Serverless

The Future of the Cloud?!

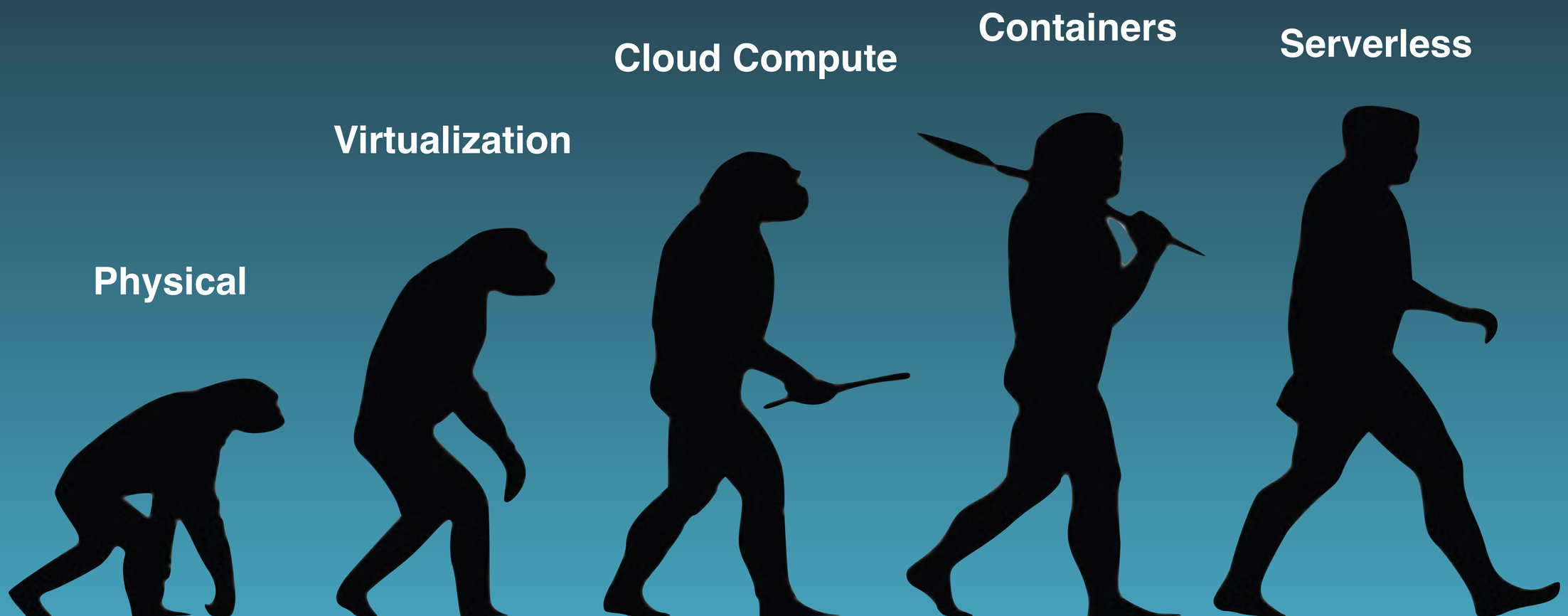
by Bert Ertman

luminis
Conversing worlds

@BertErtman

- Fellow, Director of Technology Outreach at Luminis
- Background in all things Java since 1995
- Java Champion, JavaOne Rockstar Speaker, and a Duke’s Choice Award Winner
- Involved in architecting and implementing dozens of large scale systems over the past 20 years or so
- Book author for O’Reilly, speaker at many conferences

The Evolution of Compute



The case for Serverless

- So far, the cloud is just someone else's computer
- Servers should be treated as cattle, not pets
- PAYGO? or PAYGO&aAYDG?

“No server is easier to manage than no server”

- Werner Vogels
CTO, Amazon

Serverless - what's in a name?

- Mostly describes what its not…
- Also known as Functions as a Service (FaaS)
- or Function PaaS (fPaaS) as defined by Gartner

- Some people refer to it as Back-end as a Service (BaaS)

Wait! Did someone just say “Back-end as a Service”?


Back-end as a Service??

Serverless Manifesto

- Functions are the unit of deployment and scaling
- No Machines, VMs, or Containers are visible in the programming model
- Permanent storage lives elsewhere
- Scales per request. Users cannot over- or under-provision capacity
- Never pay for idle (no cold servers/containers or their costs)
- Implicitly fault-tolerant because functions can run anywhere
- BYOC - Bring Your Own Code
- Metrics and Logging are a universal right

Main Benefits

- No servers to administer
- Pay for code execution only
- Automatic Scaling

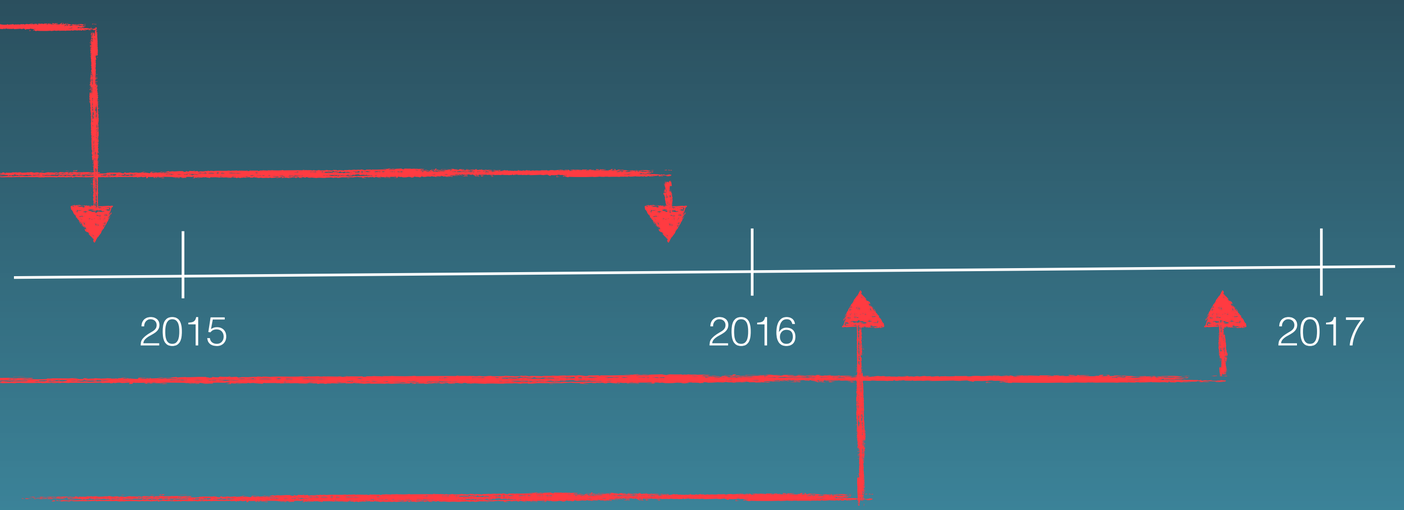
In Other Words…

- We don’t (have to) care about Application Servers
- We don’t (have to) care about Docker

So far so good, right?

Serverless implementations

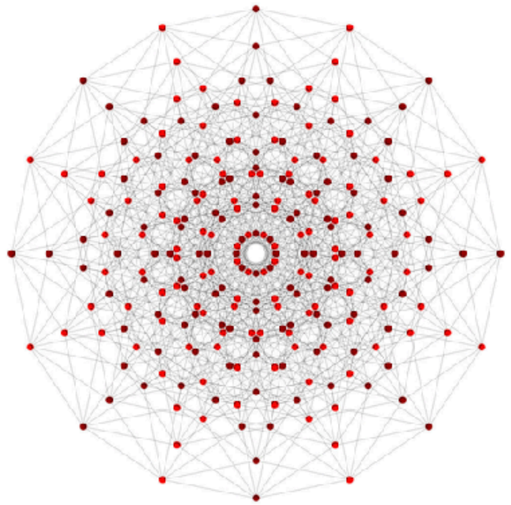
- Several Cloud vendors have implementations:
  - AWS Lambda
  - Google CloudFunctions
  - Azure Functions
  - IBM BlueMix OpenWhisk



Serverless implementations

CLOUD, DEVELOPERS, DEVOPS, JAVA, JAVASCRIPT, OPEN SOURCE | October 2, 2017

Announcing Fn–An Open Source Serverless Functions Platform



By: Shaun Smith
Director of Product Management

We are very excited to announce our new open source, cloud agnostic, serverless platform–Fn.

The Fn project is a container native Apache 2.0 licensed serverless platform that you can run anywhere–any cloud or on-premise. It's easy to use, supports every programming language, and is extensible and performant.

We've focused on making it really easy to get started so you can try it out in just a few minutes and then use more advanced features as you grow into it. Check out our quickstart to get up and running and deploying your own function in a few minutes.

Share

History

The Fn Project is being developed by the same team that created IronFunctions. The team pioneered serverless technology and ran a hosted serverless platform for 6 years. After running billions of containers for thousands of customers, pre and post Docker, the team

https://github.com/fnproject/fn

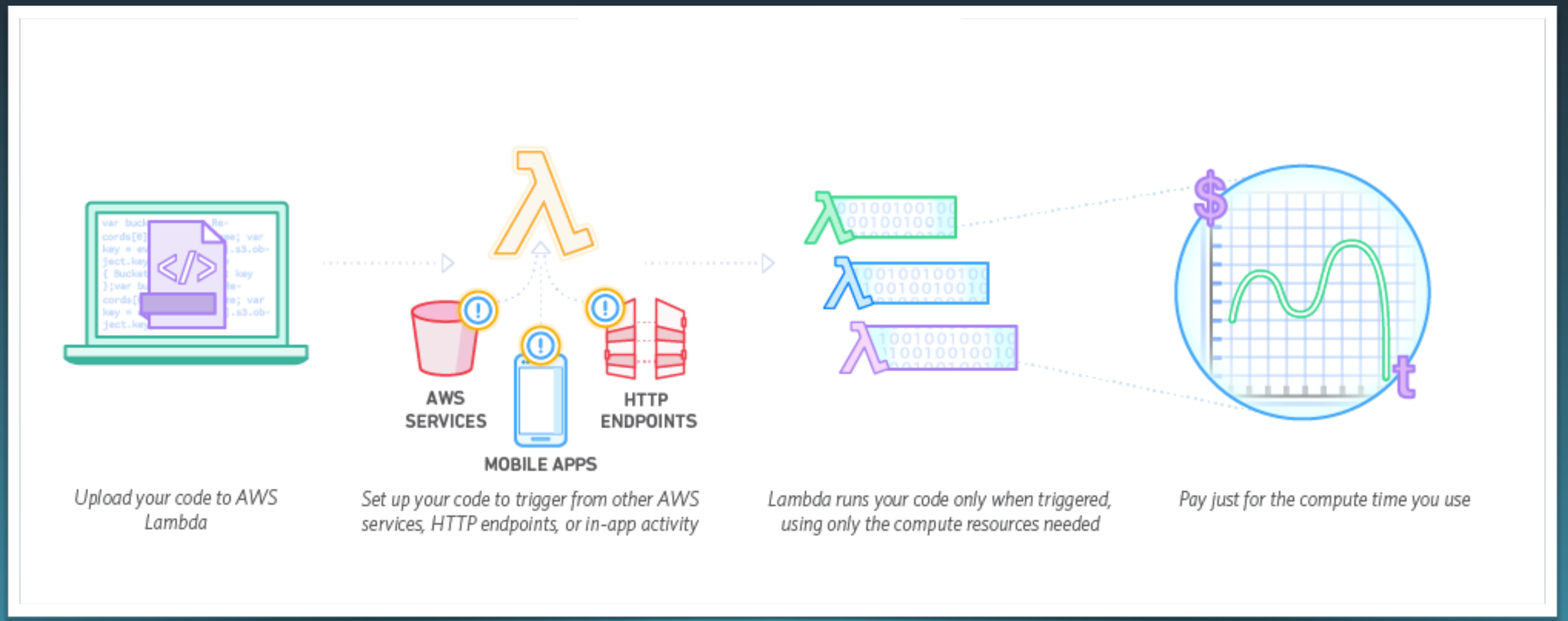
AWS Lambda

- Event-driven, serverless computing platform provided by Amazon
- Runs code in response to events and automatically manages the compute resources required by that code
- First introduced in Nov 2014
- Part of the Amazon Web Services offerings

Runtime Support

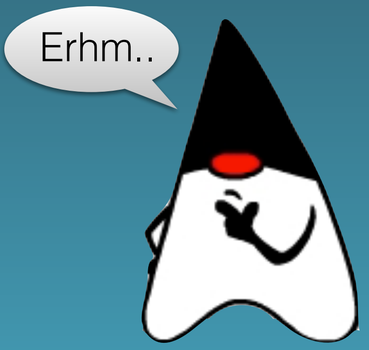
- Python 2.7
- NodeJS 4.3
- Java 8
- C#
- Go

How it works



Upload **Trigger** **Run** **Pay**

lambda



demo

Lambdas are event-driven

Many Event Sources

API Gateway

Amazon S3

Amazon DynamoDB

Amazon Aurora

Amazon Simple Notification Service

Amazon Simple Email Service

Amazon Cognito

Amazon CloudWatch

Amazon Kinesis Streams

AWS CodeCommit

AWS CloudFormation

AWS Config

Amazon Lex

Many Event Sources

API Gateway

HTTP(s) requests

Amazon S3

CRUD events on data sources

Amazon DynamoDB

Amazon Aurora

Amazon Simple Notification Service

messaging

Amazon Simple Email Service

events

Amazon Cognito

Amazon CloudWatch

log/stream processing

Amazon Kinesis Streams

commit hooks / AWS services

AWS CodeCommit

AWS CloudFormation

setup

AWS Config

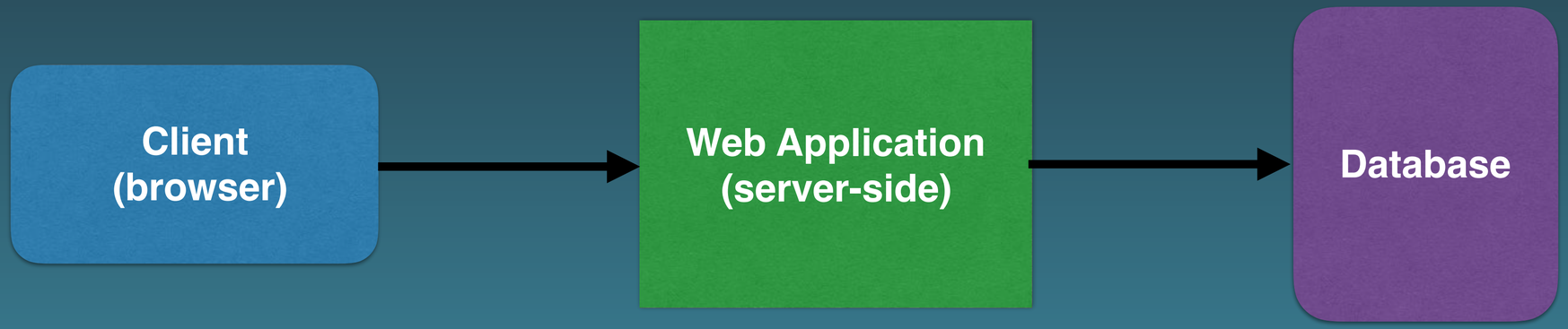
configuration mgmt

Amazon Lex

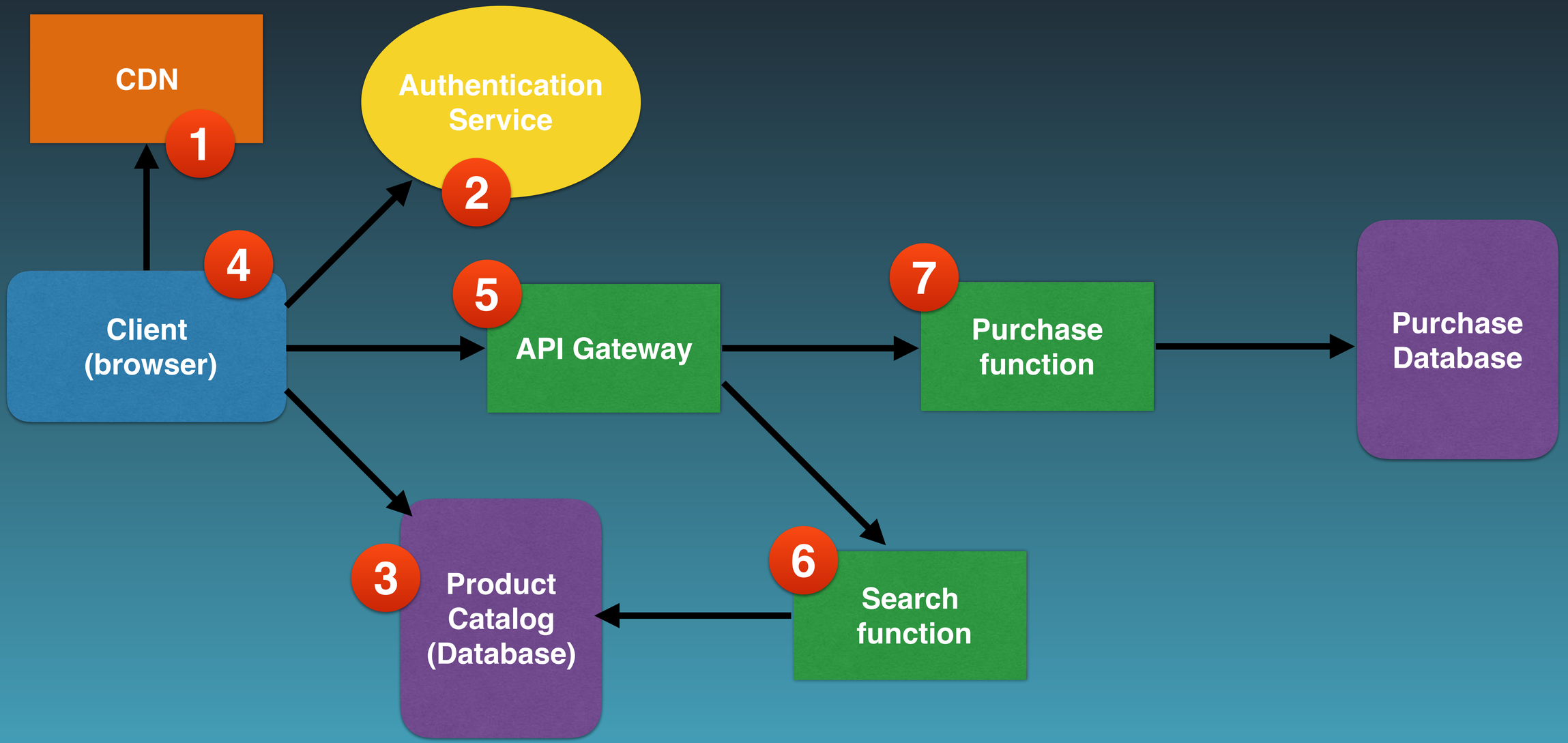
voice & text

Rethinking Traditional Architectural Concepts

Rethinking Traditional Architectural Concepts



Rethinking Traditional Architectural Concepts

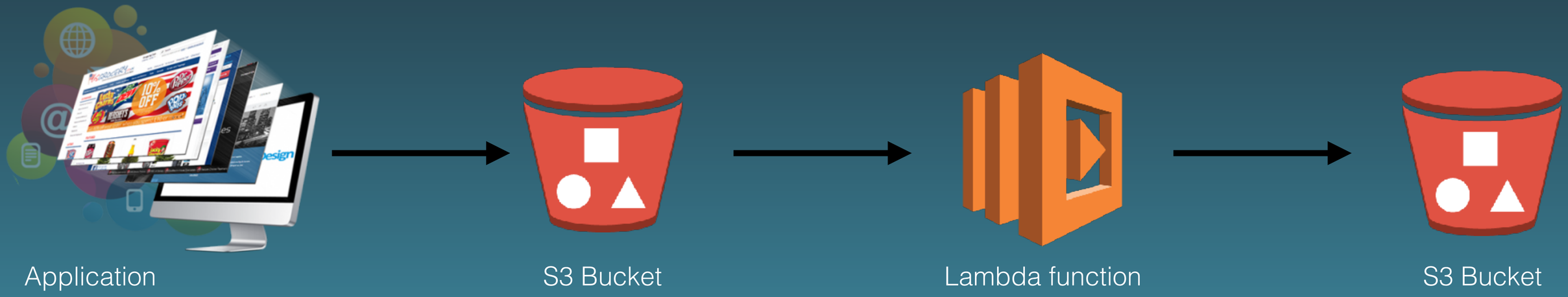


Just because you can might not always be the right reason

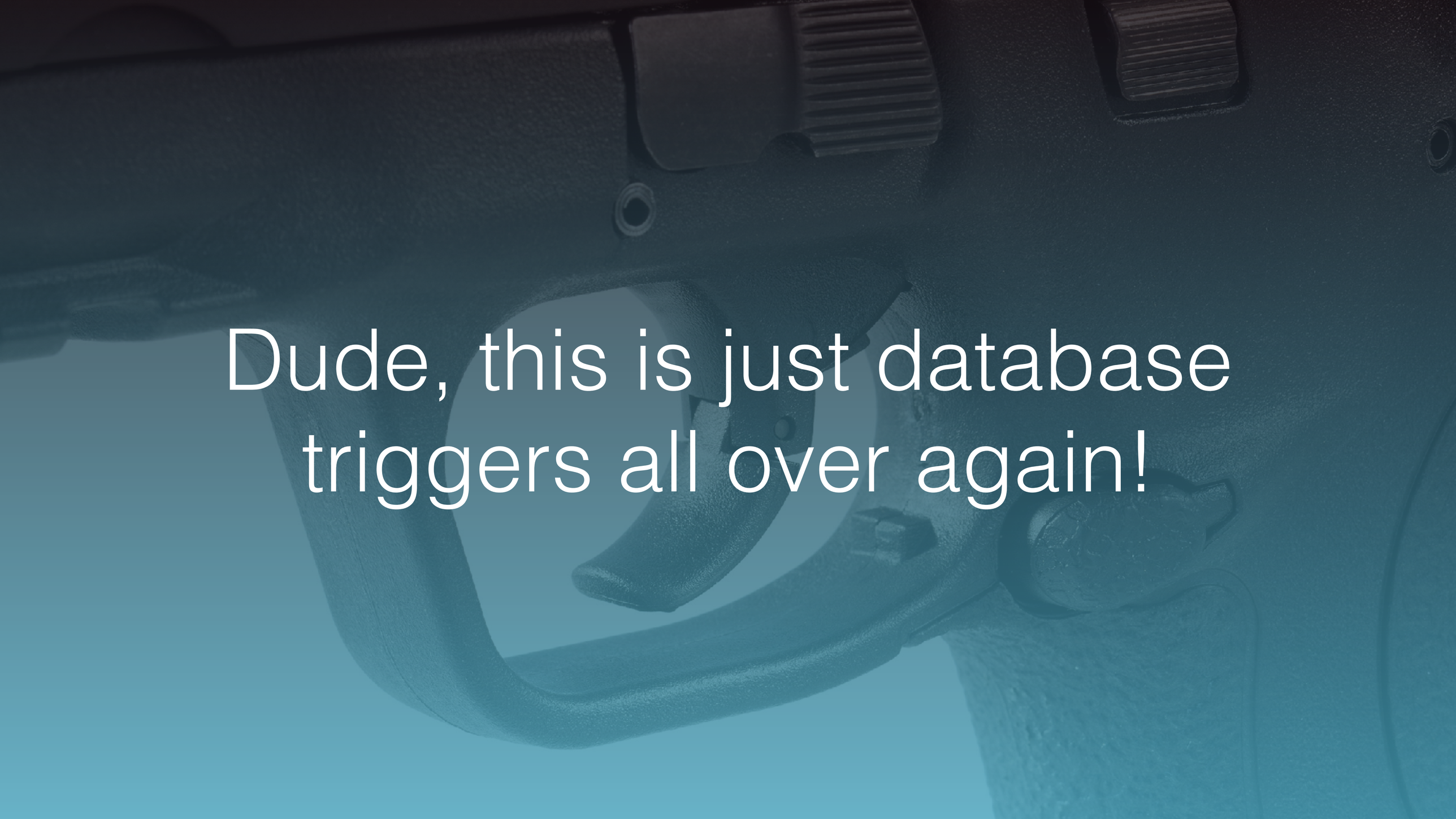
Example: Event-based processing

- Respond to incoming data, such as an S3 Bucket insert
- Useful for data/stream processing, MapReduce, or batch processing

Typical Scenario



demo


Dude, this is just database
triggers all over again!

Another Example: BaaS

- Backend-as-a-Service for a (mobile/web) app
- Responds to incoming HTTP GET/POST requests
- Stateless

Typical Scenario

Mobile App

Web application

API Gateway

Lambda function

DynamoDB

demo

Other Use-Cases

- Implement custom CI/CD pipeline on AWS
- Bots
- Voice Control :)

Typical Scenario

Amazon Echo

Alexa Skill

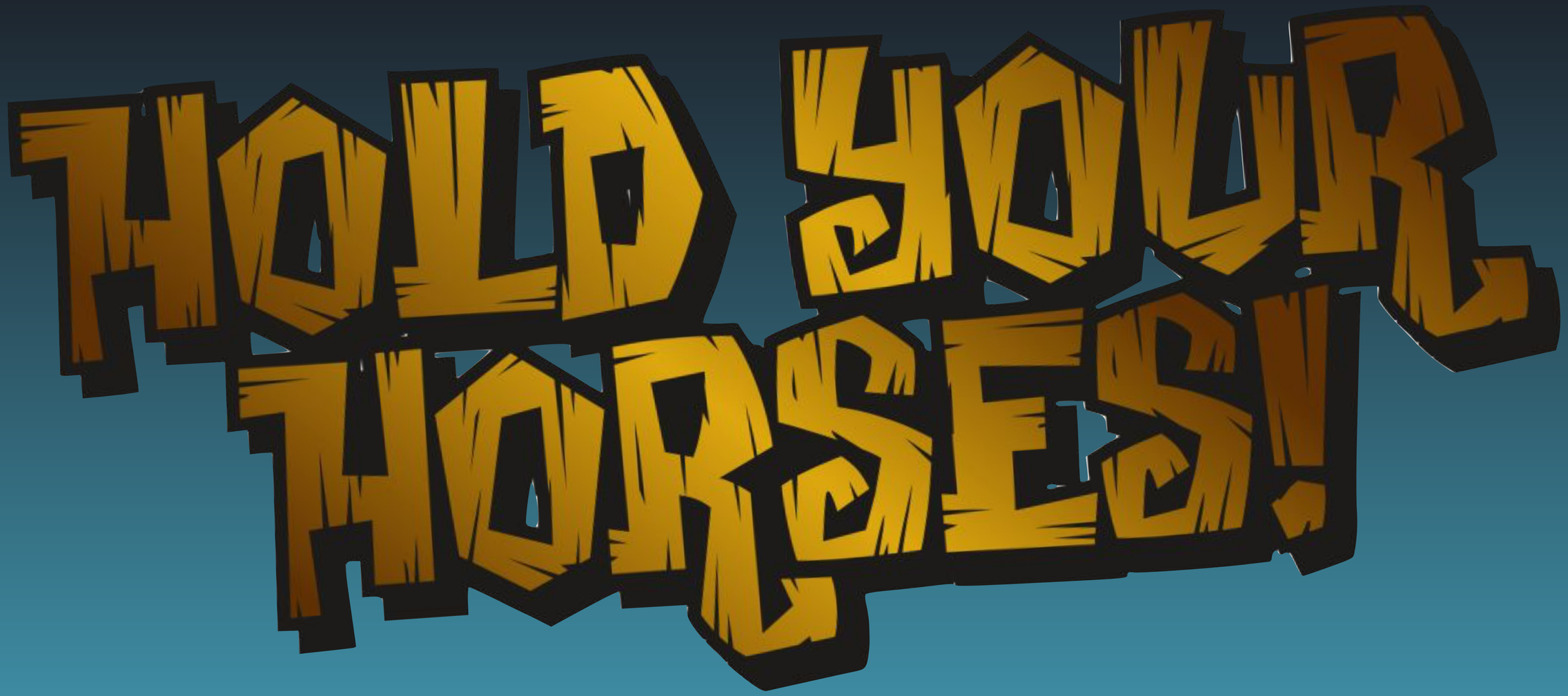
Lambda function

demo

What Expedia is doing with Lambda

beyond Hello World example

source: AWS re:invent 2016: Serverless Computing Patterns at Expedia (SVR306)


HOLD YOUR
HORSES!


Yeah, nice and all, but all of this
$**t is running in the cloud…


fn

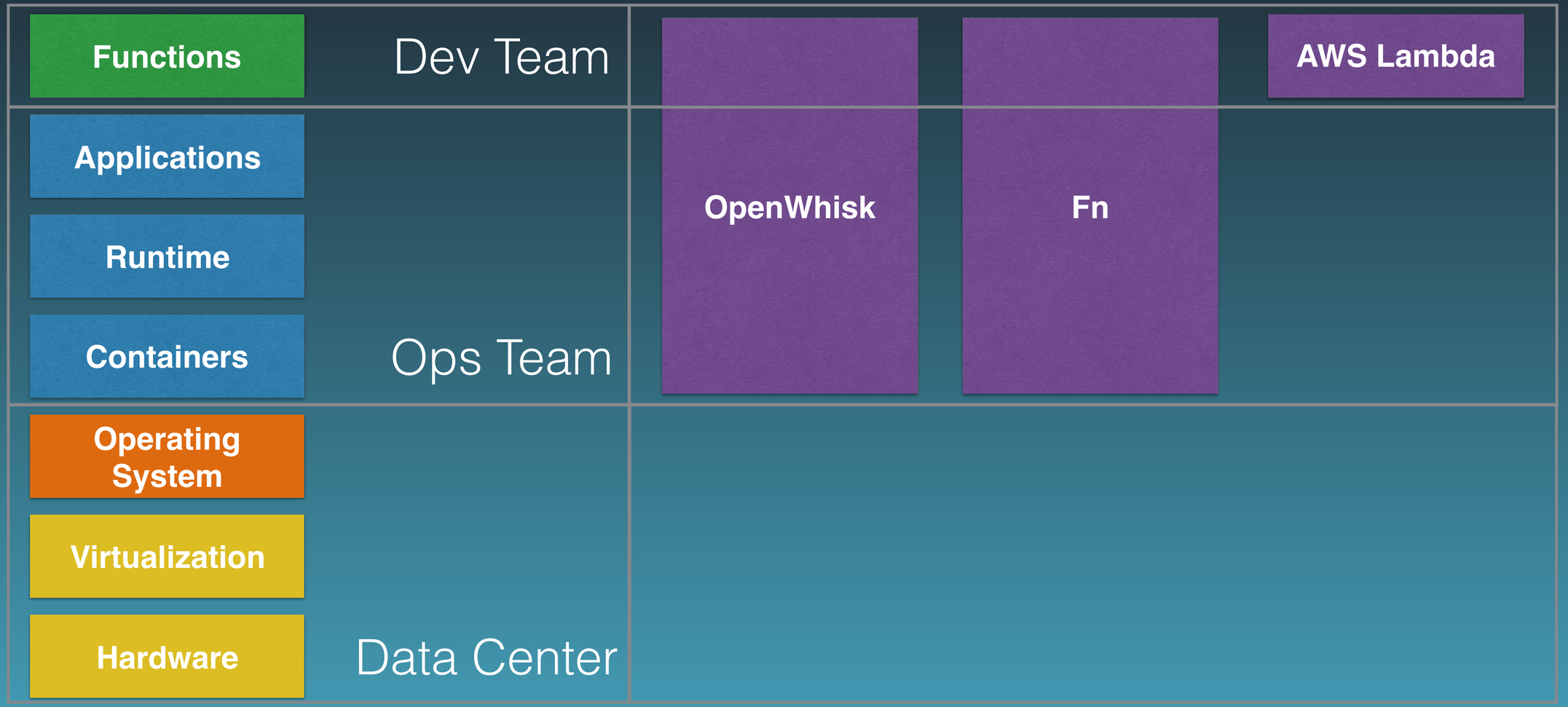
Apache OpenWhisk

- Runs in IBM's and Red Hat's cloud offerings, but can also run on-premise
- OpenWhisk is Apache licensed and Open Source
- Currently supports: JavaScript, Java, Python, and Swift(!)
- Possible to run functions in provided Docker images

- Functions are the unit of deployment and scaling
- No Machines, VMs, or Containers are visible in the programming model
- Permanent storage lives elsewhere
- Scales per request. Users cannot over- or under-provision capacity
- Never pay for idle (no cold servers/containers or their costs)
- Implicitly fault-tolerant because functions can run anywhere
- BYOC - Bring Your Own Code
- Metrics and Logging are a universal right

Is this still Serverless?

Is this still Serverless?



Now Serverless is cool, but there are some drawbacks too…

Drawbacks

- Vendor control and lock-in
- Multi-tenancy
- Security concerns (increasing the attack surface)
- Loss of server optimizations
- Execution time is limited
- Start-up latency
- Testing
- Discovery


summary

Serverless

- ..is rapidly being embraced by major cloud players
- ..is promoting functions as first class citizens
- ..is event-based, stateless, and transient
- ..is infinitely scalable (in theory)
- ..is different from traditional deployment models
- ..is giving the cloud a run for its money
- ..is lots of bang for the buck
- ..is still very much proprietary, so lock-in is your choice!

Serverless, the future of the Cloud!

Thank you!

@BertErtman