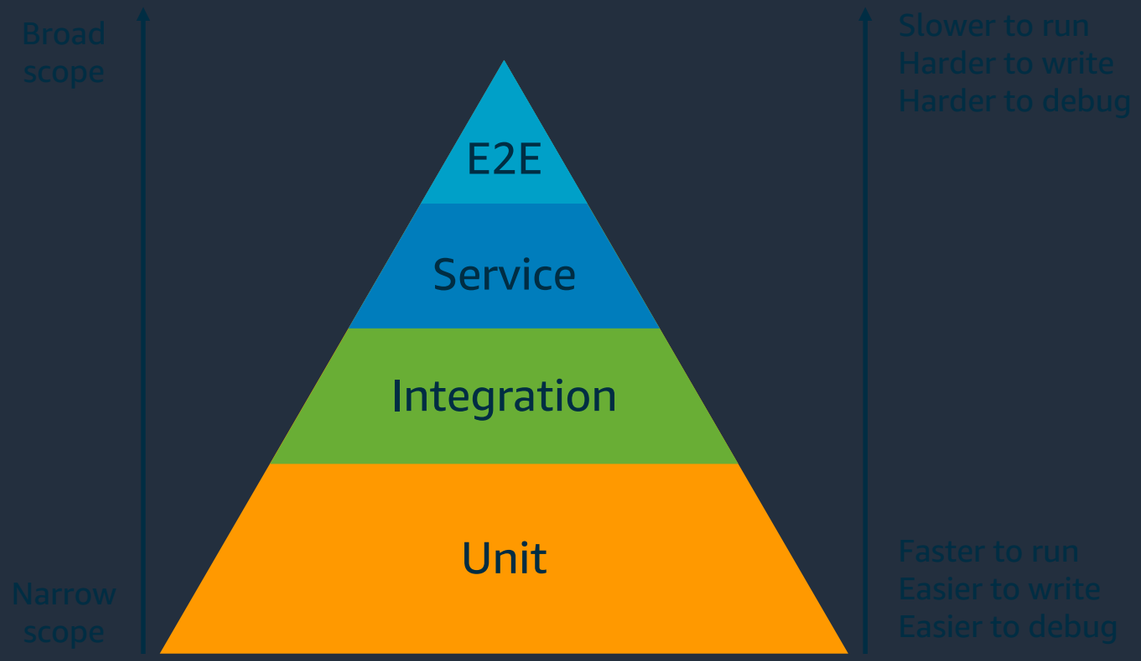
Keep broad-stack testing to a minimum

Use smaller, more focused tests wherever possible

We still need E2E tests to ensure everything works together in the cloud

Minimize:

- Number of broad-stack tests
- What they check
- How often you're waiting on their feedback

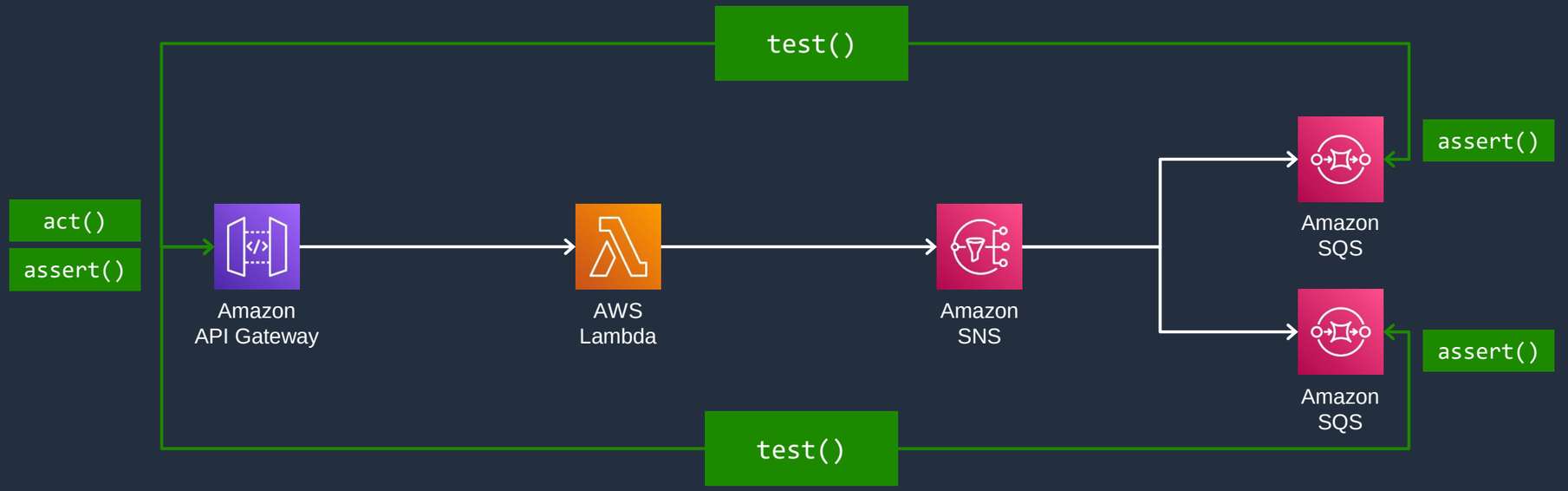


aws modern apps

Keep broad-stack testing to a minimum

1 x E2E test per user journey or business process

1 x broad-stack test per event pathway or data flow



Balance adding new tests vs extending existing ones

Only run after all other tests pass

aws modern apps

Managing Cloud Environments for Developers

aws modern apps

Anti-patterns

Only test Serverless applications against local cloud emulations

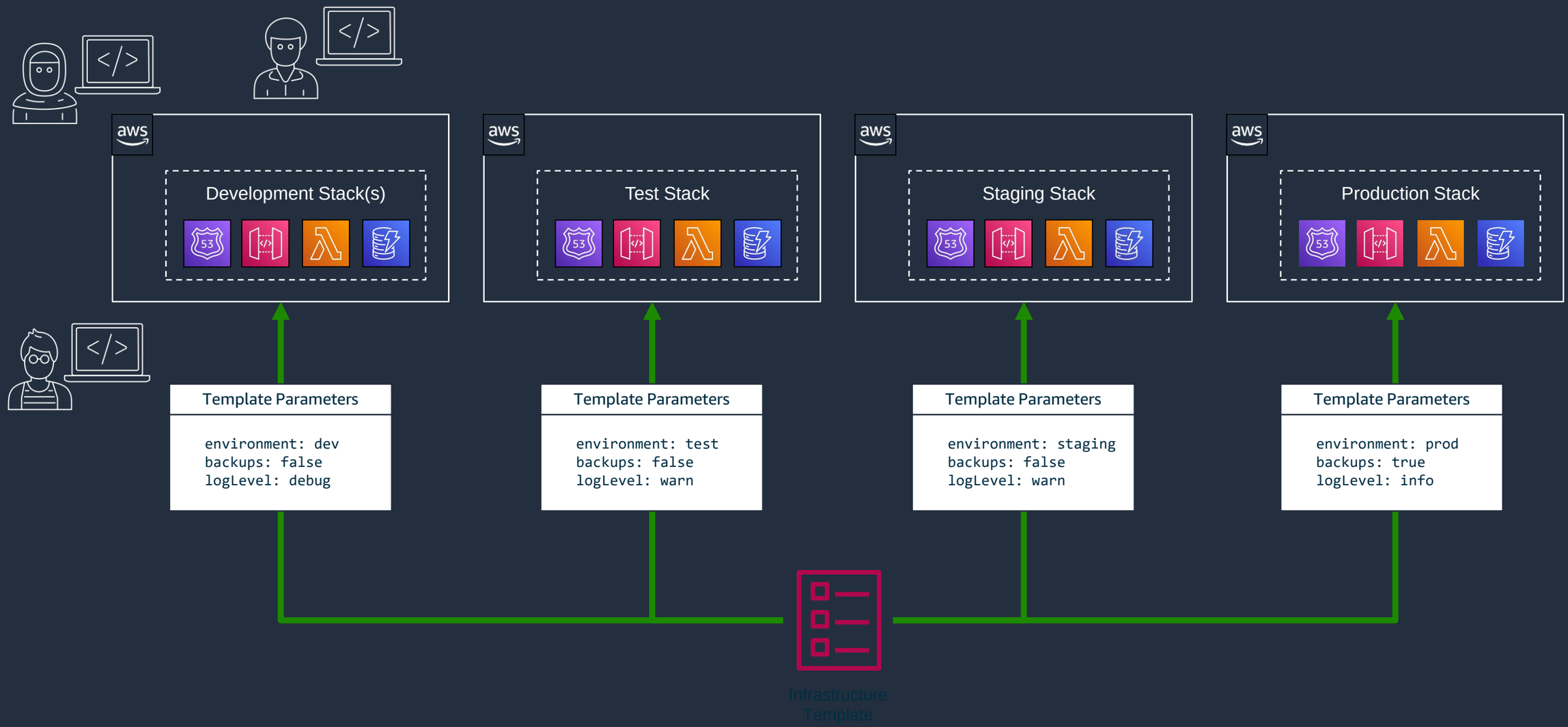
Restricting AWS account access to a few employees

Sharing a single environment across all developers

Sharing an AWS account across Development & Production

aws modern apps

AWS account per environment

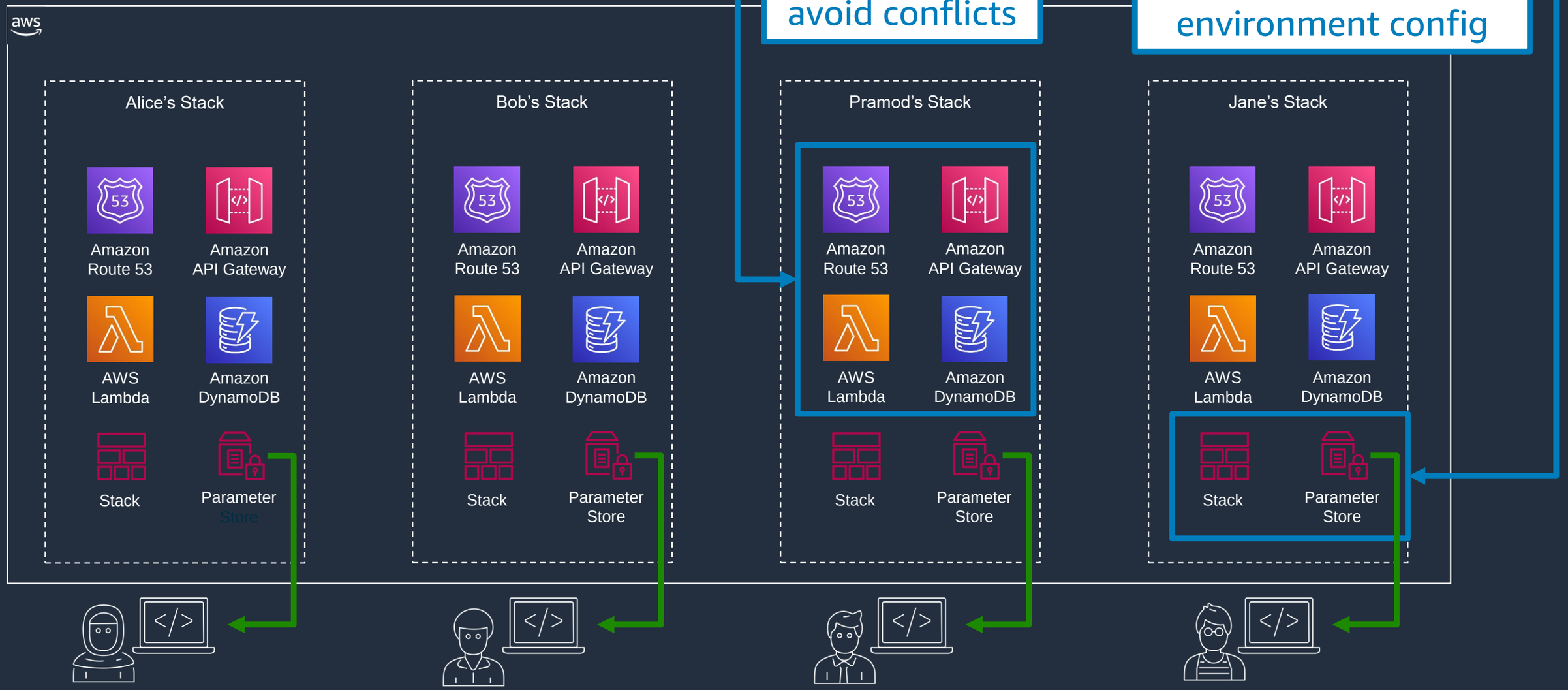


aws modern apps

Environment per developer

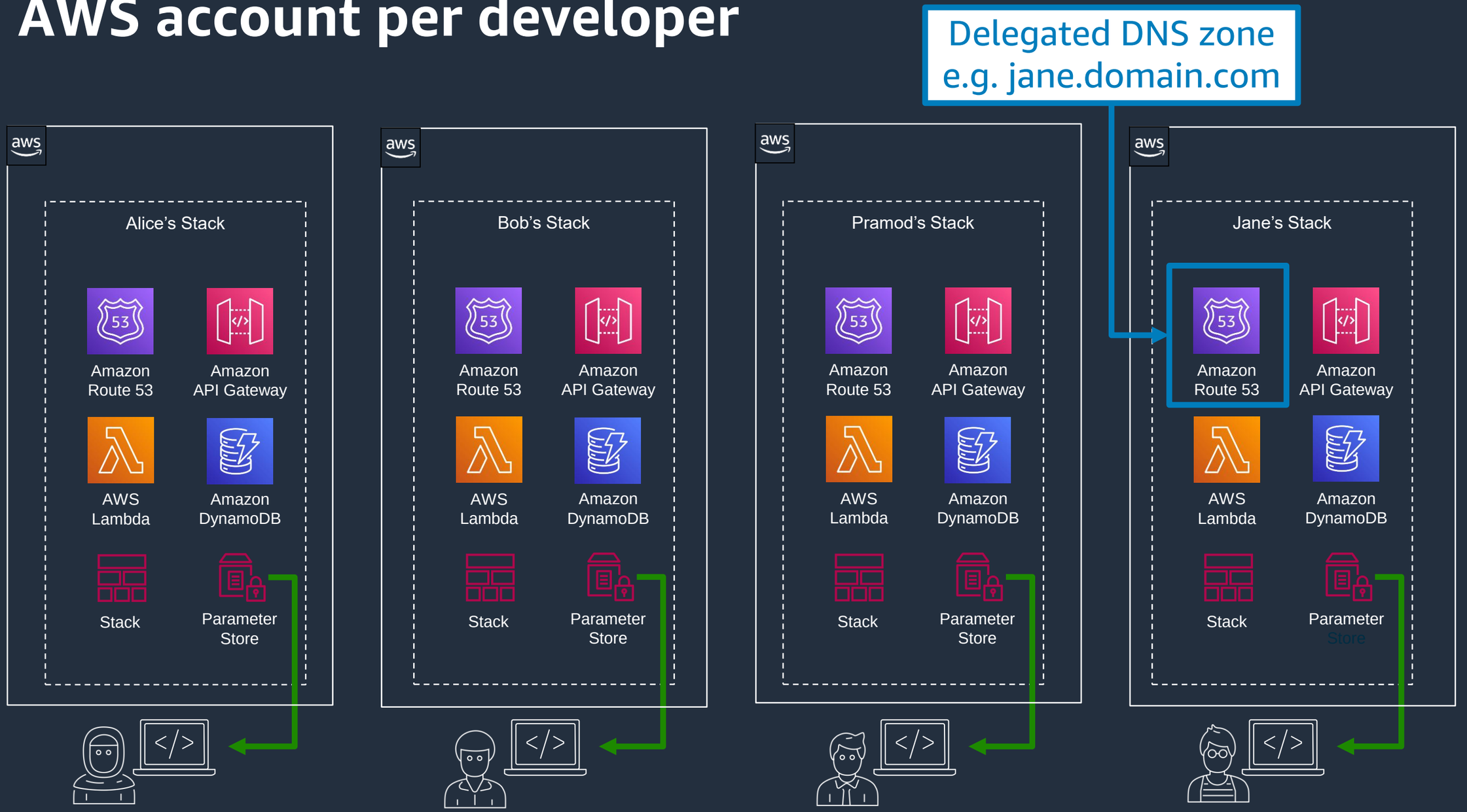
Prefixed or auto-generated resource names avoid conflicts

Tests retrieve URLs & resource ARNs from environment config



aws modern apps

AWS account per developer



aws modern apps

Key takeaways

Use infrastructure-as-code for parity across environments

1 x cloud environment per developer

Automate cloud environment creation and synchronization

Prefer short-lived stacks to avoid configuration drift

Lean on AWS services for account provisioning, access management, auditing, and guardrails