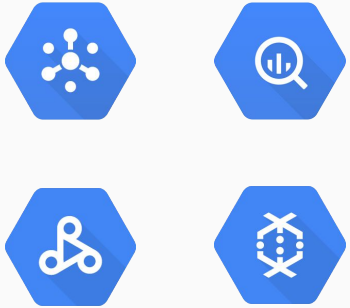
Serverless, FAAS and Event-driven architectures

Serverless

Serverless offerings

Storage



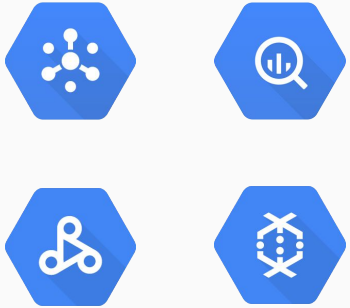
Services

Compute

**categorized this way only for better understanding

Serverless offerings

Storage

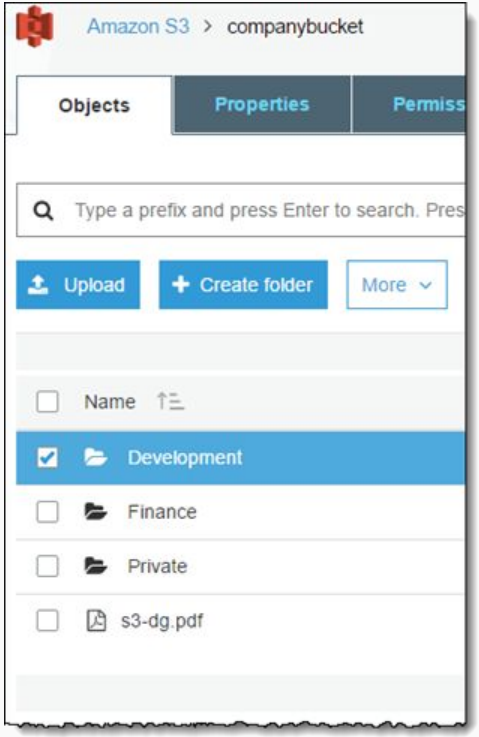


Services

Compute

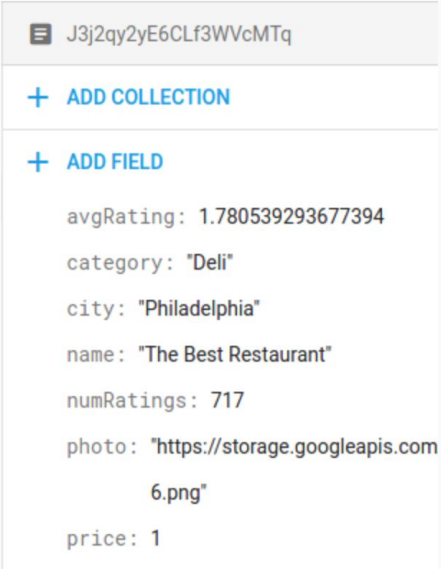
**categorized this way only for better understanding

Object Storage - AWS S3



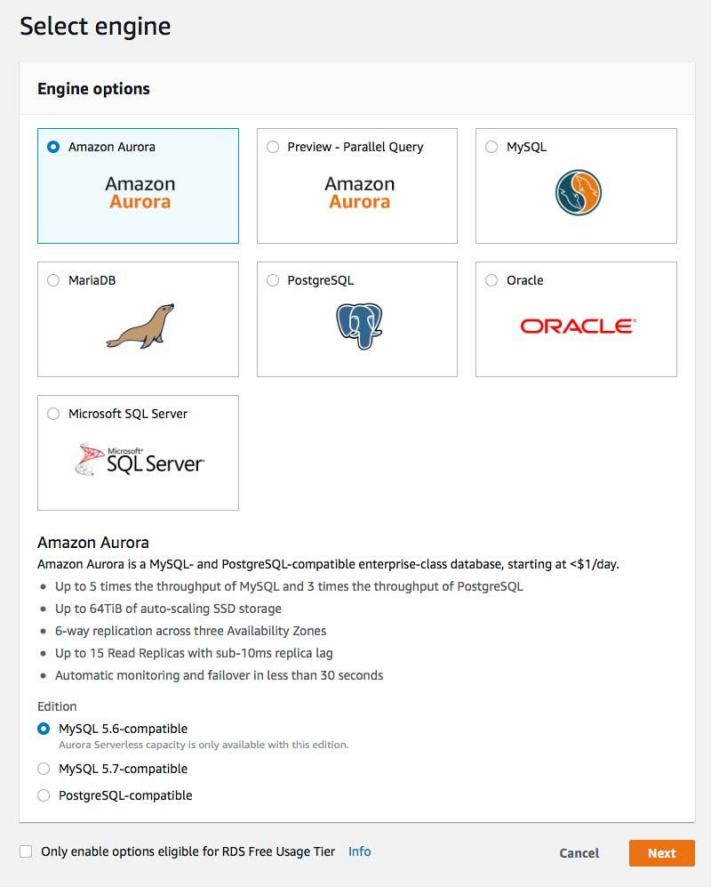
- No upfront provisioning
- Pay per amount of data stored & accessed
- SDK/HTTP for access (multiple languages)
- Data durability guarantee (99.999999999%)
- High availability (99.99%)
- Highly scalable (multi-petabytes)
- Store any kind of files (txt, json, pdf, jpg, mp4)
- Similar offerings in GCP, Azure etc.

NoSQL Storage - Firestore



- No upfront provisioning
- Pay per amount of data stored & accessed
- SDK for access (multiple languages)
- High availability (99.999%)
- Highly scalable (millions of documents)
- Store JSON document
- Offline, Sync, Caching etc.
- Similar offerings in AWS (DynamoDB), Azure etc.

SQL - AWS Aurora



- No upfront provisioning
- Pay per amount of data stored & accessed

  (Don't pay when not in use, even for RDBMS!)
- DB drivers or HTTP for access!
- Highly Resilient (6 way replication)
- Auto-Scales compute as well as storage
- Yes, RDBMS

Common theme

What was common in all these storage services?

- No provisioning upfront
- Pay per use
- Auto scale (both up & down)
- Resilient / Highly Available
- SDK for access
- Complete abstraction of servers (for developers)

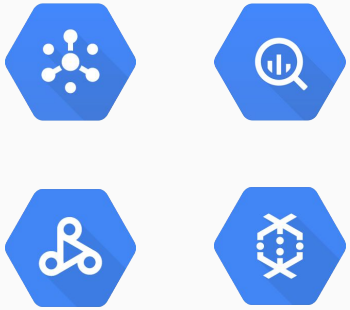
Common theme

Key characteristics of serverless

- No provisioning upfront
- Pay per use
- Auto scale (both up & down)
- Resilient / Highly Available
- Server management handled by provider

Serverless offerings

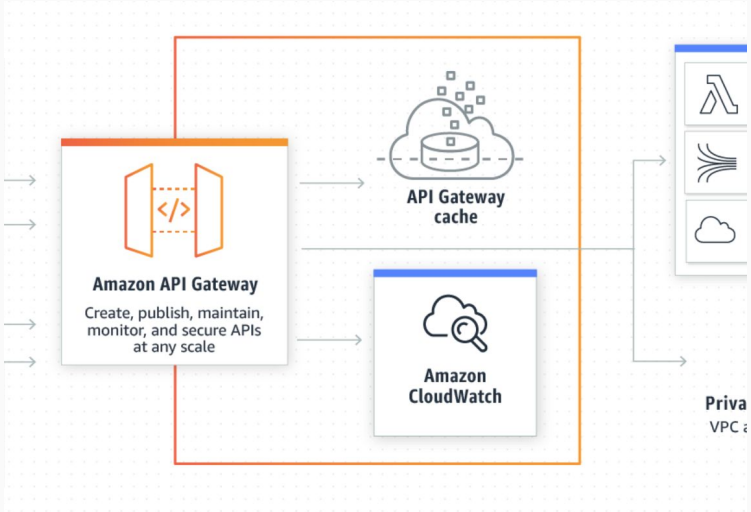
Storage



Services

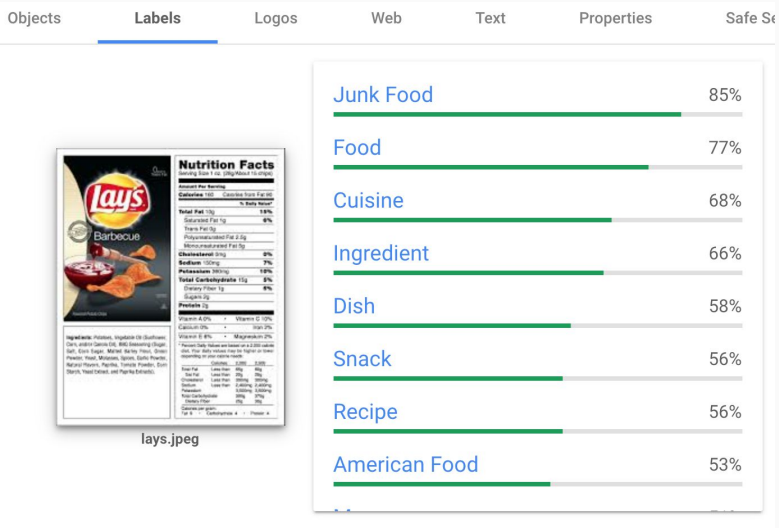
Compute

API Gateway as a service



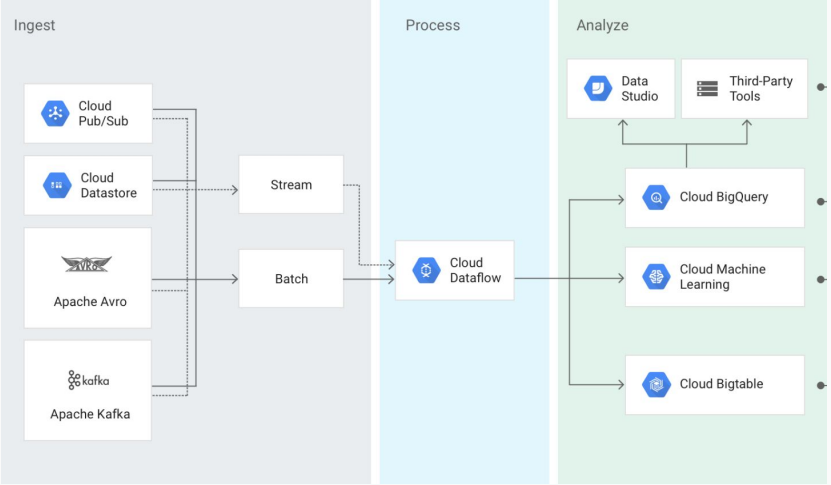
- No instance provisioning
- Auto-scale
- Pay only for requests passing through
- Feature rich - Security / Caching / Monitoring
- High availability
- Complete abstraction of servers (for developers)

Machine Learning as a service



- No provisioning of CPU/GPU/TPU
- Auto-scale
- Pay per image scanned
- Feature rich - Labels / Logo / Text etc.
- High availability
- Complete abstraction of servers (for developers)

Stream data processing as a service



- No provisioning
- Auto-scale
- Pay only for data stored & processed
- High availability
- Lightweight data stream processing
- Complete abstraction of servers (for developers)

Common theme

Key characteristics of serverless

- No upfront provisioning
- Pay per use
- Auto scale (both up & down)
- Resilient / Highly Available
- Server management handled by provider

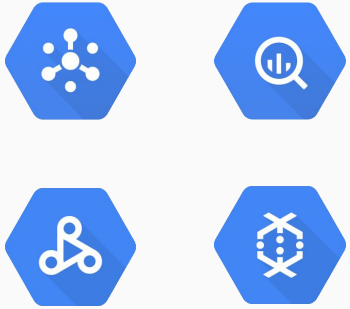
Alternate name for serverless?

"Fully managed service with no upfront provisioning which scales up automatically, scales down to zero, with pay per use billing model"

** doesn't look good on marketing pamphlets :)

Serverless offerings

Storage



Services

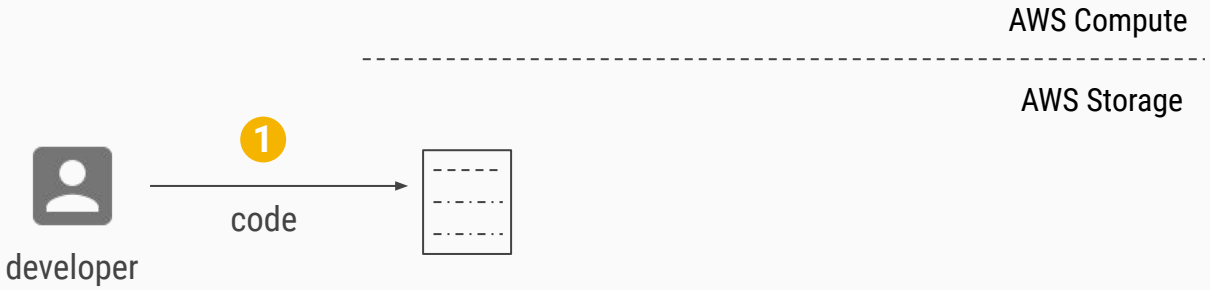
Compute

Function as a service

True Serverless

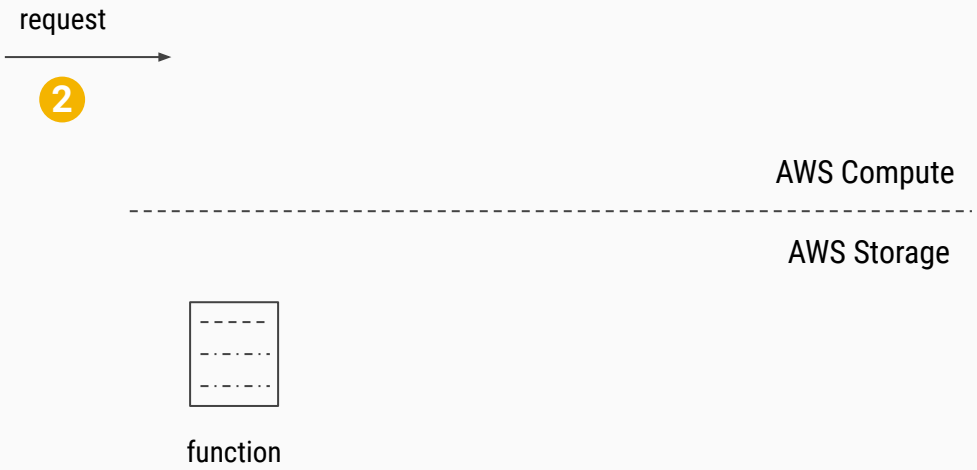
AWS Lambda

Launched in 2014



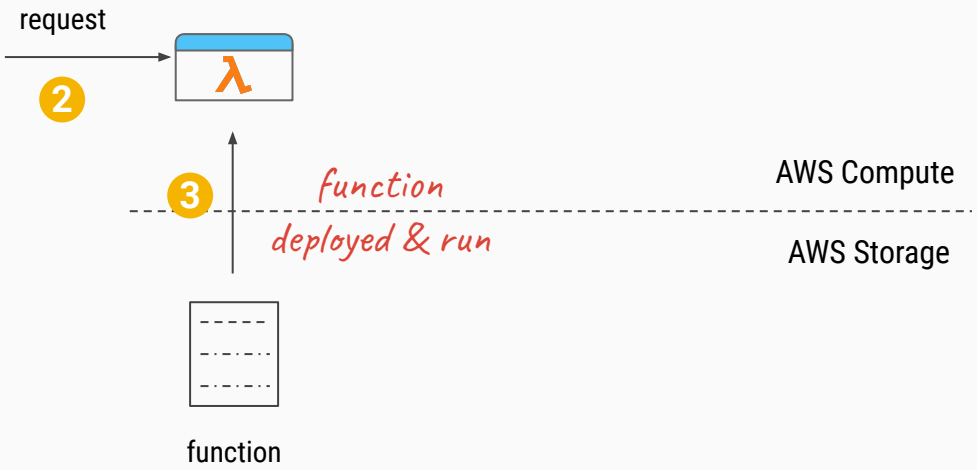
function to be executed on a request
(more granular than a microservice)

AWS Lambda

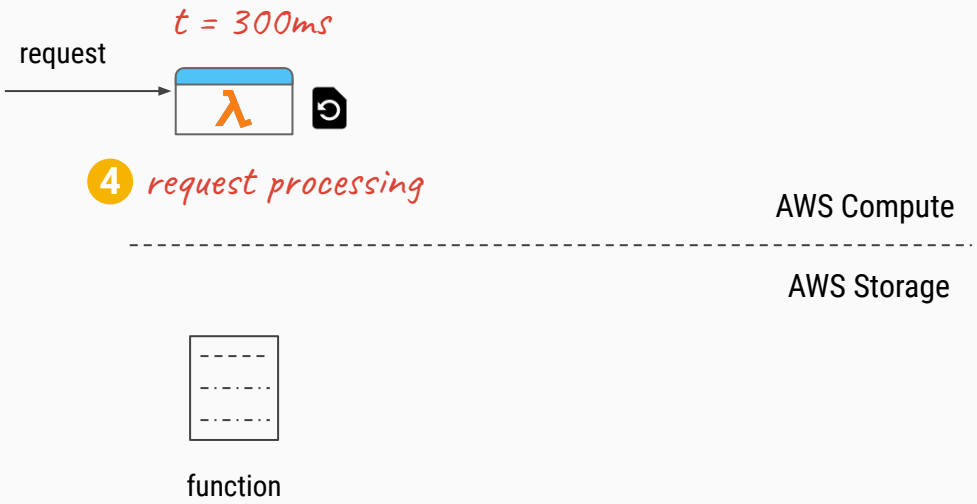


AWS Lambda

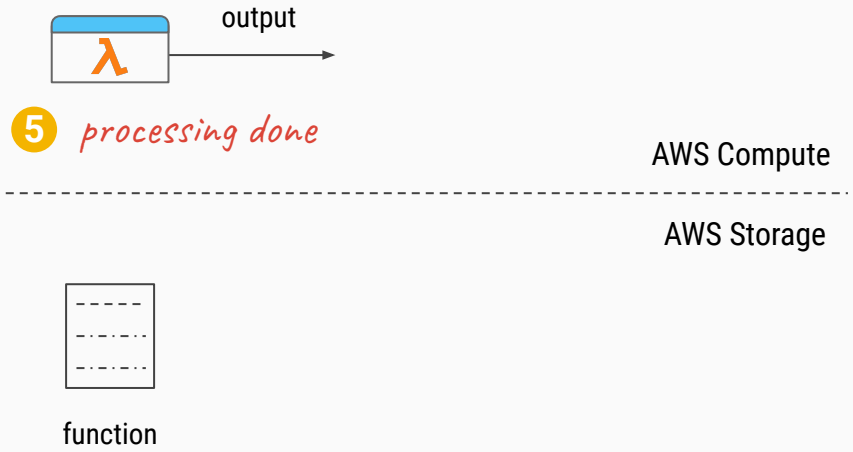
1. Auto provisioning



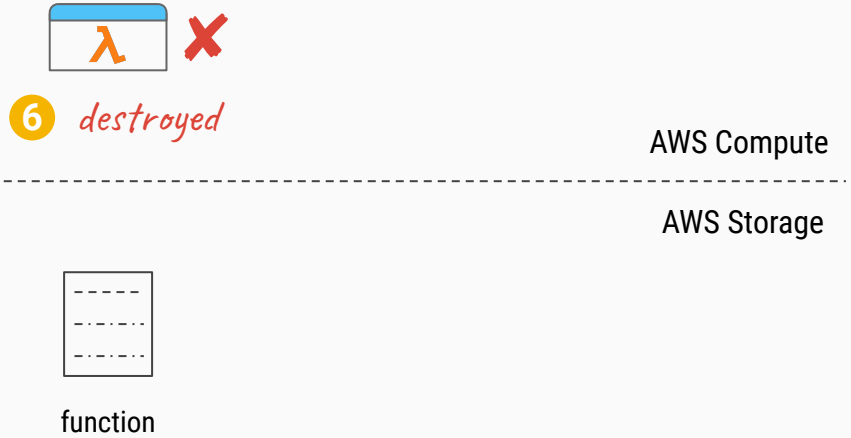
AWS Lambda


t = 300ms
request
4
request processing
AWS Compute
AWS Storage
function

AWS Lambda



AWS Lambda



Not immediately destroyed but good to understand

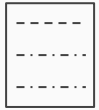
AWS Lambda

1. No upfront provisioning
2. Pay per use billing

Billed for 300ms
(duration your code was running in 100ms increments)

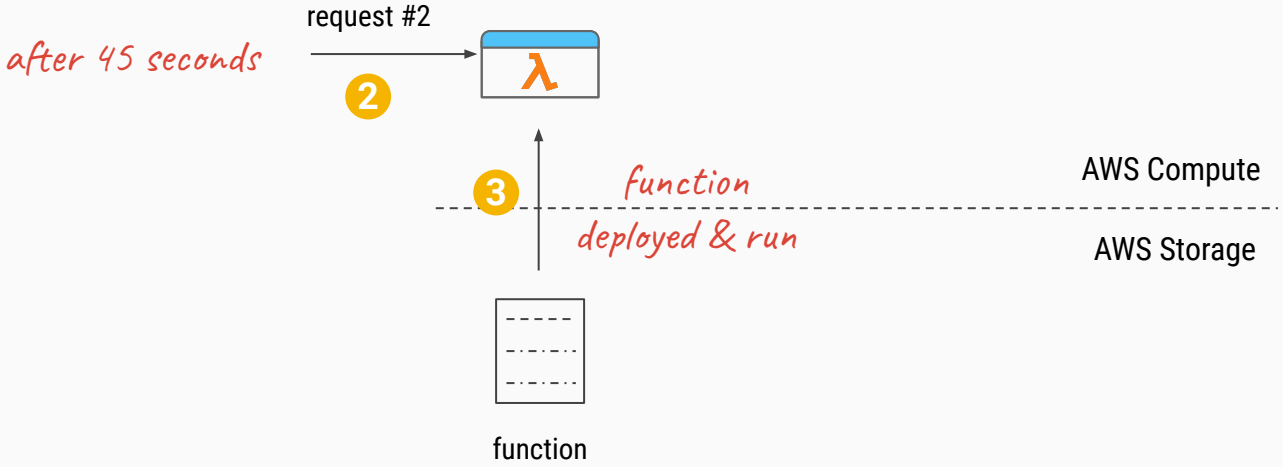
AWS Compute

AWS Storage

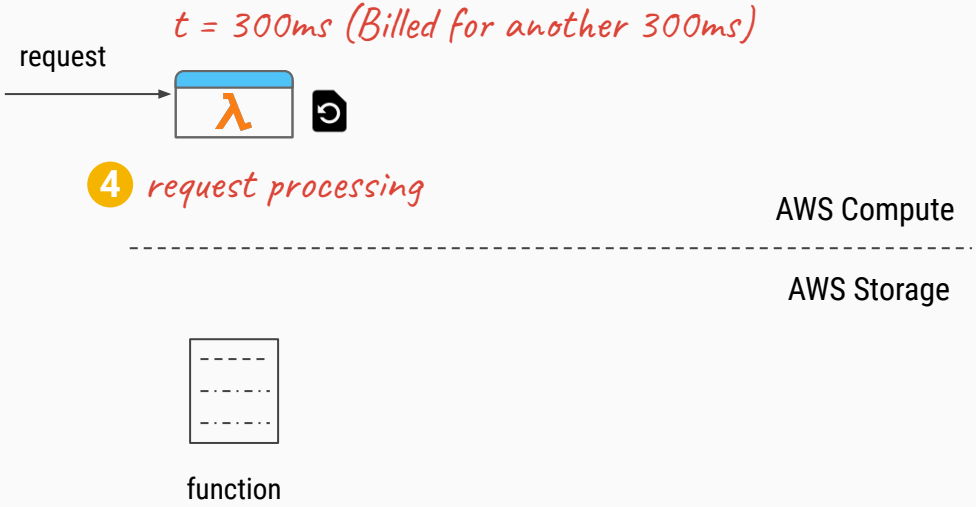


function

AWS Lambda

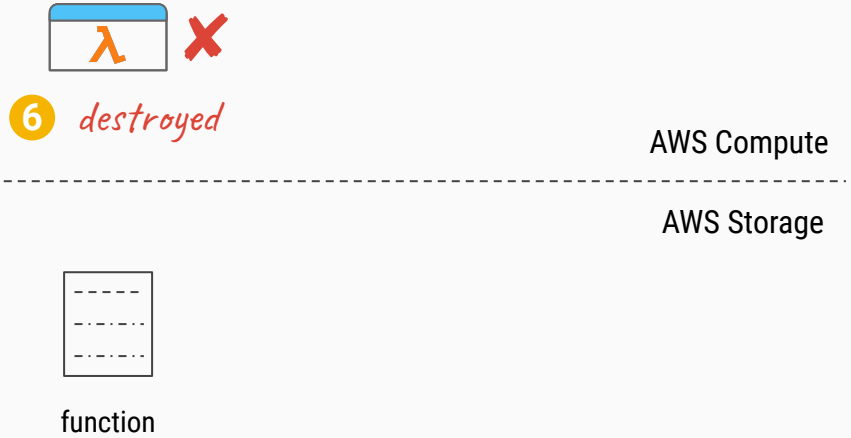

after 45 seconds
request #2
2
3
function
deployed & run
AWS Compute
AWS Storage
function

AWS Lambda


t = 300ms (Billed for another 300ms)
request
4 request processing
AWS Compute
AWS Storage
function

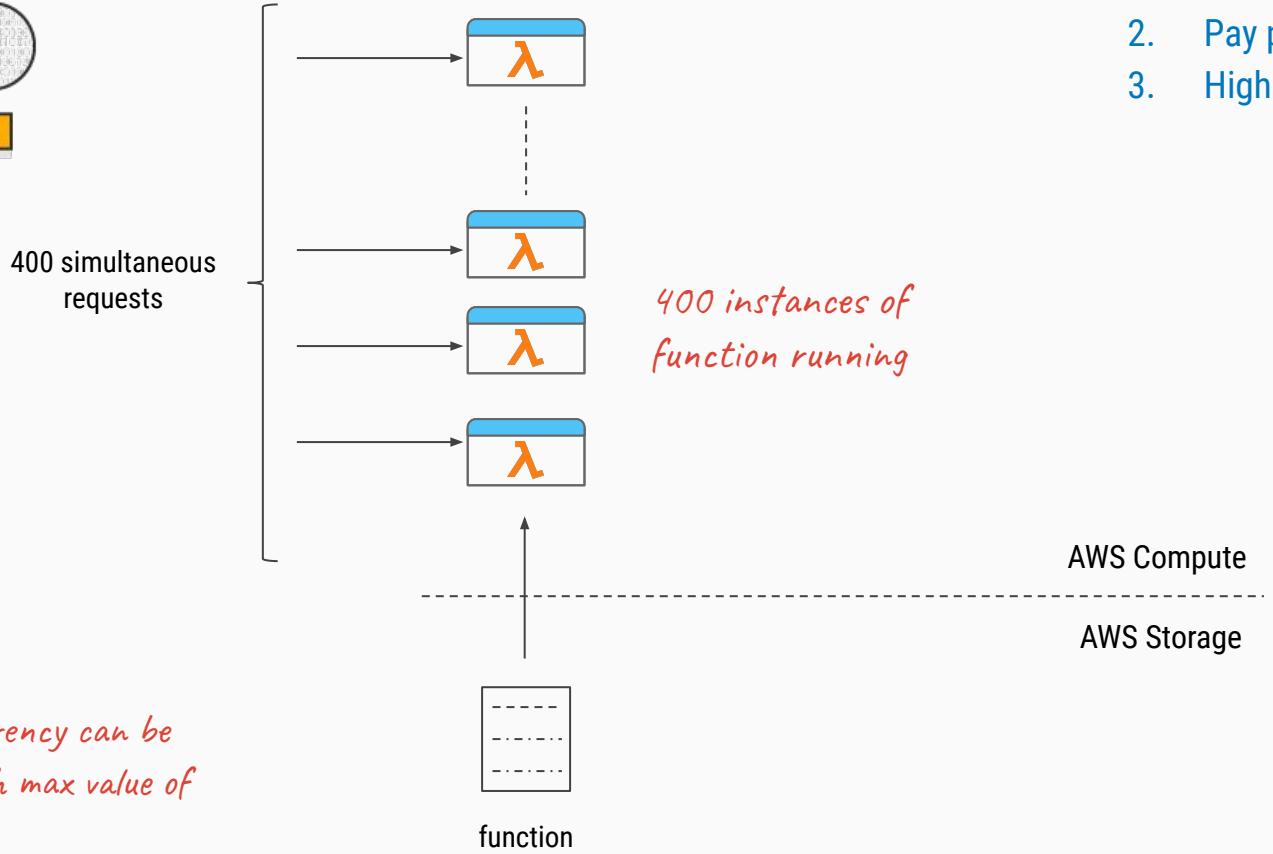
AWS Lambda

1. No upfront provisioning
2. Pay per use billing



FAAS Scaling

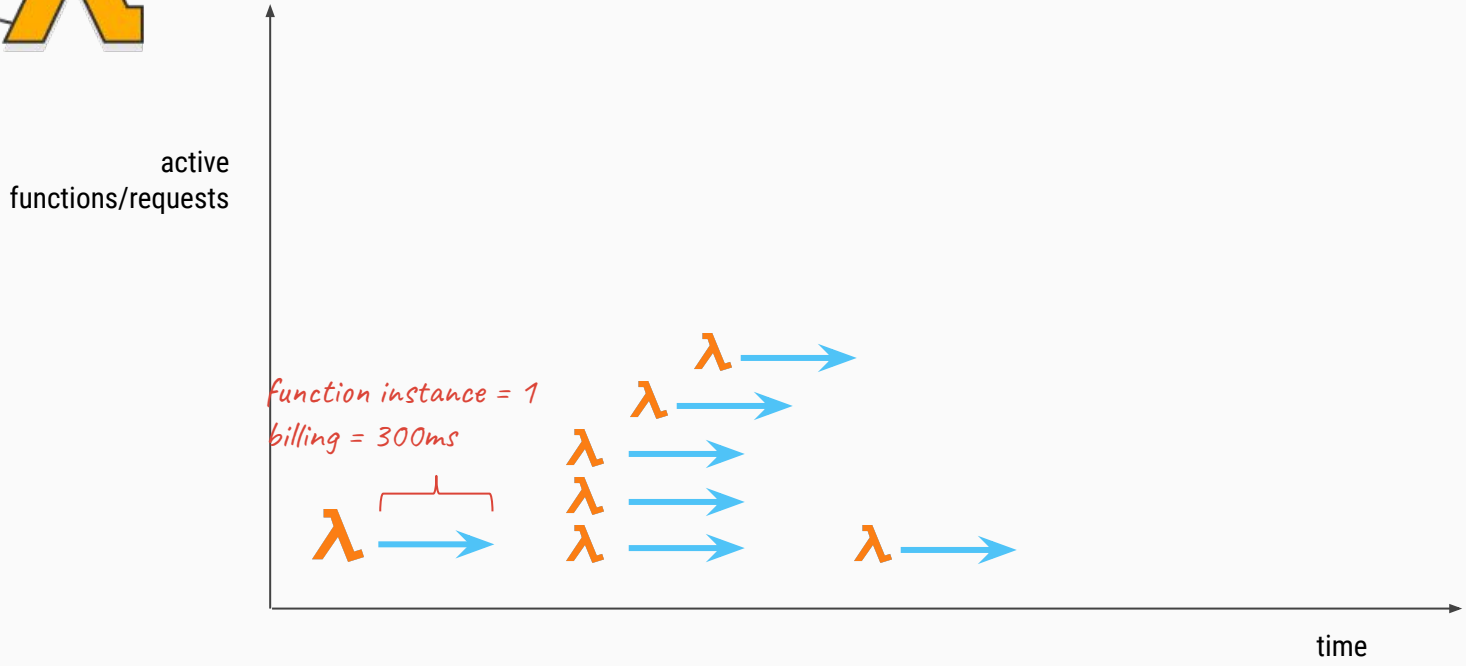
1. No upfront provisioning
2. Pay per use billing
3. Highly scalable



**This concurrency can be configured with max value of 1000

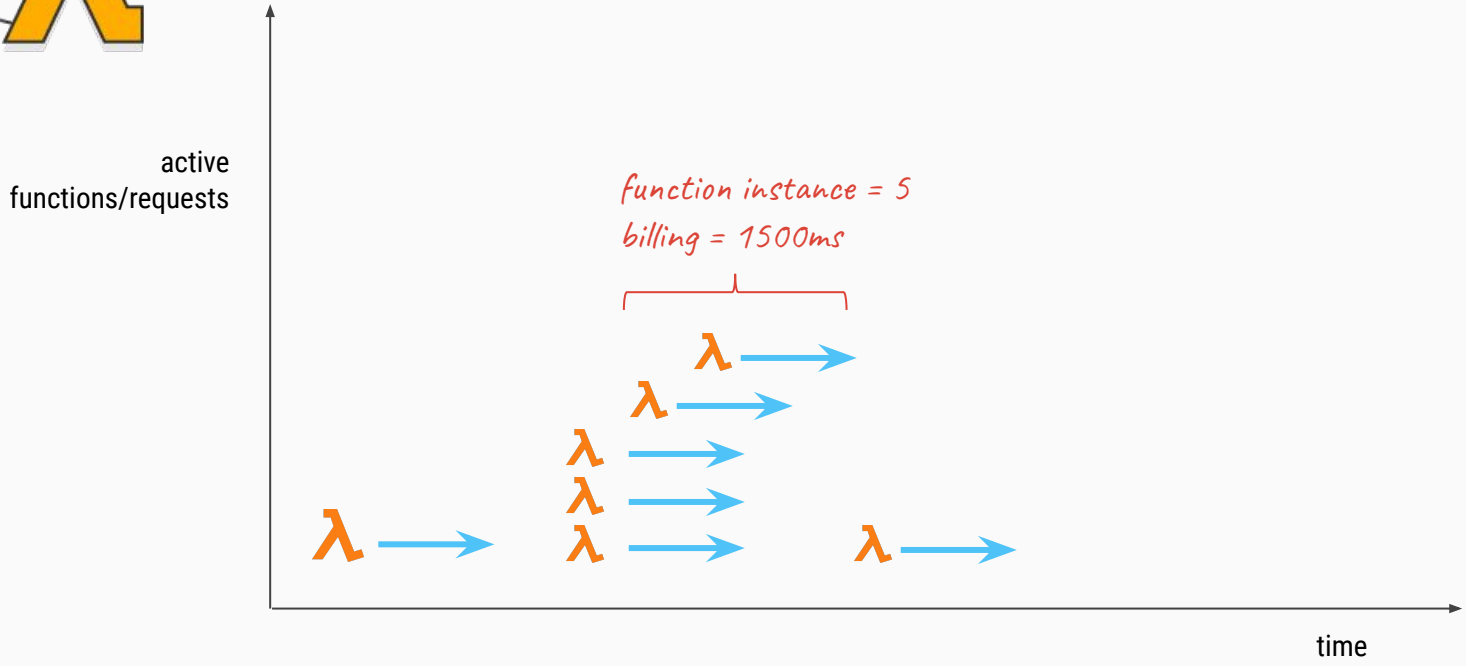
FAAS Scaling

1. No upfront provisioning
2. Pay per use billing
3. Highly scalable



FAAS Scaling

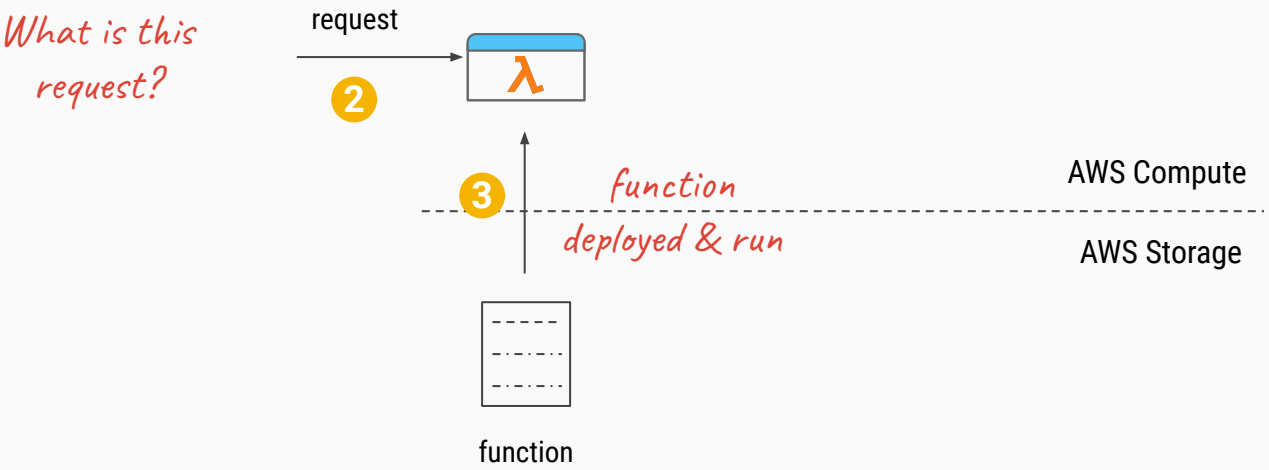
1. No upfront provisioning
2. Pay per use billing
3. Highly scalable



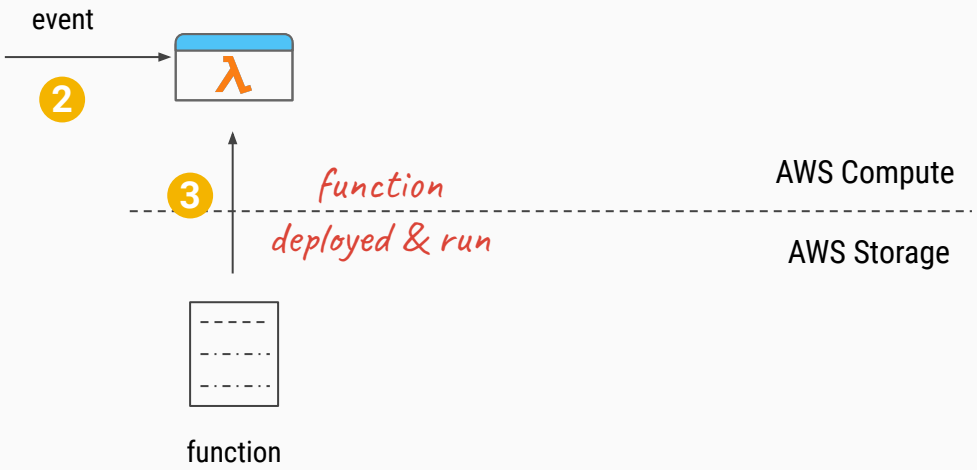
Function as a service features

- ✓ Provisioning
- ✓ Scaling
- ✓ Billing
- Events

Event driven

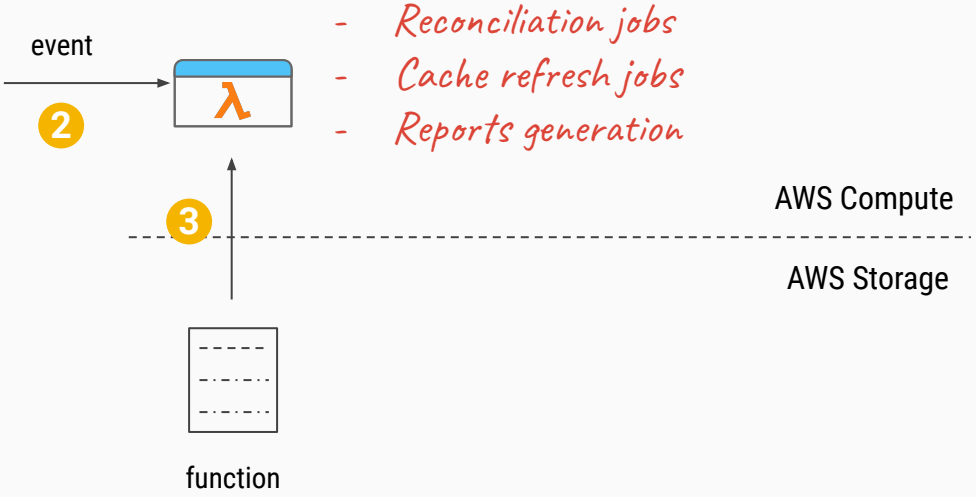


Event driven



Event driven

Scheduler
(run each day at 5pm)



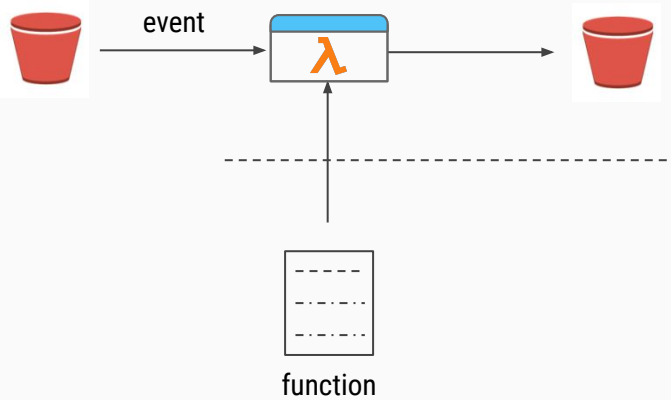
- *Reconciliation jobs*
- *Cache refresh jobs*
- *Reports generation*

AWS's responsibility to trigger this event

Event driven

Resize and make copies for multiple devices and store in another S3 bucket

S3 event
(triggered on image upload to a bucket)



Output resized images stored in another bucket

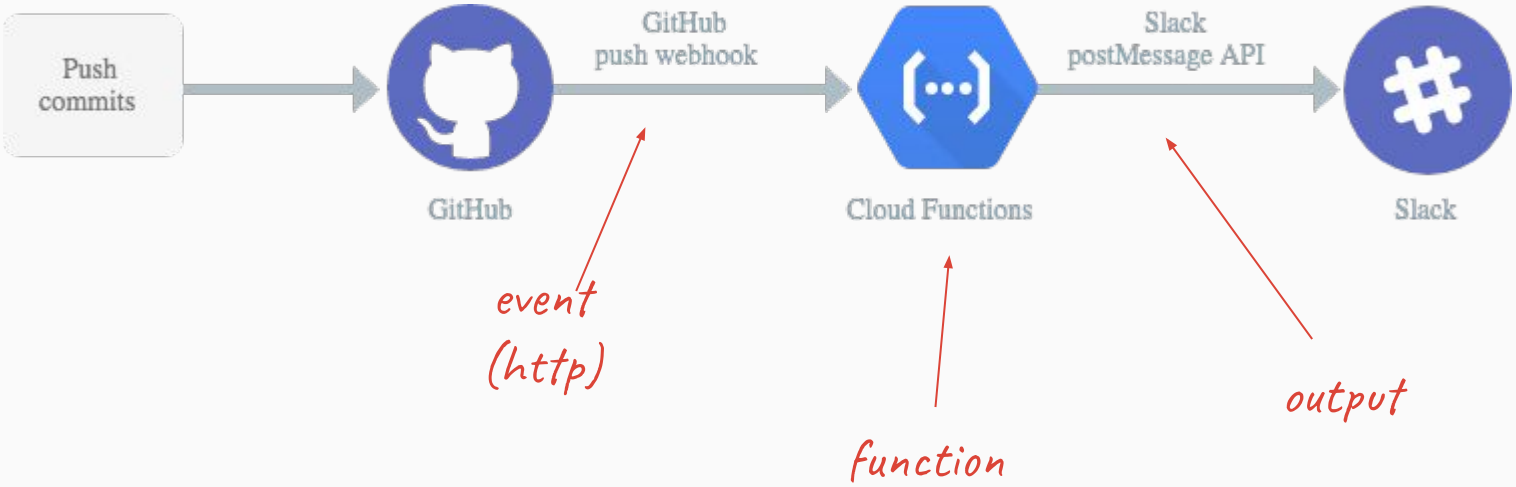
AWS Compute

AWS Storage

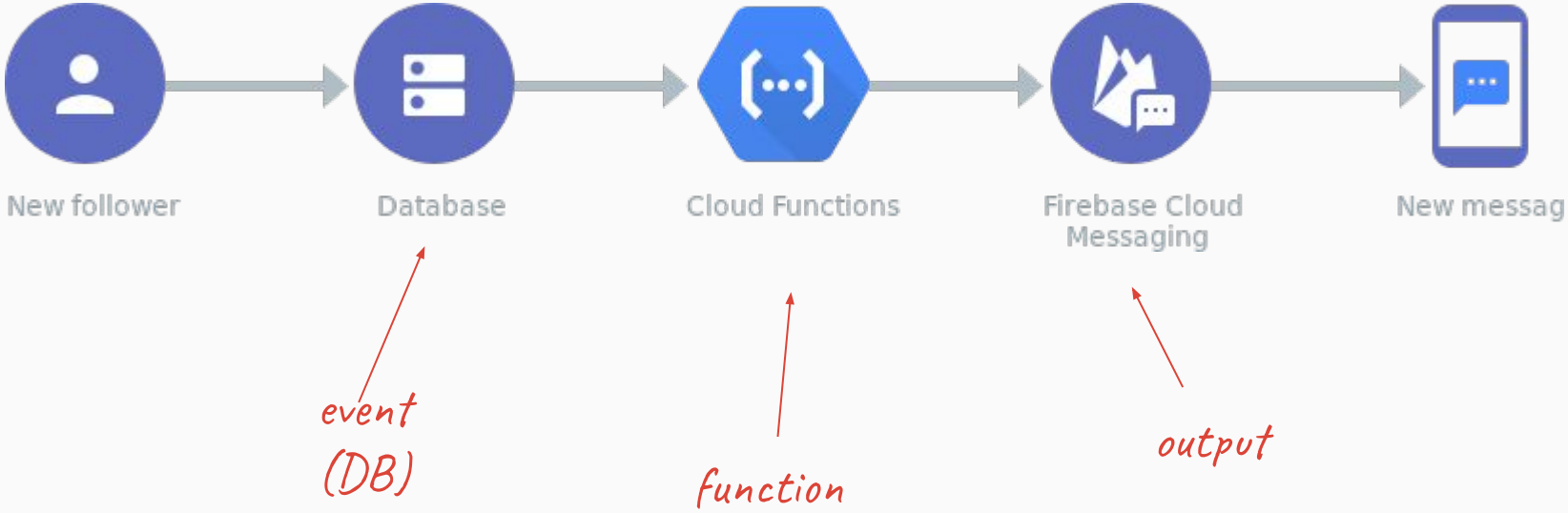
Sample S3 event JSON passed to the function


```
{
   "Records":[
      {
         "eventVersion":"2.1",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"127.0.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/JRWeUWerMUE5JgHv
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
               "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
               "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
}
```

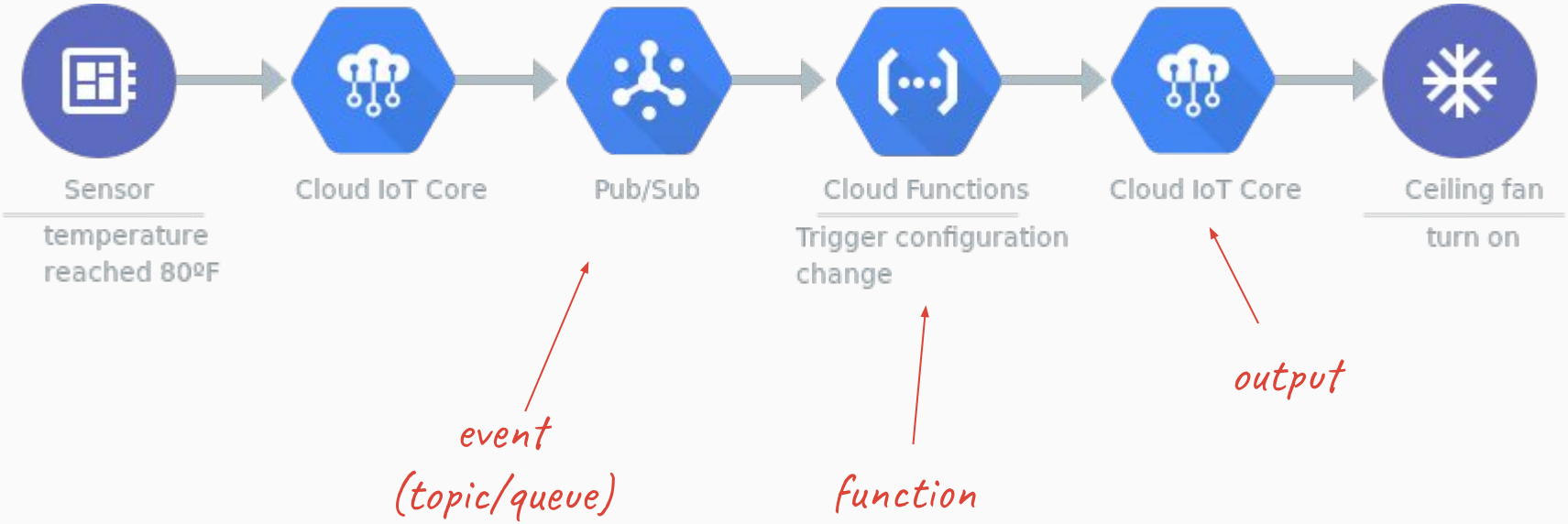
Sending Slack message on every Git push



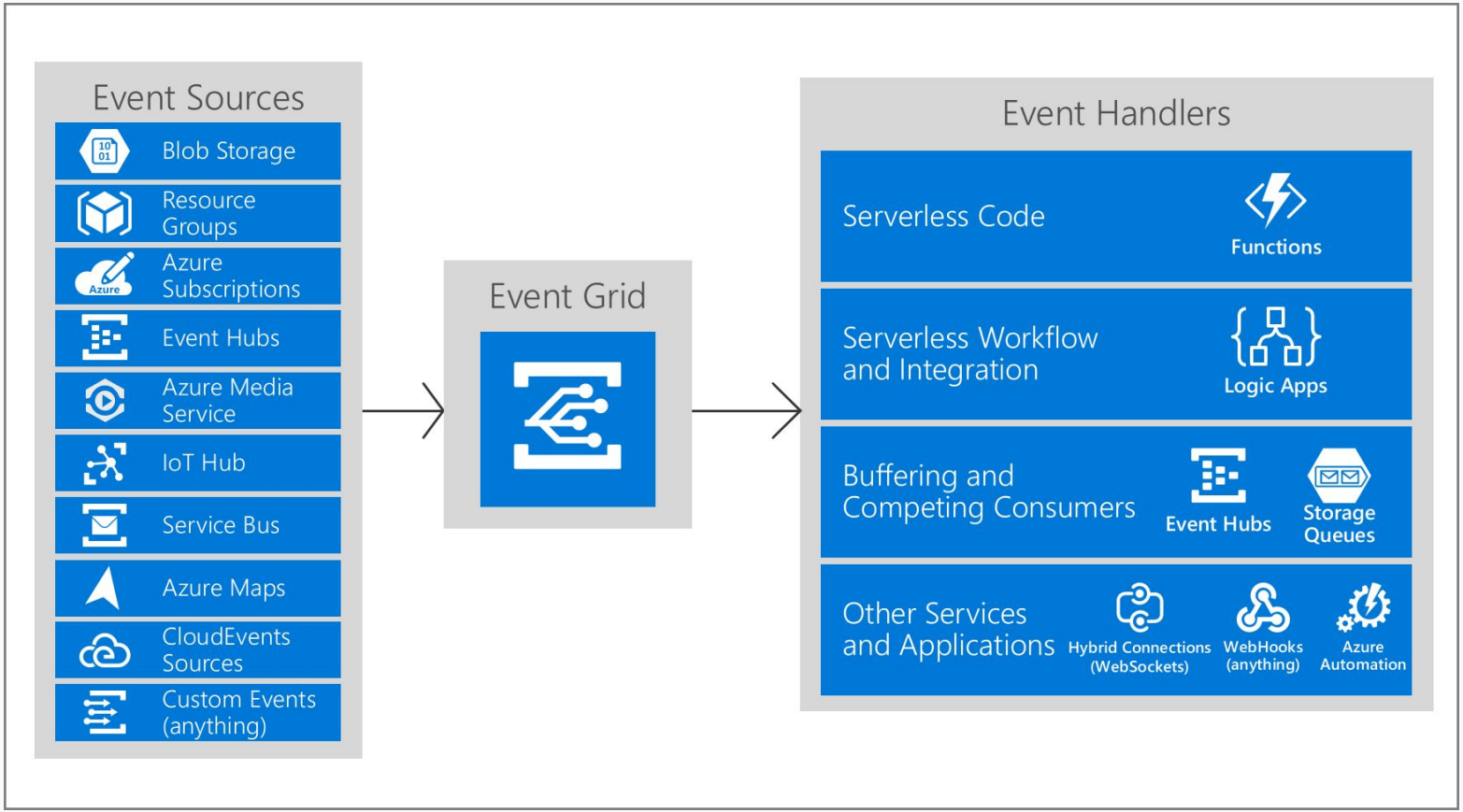
Push notification to user on a new follower (insert into DB)



IoT devices


Sensor
temperature reached 80ºF
Cloud IoT Core
Pub/Sub
Cloud Functions
Trigger configuration change
Cloud IoT Core
Ceiling fan
turn on
event
(topic/queue)
function
output

Event Sources



Function as a service features

- ✓ Provisioning (auto)
- ✓ Scaling (auto)
- ✓ Billing (100ms)
- ✓ Event driven

Ideal Workloads

Provisioning (auto)

Scaling (auto)

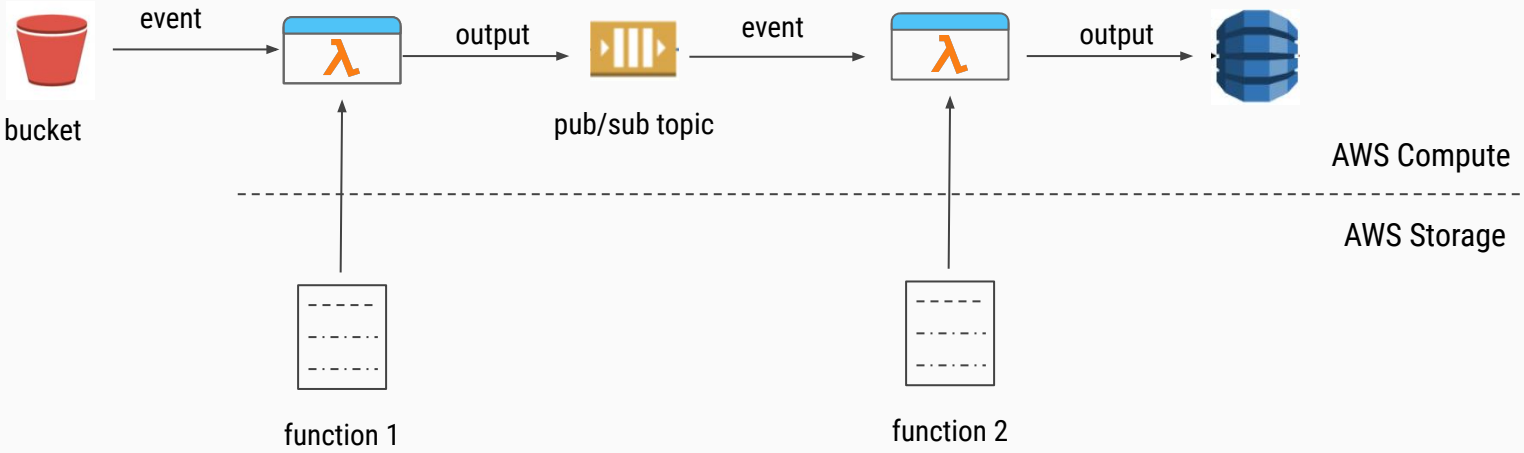
Billing (100ms)

Event driven

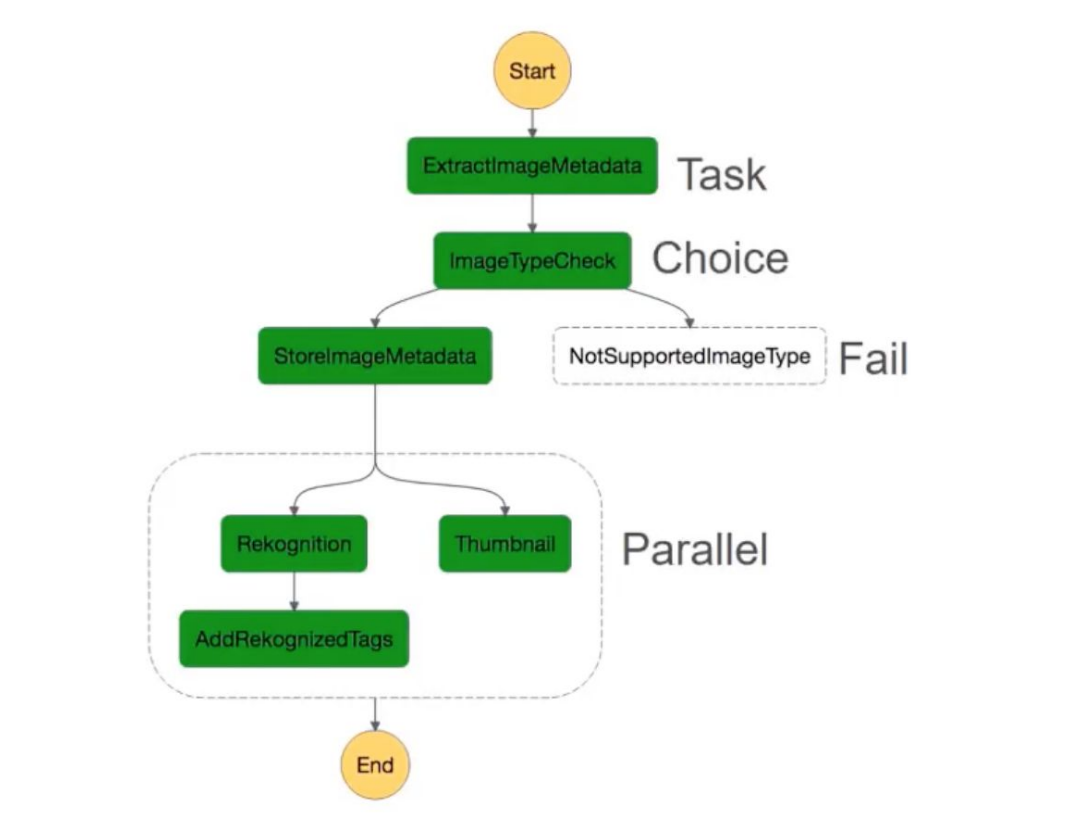
- ✓ Stateless**
- ✓ Parallel
- ✓ Asynchronous**
- ✓ Short-lived
- ✓ Unpredictable / Bursty

Demo

Event driven architecture (flow of events)



AWS Step functions



Azure has similar feature called Durable functions

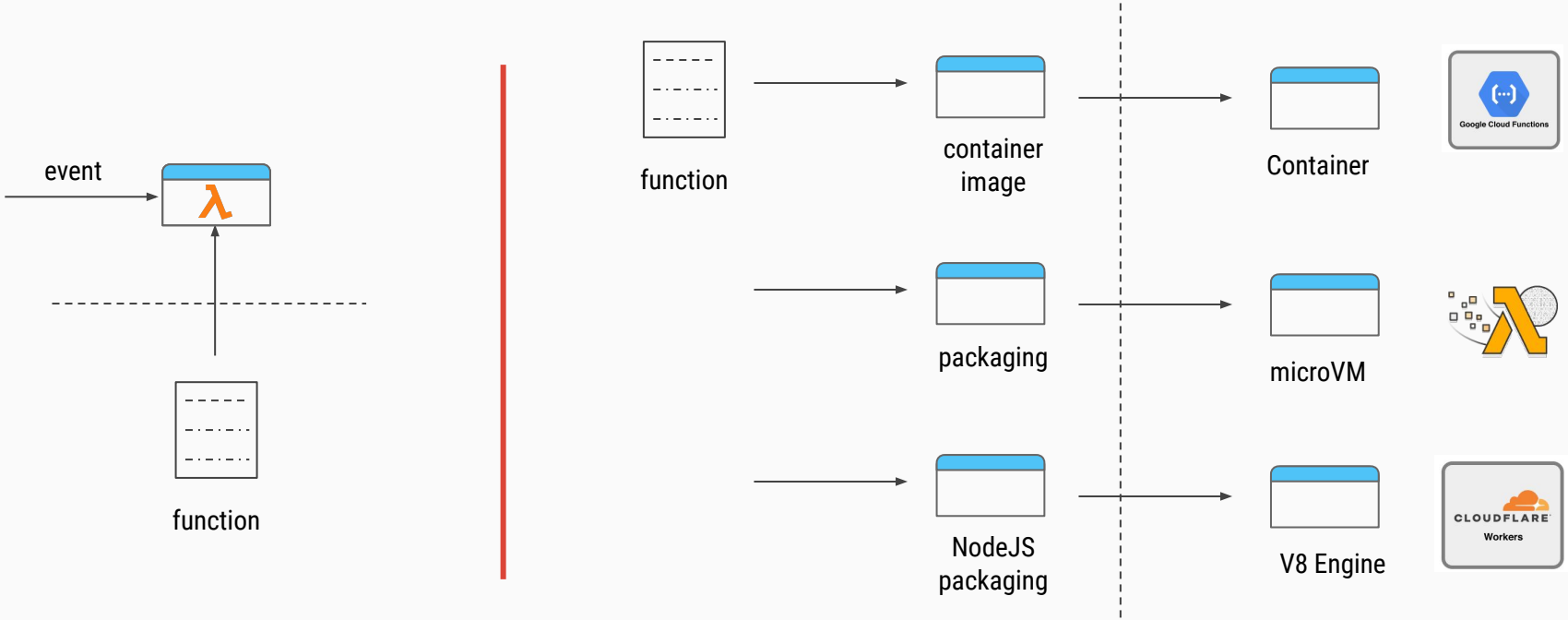
Providers

Major cloud providers

On-prem versions

Specialized on-cloud versions

Behind the scenes



Abstracted from developers

Real World Use Cases

Uses AWS Lambda to build rule-based self-managing infrastructure (create EC2 instances, monitor infra, validate backups etc), and automate the encoding process of media files

Uses AWS Lambda with Kinesis for data ingestion pipeline for analytics service with ingestion rate of 4,000 events per second (expected to reach 10,000 by year end).

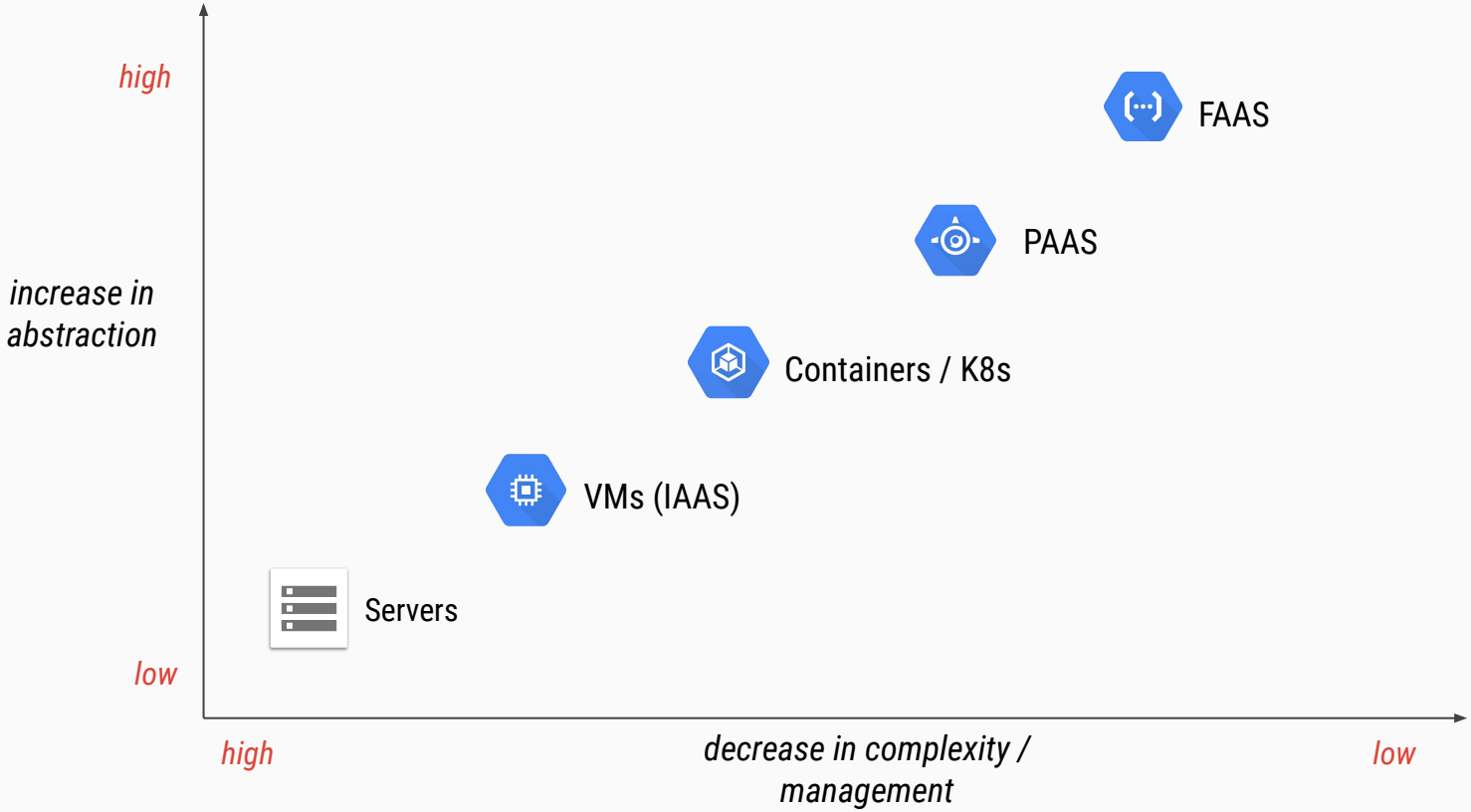
FINRA analyzes 75 billion market events every day to identify fraud and insider trading using AWS Lambda.

Uses AWS Lambda to process in-game screenshots (Creates thumbnails, adds watermark to the images and shares with other players).

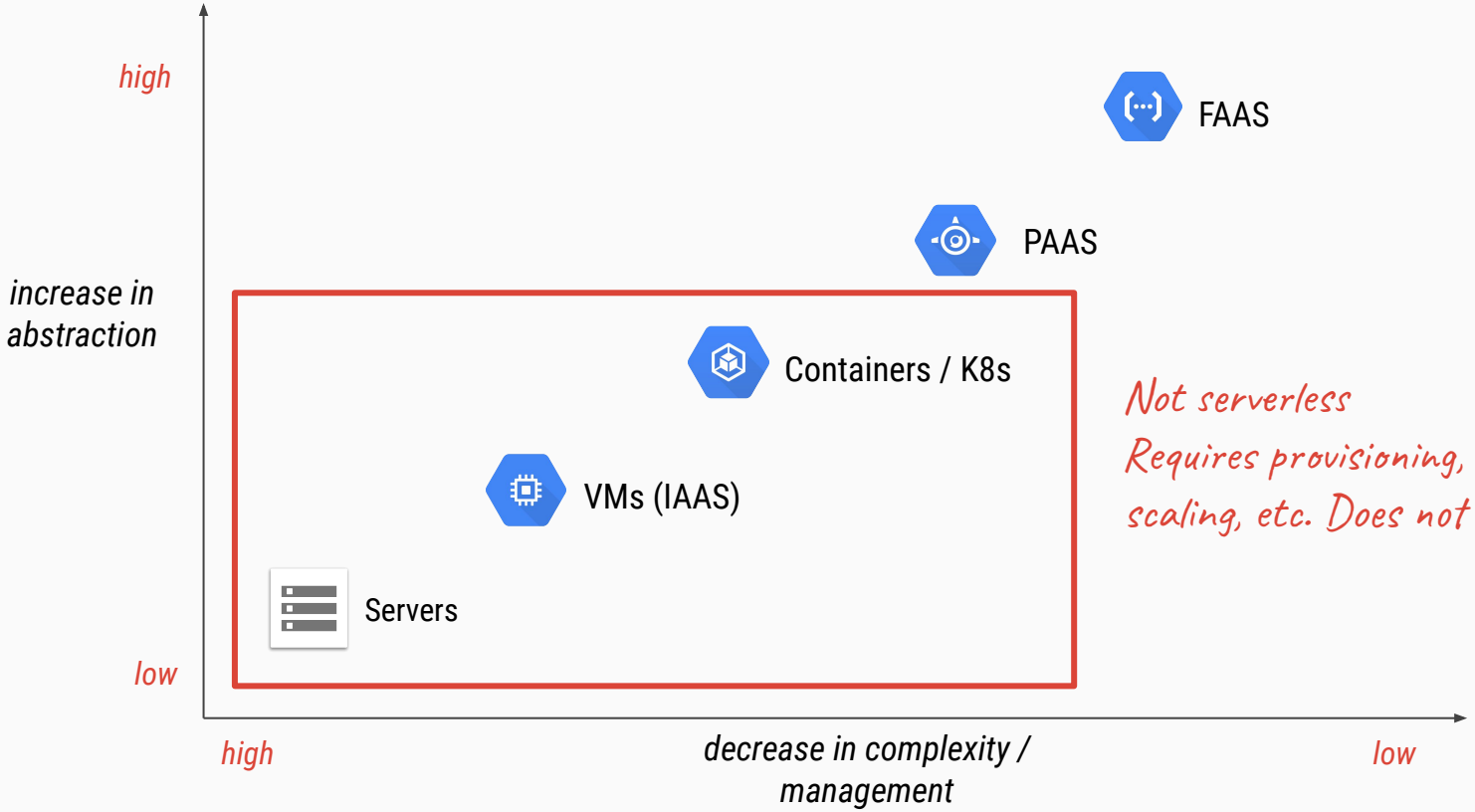
Uses AWS Lambda and S3 to create election county results application which runs on 1 day a year, seeing 3,000 concurrent users. Total bill was $25.

Computing Abstractions

Compute offerings



Compute offerings



Not serverless
Requires provisioning, maintenance, scaling, etc. Does not scale down to 0

Platform as a service

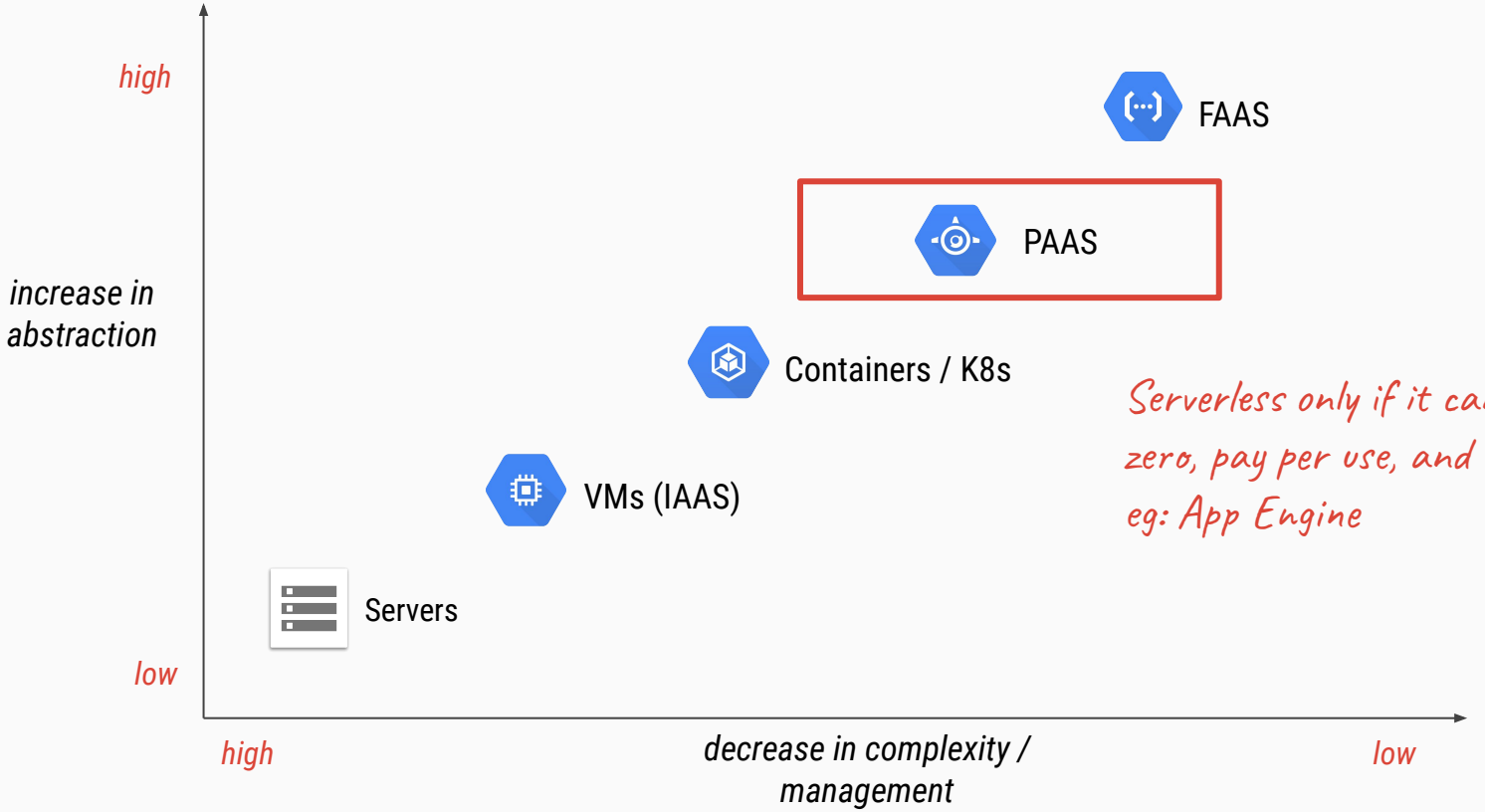
App Engine

Heroku

- Deploy your code (WAR file)
- Auto-scales
- Pay per *hours* of instance running time
- Stops when not in use
- Fault tolerant (ensures instance remains up)
- No server maintenance by developers
- Services like Caching, Queues, Cron, etc.
- Multiple language runtimes available
- Cold starts take few seconds

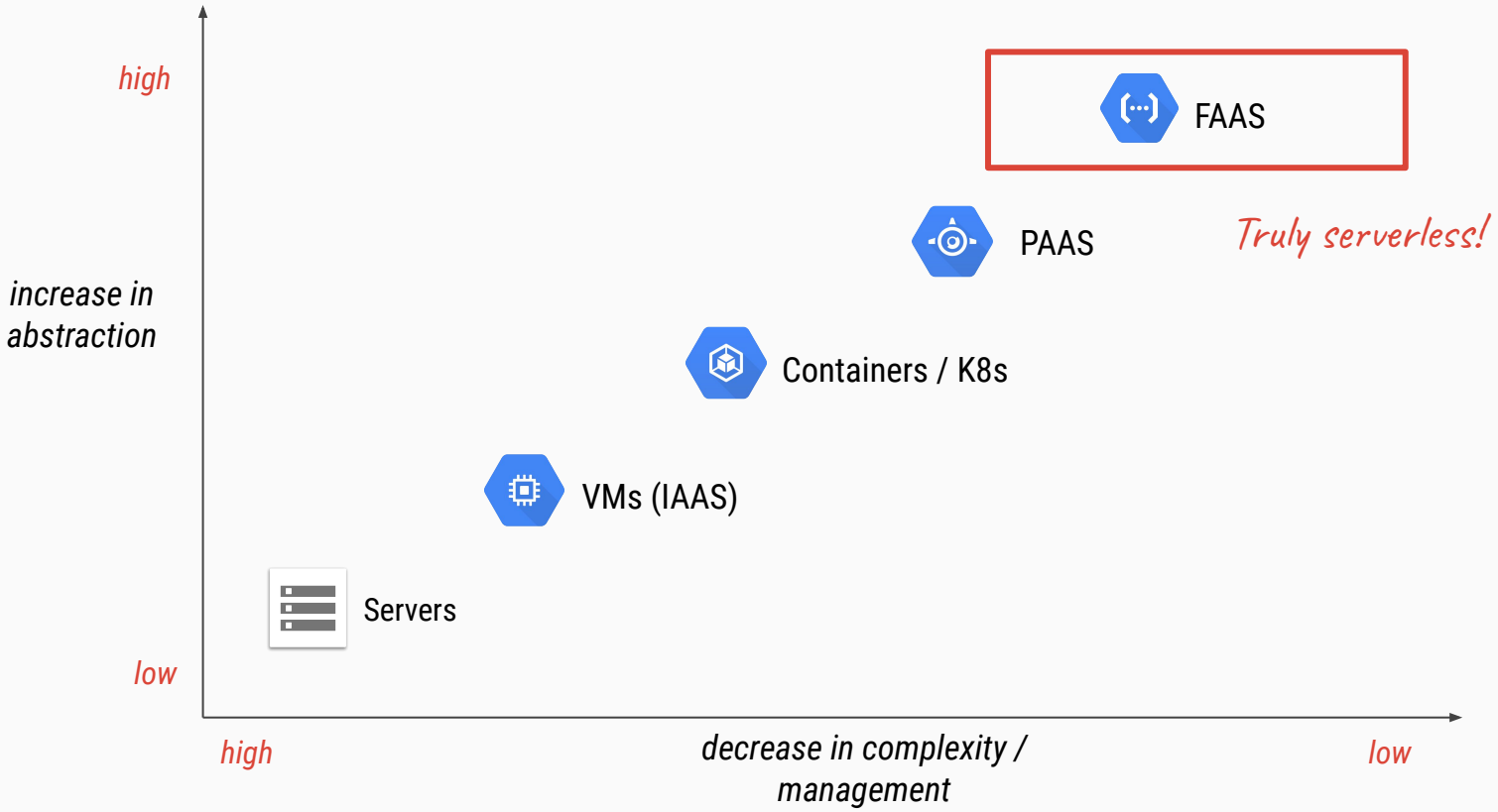
**Debate whether its truly serverless

Compute offerings



Serverless only if it can scale down to zero, pay per use, and auto-scale
eg: App Engine

Compute offerings



Thank you!

FAQ

- How is the tooling support?
- Can I version the functions?
- Can I do traffic splitting, blue/green etc?
- Can I make it work multi-cloud?

Simple use case

Virtual Machines (IAAS)	Containers / K8s (CAAS)
App Engine (PAAS)	FAAS

- Provision the instance
- Setup / Configure / Deploy
- Patching / Resiliency / Maintenance
- 24/7 running (cost + inefficient)

No an ideal solution

```
@SpringBootApplication
@EnableScheduling
public class DemoApplication {

        public static void main(String[] args) {
                SpringApplication.run(DemoApplication.class, args);
        }

        @Scheduled(cron = "0 17 * * *")
        public void reconcile() {
                // ...
                // ...
        }
}
```

Simple use case

Virtual Machines (IAAS)	Containers / K8s (CAAS)
App Engine (PAAS)	FAAS

- Build / Setup / Configure / Deploy
- Configure for resiliency
- 24/7 running (cost + inefficient)
- No ease of use for developers

No an ideal solution

```java
@SpringBootApplication
@EnableScheduling
public class DemoApplication {

	public static void main(String[] args) {
		SpringApplication.run(DemoApplication.class, args);
	}

	@Scheduled(cron = "0 17 * * *")
	public void reconcile() {
		// ...
		// ...
	}
}
```

Simple use case

Virtual Machines (IAAS)	Containers / K8s (CAAS)
Cloud Foundry (PAAS)	FAAS

- Deploy code
- Auto maintenance / monitoring
- 24/7 running (cost + inefficient)

Better than previous, but can we do better?

```
@SpringBootApplication
@EnableScheduling
public class DemoApplication {

        public static void main(String[] args) {
                SpringApplication.run(DemoApplication.class, args);
        }

        @Scheduled(cron = "0 17 * * *")
        public void reconcile() {
                // ...
                // ...
        }
}
```

*** App Engine is PAAS which scales down to 0*

Compute offerings

Physical Servers Virtual Machines (IAAS)	- Provisioning - Server Maintenance / Patching / Resiliency - Manual scaling (or configure) - Pay for whole duration (even when not in use)
Containers / K8s (CAAS)	- Provisioning - Configure scaling -
App Engine (PAAS) Heroku Cloud Foundry	- Pay for at least 1 instance - Scaling has to be configured - No server maintenance - Services for integration (DB, Caching, Pub/Sub, etc.)
FAAS (Function as a service)	?

Serverless, FAAS, Event driven, Java, Micronaut, GraalVM, ZeroGC,

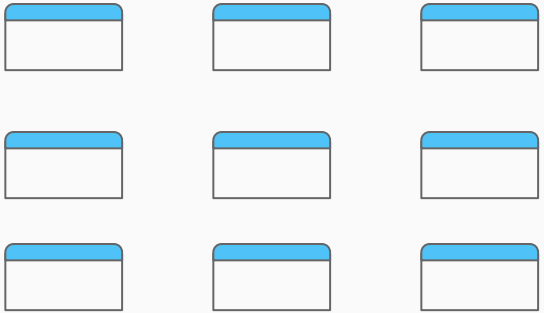
- What is serverless?
- UseCase #1 Cron/Batch jobs. If using Spring Boot, have to keep application up for 24/7. Wasting precious resources.
- UseCase #2 Image resizing. In addition to above, who will trigger or tell our application that image is uploaded if app is off.
- FAAS - On some event happening (image upload) we want someone (Cloud Provider) to call our function. When function's job is over (how does it know) then stop billing...
- What types of workloads are suited to FAAS?
    - Embarrassingly parallel. Bursty/Unpredictable. Event-Driven / Asynchronous. Short-lived.
- Use Cases:
    -
    - Data transformation
    - CI/CD
- Providers
    - AWS Lambda (pioneers)
    - GCP, Azure etc. Fn, KNative, Cloudfare workers etc.
- Language runtimes
    - Cloud providers (screenshots)
    - Also option to upload Docker containers. Fargate/Cloud Run/Fn etc.
- Internals - MicroVM (firecracker), gVisor (container), Isolates (V8, cloudfare), WASM... (some will be applicable only for JS/Node runtimes).... for developer it shouldn't matter.
- Sometimes, serverless is conflated with FAAS.
    - Serverless = no provisioning capacity, scales down to 0, pay per use, auto infinite scale
-

Slides

- Pyramid of abstractions
- ~~Characteristics - No server management, auto scale, auto provision, billing on per use, high availability~~
- Serverless - As a developer (left) you are abstracted from (cloud/serverless platform - right), provisioning, scaling & management
- Compute - BAAS, FAAS. Storage - S3, Aurora. Services - Auth0, Firebase Auth, Vision API, BigQuery etc.
- FAAS - Can also be thought as further breaking down microservices, which talk to each other using events (instead of direct calls).
- Ideal workloads - Embarrassingly parallel, stateless, asynchronous, short-lived
- Everything can produce an event, and more & more full fledged functionalities are being provided as service. Eventually its only glueing.
- This leads to event driven architecture. Many workloads are glove fit to this architecture (image, video, IoT etc.)
- How to create a function? Scheduling, http, message.
- Is PAAS serverless? No, it has to start instantly, and scale down to 0. Cloud Foundry doesn't scale down to 0 when not in use. If you are paying even when not using (no requests are served for compute, nothing stored for storage), it's not serverless.
- BAAS - Auth0, Firebase (authentication)
- Cost - 100ms billing. Can be incredibly cheap depending on the workloads.
- But its not about cost. Economy of time and money. Its about developer freedom. Its about more focus on business value (app code) and less on infrastructure.
- Cons -
    - Cold Starts. Might not be best fit for synchronous APIs. Though, it is improving fast, and Azure allows ways around it.
    - Step functions / Durable functions for complex stateful logic (can't just depend on state being in cache for a user, which actually if you think about Cloud native, you should not anyways, you should use external cache like Redis)
    - Tooling still not mature.
- Stateless? yes, but container loaded could be reused for next request. So architect for share nothing instead of stateless. Caching is ok.
- screenshot of associating event (source) to a lambda.
- Microservices vs FAAS (PAAS vs Event-driven).
- Event driven architecture - One fn calls 3 other functions, it will cost. Instead one fn can push event/message on queue, or add record to

AWS Lambda

Launched in 2014



One long running application (monolith/microservice)

Smaller units of code each responding to an type of event