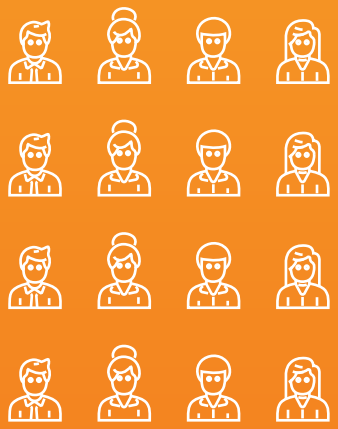
# Modern App Development

## Serverless First

Adrian Cockcroft
@adrianco
AWS VP Cloud Architecture strategy

# AWS Re:Invent
# Non-profit Hackathon for Social Good

Several teams
of ~four people



+

Non-profit organization
with a problem

Best Friends SAVE THEM ALL

SKYTRUTH

URBAN INSTITUTE

V!brant

=

Monday, 10 AM

“ Here’s a set of problems. Pick one and build a solution by 8 PM this evening…

# AWS Re:Invent
# Non-profit Hackathon for Social Good

Several teams of four people

Non-profit organization with a problem

Monday, 10 AM

Best Friends SAVE THEM ALL

SKYTRUTH

URBAN INSTITUTE

V!brant

## Where would you start?

> Here's a set of problems. Pick one and build a solution by 8 PM this evening…

# 2016 AWS Re:Invent Hackathon

## Adrian is a judge

**1** | **Every team** used AWS Lambda

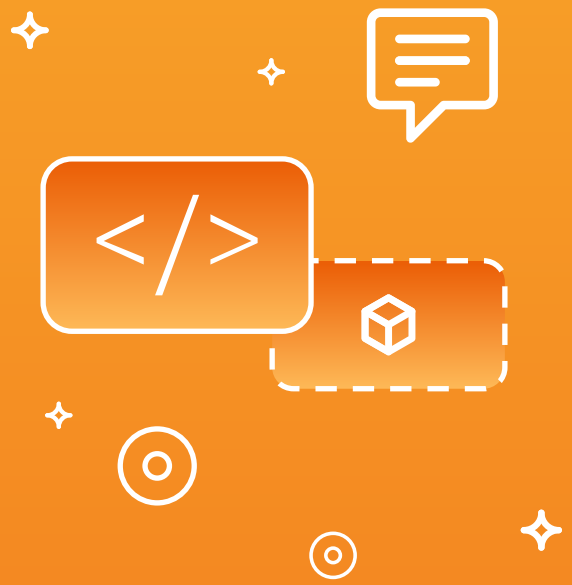
**2** | Less than **a day** total

**3** | Many team members **first time** using Lambda

**4** | Extremely **functional and scalable** prototypes

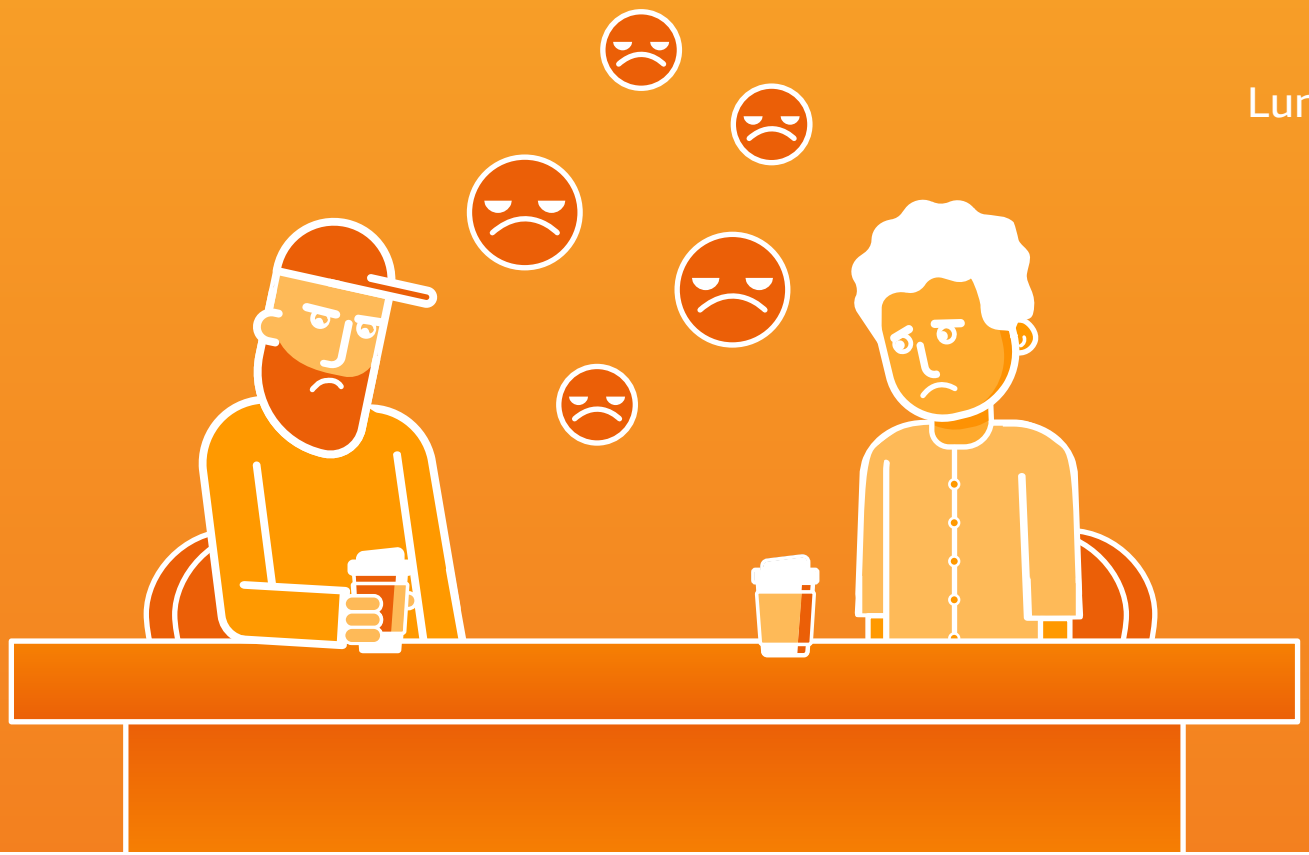
**“The plural of anecdote is not data”**

But I collect anecdotes…



# Conventional Development

# Conventional Development



2 months left

Friday

Lunchbreak

Progress

# Conventional Development

# Serverless Development

2 months left

Saturday

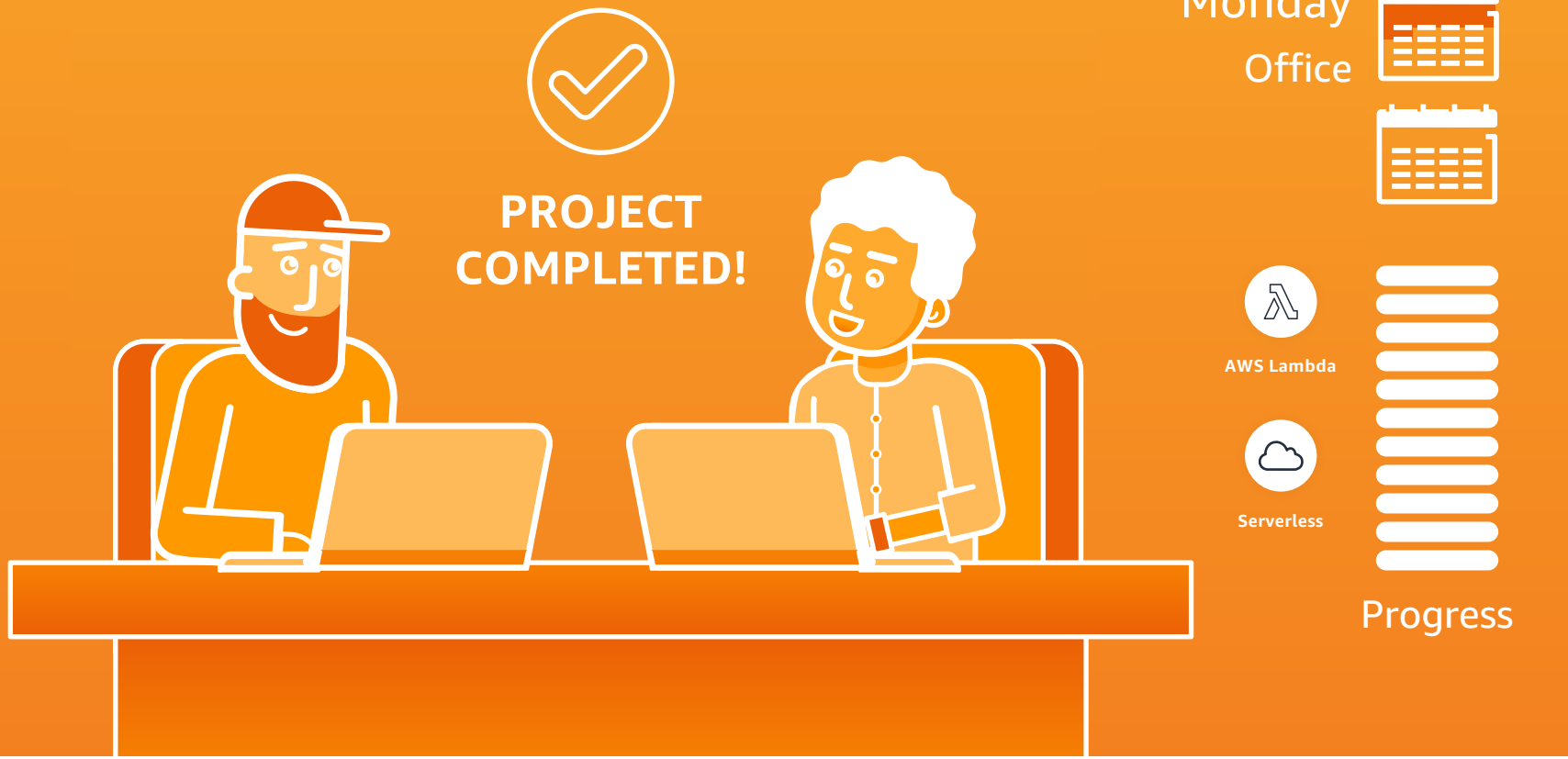
Home

AWS Lambda

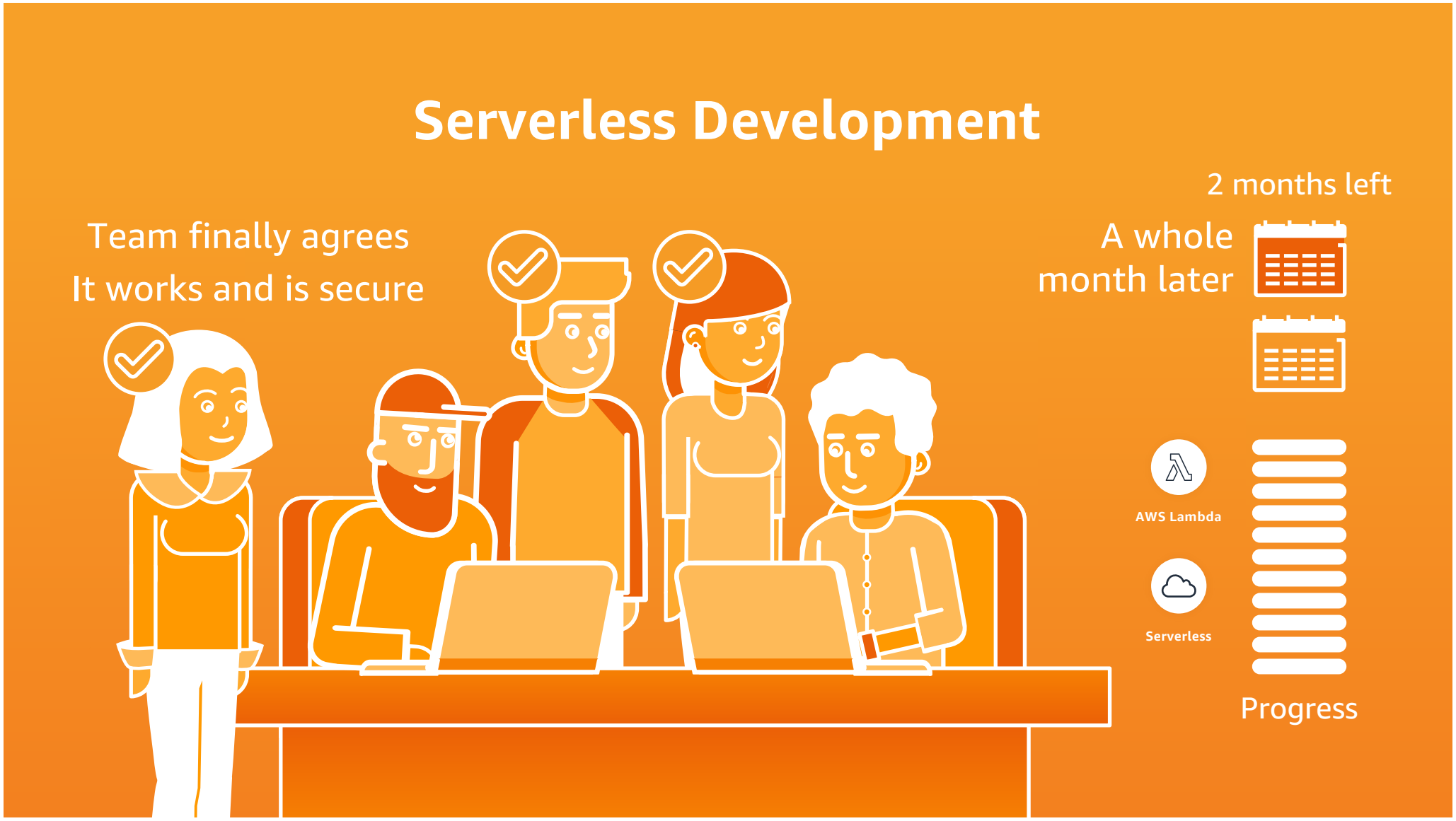
Serverless

Progress

# Serverless Development



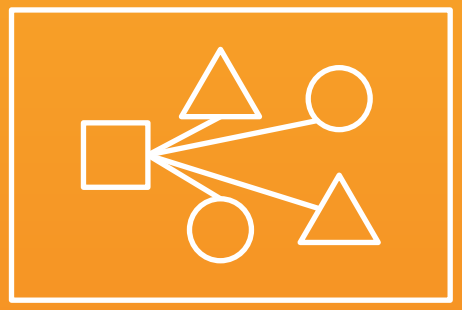
# Serverless Development



# Serverless Development

**Shipped application 1 month early**

# Serverless sizing discussion



We have this small but critical app running on an IBM Mainframe.

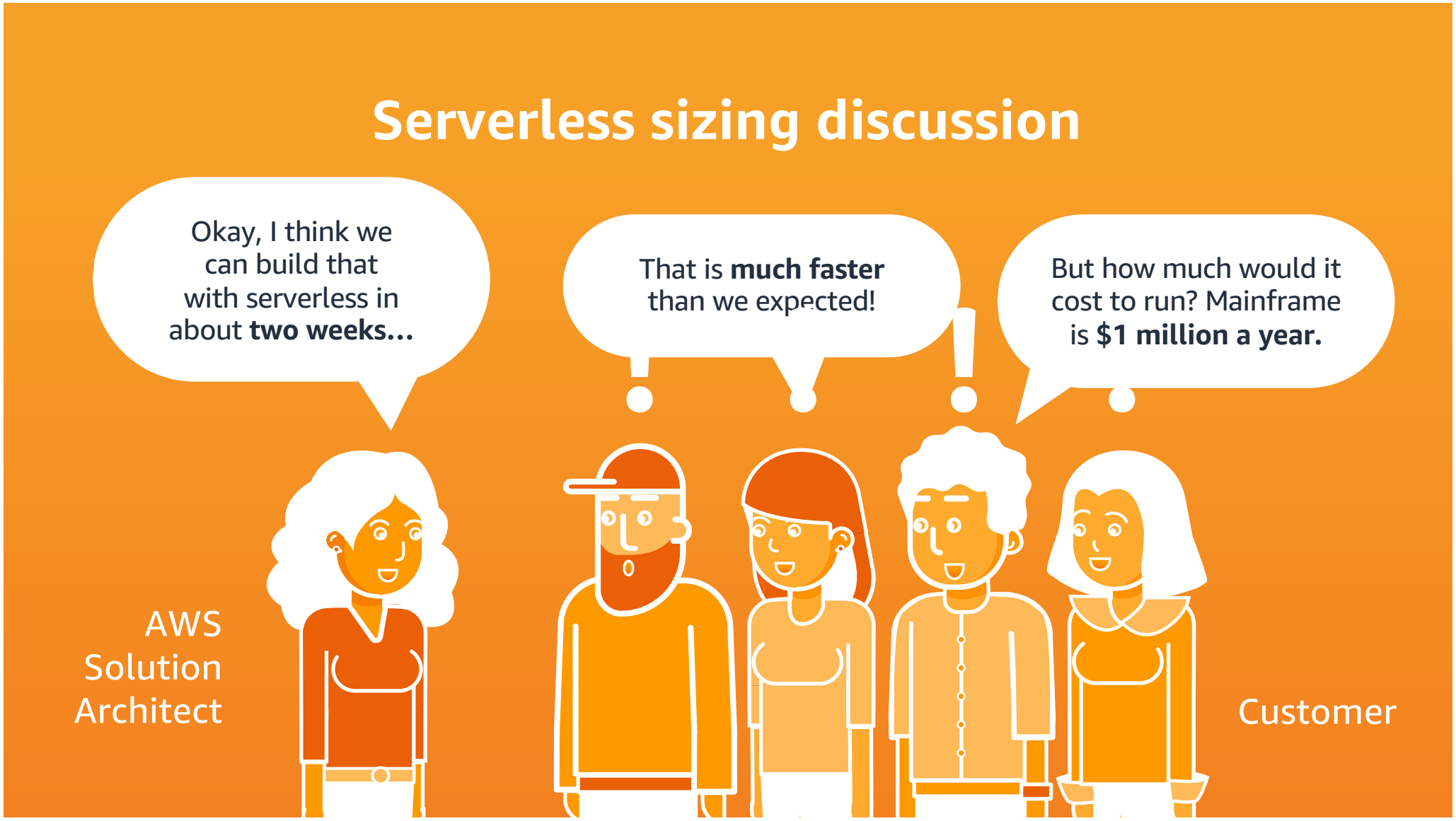
We would like to re-write it.

It looks like this.

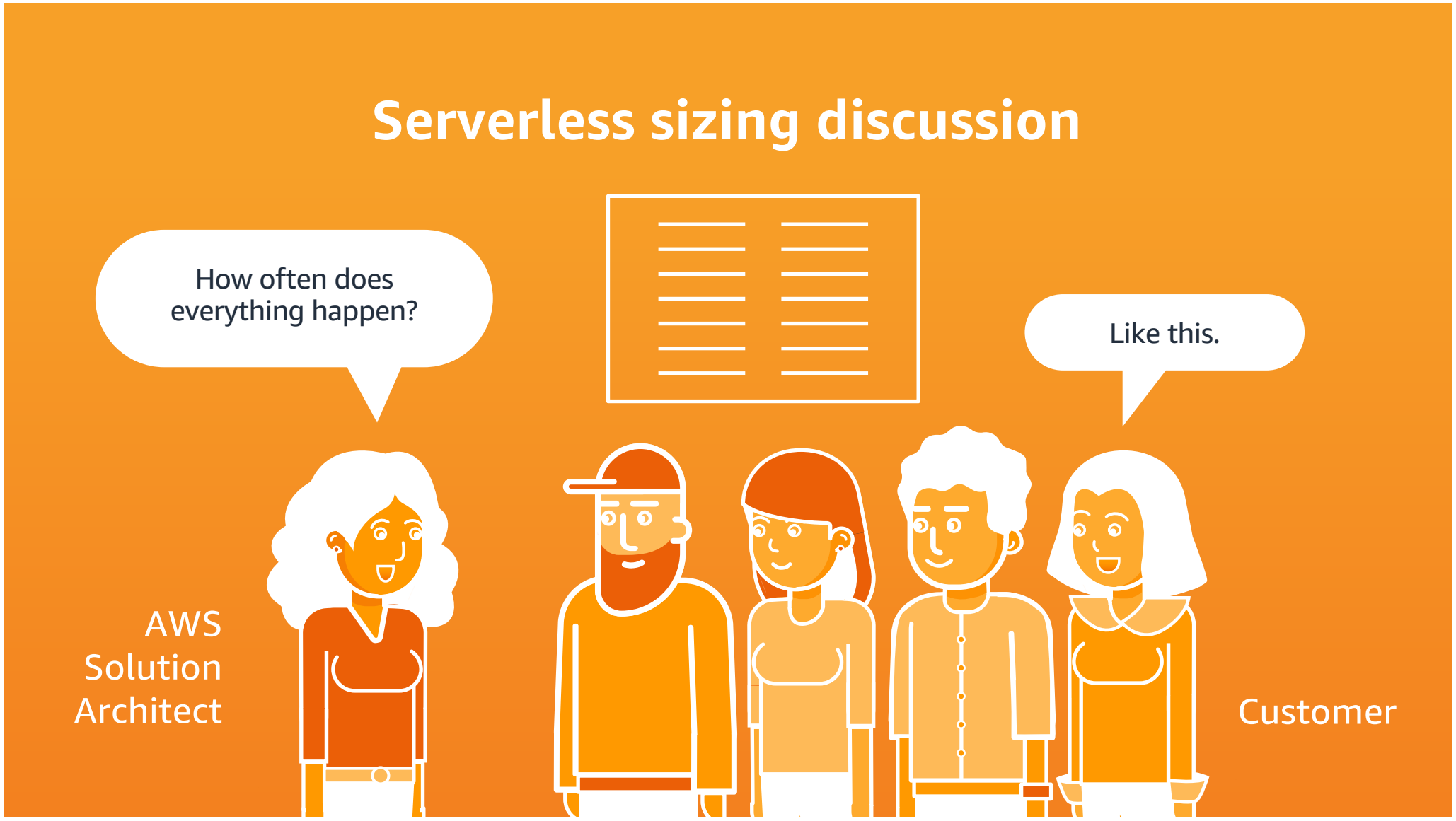
AWS Solution Architect

Customer

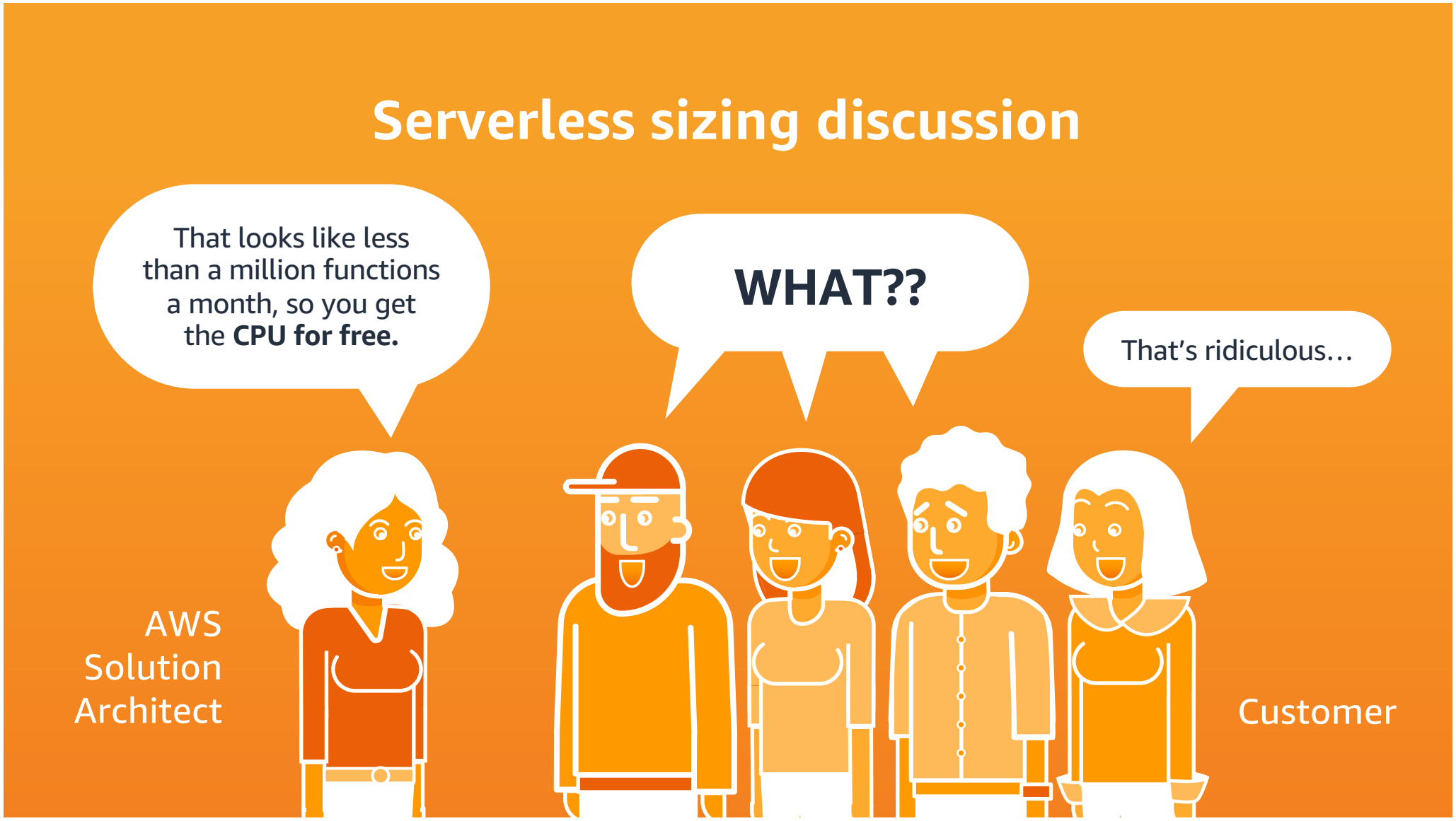
# Serverless sizing discussion



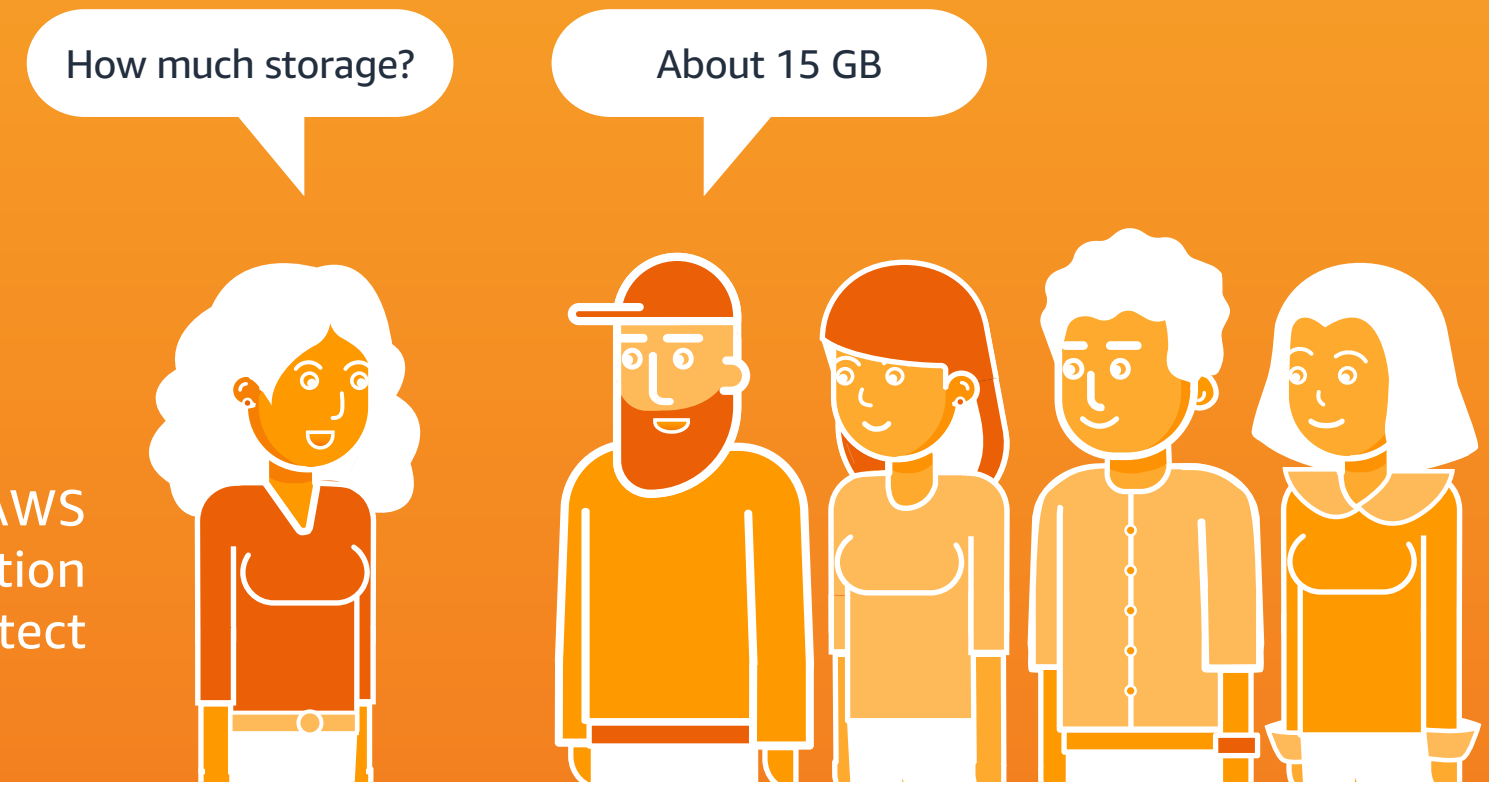
# Serverless sizing discussion



# Serverless sizing discussion



# Serverless sizing discussion



# Serverless sizing discussion

# Small but critical
# IBM Mainframe App Sizing Exercise

Cost to run today

**$1,000,000** per year

## Serverless Lambda Functions

2 weeks of work to build and test

Less than 1 million executions per month

Runs **in the free tier,** a few $ for storage, API calls, etc.

https://medium.com/@gary.crook/ibm-mainframe-to-microservices-on-aws-lambda-done-in-60-seconds-6d07024c23d3

Lots of corporate IT apps are very low utilization spiky workloads. Complex business event handling logic and workflow.

Lots of corporate IT apps are very low utilization spiky workloads. Complex business event handling logic and workflow.

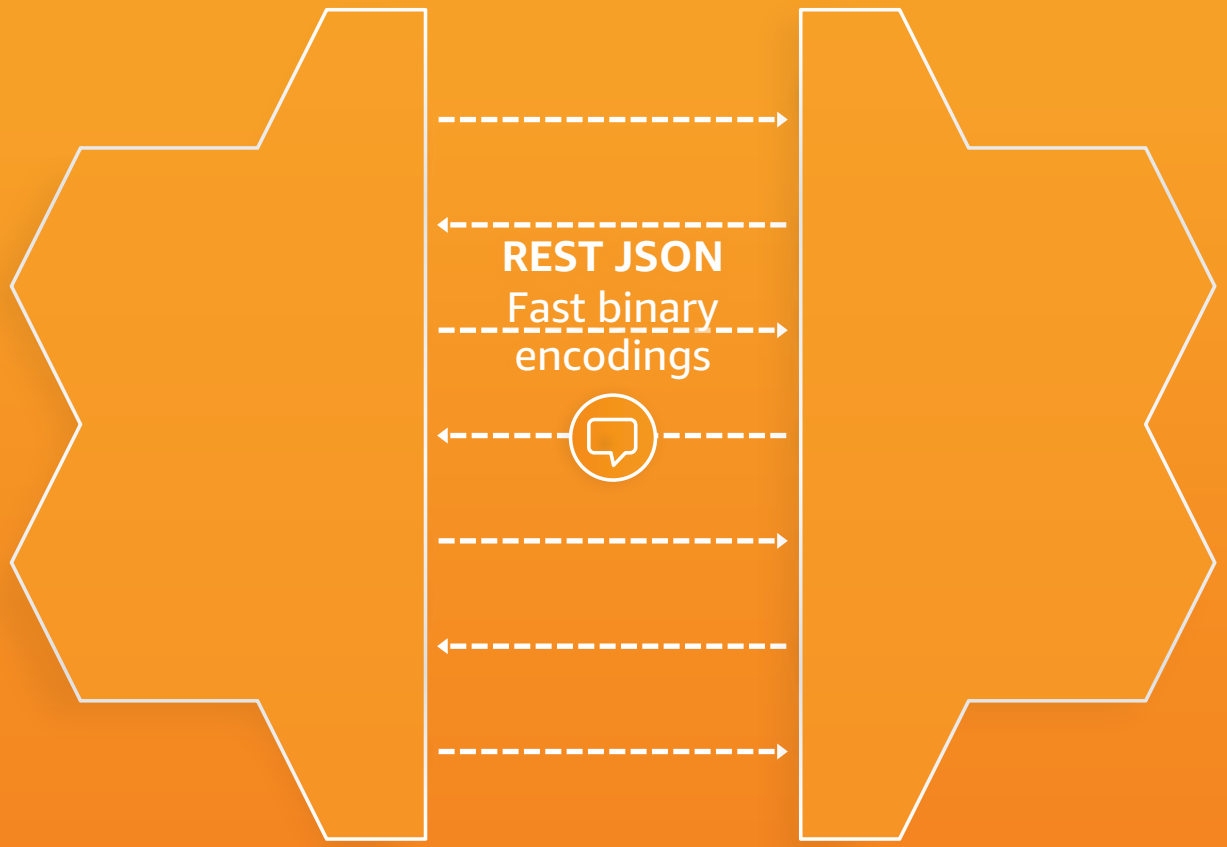
---

**Ideal to re-write as serverless!**

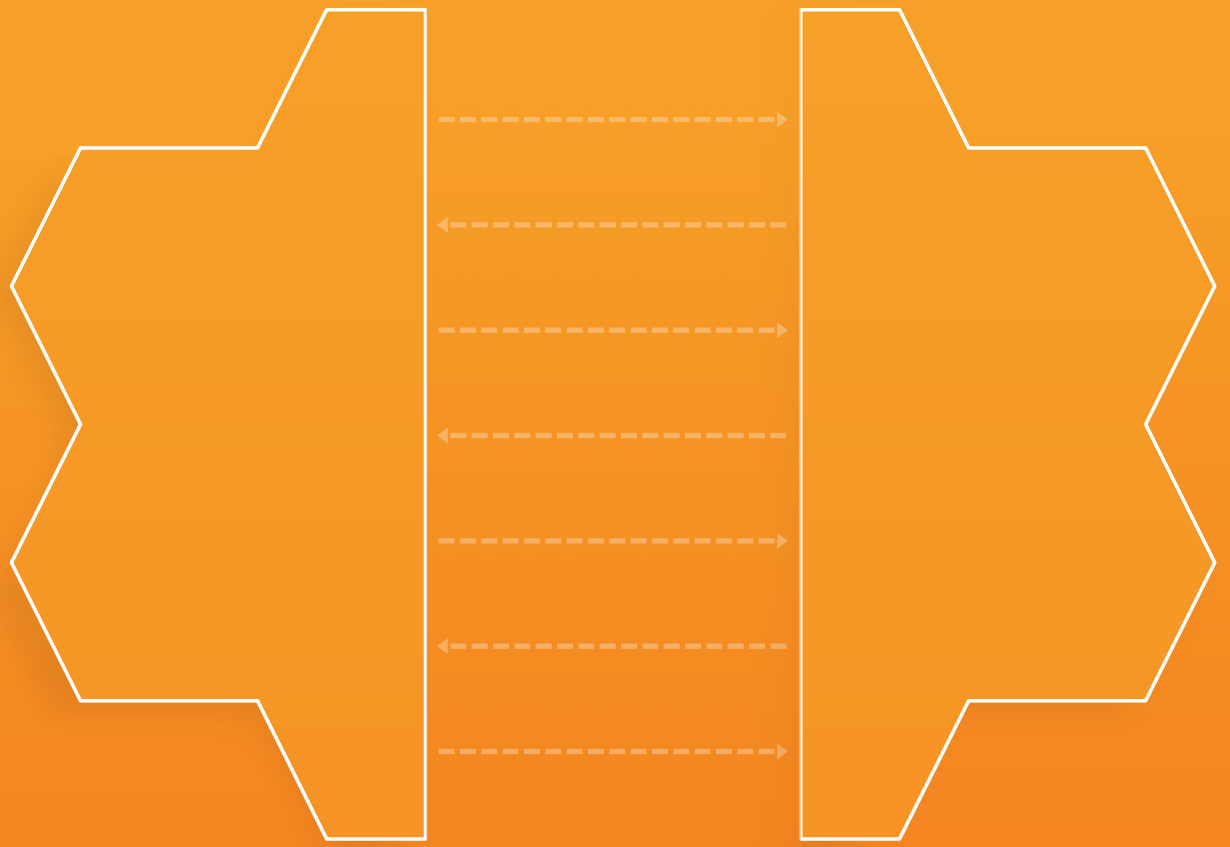
---

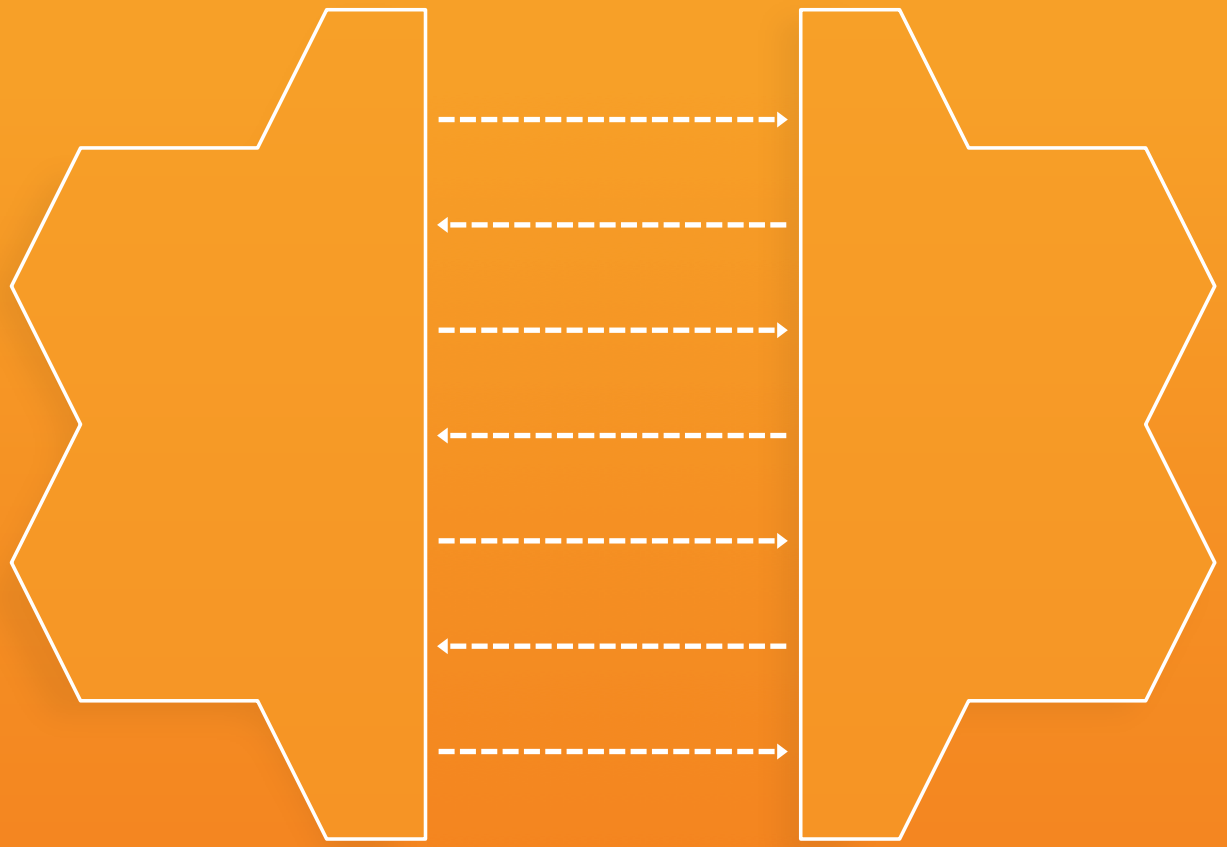
# What is different about serverless?

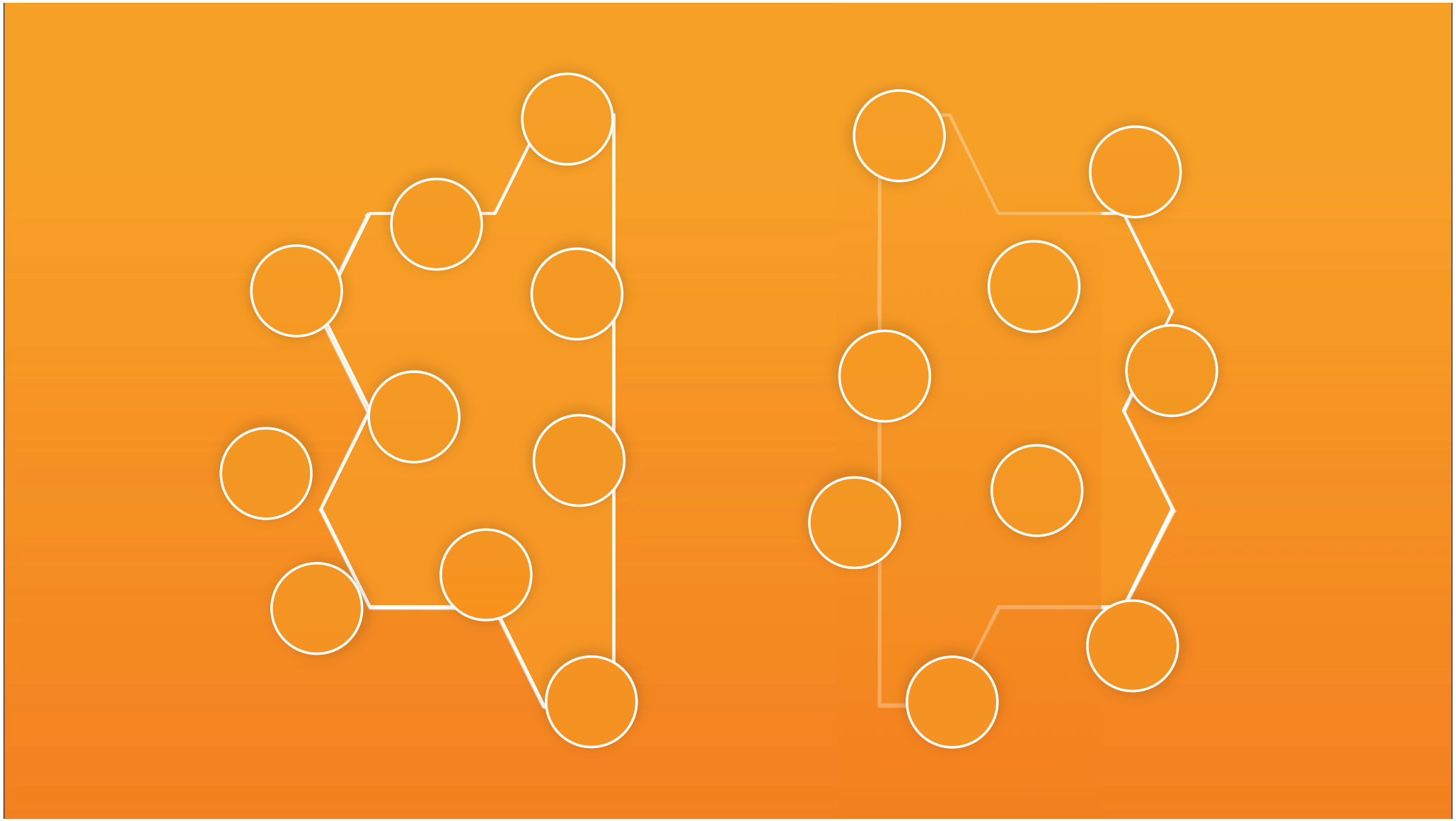
# Splitting Monoliths

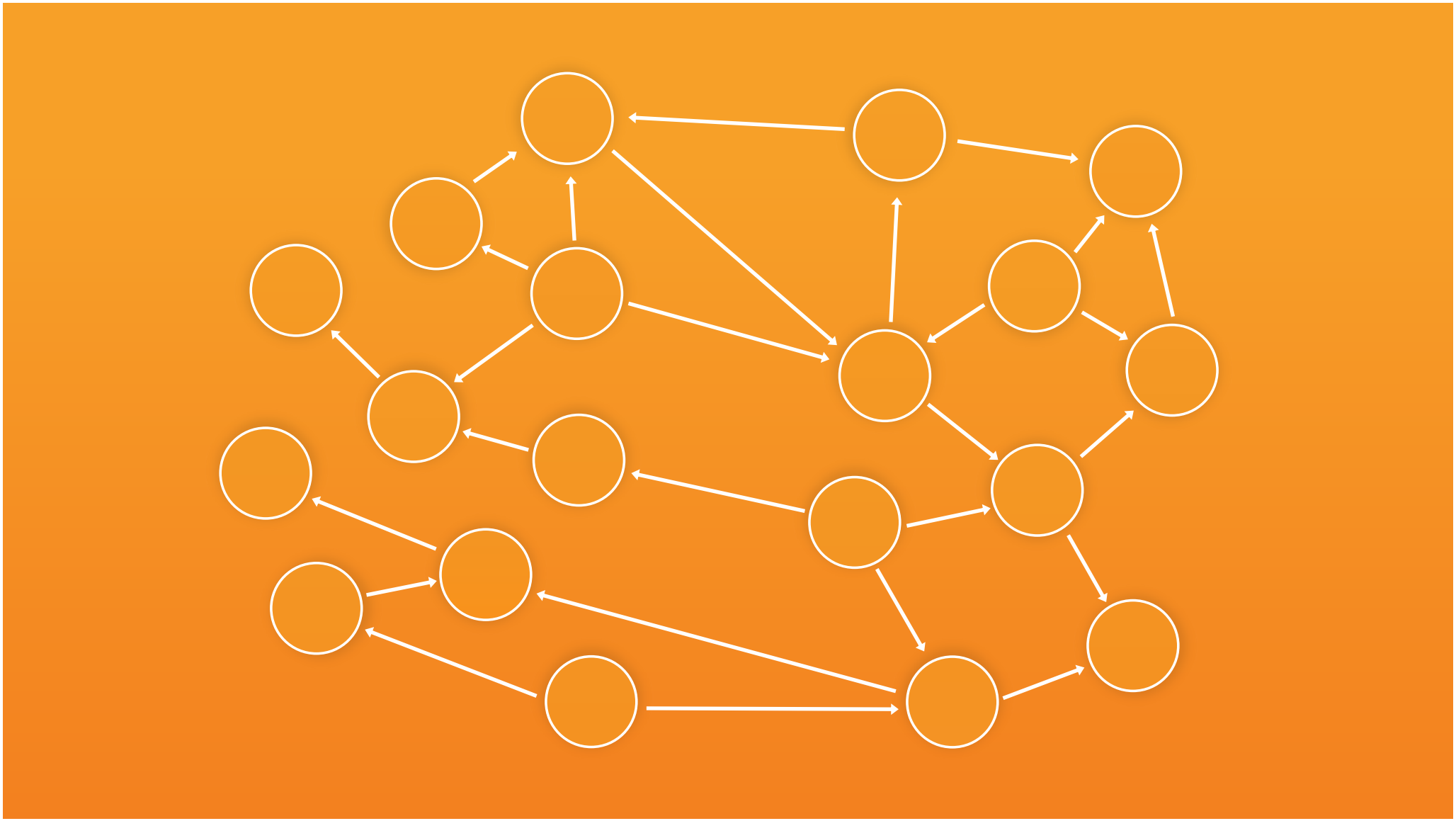


# Splitting Monoliths









# Microservices

# Microservices to Functions

Standard building brick services provide standardized platform capabilities

Amazon API Gateway

Amazon SQS

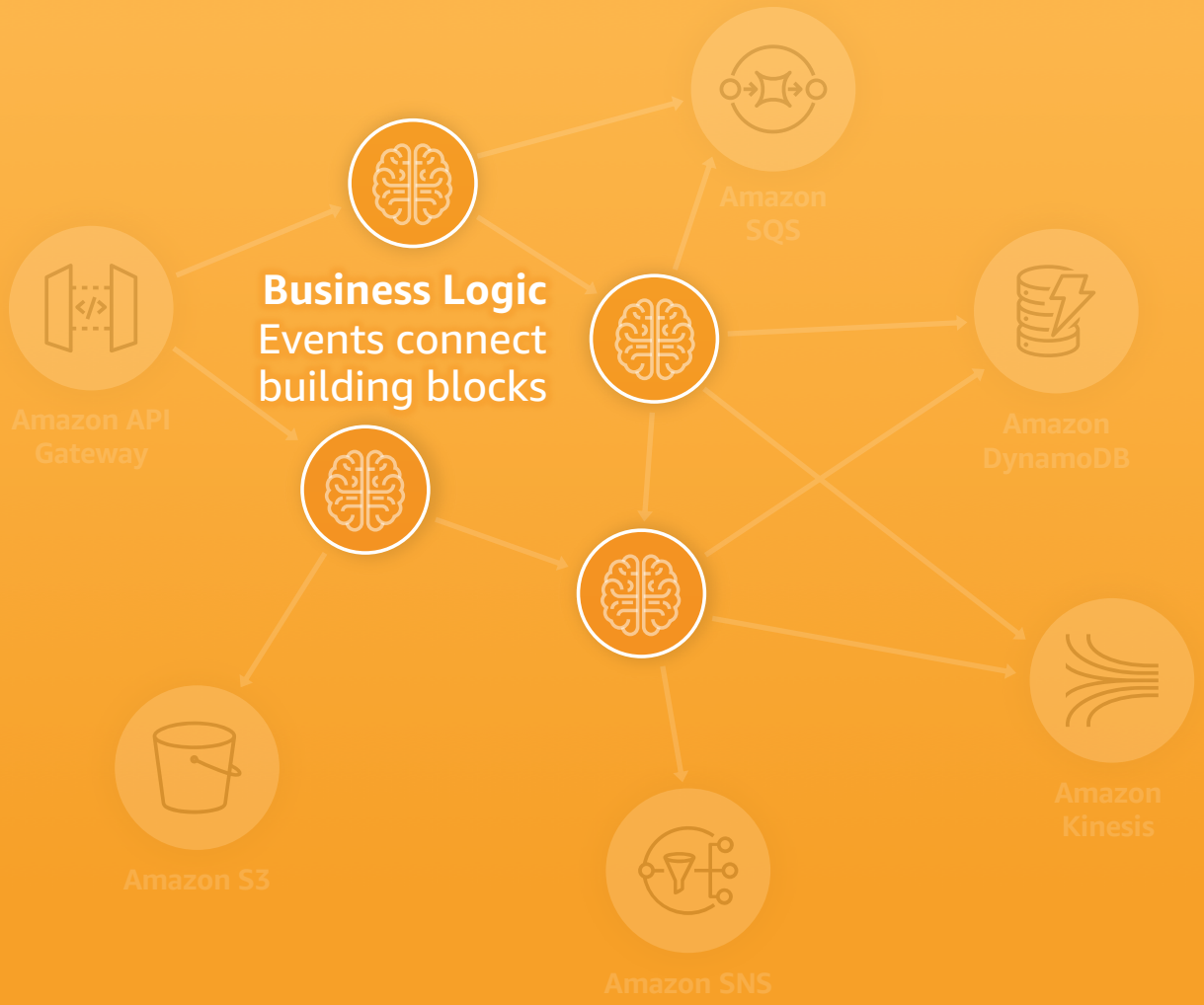
Amazon DynamoDB

Amazon S3

Amazon SNS

Amazon Kinesis

# Microservices to Functions



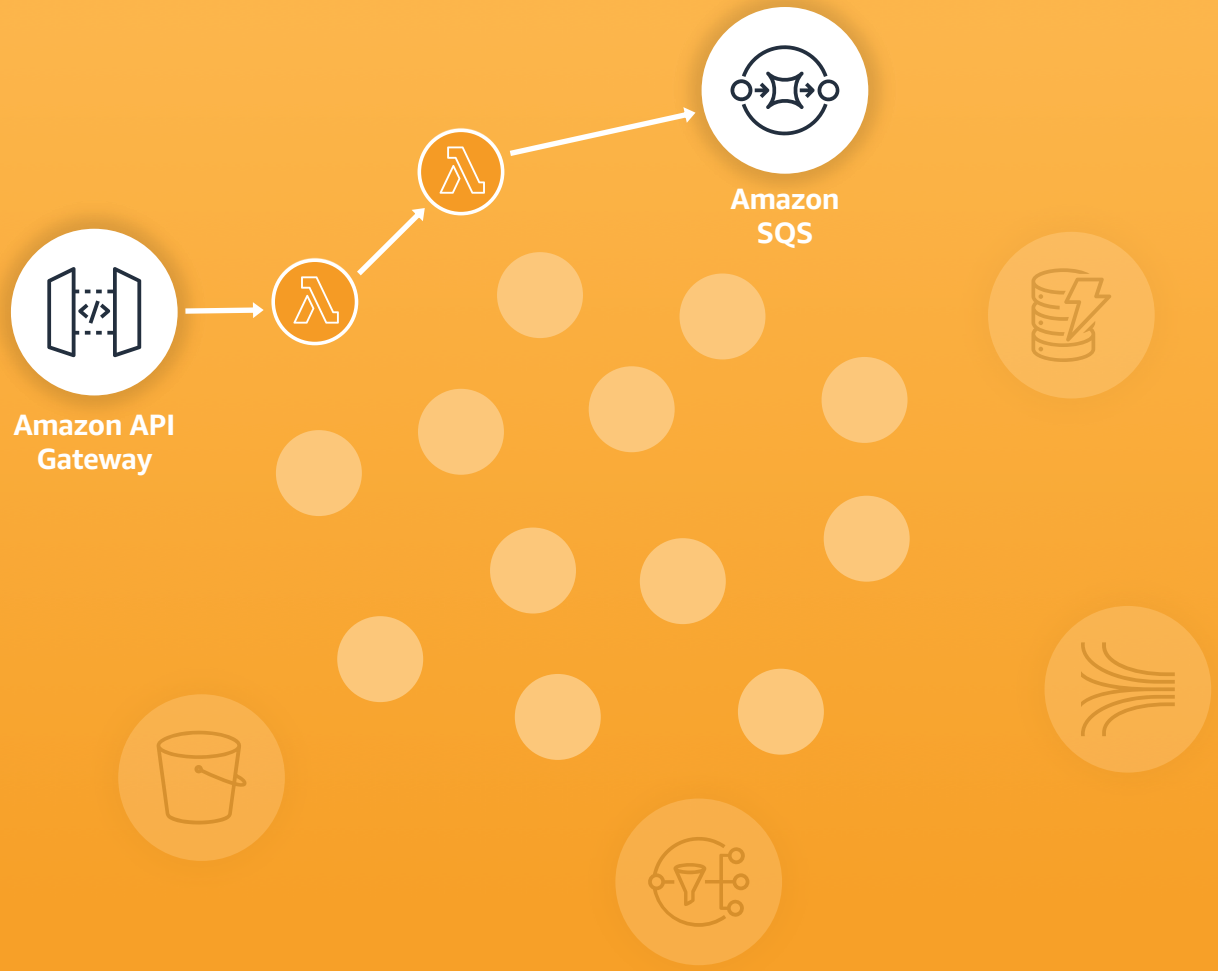
# Microservices to Functions

# Microservices to Functions

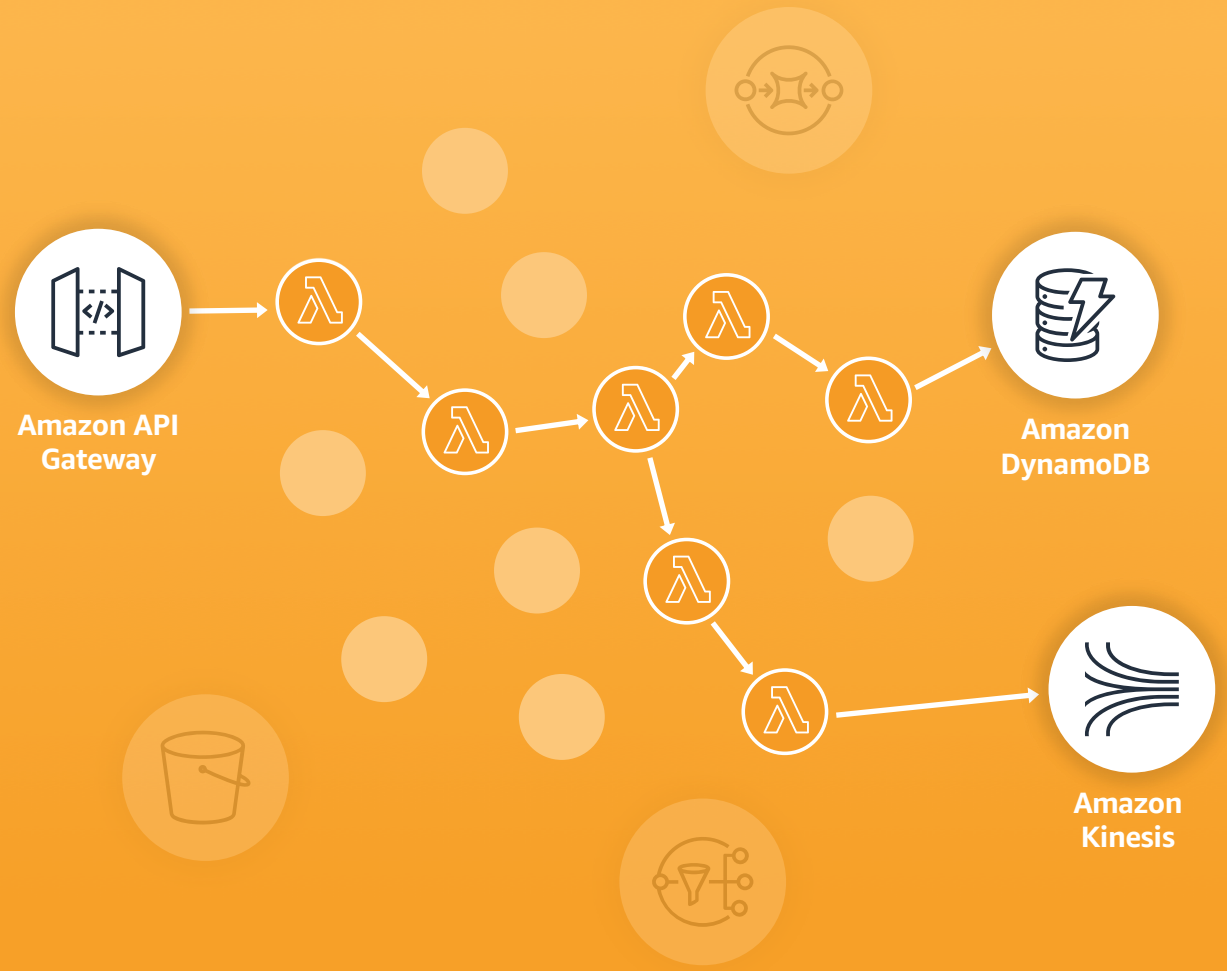
# Microservices to Ephemeral

# Microservices to Ephemeral Functions

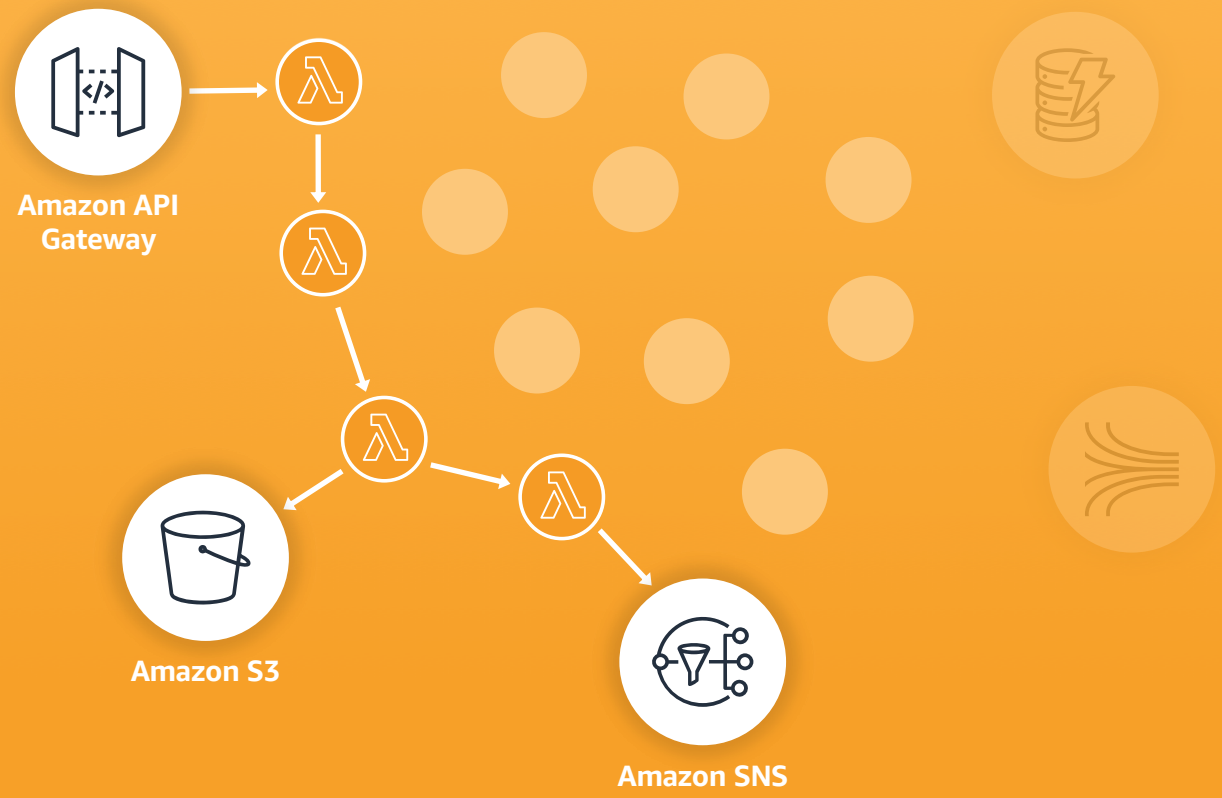
# Microservices to Ephemeral Functions



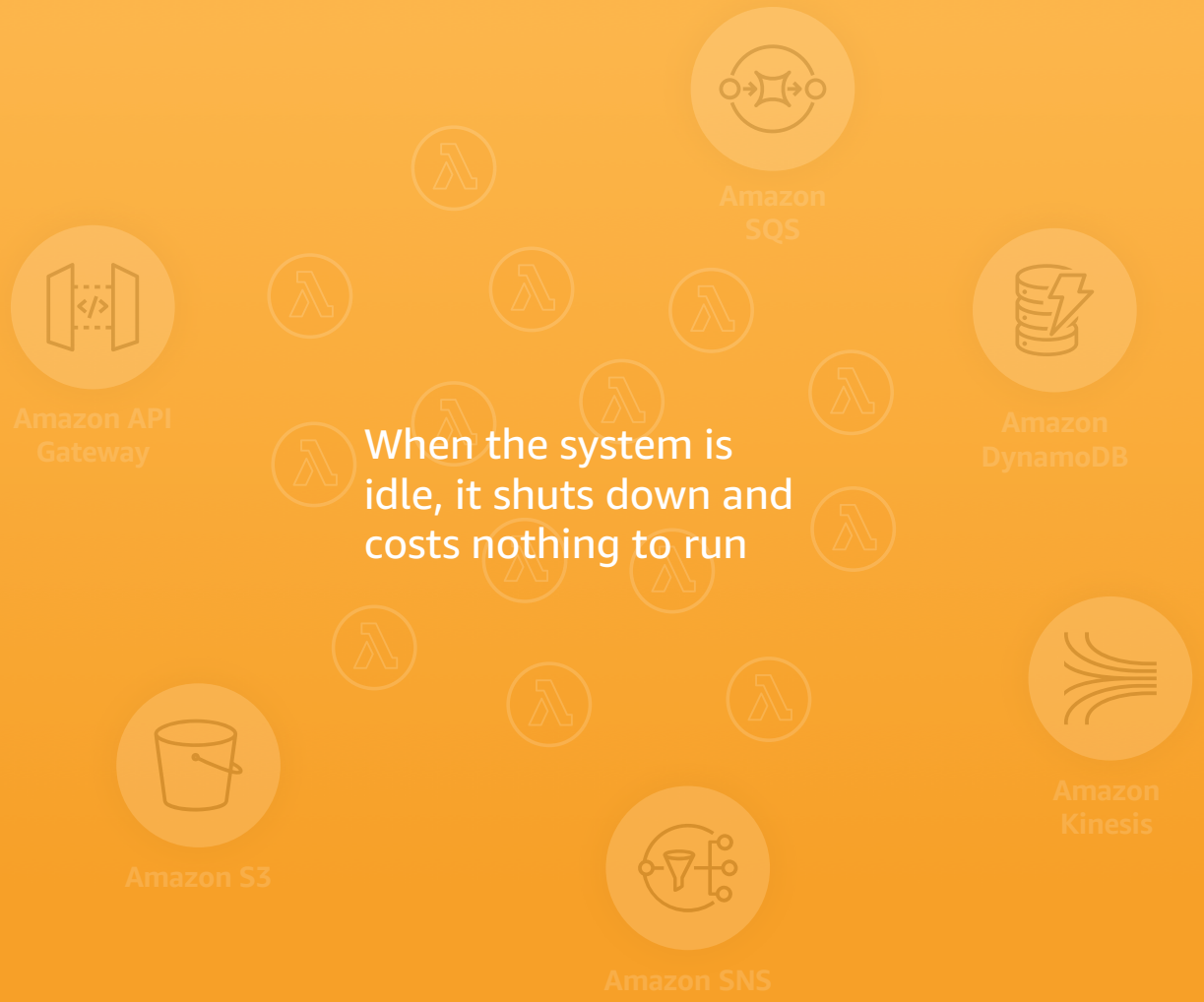
# Microservices to Ephemeral Functions



# Microservices to Ephemeral Functions



# Microservices to Ephemeral Functions

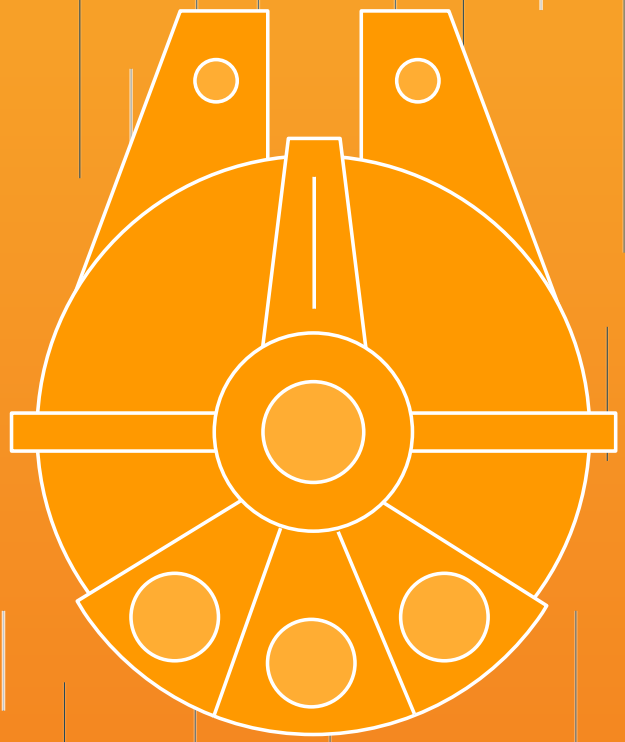



When the system is idle, it shuts down and costs nothing to run

**So WHY is it so fast to write a serverless app?**

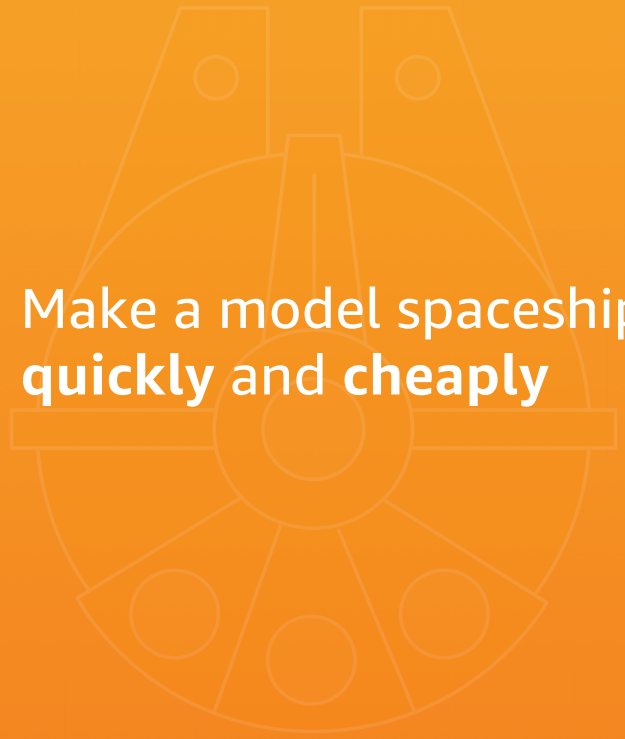
An analogy…

# What is the user need?



**What is the problem you are trying to solve?**

Make a model spaceship **quickly** and **cheaply**



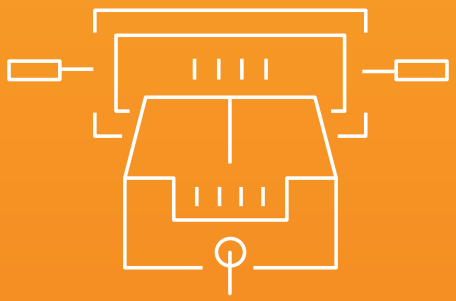
# Traditional Development

Design a prototype

# Traditional Development

Carve from
modelling clay

# Traditional Development



Make molds

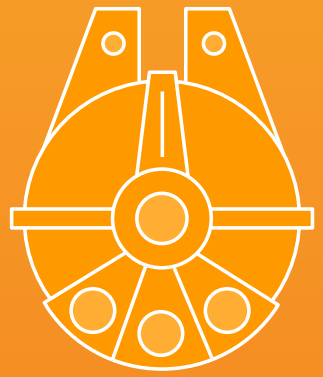
# Traditional Development

Produce injection molded parts

# Traditional Development

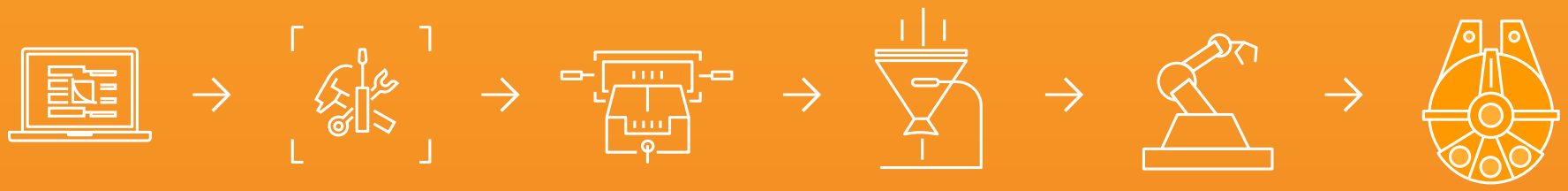
Assemble parts

# Traditional Development



Sell finished toy

# Traditional Development



Design a prototype

Carve from modelling clay

Make molds

Produce injection molded parts

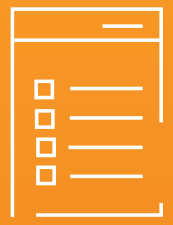
Assemble parts

Sell finished toy

# Rapid Development

Big bag of blocks

+



Instructions

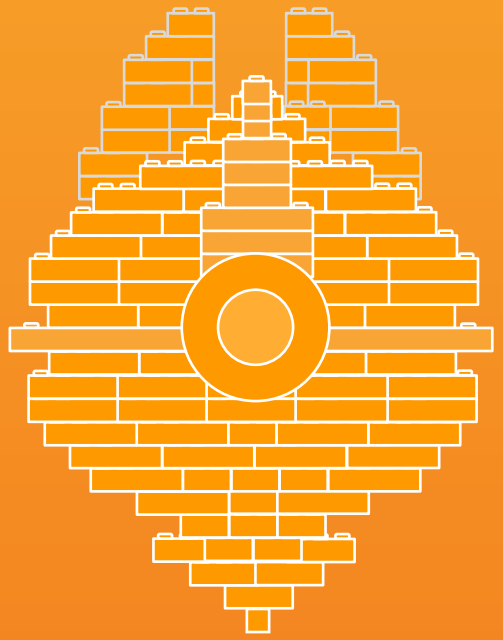
+

A few hours

# Rapid Development

A finished toy

# Rapid Development

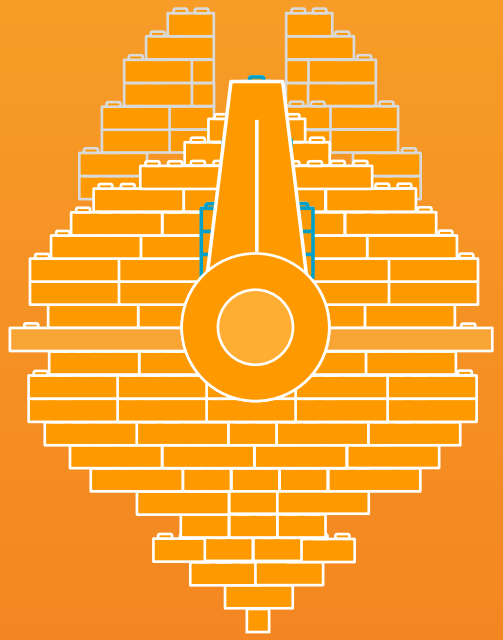


Lacks fine detail

Recognizable, but not exactly what was asked for

Easy to modify and extend

# Optimization



Take a group of Lego bricks…
…and form a new custom brick

A more specialized common component

| Traditional | Rapid Development |
| --- | --- |
| Full custom design | Building blocks assembly |
| Months of work | Hours of work |
| Custom components may be fragile and need to be debugged and integrated | Standard reliable components scale and are well understood and interoperable |
| Too many detailed choices | Need to adjust requirements to fit the patterns available |
| Long decision cycles | Constraints tend to reduce debate and speed up decisions |

| Containers | Serverless |
| --- | --- |
| Custom code and services | Serverless events and functions |
| Lots of choices of frameworks and API mechanisms | Standardized choices |
| Where needed, optimize serverless applications by also building services using containers to solve for anything serverless doesn't do well… yet. | Combine building blocks including:<br>λ AWS Lambda<br>API Gateway, EventBridge<br>Amazon SNS, SQS<br>Amazon DynamoDB<br>AWS Step Functions |

# So… why doesn’t everyone use serverless first?

## Objections and limitations

Note: See Re:Invent 2019 SVS343

Objection

# My favorite programming language isn’t supported by AWS Lambda

**AWS supported languages:**

Java, Go, PowerShell, Node.js,

C#, Python, and Ruby

C++ example BYOL Runtime API

**Third-party language support:**

JVM based—Clojure, Scala etc.

Rust, PHP, Erlang, Elixir,

COBOL, FORTRAN… etc.

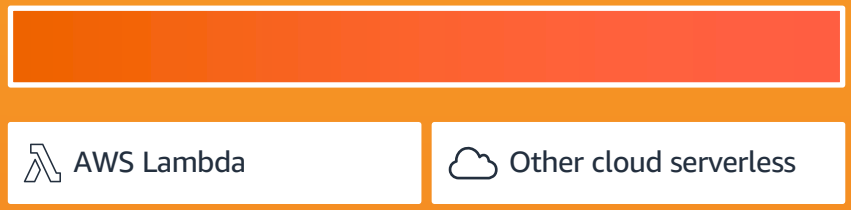
Objection

# Serverless apps aren't portable

**1** **week** on AWS Lambda

**2** **MONTHS** to build **portable** with containers

Time spent **arguing** about Kubernetes features




Time spent **building** serverless app **TWICE**

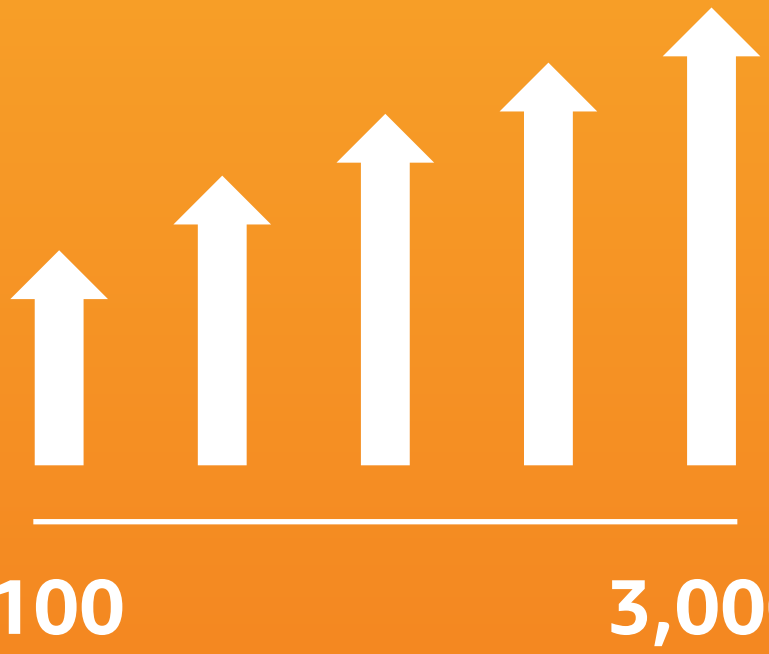
Objection

# Serverless can't scale

## Since launch

AWS Lambda **DEFAULT** concurrent execution limit increased from 100 to 3,000 CPUs

Virginia, Oregon, Dublin (500–1,000 elsewhere)




https://docs.aws.amazon.com/lambda/latest/dg/scaling.html

Objection

# Serverless can't scale

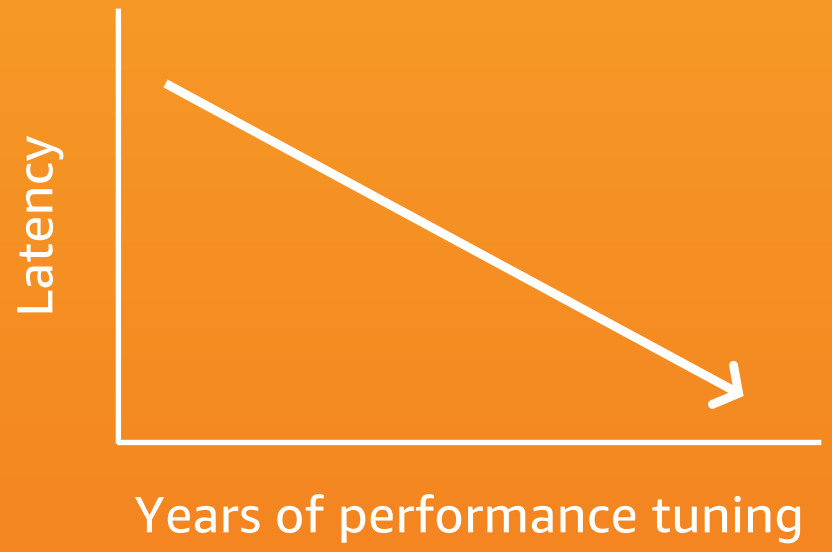
Request increase to AWS Lambda concurrent execution limit

**3,000** **10,000+**

**Limit** can be increased to well **over 10,000** CPU's all running your workload **at once!**

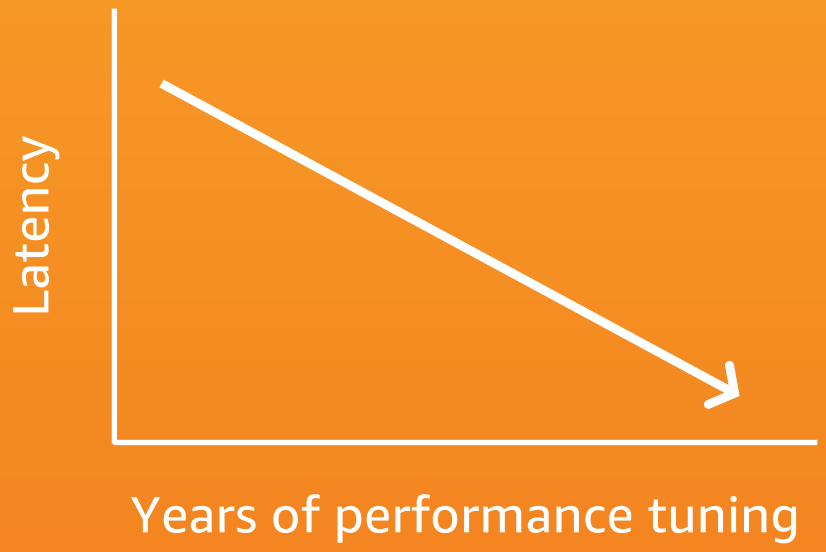
Objection

# Startup latency is too high



Objection

## Startup latency is too high




**NEW**

Provisioned concurrency lets you **pre-initialize functions**

Up to **300** in 1–2 minutes

Then **500** per minute

**10,000** in 15 minutes

Useful for “flash-sale” workloads

Objection

# Network setup takes too long

It used to be slow to attach a function to a VPC network

**September 2019**

**FIXED**

Now **secure access to private VPC** networks isn't an issue

Objection

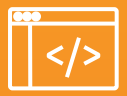
# Routing traffic to Lambda is expensive

## API gateway

Reduced cost/faster transforms, throttles

## ALB to Lambda path based routing

Simpler, high throughput




## AppSync GraphQL

Provides rich query language to support more work in fewer calls

Note: See Re:Invent NET413, MOB307, MOB402

Objection

# Serverless isn’t secure

Every function runs in **its own** virtual machine

**Minimal attack surface**

Initialized Lambda’s are only reused for **the same account**

**NO open ports**

**Initialised functions are re-cycled** after a few hours

**NO long running services to get infected**

Objection

# Lambda doesn’t handle state

AWS Step Functions provides a Lambda based **state machine**

Flows can run for **a year**

**Visual creation** of business logic

Objection

# Step Functions is a toy

Based on
**Amazon Simple Workflow**

**SWF Used** throughout
Amazon **for many years**

Very **reliable**
and **scalable**

**Since launch**

Step
rates

Cost/
step

**NEW**

AWS Step Functions Express Workflows supports over **100,000 events per second!**

Objection

# Need to run for longer

**2018**

AWS Lambda's max duration increased

**5 min** **15 min**

Objection

## Need to run for longer

**Since launch**

AWS Lambda's max duration

**5 min** **15 min**

Use **AWS Batch** or **Fargate** to create serverless containers for long running jobs, or to access specific instance types

Objection

# AWS Lambda can’t do sophisticated event processing

**NEW**

**1** | Amazon Kinesis Data Streams, SQS FIFO, SNS Dead Letter Queue

**2** | Lambda Destinations

**3** | Detailed controls for streamed events and async events

**4** | EventBridge schema registry and discovery. One to many fan-out

Note: See Re:Invent API304, API315, API320, SVS308, SVS317

Objection

# It’s hard to configure Lambda for complex production use cases

AWS Systems Manager Parameter Store

AWS Lambda Environment variables for feature flags

KMS for encryption and integration with secrets manager—no hard coded credentials

Objection

# It’s hard to configure Lambda for complex production use cases

**2018**

AWS Lambda Layers for common code

Stacks up in reusable modules

Objection

# My debug tools don’t know how to monitor Lambda

AWS X-Ray Integration trace maps

CloudWatch Service Lens

CloudWatch Synthetics canary test

CloudWatch Contributor Insights

**Third-party tools for serverless:**

Datadog, I0pipe, SignalFX,

New Relic, Thundra, Epsagon,

Dashbird, AppDynamics, etc.

Objection

# Lambda doesn't handle SQL databases well

Per function concurrency limits to prevent Lambda from overrunning SQL database interfaces

**NEW**

RDS Proxy for serverless connection pooling and failover, using AWS Secrets Manager to keep credentials out of application code

Objection

# Too hard to get started

1 | AWS Serverless Application Model—simple templating for common cases including IAM roles management

3 | AWS CDK support for serverless applications

**Third-party**

2 | Serverless Framework and Terraform support

**NEW**

4 | SAM CLI—lots of new features in 2019. Local support for testing with Docker Containers

https://aws.amazon.com/blogs/compute/improving-the-getting-started-experience-with-aws-lambda/

Demo

# Serverless application AircraftML

Jerry Hargrove
@awsgeek

# What's this?

# AircraftML

## a serverless bot

# It started with a **BIG** idea...

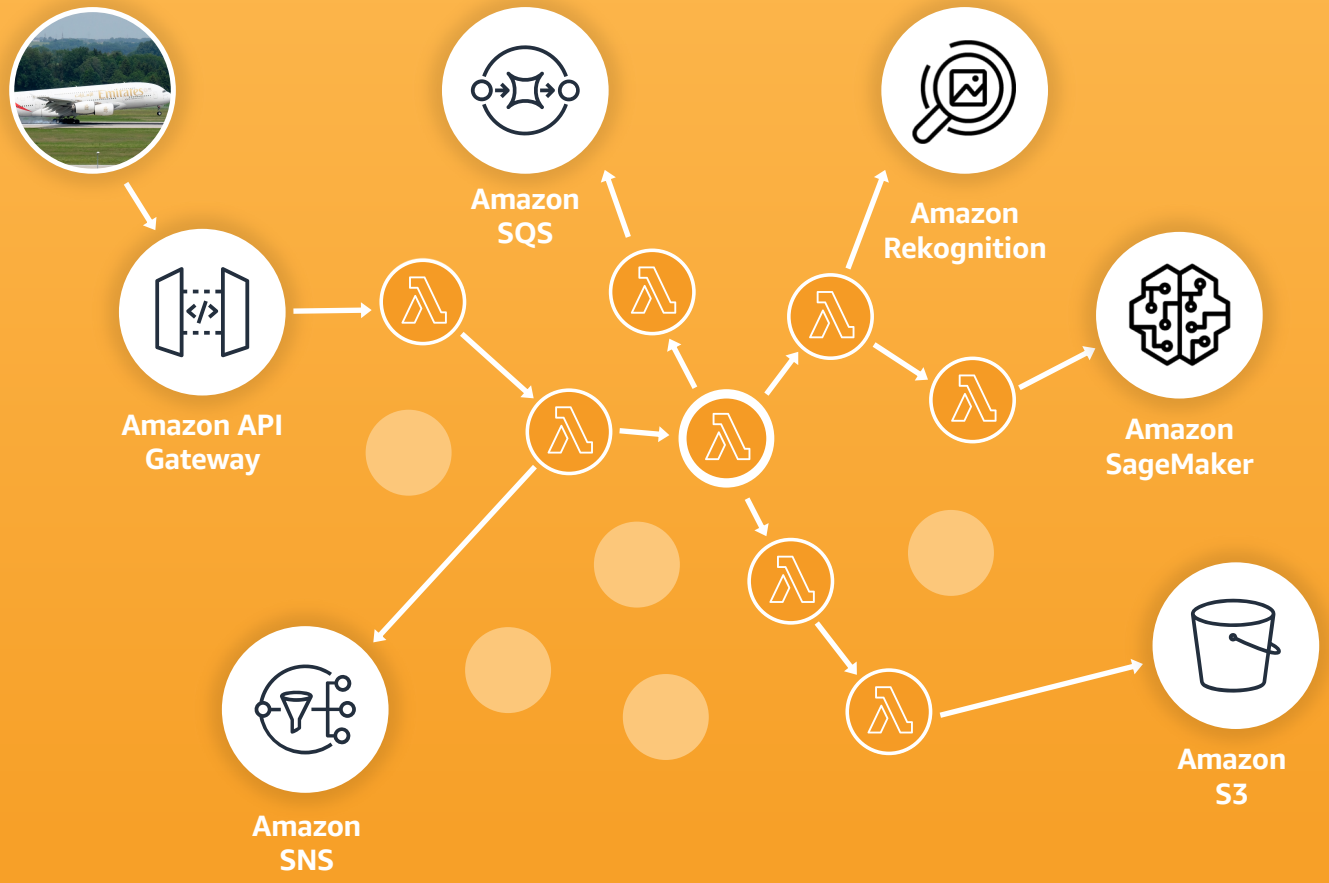
# It started with a BIG idea...


# ...and a little bit of code

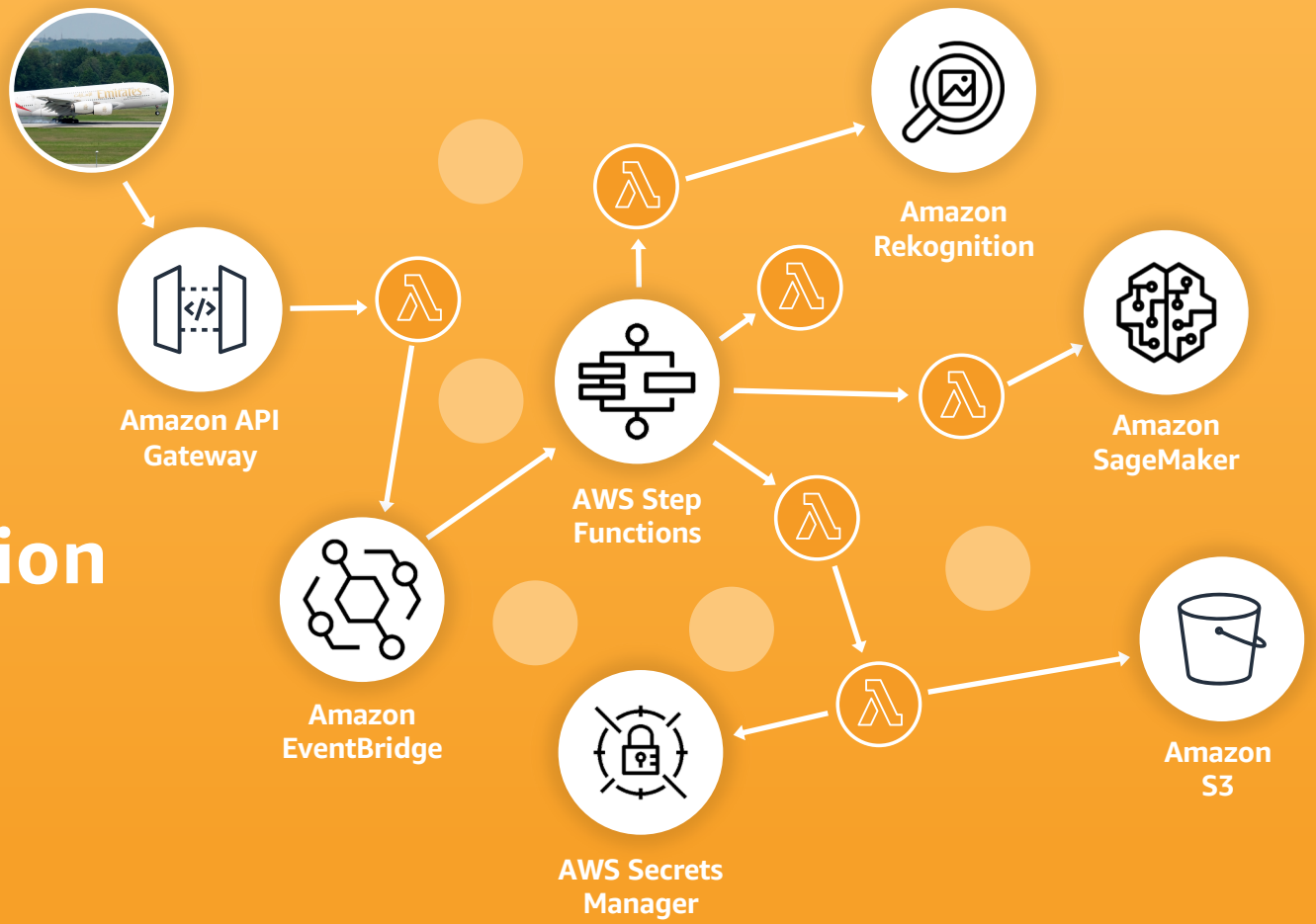
```
def handler(event, context):

  labels = recognition.detect_labels(image)
  return parse_labels(labels)
```

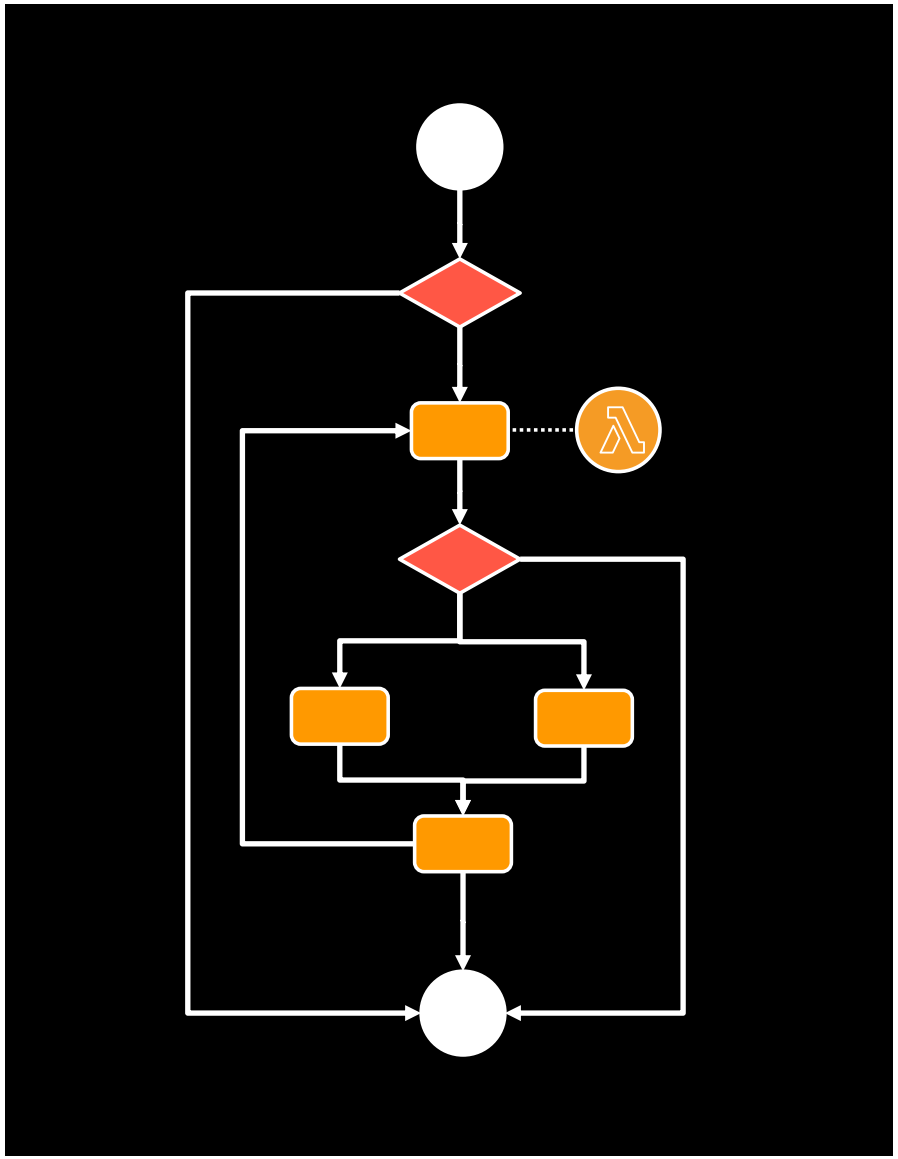
# It EVOLVED over time…

# …and a little MORE code



# Into a MODERN cloud application





# Orchestrated with AWS Step Functions

## Business logic in Lambda

## Control flow in Step Functions

Error conditions

Looping

Branching

# Built entirely with the AWS CDK

```
# Step 5 - classify aircraft
classify_aircraft_job = aws_stepfunctions.Task( self,
  'Classify aircraft',
  task = sft.InvokeFunction(classify_aircraft_func))

# Step 4 - Scan for tail numbers
detect_tailnumber_job = sf.Task( self,
  'Detect tailnumber',
  task = sft.InvokeFunction(detect_tailnumber_func))

detect_tailnumber_job.next(classify_aircraft_job)

# Step 3 - Crop aircraft from image
crop_aircraft_job = sf.Task( self,
  'Crop aircraft',
  task = sft.InvokeFunction(crop_image_func))

crop_aircraft_job.next(detect_tailnumber_job)
```

1 | Extensible

3 | Scalable

2 | Cost efficient

4 | Portable

**Bottom line—**

Serverless is the **fastest way** to build a modern application

If you object, let us know what we should work on next!

# Modern App Development

## Serverless First

Adrian Cockcroft
@adrianco
AWS VP Cloud Architecture strategy

# Thank you!