University of BRISTOL

Serverless

Dan Schien and Alan Forsyth

COMSM0010 Lecture 10

bristol.ac.uk

Objectives/Learning Outcomes

- Explain the principles of serverless architecture
- Name the serverless architectural layers
- Describe the four ‘pillars’ of serverless architecture
- Introduce the AWS Lambda FaaS service
- Describe the four ‘stumbling blocks’ of serverless
- Illustrate common serverless use cases using AWS Lambda
- Review some methods of getting hands-on access to AWS

First – to Dispel the Myth…

First – to Dispel the Myth…

Serverless Uses Servers!

Serverless Definition 1: NoAuth

"The essence of the serverless trend is the **absence** of the server concept during software development"

https://auth0.com/blog/what-is-serverless/

Server Management

"I've hugged a lot of servers in my life, and believe me, they do not hug you back. They hate you."

Werner Vogels, CIO, AWS

** Undifferentiated Heavy Lifting*

Serverless Definition 2: AWS

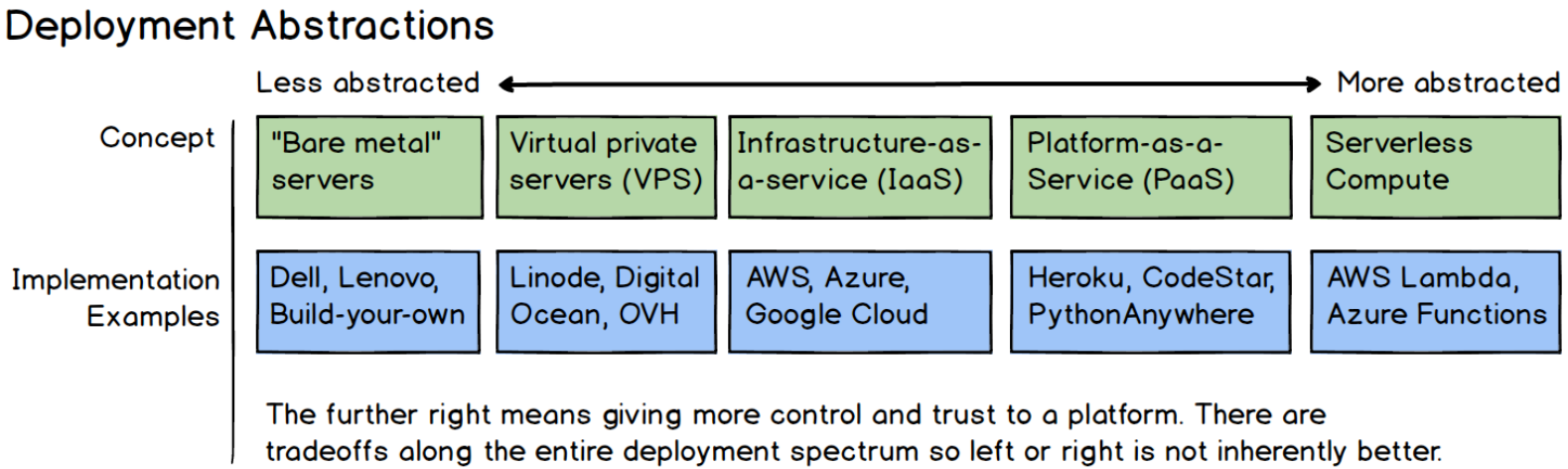
"Serverless most often refers to serverless applications. Serverless applications are ones that **don't require you to provision or manage any servers.** You can focus on your core product and business logic instead of responsibilities like operating system (OS) access control, OS patching, provisioning, right-sizing, scaling, and availability. By building your application on a serverless platform, the platform manages these responsibilities for you."

https://d1.awsstatic.com/whitepapers/serverless-architectures-with-aws-lambda.pdf

Abstractions of App Deployment

- Evolution in deployment has seen more and more abstraction
- There are always tradeoffs:

: more control and trust given to platform



Serverless Layers

– Function as a Service (FaaS): AWS Lambda, Google Cloud Functions, Azure Functions

Serverless Layers

– Function as a Service (FaaS): AWS Lambda, Google Cloud Functions, Azure Functions

– AWS S3 / DynamoDB / Aurora, Google Firebase, Fauna

Serverless Layers

– Function as a Service (FaaS): AWS Lambda, Google Cloud Functions, Azure Functions

– AWS S3 / DynamoDB / Aurora, Google Firebase, Fauna

– AWS SNS / SQS, PubNub

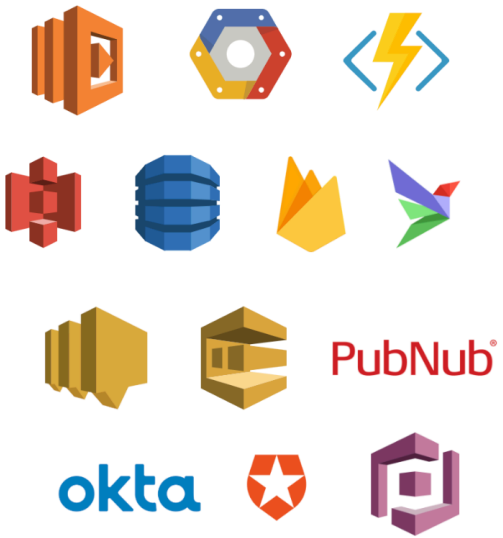
Serverless Layers

– Function as a Service (FaaS): AWS Lambda, Google Cloud Functions, Azure Functions

– AWS S3 / DynamoDB / Aurora, Google Firebase, Fauna

– AWS SNS / SQS, PubNub

– Okta, Auth0, AWS Cognito



Serverless Layers

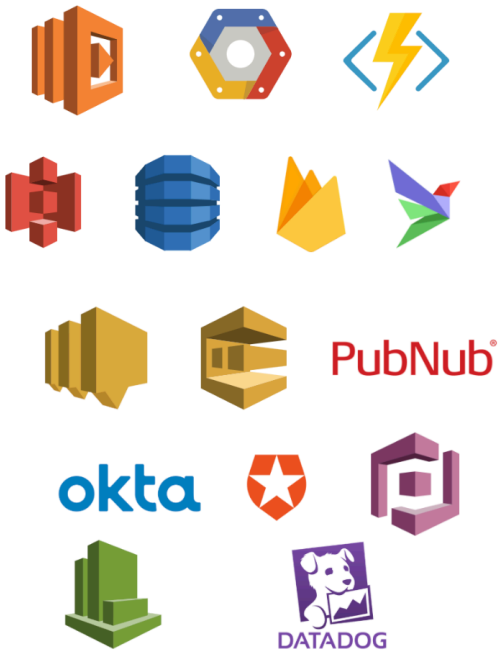
– Function as a Service (FaaS): AWS Lambda, Google Cloud Functions, Azure Functions

– AWS S3 / DynamoDB / Aurora, Google Firebase, Fauna

– AWS SNS / SQS, PubNub

– Okta, Auth0, AWS Cognito

– AWS CloudWatch, DataDog



Serverless Layers

– Function as a Service (FaaS): AWS Lambda, Google Cloud Functions, Azure Functions

– AWS S3 / DynamoDB / Aurora, Google Firebase, Fauna

– AWS SNS / SQS, PubNub

– Okta, Auth0, AWS Cognito

– AWS CloudWatch, DataDog

– AWS API Gateway / CloudFront, Netlify

The Four Pillars of Serverless

No Server Management

No need to know the number of servers or their configuration

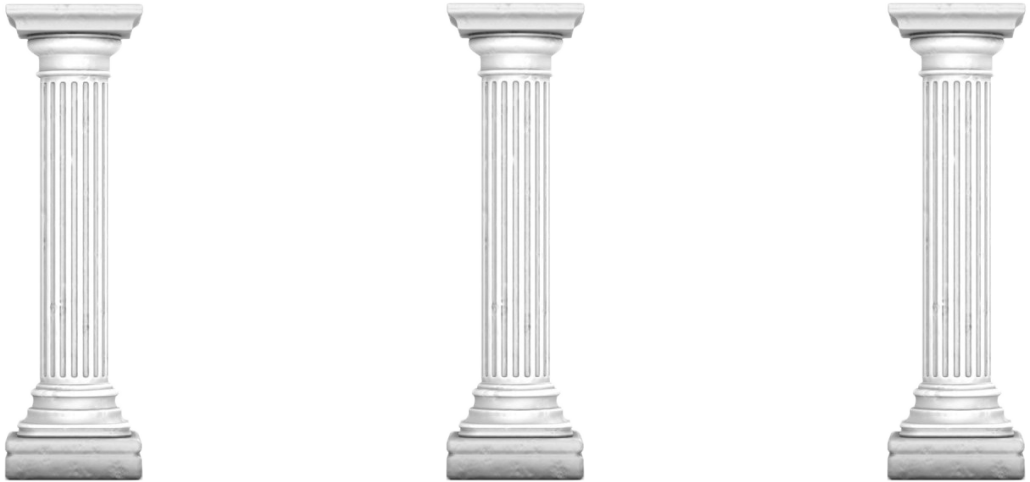
The Four Pillars of Serverless

Flexible
Scaling

More resources are allocated if you need them

The Four Pillars of Serverless

High Availability



Redundancy and Fault Tolerance built in

The Four Pillars of Serverless

Never Pay for Idle

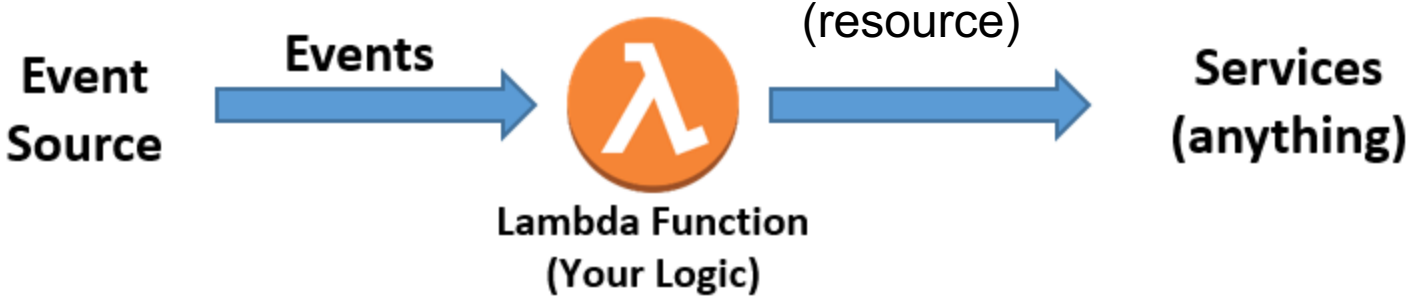



No charge for unused resources

Serverless FaaS: AWS Lambda

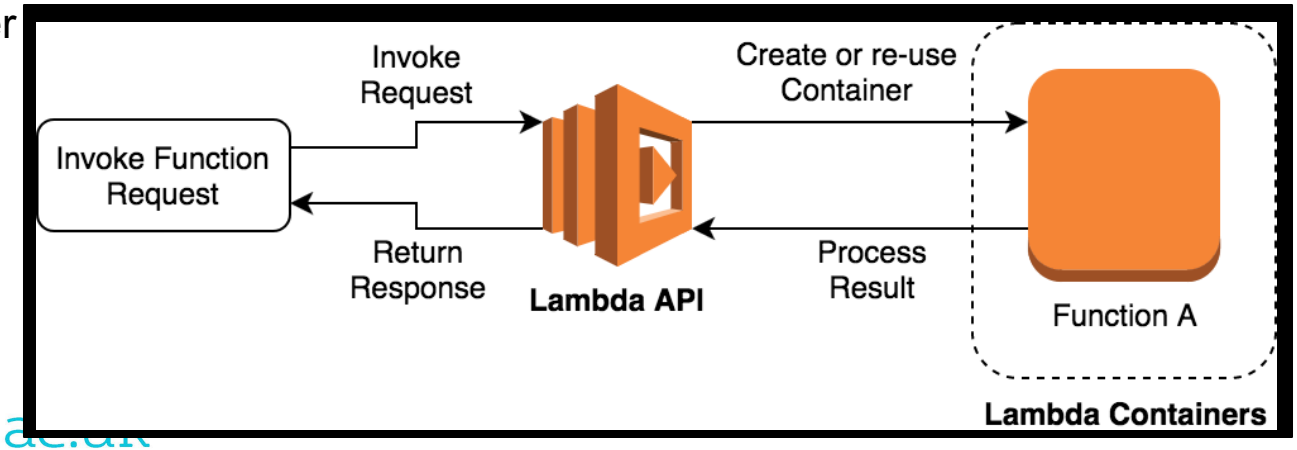
- Fully launched in 2015
- Currently supports:
  - Node JS, Python 2/3, C# (core), Java, Go, PowerShell (core) [more info]
- Triggered by an event
  - e.g. notification, message, cron

- Minimal Configuration:
  - **Code** (zip – max 50MB)
  - **Memory** (128MB – 3GB)
  - **Timeout** (max 300 secs)
  - Invocation **Event Source** or schedule
  - 2 sets of **permissions**: event source (invocation) & service (resource)



How Does Lambda Work?

- Lambda function is invoked via event or schedule:
  - Minimal **Amazon Linux** container is run containing the required runtime and function
  - Function provided with JSON invocation context and parameters
  - Function executes and completes or times out, result returned if defined
  - Logs are created if function has been given logging resource permissions
- Typically invoked in a few ms (warm start) – default max is 1000 simultaneous invocations
- Container



Lambda Billing

- Billed per request (invocation) for:
  - amount of **memory** allocated
  - **duration** (in 100ms increments)
- Cost/month (after Free Tier):
  - $0.20 per 1M requests
  - $0.00001667 per GB-second

- **Lambda Free Tier** (permanent!)
  - 1M requests free
  - 400,000 GB-seconds per month (up to 3.2M seconds at 128MB)
- Cost calculators:
  - http://serverlesscalc.com/
  - servers.lol

Demo 1 – Configuring and Testing a Lambda Function

[if time allows]

The Four Stumbling Blocks of Serverless

Performance Limitations

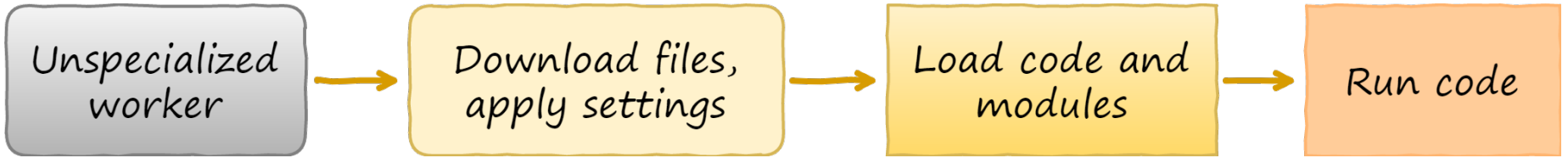
Cold starts, max execution time and strict resource limits

Cold Start Issue

- Auto-provisioning and auto-scaling require dynamic provisioning

code that hasn't been used for a while takes longer to start

longer response latencies from serverless application

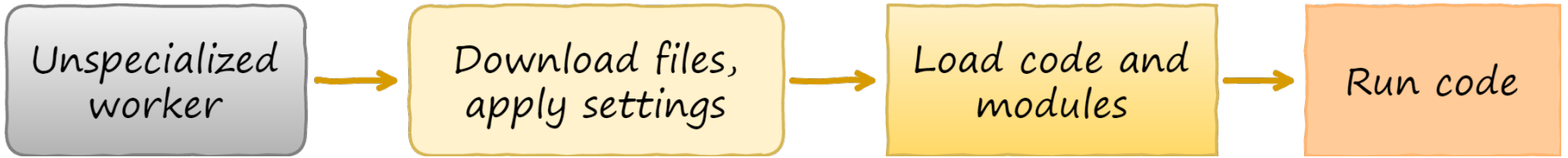


Cold Start Issue

- Auto-provisioning and auto-scaling require dynamic provisioning

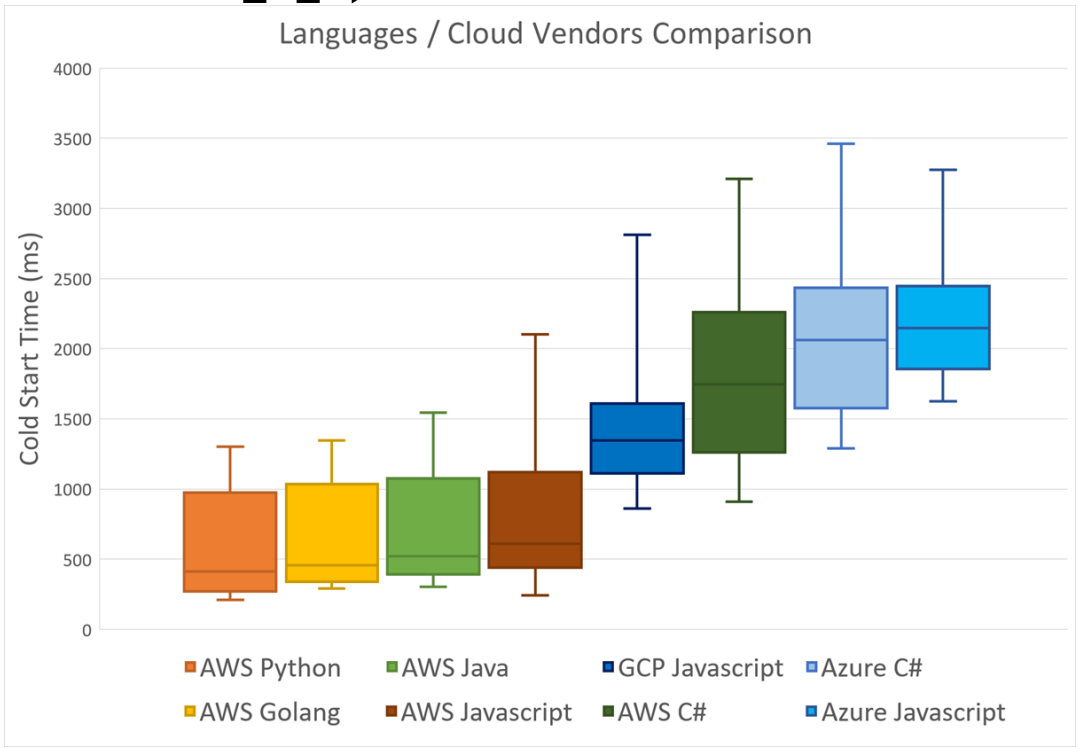
  code that hasn't been used for a while takes longer to start

  longer response latencies from serverless application



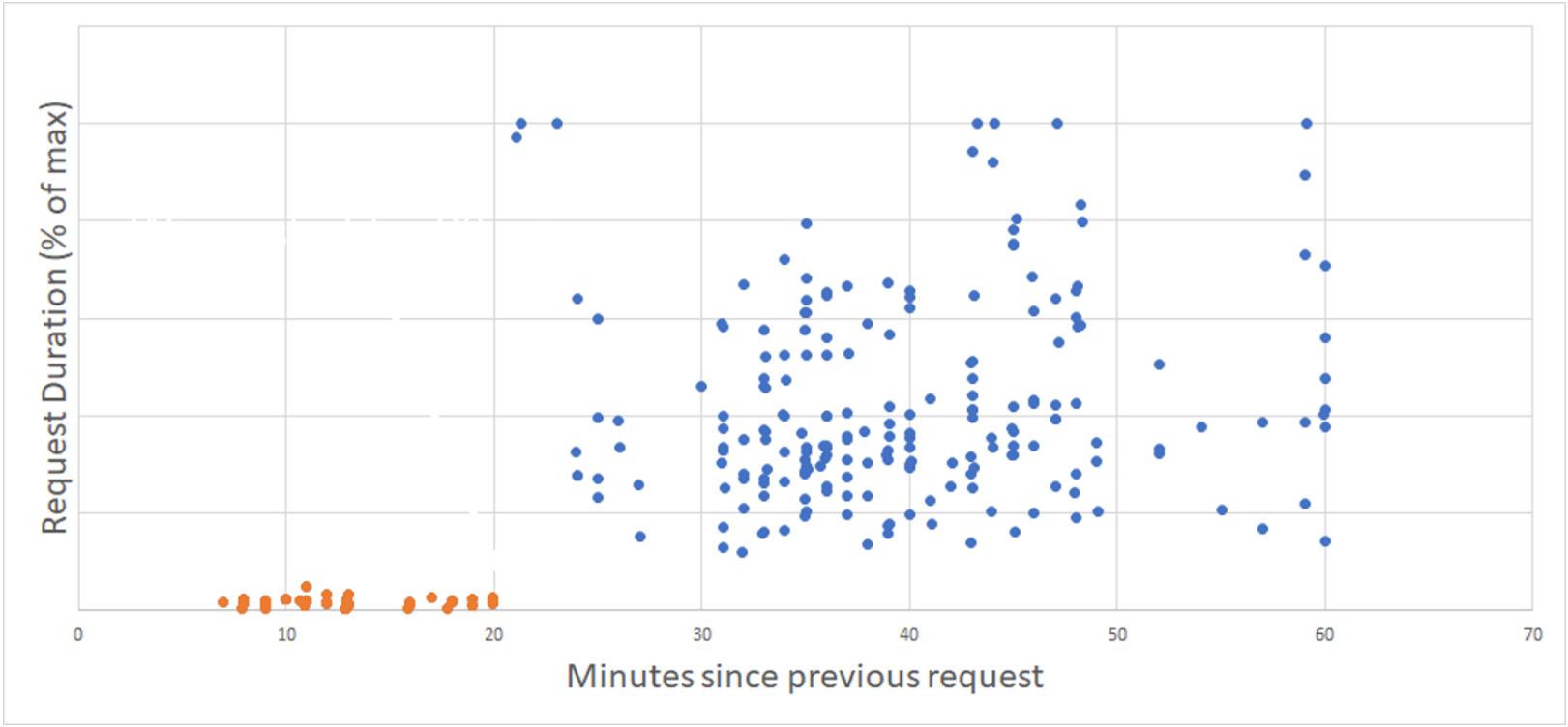
- Providers keep running containers 'warm' for a certain time for quick reuse

FaaS Platform Cold Start Comparison ('Hello World' app)



Reference: https://mikhail.io/2018/08/serverless-cold-start-war/

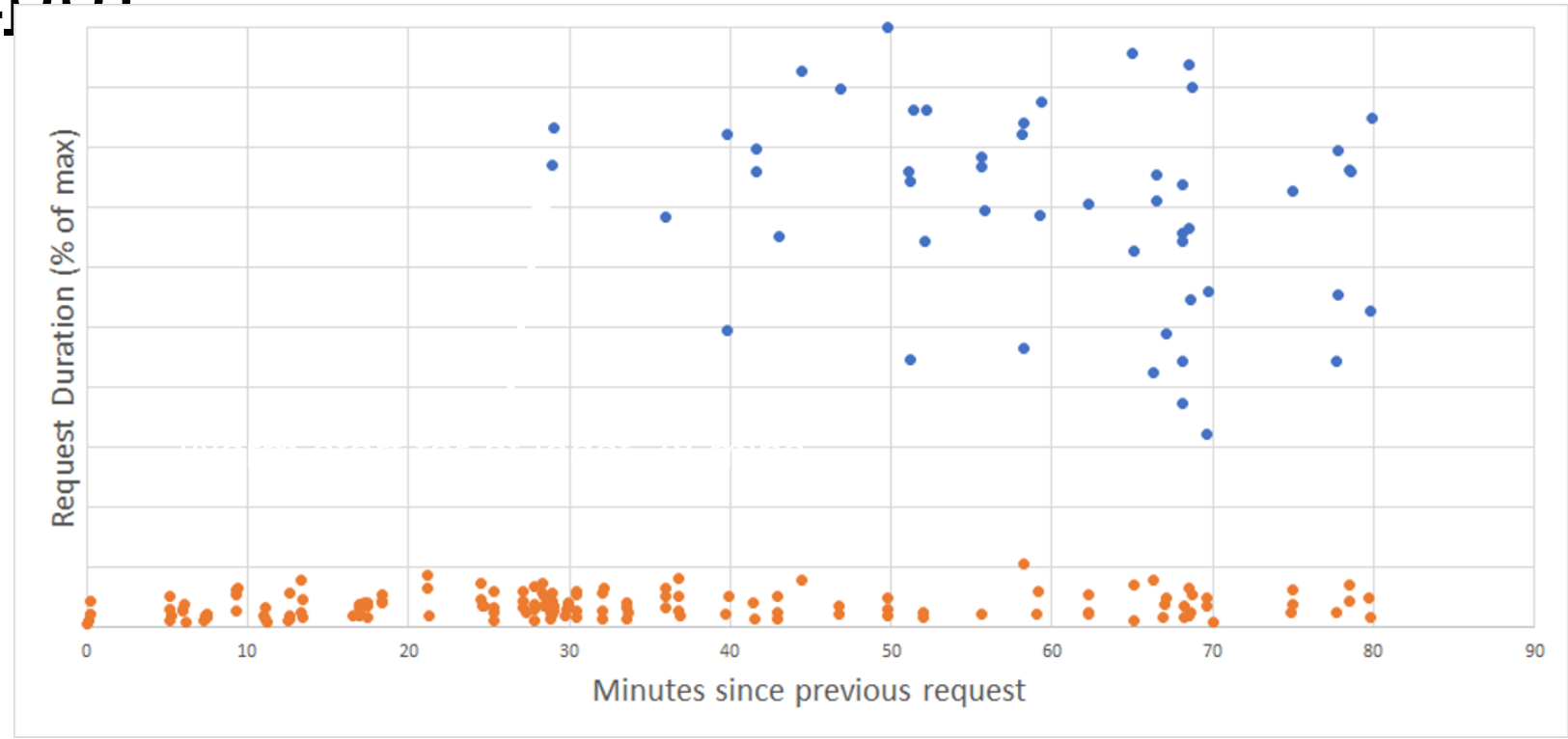
Azure – Fixed Cold Start ('Hello World' app)



bristol.ac.uk

Reference: https://mikhail.io/2018/08/serverless-cold-start-war/

AWS – Variable Cold Start ('Hello World' app)



Reference: https://mikhail.io/2018/08/serverless-cold-start-war/

The Four Stumbling Blocks of Serverless

Vendor
Lock-in

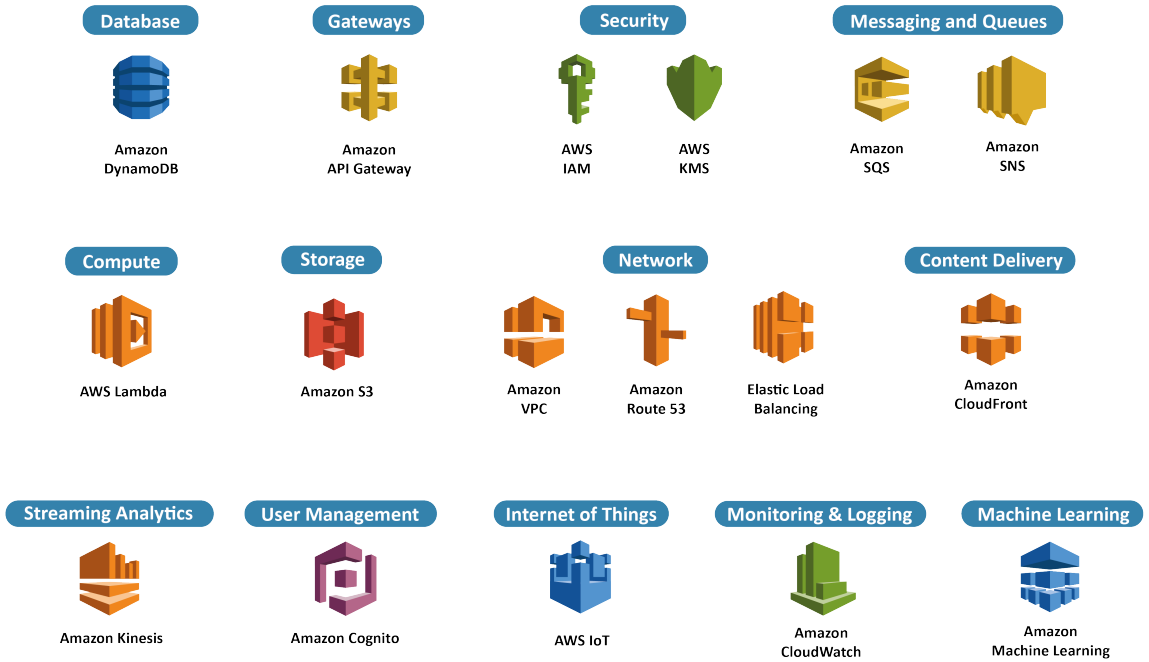
Data
Applications
Infrastucture
HR

Trade off: deep integration vs agnostic architecture (open source)

AWS Serverless Ecosystem

- Rapid development
- Easy integration
- Cost-effective (low TCO)
- Large community & support

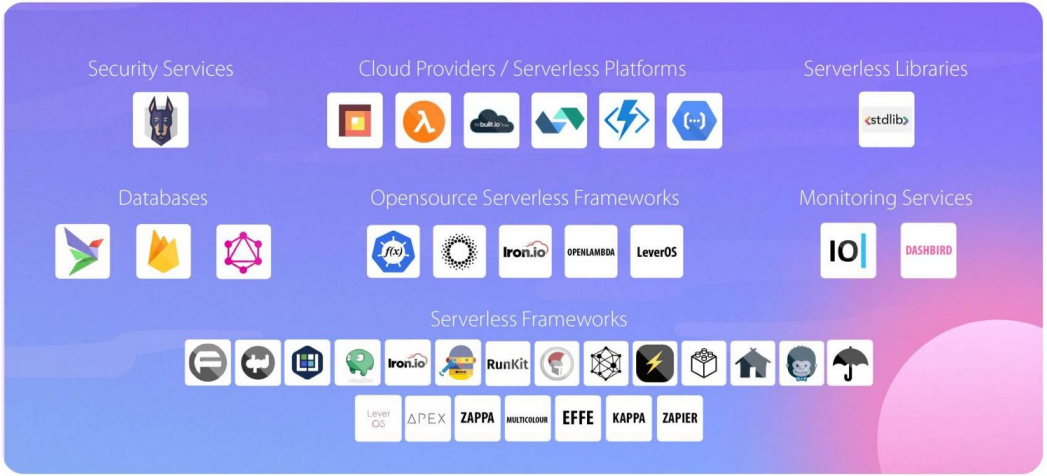
- Single vendor dependency
- Proprietary tooling & APIs



Open Source Serverless Ecosystem

- Open code & architecture
- Portability
- Heavy customisation possible
- No vendor dependency

- Higher TCO ('UDH')
  - Specialists & admin
  - Updates & patches
- Variable documentation & support
- Harder integration



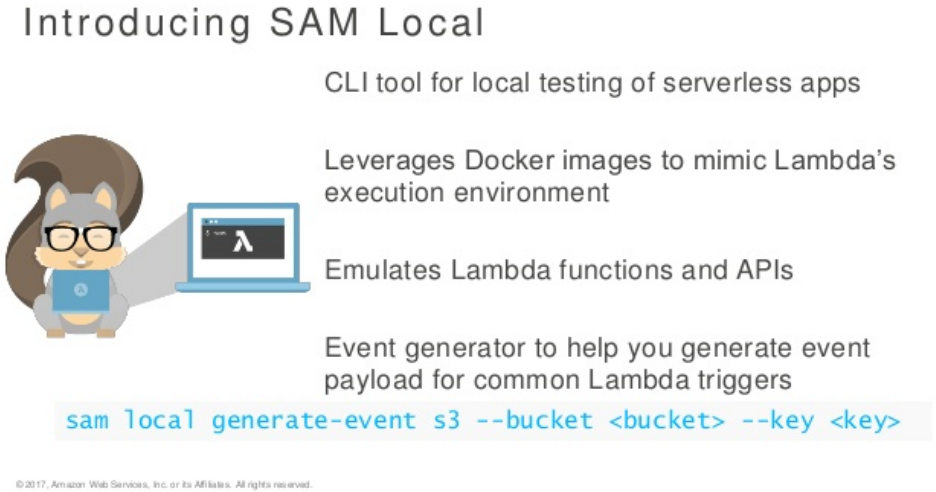
The Four Stumbling Blocks of Serverless

Monitoring & Debugging

Challenging to troubleshoot provider-controlled resources

Serverless Monitoring & Debugging – Work In Progress

- Many emerging tools & frameworks
  - e.g. serverless.com, Zappa
- Local testing of Lambda functions:
  - SAM Local (open source container)
- Cloud-based IDE
  - Cloud9
- Watch this space!



The Four Stumbling Blocks of Serverless

Security & Privacy

Multi-tenancy, opacity and resource reuse add extra challenges

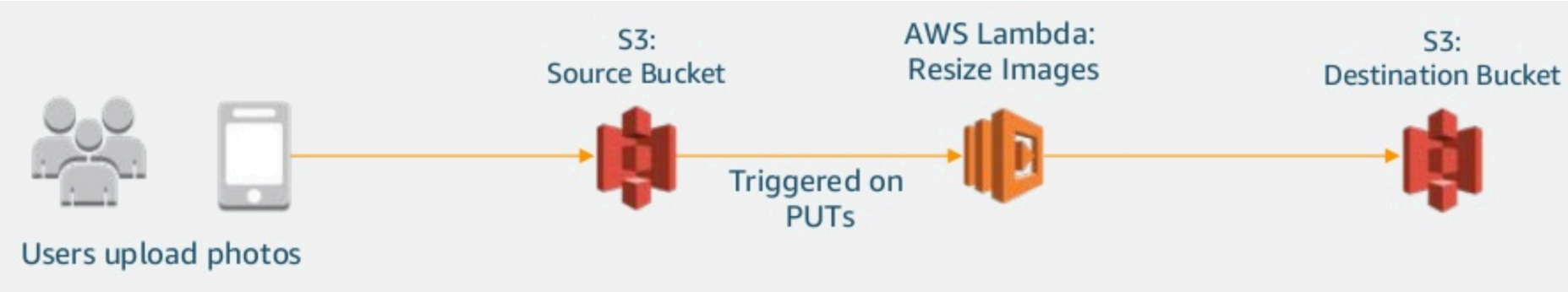
Serverless Security

- **Strengths**
  - OS patched and managed by platform
  - Attack surface area can be reduced
    - No long-running processes
    - Granular permissions can be assigned

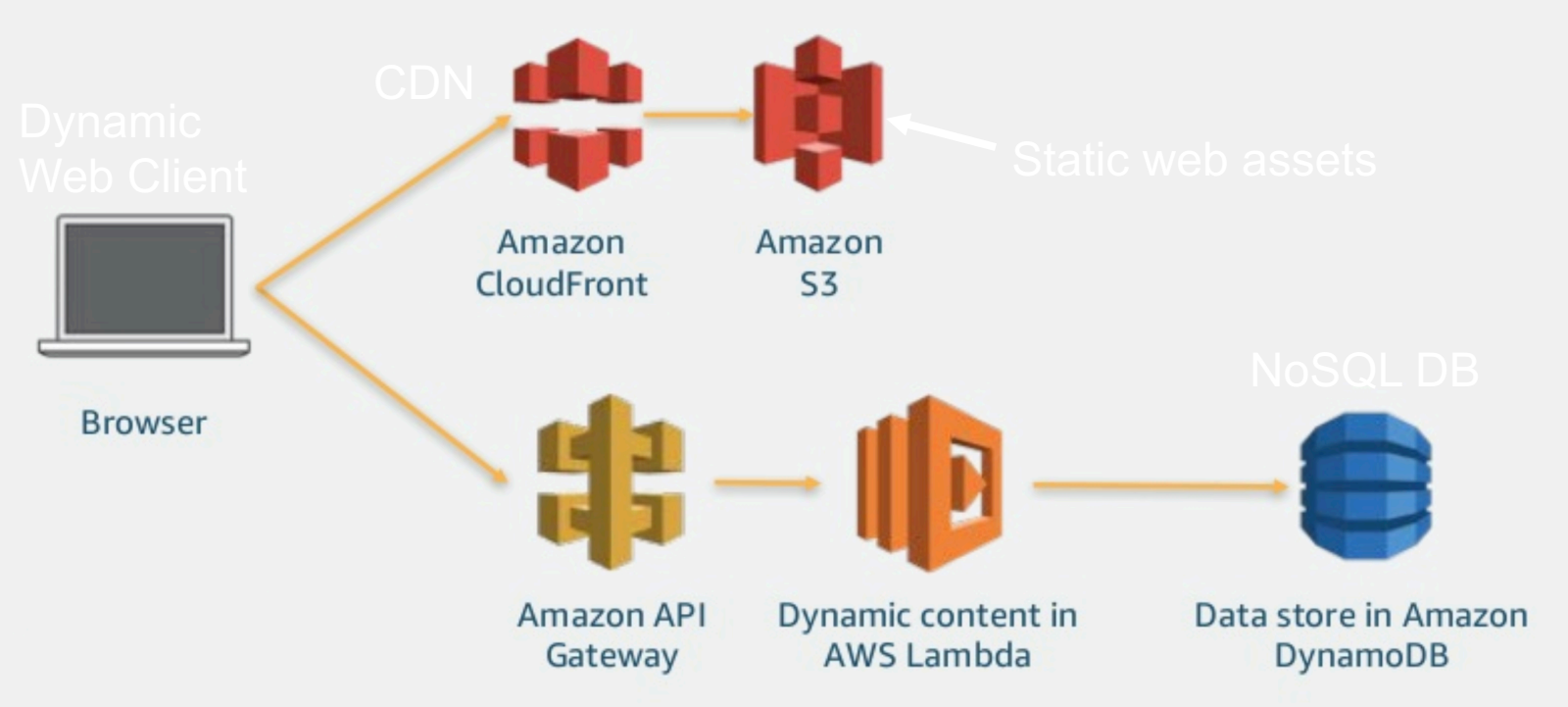
- **Weaknesses**
  - App code still vulnerable to library dependencies
  - OWASP top 10 vulnerabilities still relevant
    - Injection, Cross-site scripting, Sensitive data exposure
  - Permissions are often too lax
  - Wider and more diverse ecosystem (microservices)
    - Increases possible surface area for attack

OWASP Top 10 - 2017
A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE)
A5:2017-Broken Access Control
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

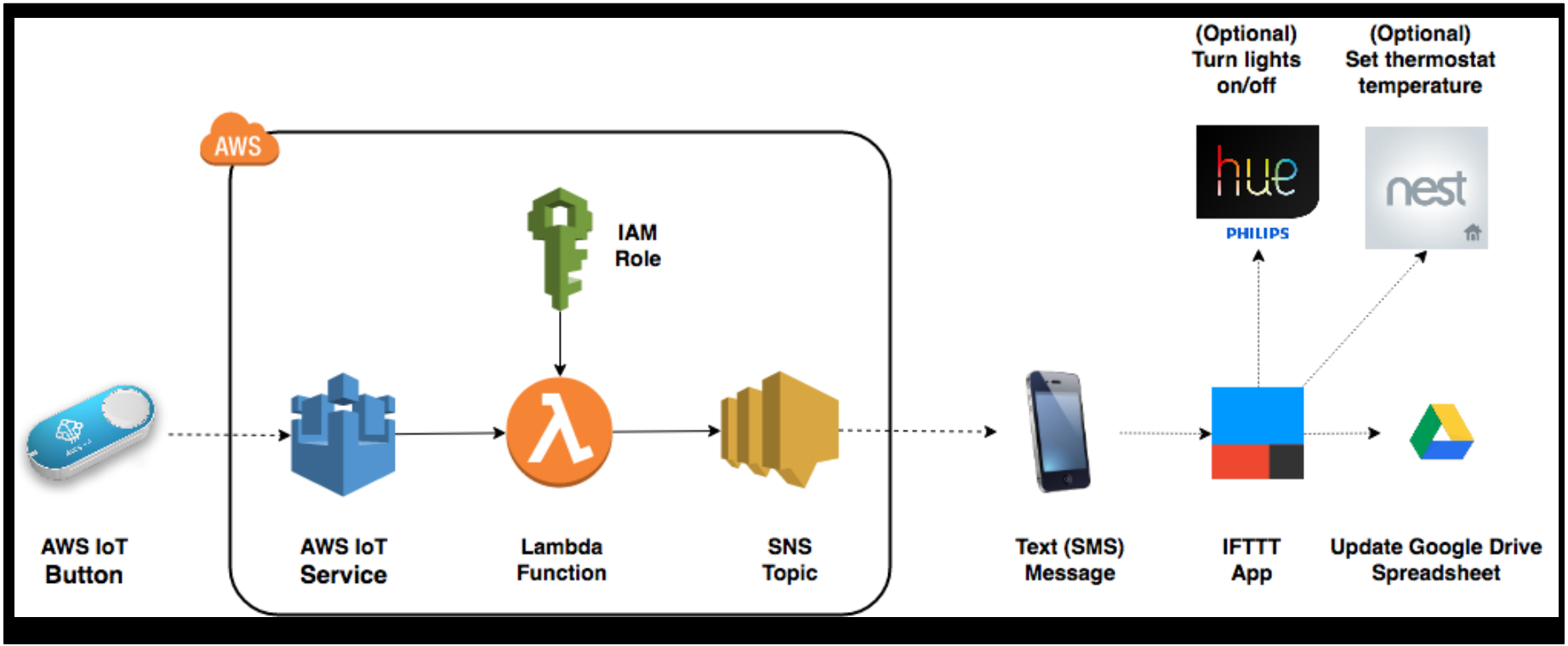
Use Case #1: Event-driven data processing (resize uploaded images)



Use Case #2: Serverless Web Applications (simple 3-tier app)



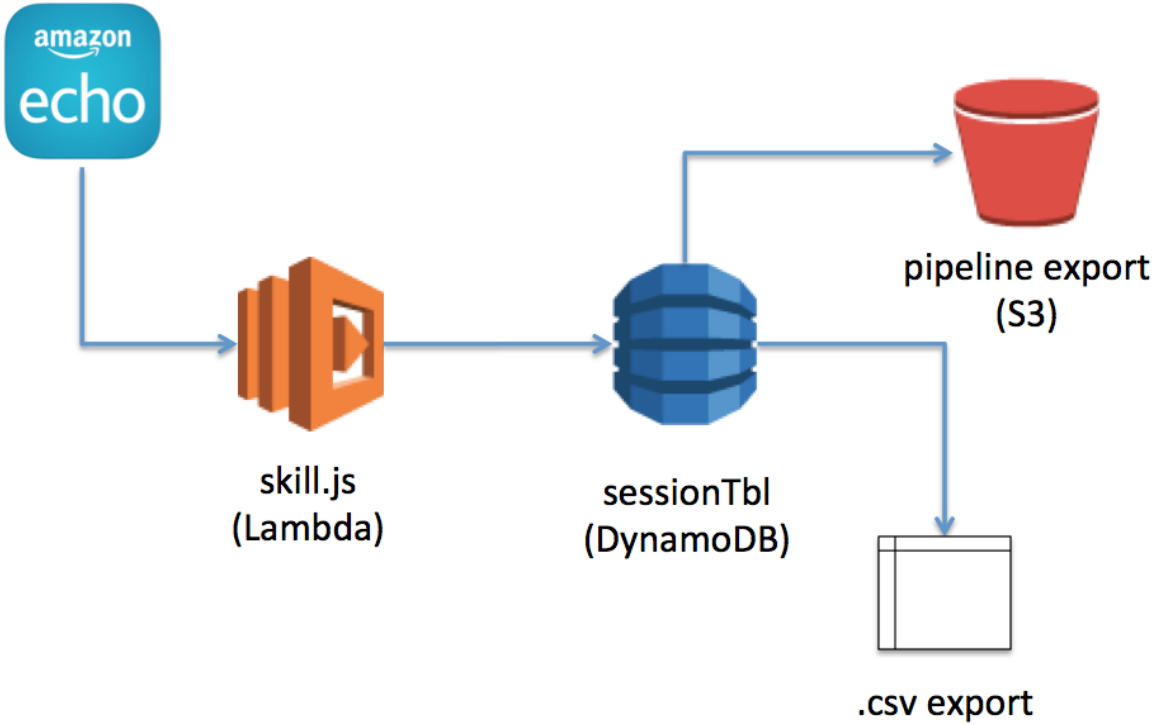
Use Case #3: Mobile & IoT Apps (Airbnb smart home)



Demo 2 – Shutting down EC2 VMs with an IOT Button

[if time allows]

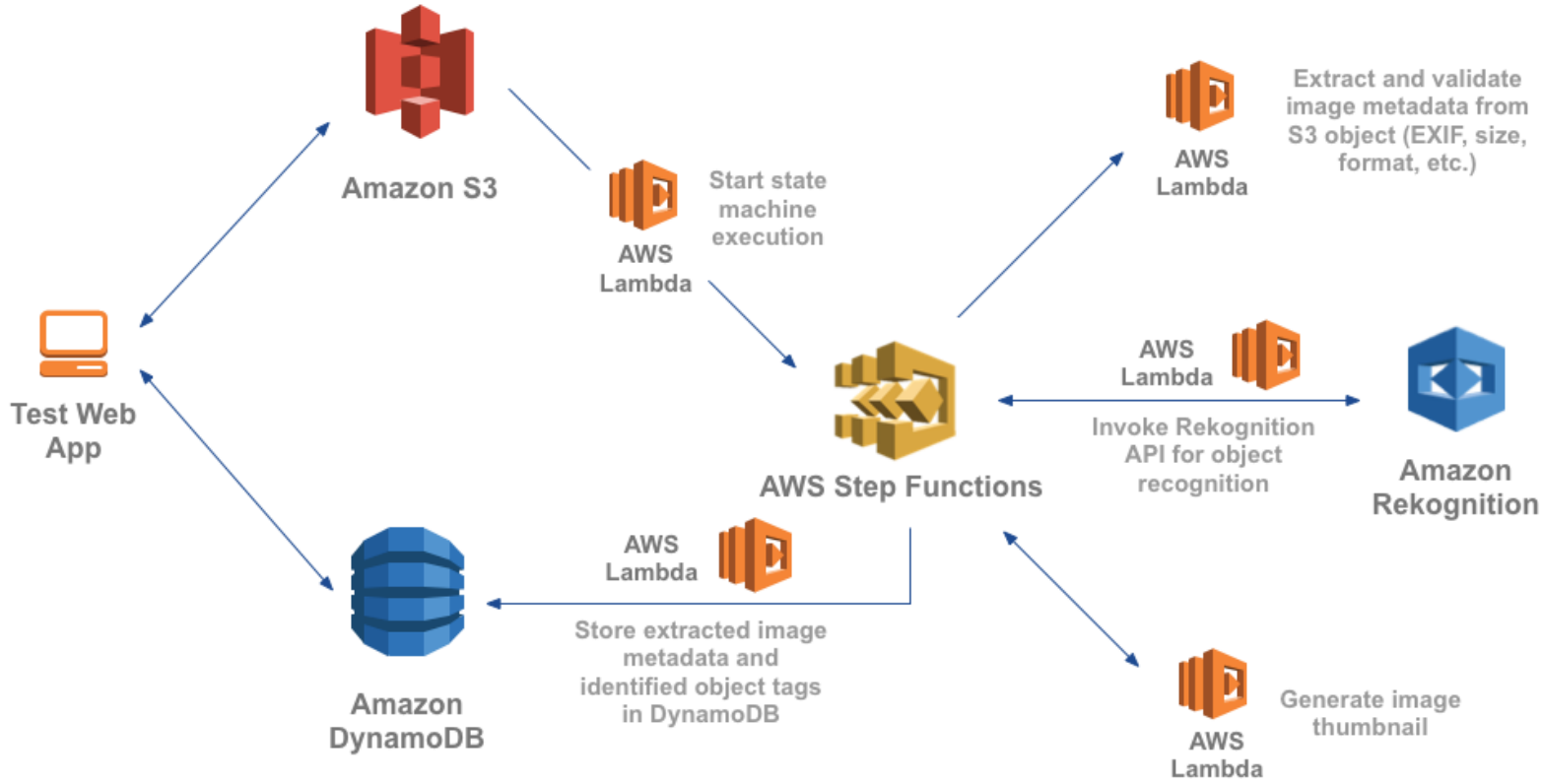
Use Case #4: Application Ecosystems (Alexa Skill)



Demo 3 – Running an Alexa Skill powered by Lambda

[if time allows]

Use Case #5: Event Workflow (image recognition & processing)



Demo 4 – Running a serverless image processor

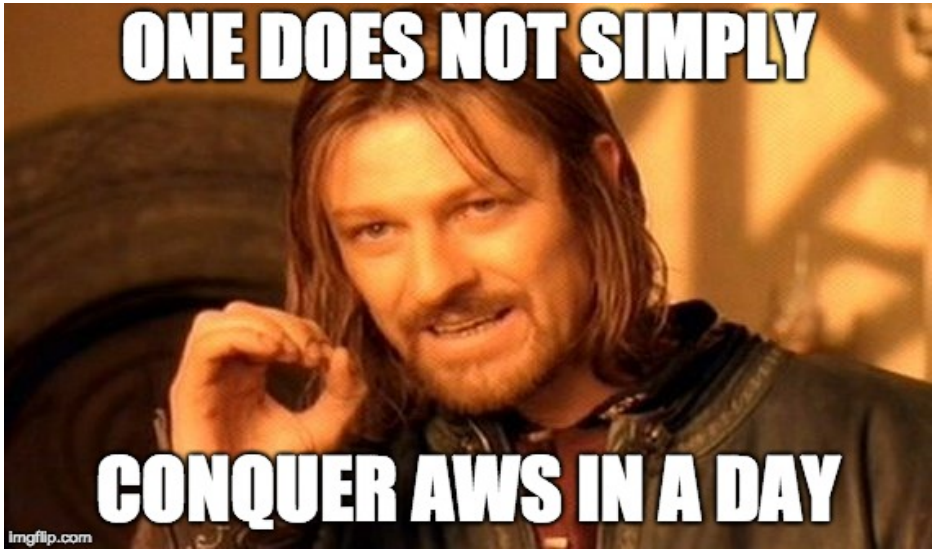
[if time allows]

Experiment with AWS!

- Run a quota of most services free
  - e.g. T2.micro VM 24/7 for a year
- Account requires credit card
- Ensure you set up billing alerts!

- We can create a limited account for you with a budget of $50
- Please send your email address to alan.forsyth@bristol.ac.uk

- Request GitHub Student Credits: https://education.github.com/pack



Terminology comparison AWS, Oracle and

Service	AWS	Oracle	Google
IaaS	Amazon EC2	Compute	Google Compute Engine
	Amazon EC2 Container Service	Container	Google Container Engine
PaaS	AWS Elastic Beanstalk	Application Container Cloud	Google App Engine
FaaS (Serverless)	AWS Lambda	Oracle Functions (Fn)	Google Cloud Functions
Key Value	Simple DB and Amazon DynamoDB	Oracle NoSQL Database	Google Cloud Datastore or Google Cloud Bigtable
Database	Amazon RDS	Oracle Database (DBaaS)	Google Cloud SQL and Spanner
Storage	Amazon S3	Object Storage	Google Cloud Storage Standard
	EBS	Block Storage	Persistent Disk
	Amazon EC2 Container Registry	Oracle Container Registry	Google Container Registry
Monitoring	Amazon CloudWatch	Application Monitoring	Google Cloud Monitoring and Google Cloud Logging
Networking	Amazon Elastic Load Balancing	Oracle Load BalancerÊ	Google Cloud Load Balancing
	Amazon Route 53	Dyn	Google Cloud DNS and Google Domains
Big data	Amazon EMR and AWS Data Pipeline	Big Data	Google Cloud Dataflow and Google Cloud Dataproc
	Amazon Kinesis and Amazon Simple Queue Service (SQS)	Messaging	Pub/Sub

Review

Example Exam Questions

- 1) Describe 2 benefits and 2 drawbacks to serverless technologies.
- 2) List the serverless architectural layers.
- 3) Explain the concept of ‘cold start’ in a serverless application.
- 4) “You should avoid AWS due to vendor lock-in!”. Discuss.
- 5) Name 4 parameters that must be specified when configuring a Lambda function.
- 6) Name three serverless use cases.

Questions?