Serverless Patterns
(The awkward early years)

@mikebroberts
https://symphonia.io/
So we’re on the same page...
Serverless Computing is about building systems using Serverless Services
Serverless Services = FaaaS + BaaS
FaaS
Functions as a Service
BaaS
Backend as a Service
“What is Serverless?”

https://symphonia.io/
What are Patterns?

@mikebroberts
"Patterns ... identify common solutions to recurring problems."

Patterns are a proven way to capture experts' knowledge in fields where there are no simple ‘one size fits all’ answers.

Hohpe & Woolf : Enterprise Integration Patterns
In other words, patterns are not best practices!
Key parts of a pattern

1. A name! (Groups of patterns form a *vocabulary*)
2. Implementation
3. What problem it solves: *when*, and *when NOT* to use it
“The awkward early years”
Serverless is new enough that we’re still learning new ways of doing things.
Serverless is new enough that we’re still forming opinions of what solutions are common
But we DO know enough to get started!
Serverless Pattern
Categories

@mikebroberts
Serverless impacts many activities

Solution
Choice

Feature Patterns
Serverless impacts many activities

Solution Choice

Design

Architecture Patterns

Service and Function Patterns
Serverless impacts many activities

Development Patterns

Solution Choice
Design
Develop
Serverless impacts many activities

- Solution Choice
- Design
- Develop
- Operate

Deployment Patterns
Monitoring Patterns
Feature Patterns

@mikebroberts
Image Resizer
How do I have a highly scalable capability of resizing images without having to constantly run servers?
Image Resizer
Image Resizer

When not to use?

- Need a lower latency solution
- Don’t want to store every combination of possible sizes - wan’t to provide just-in-time capability instead
- If you have significant throughput and costs would be an issue (but take heed, Lambda is often surprisingly inexpensive!)
Architecture Patterns

@mikebroberts
Serverless API
Serverless API

How do I build an HTTPS API without having to consider resource management or scaling concerns?
Serverless API
Serverless API

When not to use?

- Where latency concerns are not compatible with present-day capabilities
- Where throughput makes total costs too high in comparison with traditional techniques
Serverless Data Pipeline
Serverless Data Pipeline

How do I build a data pipeline that has great scaling capabilities, and also allows new components to be added without significant infrastructure knowledge?
Serverless Data Pipeline

When not to use?

- Use a hybrid approach (Serverless / traditional) where certain transform stages are not suitable for FaaS.
- If you have staggeringly high throughput, to the point that Serverless techniques would be far too expensive, then maybe consider a fully traditional approach.
Service & Function Patterns

@mikebroberts
Idempotent Function
Idempotent Function

How do I guarantee that there is at most one side-effect for each event that triggers a FaaS function?
Idempotent Function
Idempotent Function

When not to use?

- If your side-effect can be implemented in an idempotent way
- If duplicated side-effects are infrequent enough, or unimportant enough, that the fact that they happen can be lived with
- If your system’s throughput is such that DynamoDB costs would be prohibitively expensive
Development Patterns

@mikebroberts
FaaS-Free
Unit Test
FaaS-Free Unit Test

How do I write unit tests for my FaaS Function that don’t suffer from the startup time of local or remote FaaS runtimes?
FaaS-Free Unit Test
**FaaS-Free Unit Test**

When not to use?

- If you are exploring the event that is sent (e.g. TDD FaaS development, in some cases)
- If your unit test is more like a functional test, requiring access to cloud resources
Deployment Patterns
Serverless Application
Serverless Application

How do I group number of different functions and configured services within one coherent, deployable, unit
Serverless Application
Serverless Application

When not to use?

- If you’re performing a small integration between 2 existing components (“Serverless Spackle / Polyfilla”) then don’t force arbitrary grouping of Lambda functions / components. Sometimes one component is enough!
Monitoring Patterns

@mikebroberts
Log Sourced Metric
Log Sourced Metric

How do we capture application metrics when a metric collection service may not be able to handle the scaling behavior of functions deployed to a FaaS platform?
Log Sourced Metric
Log Sourced Metric

When not to use?

- If you’re using a vendor metric system that *does* scale (e.g. through sampling)
- If you don’t want to capture application metrics from your Lambda function
Sourcing your own patterns

@mikebroberts
What solutions do you see repeated throughout your organization?
Use the categories to narrow down your search
Remember to think about when, and when NOT, to use a solution
Summing up
We’re still figuring out common patterns of Serverless
But we do have some ideas now
Useful to break Serverless into different activity categories in order to source patterns.
And I showed you a few to get started!
Thanks!
Slides and links at

https://go.symphonia.io/sacon-nyc-patterns

@mikebroberts

https://symphonia.io/
mike@symphonia.io