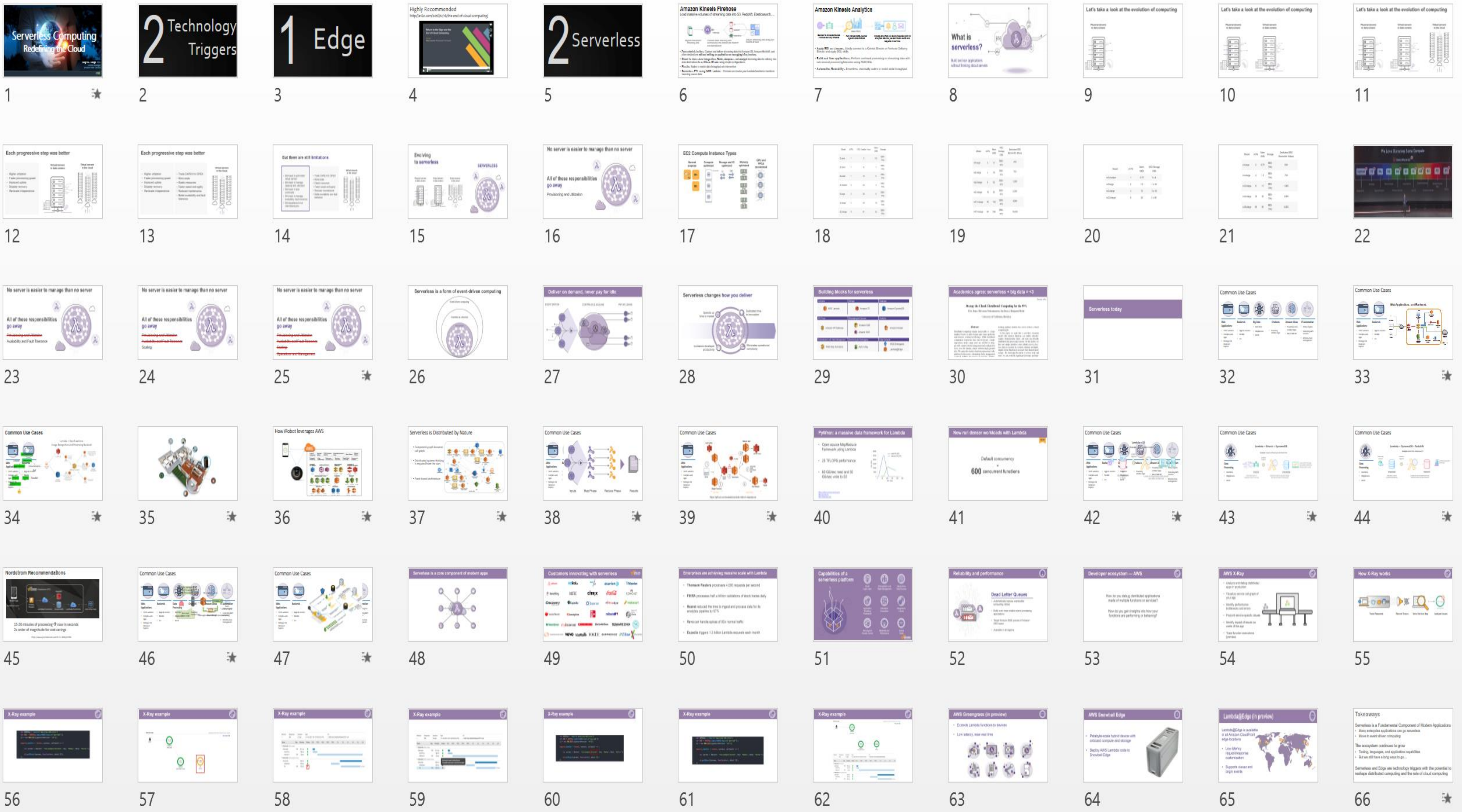


Serverless Computing
Redefining the Cloud
1
2 Technology Triggers
2
1 Edge
3
Highly Recommended
4
2 Serverless
5
Amazon Kinesis Firehose
6
Amazon Kinesis Analytics
7
What is serverless?
8
Let's take a look at the evolution of computing
9
Let's take a look at the evolution of computing
10
Let's take a look at the evolution of computing
11
Each progressive step was better
12
Each progressive step was better
13
But there are still limitations
14
Evolving to serverless
15
No server is easier to manage than no server
All of these responsibilities go away
16
EC2 Compute Instance Types
17
18
19
20
21
22
No server is easier to manage than no server
All of these responsibilities go away
23
No server is easier to manage than no server
All of these responsibilities go away
24
No server is easier to manage than no server
All of these responsibilities go away
25
Serverless is a form of event-driven computing
26
Deliver on demand, never pay for idle
27
Serverless changes how you deliver
28
Building blocks for serverless
29
Academics agree: serverless + big data = <3
30
Serverless today
31
Common Use Cases
32
Common Use Cases
33
Common Use Cases
34
35
How iRobot leverages AWS
36
Serverless is Distributed by Nature
37
Common Use Cases
38
Common Use Cases
39
PyWren: a massive data framework for Lambda
40
Now run denser workloads with Lambda
Default concurrency
600 concurrent functions
41
Common Use Cases
42
Common Use Cases
43
Common Use Cases
44
Nordstrom Recommendations
45
Common Use Cases
46
Common Use Cases
47
Serverless is a core component of modern apps
48
Customers innovating with serverless
49
Enterprises are achieving massive scale with Lambda
50
Capabilities of a serverless platform
51
Reliability and performance
Dead Letter Queues
52
Developer ecosystem — AWS
53
AWS X-Ray
54
How X-Ray works
55
X-Ray example
56
X-Ray example
57
X-Ray example
58
X-Ray example
59
X-Ray example
60
X-Ray example
61
X-Ray example
62
AWS Greengrass (in preview)
63
AWS Snowball Edge
64
Lambda@Edge (in preview)
65
Takeaways
66

Serverless Computing
Redefining the Cloud

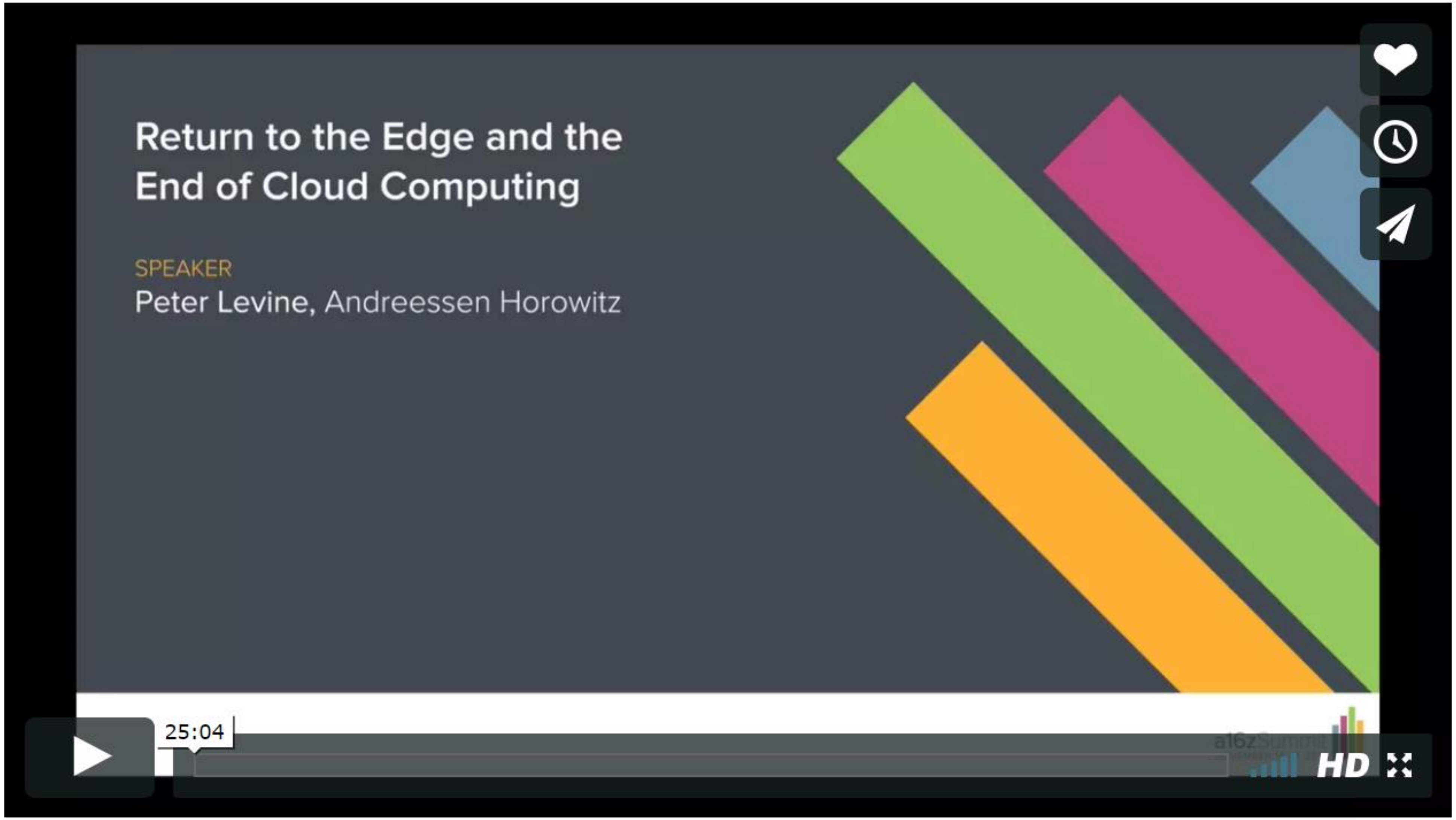
Roger S. Barga, Ph.D.
General Manager
Amazon Web Services

2 Technology Triggers

1 Edge

Highly Recommended

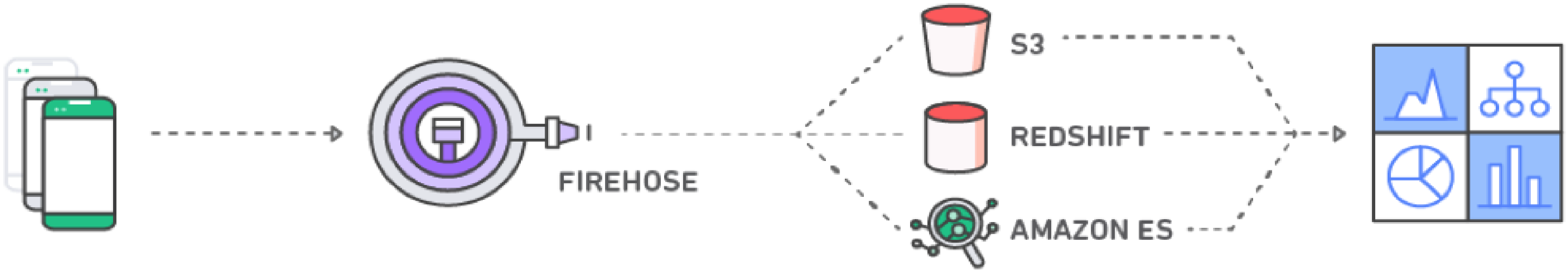
http://a16z.com/2016/12/16/the-end-of-cloud-computing/



2 Serverless

Amazon Kinesis Firehose

Load massive volumes of streaming data into S3, Redshift, Elasticsearch,…



Capture and submit streaming data

Firehose loads streaming data continuously into Amazon S3, Redshift and Elasticsearch

Analyze streaming data using your favorite BI tools

- **Zero administration:** Capture and deliver streaming data into Amazon S3, Amazon Redshift, and other destinations **without writing an application or managing infrastructure.**
- **Direct-to-data store integration: Batch, compress**, and **encrypt** streaming data for delivery into data destinations **in as little as 60 secs** using simple configurations.
- **Elastic:** Scales to match data throughput w/o intervention
- **Serverless ETL using AWS Lambda** - Firehose can invoke your Lambda function to transform incoming source data.

Amazon Kinesis Analytics

Connect to Kinesis streams, Firehose delivery streams

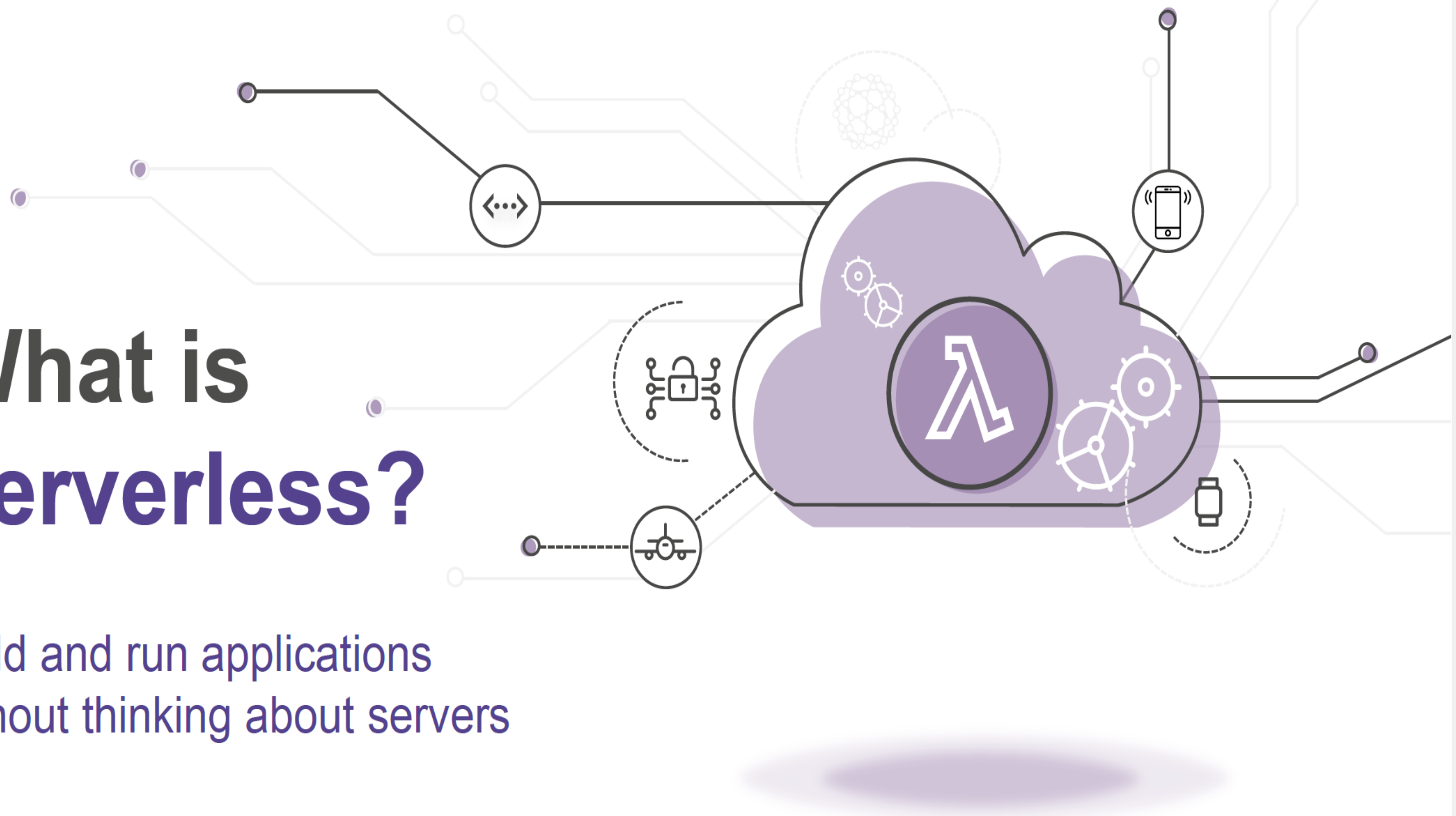
Run standard SQL queries against data streams

Kinesis Analytics can send processed data to analytics tools so you can create alerts and respond in real-time

- **Apply SQL on streams:** Easily connect to a Kinesis Stream or Firehose Delivery Stream and apply SQL skills.
- **Build real-time applications:** Perform continual processing on streaming data with sub-second processing latencies using ANSI SQL
- **Automatics Scalability :** Serverless, elastically scales to match data throughput.

What is **serverless?**

Build and run applications without thinking about servers



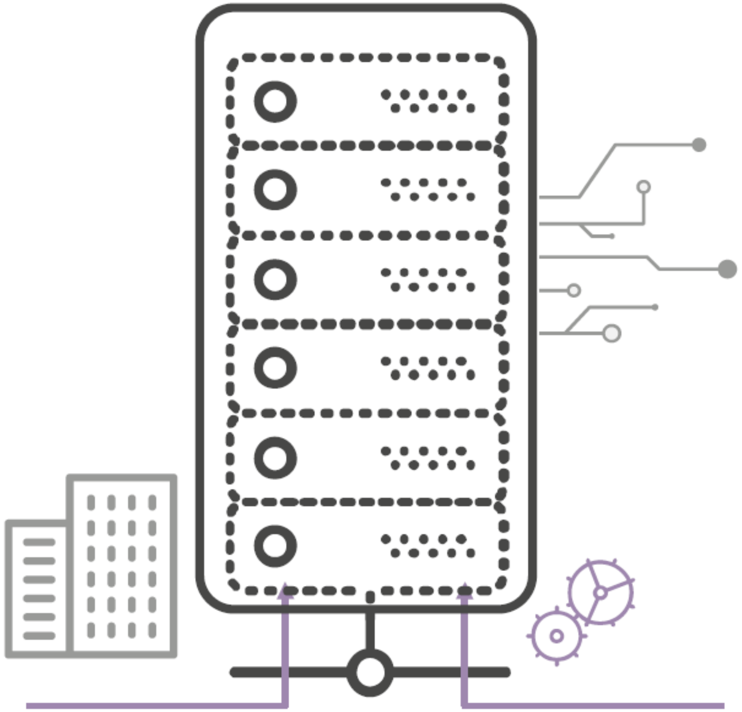
Let’s take a look at the evolution of computing

Physical servers in data centers

Let's take a look at the evolution of computing

Physical servers in data centers

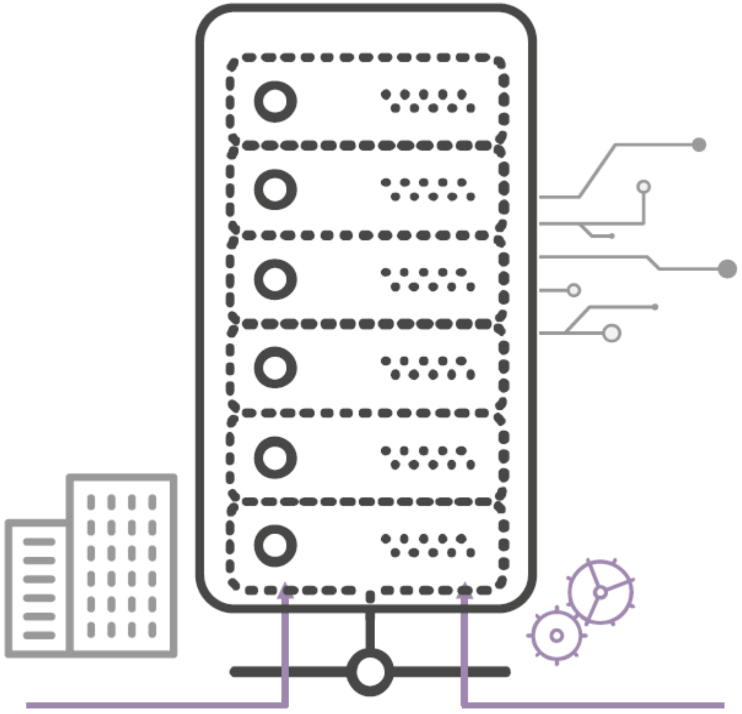
Virtual servers in data centers



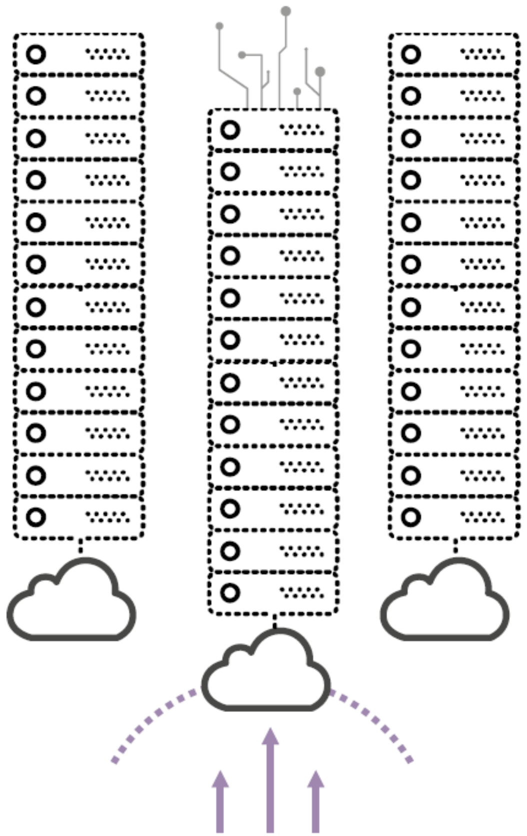
Let’s take a look at the evolution of computing

Physical servers in data centers

Virtual servers in data centers



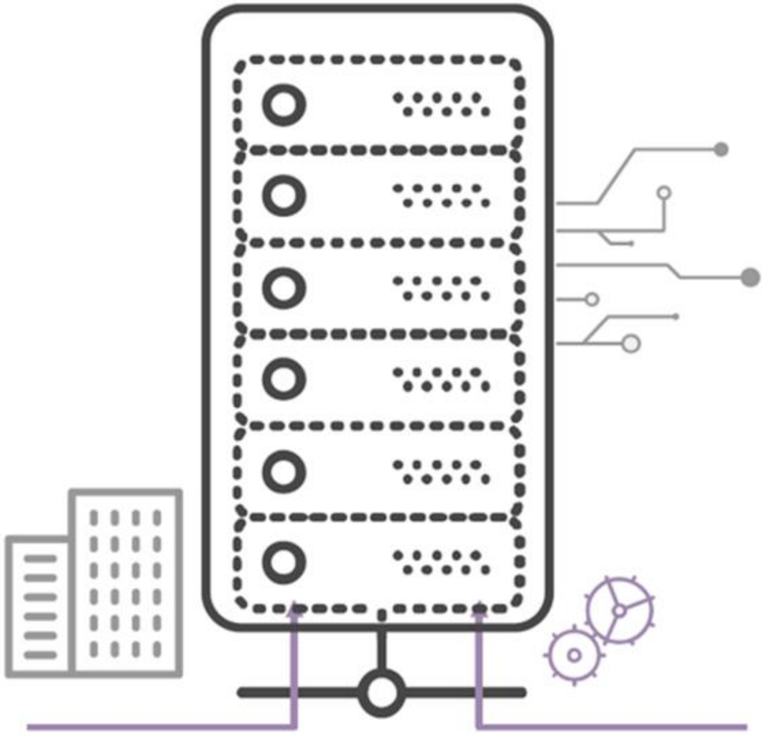
Virtual servers in the cloud



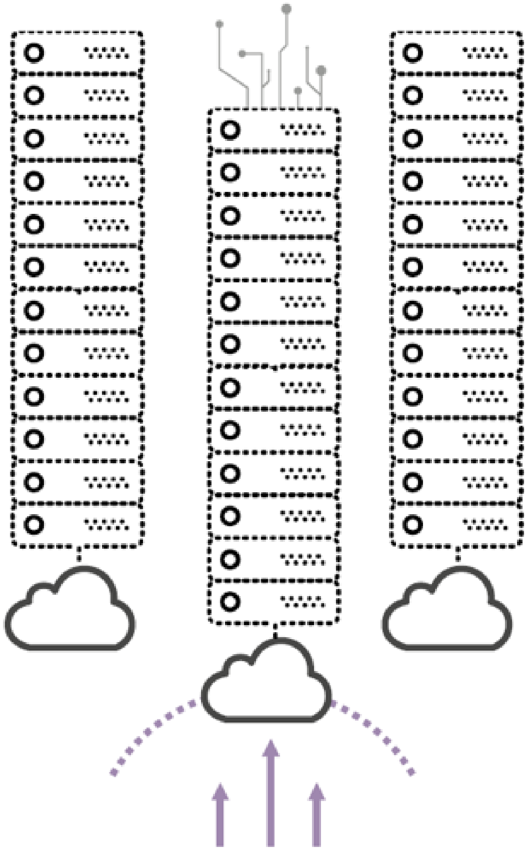
Each progressive step was better

- Higher utilization
- Faster provisioning speed
- Improved uptime
- Disaster recovery
- Hardware independence

Virtual servers in data centers



Virtual servers in the cloud

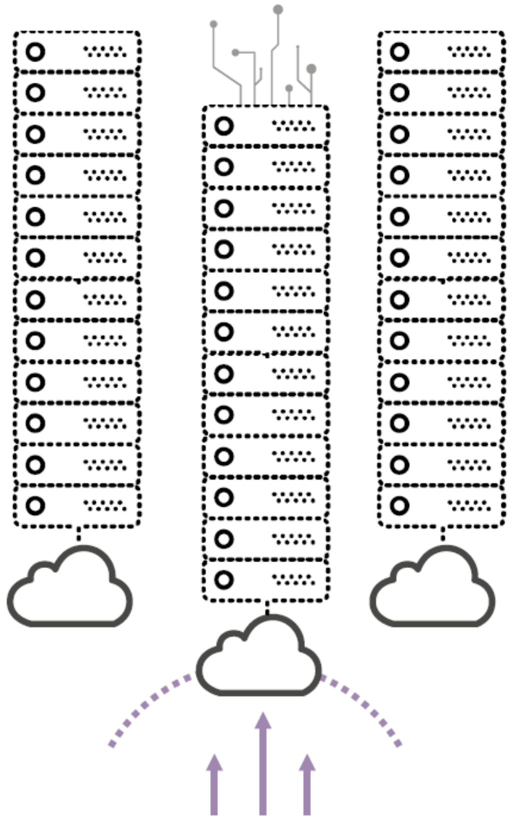


Each progressive step was better

- Higher utilization
- Faster provisioning speed
- Improved uptime
- Disaster recovery
- Hardware independence

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

Virtual servers in the cloud

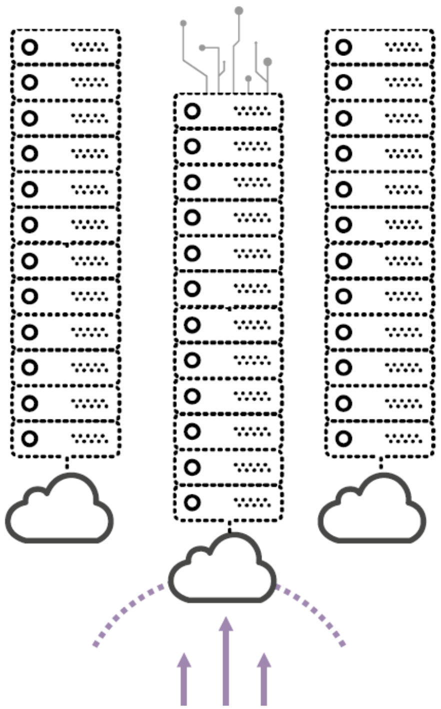


But there are still limitations

- Still need to administer virtual servers
- Still need to manage capacity and utilization
- Still need to size workloads
- Still need to manage availability, fault tolerance
- Still expensive to run intermittent jobs

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

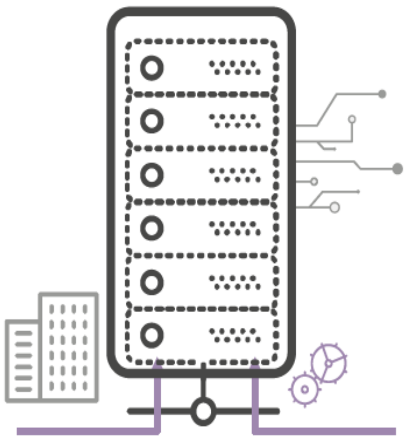
Virtual servers in the cloud



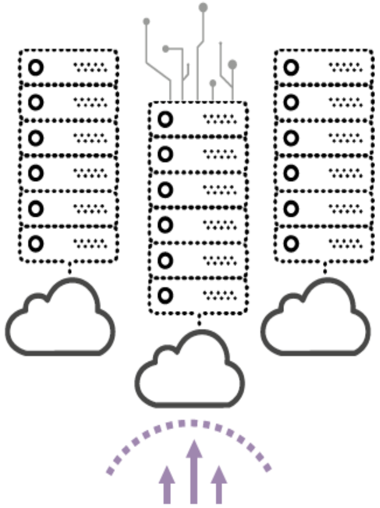
Evolving to serverless

Physical servers in data centers

Virtual servers in data centers



Virtual servers in the cloud



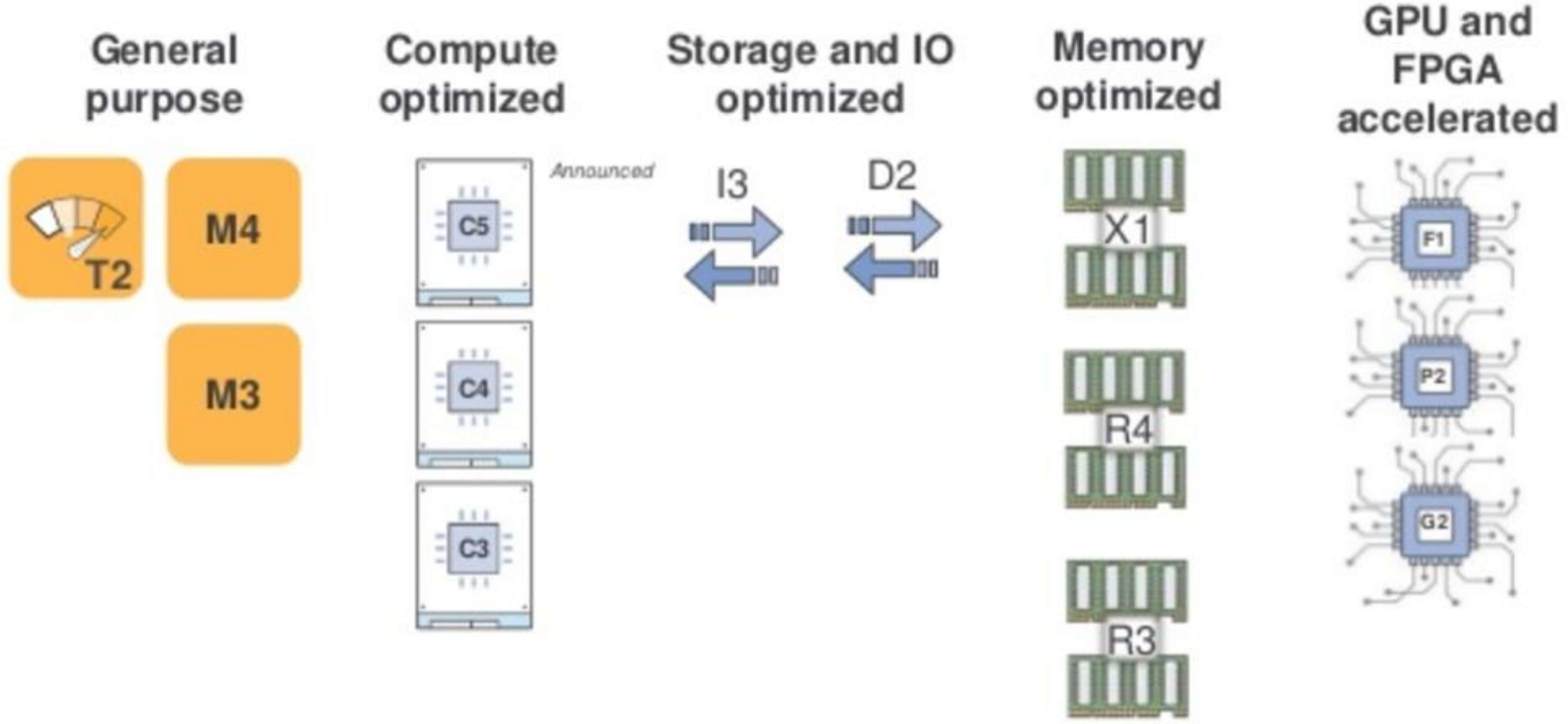
SERVERLESS

No server is easier to manage than no server

All of these responsibilities **go away**

Provisioning and Utilization

EC2 Compute Instance Types

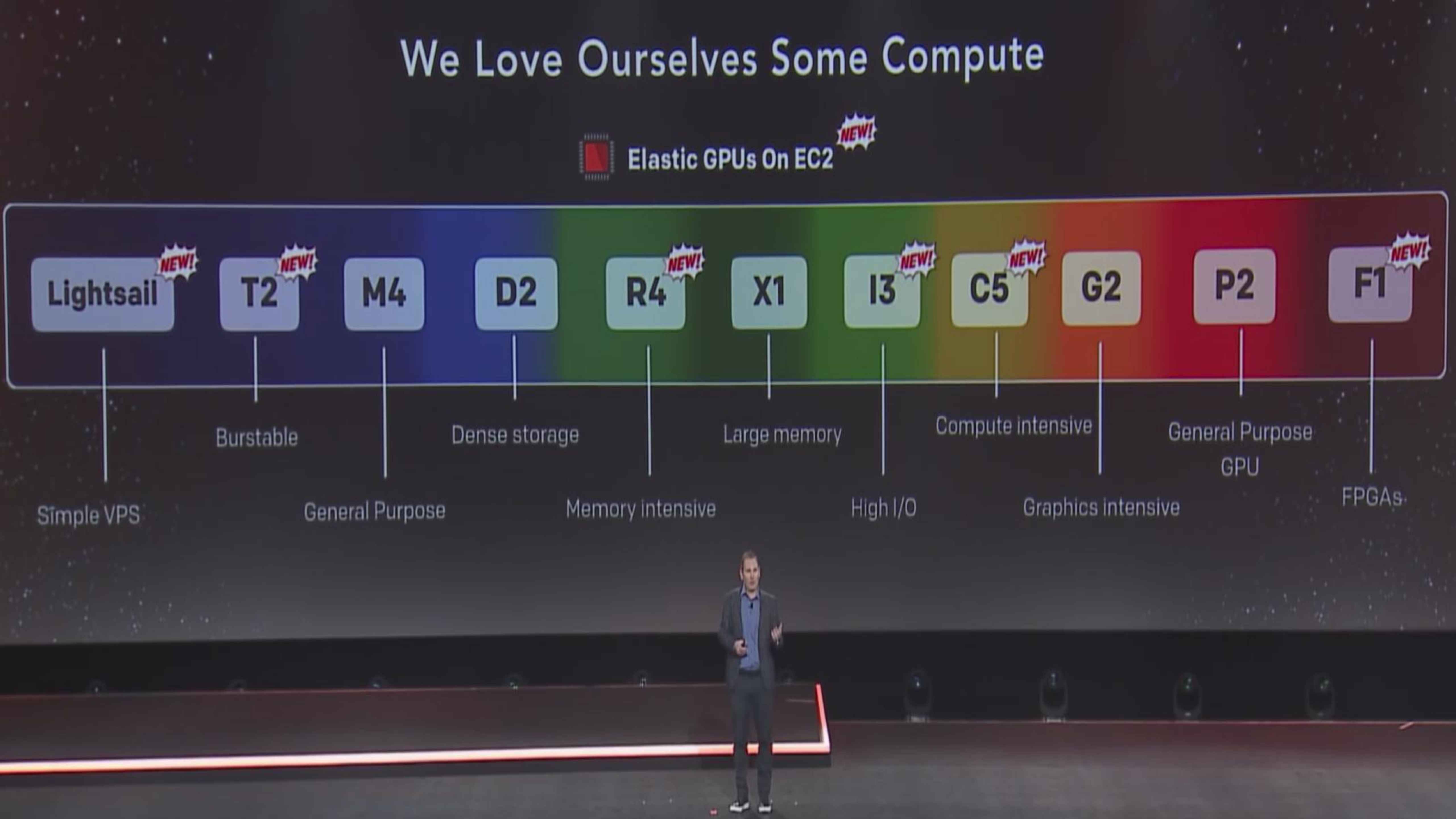


Model	vCPU	CPU Credits / hour	Mem (GiB)	Storage
t2.nano	1	3	0.5	EBS-Only
t2.micro	1	6	1	EBS-Only
t2.small	1	12	2	EBS-Only
t2.medium	2	24	4	EBS-Only
t2.large	2	36	8	EBS-Only
t2.xlarge	4	54	16	EBS-Only
t2.2xlarge	8	81	32	EBS-Only

Model	vCPU	Mem (GiB)	SSD Storage (GB)	Dedicated EBS Bandwidth (Mbps)
m4.large	2	8	EBS-only	450
m4.xlarge	4	16	EBS-only	750
m4.2xlarge	8	32	EBS-only	1,000
m4.4xlarge	16	64	EBS-only	2,000
m4.10xlarge	40	160	EBS-only	4,000
m4.16xlarge	64	256	EBS-only	10,000

Model	vCPU	Mem (GiB)	SSD Storage (GB)
m3.medium	1	3.75	1 x 4
m3.large	2	7.5	1 x 32
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80

Model	vCPU	Mem (GiB)	Storage	Dedicated EBS Bandwidth (Mbps)
c4.large	2	3.75	EBS-Only	500
c4.xlarge	4	7.5	EBS-Only	750
c4.2xlarge	8	15	EBS-Only	1,000
c4.4xlarge	16	30	EBS-Only	2,000
c4.8xlarge	36	60	EBS-Only	4,000


We Love Ourselves Some Compute
Elastic GPUs On EC2
NEW!
Lightsail
NEW!
T2
NEW!
M4
D2
R4
NEW!
X1
I3
NEW!
C5
NEW!
G2
P2
F1
NEW!
Simple VPS
Burstable
General Purpose
Dense storage
Memory intensive
Large memory
High I/O
Compute intensive
Graphics intensive
General Purpose GPU
FPGAs

No server is easier to manage than no server

All of these responsibilities go away

~~Provisioning and Utilization~~

Availability and Fault Tolerance

No server is easier to manage than no server

All of these responsibilities go away

~~Provisioning and Utilization~~

~~Availability and Fault Tolerance~~

Scaling

No server is easier to manage than no server

All of these responsibilities go away

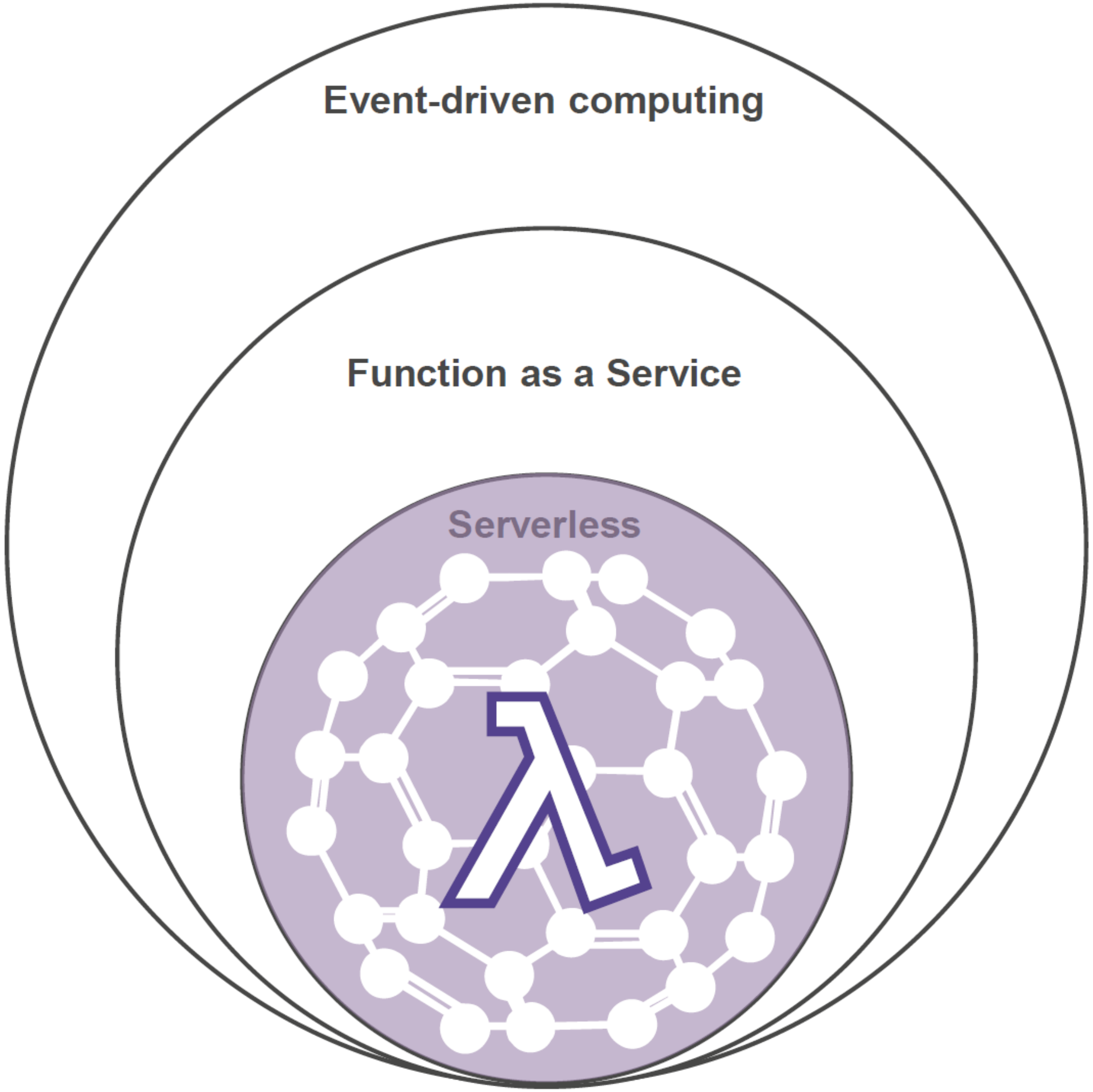
~~Provisioning and Utilization~~

~~Availability and Fault Tolerance~~

~~Scaling~~

~~Operations and Management~~

Serverless is a form of event-driven computing

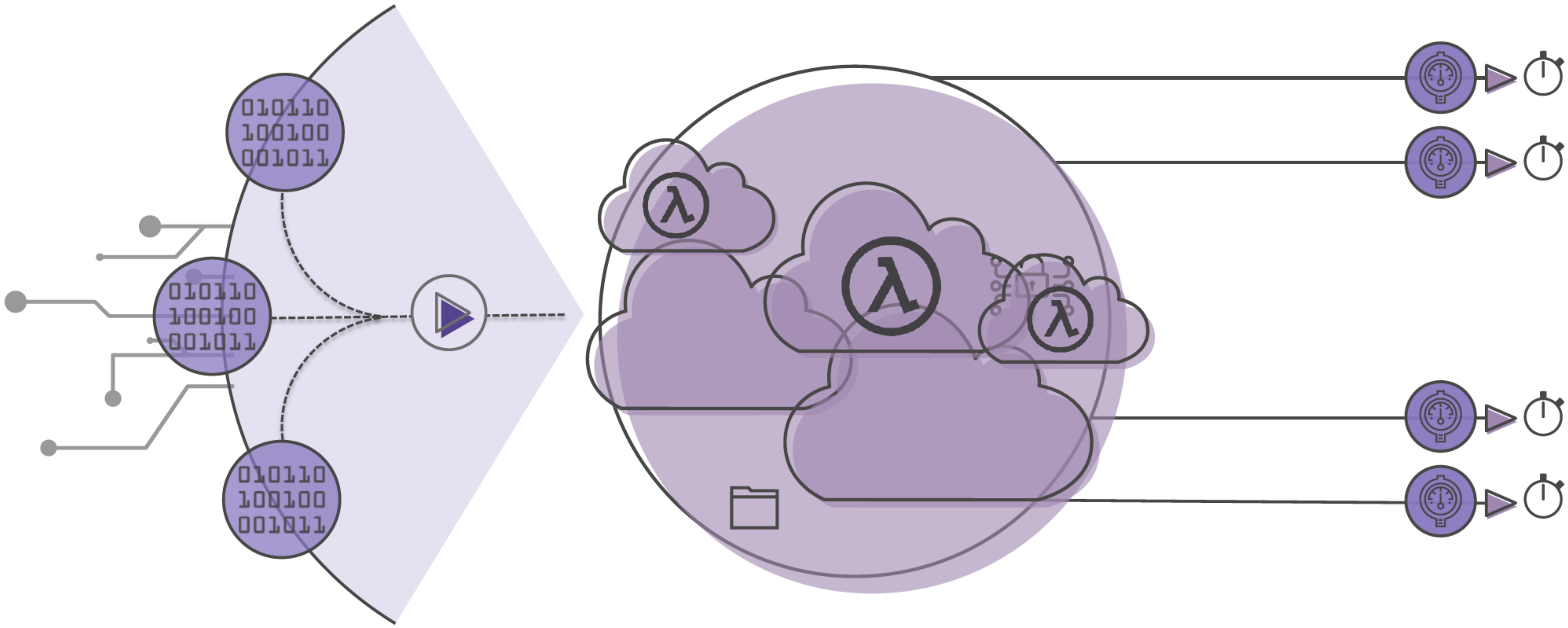


Deliver on demand, never pay for idle

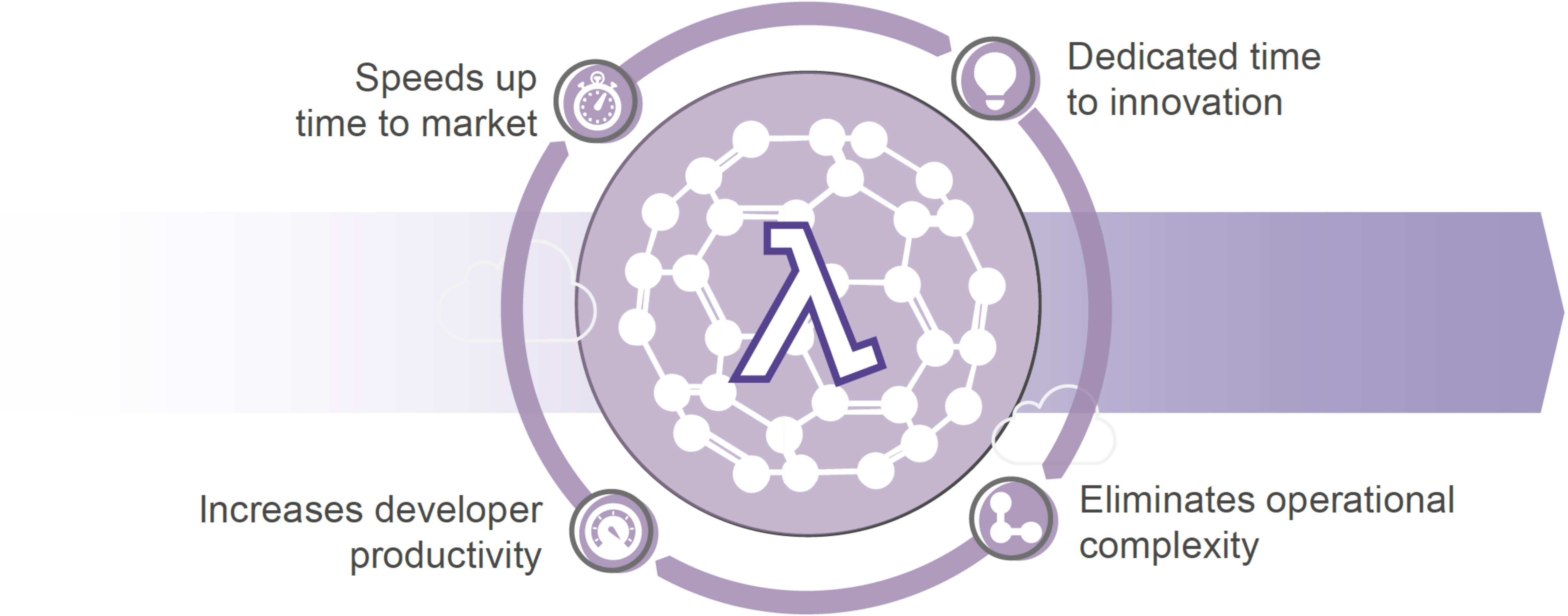
EVENT DRIVEN

CONTINUOUS SCALING

PAY BY USAGE



Serverless changes how you deliver



Building blocks for serverless

Compute
AWS Lambda

Storage
Amazon S3

Database
Amazon DynamoDB

API Proxy
Amazon API Gateway

Messaging and Queues
Amazon SQS

Amazon SNS

Analytics
Amazon Kinesis

Orchestration and State Management
AWS Step Functions

Monitoring and Debugging
AWS X-Ray

Edge Compute
AWS Greengrass

Lambda@Edge

Academics agree: serverless + big data = <3

(Source: arXiv)

Occupy the Cloud: Distributed Computing for the 99%

Eric Jonas, Shivaram Venkataraman, Ion Stoica, Benjamin Recht

University of California, Berkeley

Abstract

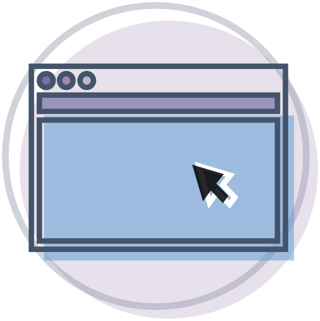
Distributed computing remains inaccessible to a large number of users, in spite of many open source platforms and extensive commercial offerings. While distributed computation frameworks have moved beyond a simple map-reduce model, many users are still left to struggle with complex cluster management and configuration tools, even for running simple embarrassingly parallel jobs. We argue that stateless functions represent a viable platform for these users, eliminating cluster management overhead, fulfilling the promise of elasticity. Further-

learning graduate students have never written a cluster computing job.

In this paper we argue that a serverless execution model with stateless functions can enable radically-simpler, fundamentally elastic, and more user-friendly distributed data processing systems. In this model, we have one simple primitive: users submit *stateless functions* that are executed in a remote container and inputs, outputs for the function are accessed from shared remote storage. By removing the notion of servers from end users, we can avoid the significant developer and man-

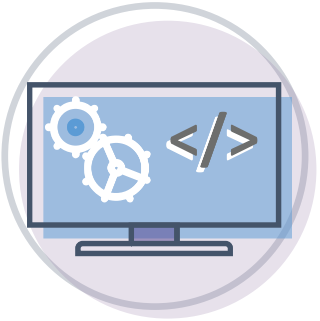
Serverless today

Common Use Cases



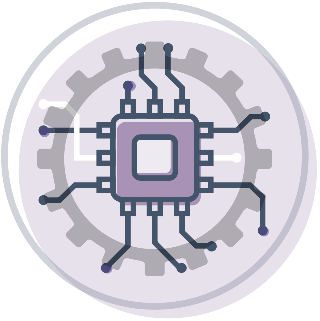
Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express



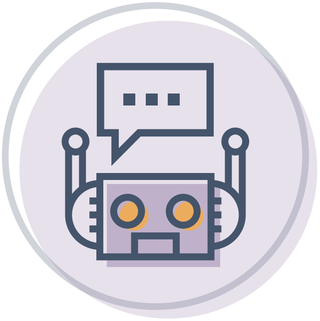
Backends

- Apps & services
- Mobile
- IoT



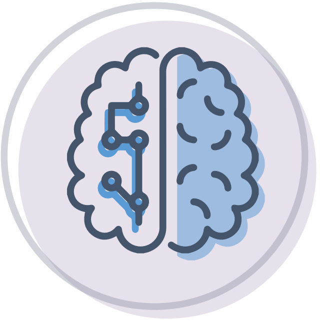
Big Data

- Real time
- MapReduce
- Batch



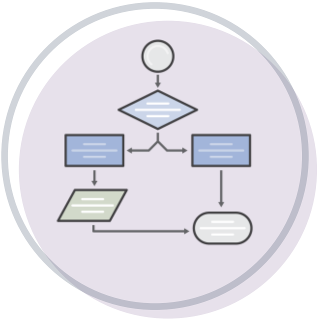
Chatbots

- Powering chatbot logic



Amazon Alexa

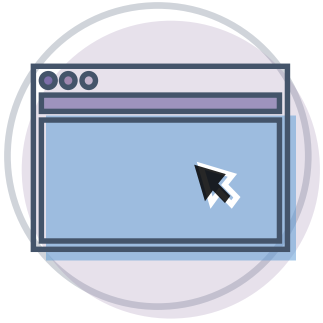
- Powering voice-enabled apps
- Alexa Skills Kit



IT Automation

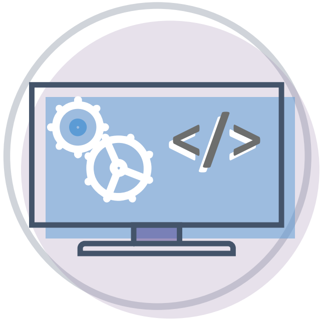
- Policy engines
- Extending AWS services
- Infrastructure management

Common Use Cases



Web Applications

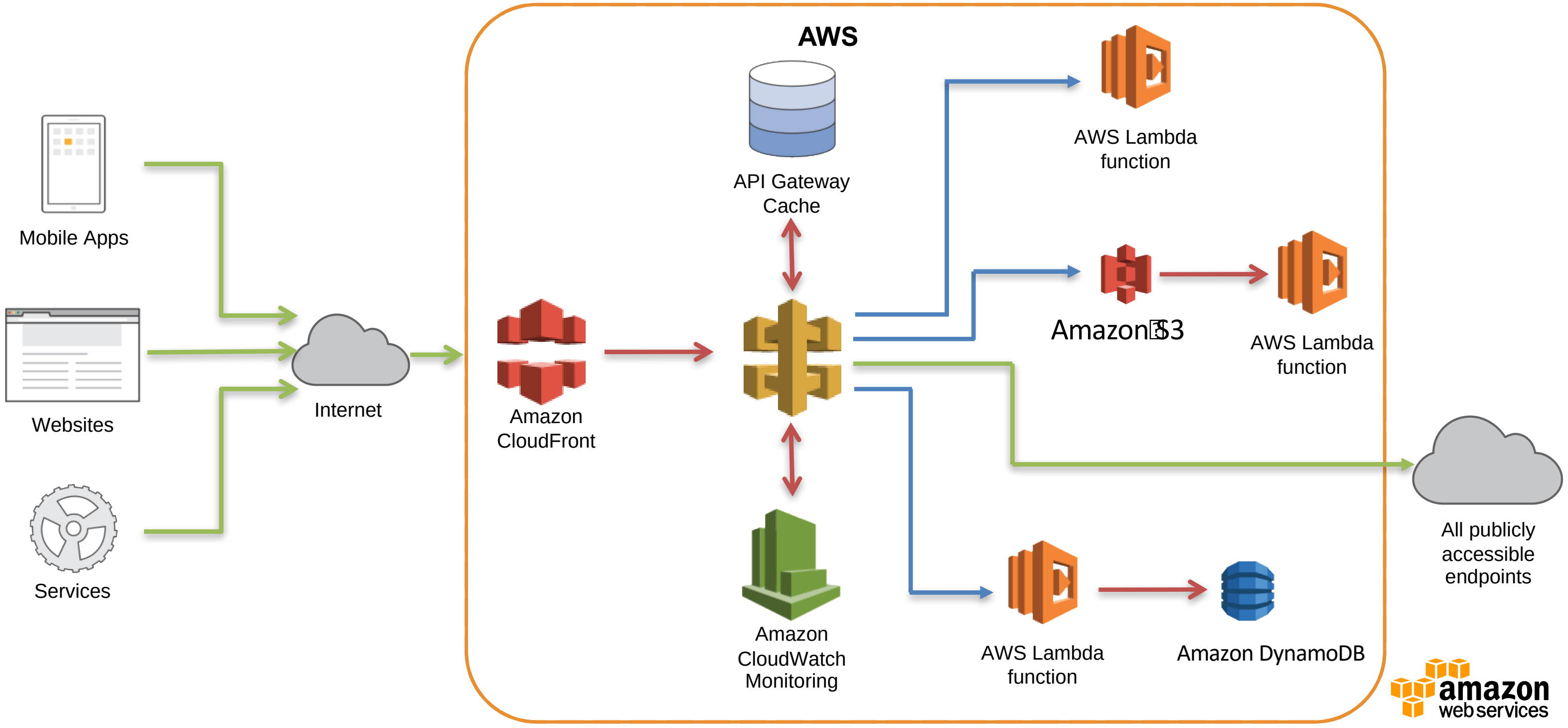
- Static websites
- Complex web apps
- Packages for Flask and Express



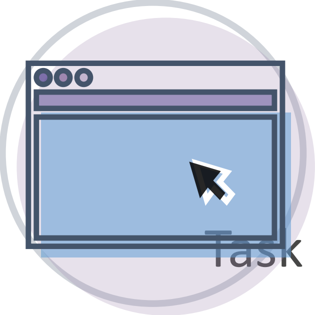
Backends

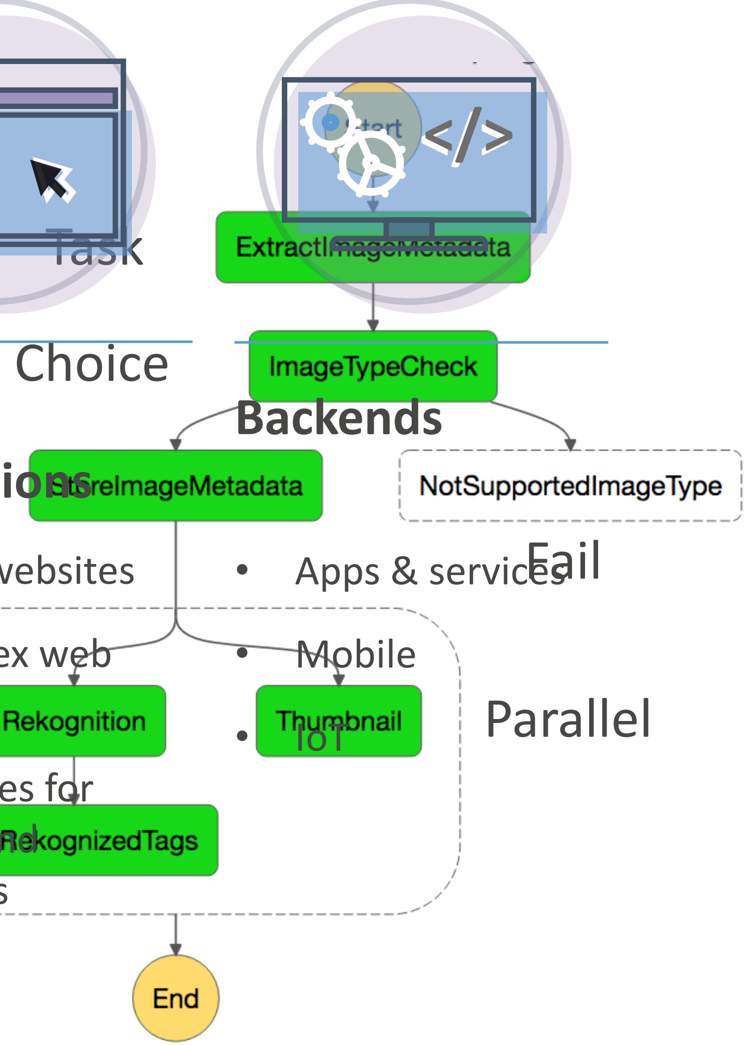
- Apps & services
- Mobile
- IoT

Web Applications and Backends



Common Use Cases





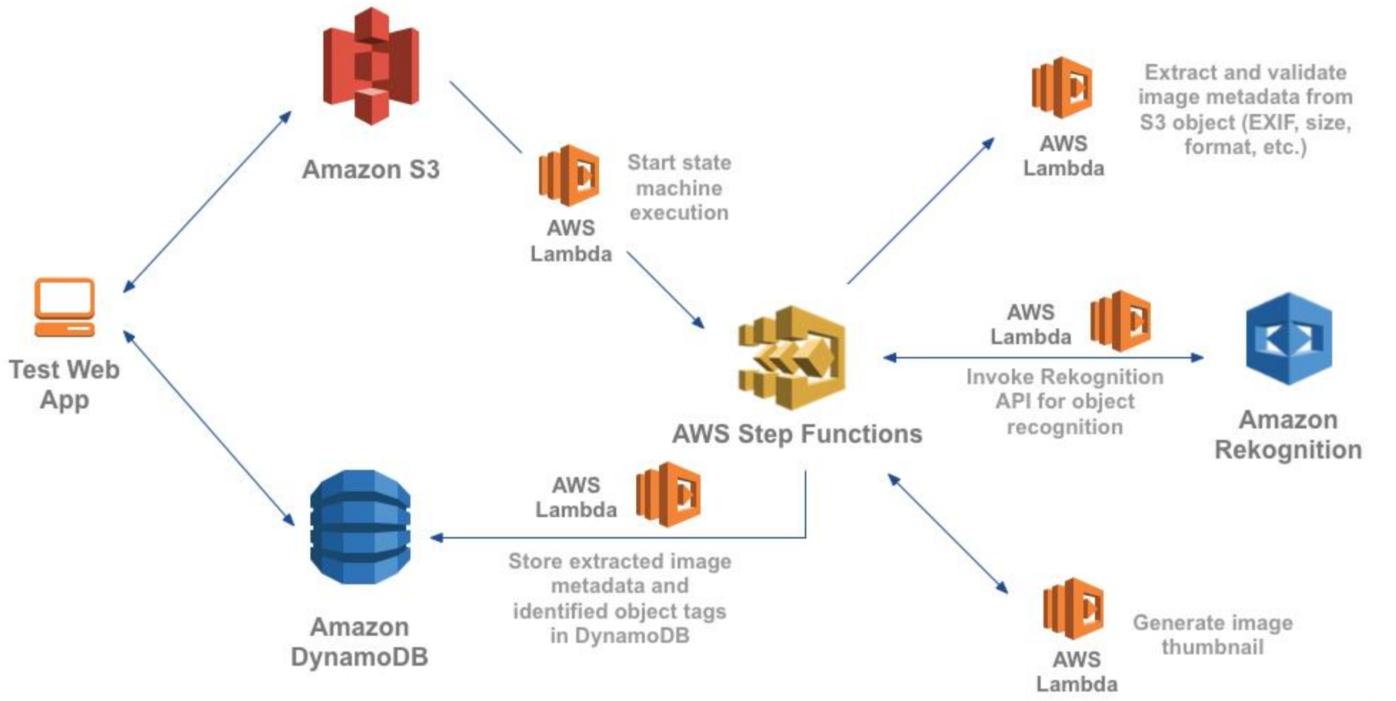
Web Applications

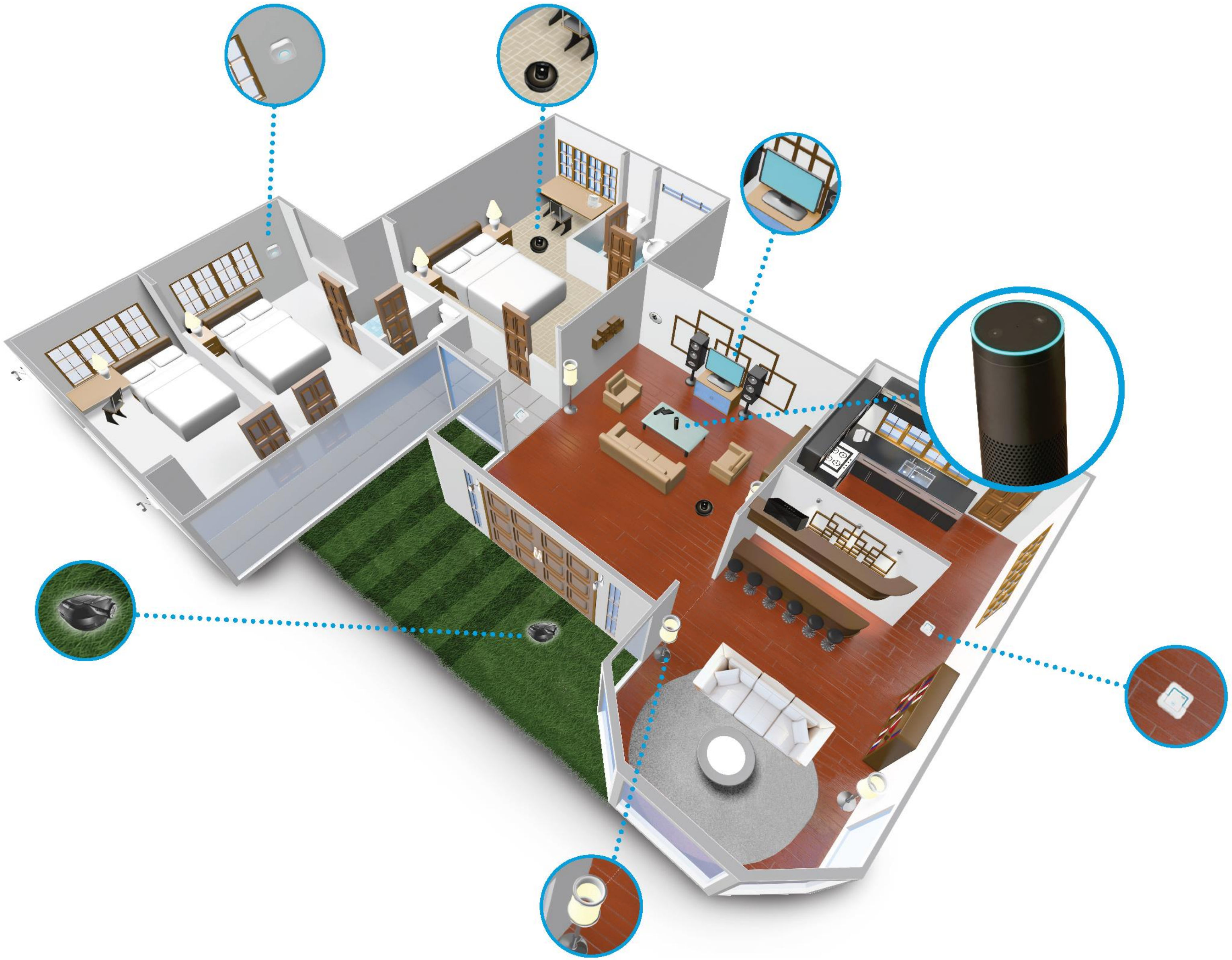
- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

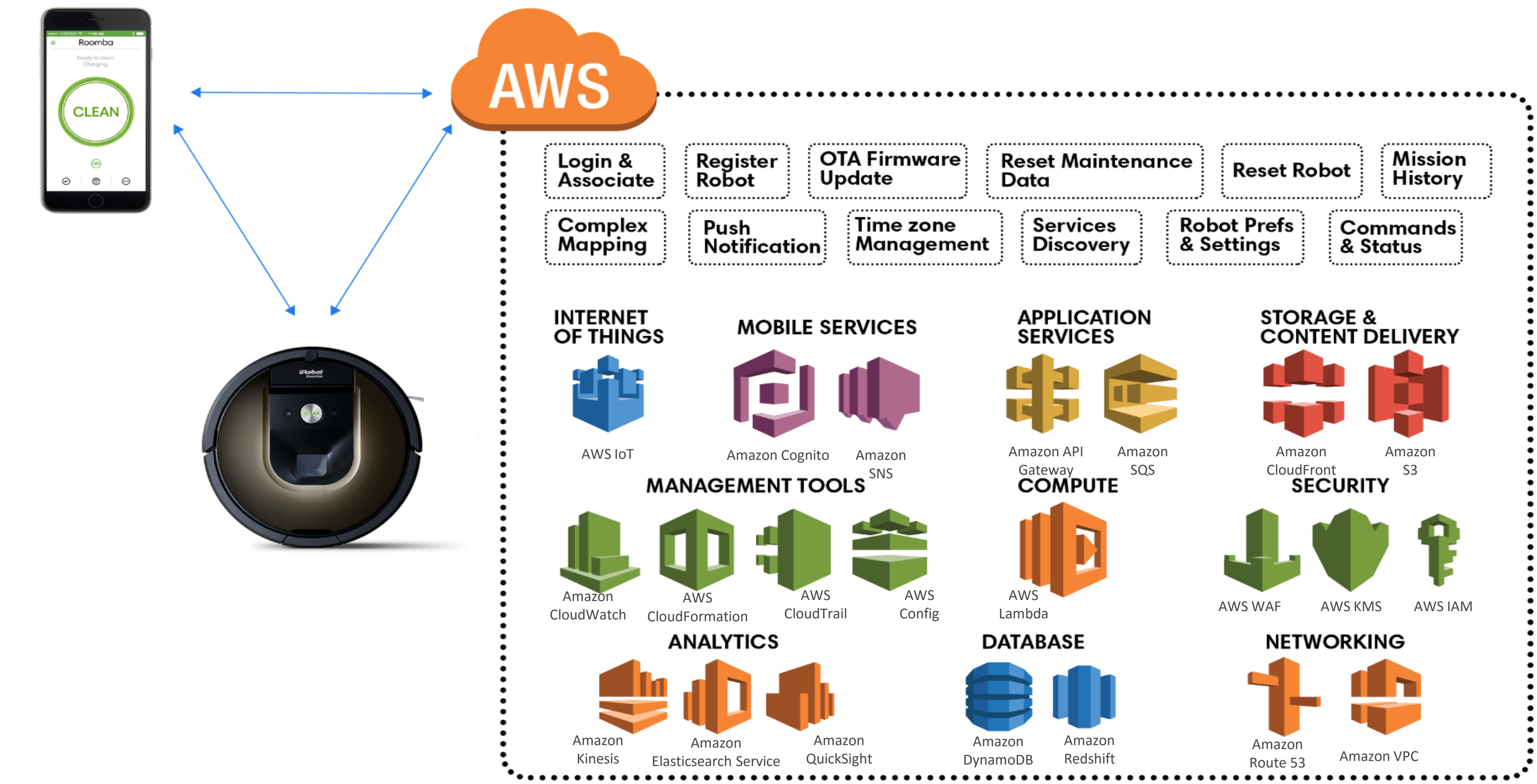
- Apps & services
- Mobile
- IoT

Lambda + Step Functions
Image Recognition and Processing Backend



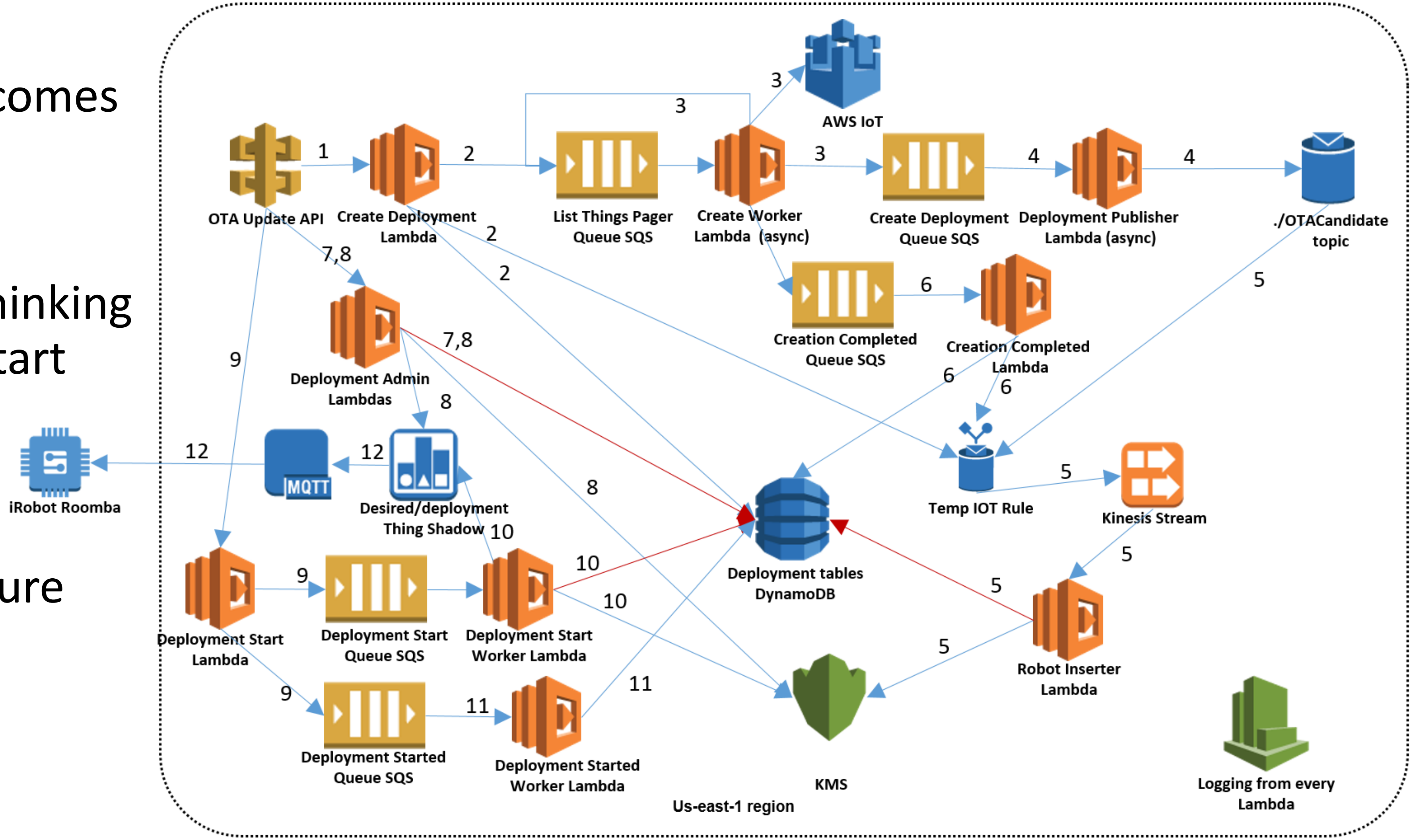


How iRobot leverages AWS

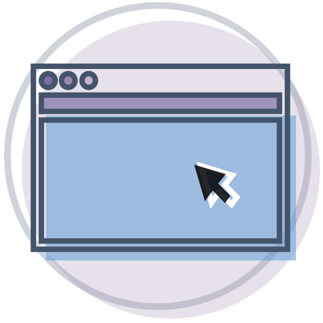


Serverless is Distributed by Nature

- Component graph becomes call graph
- Distributed systems thinking is required from the start
- Event-based architecture



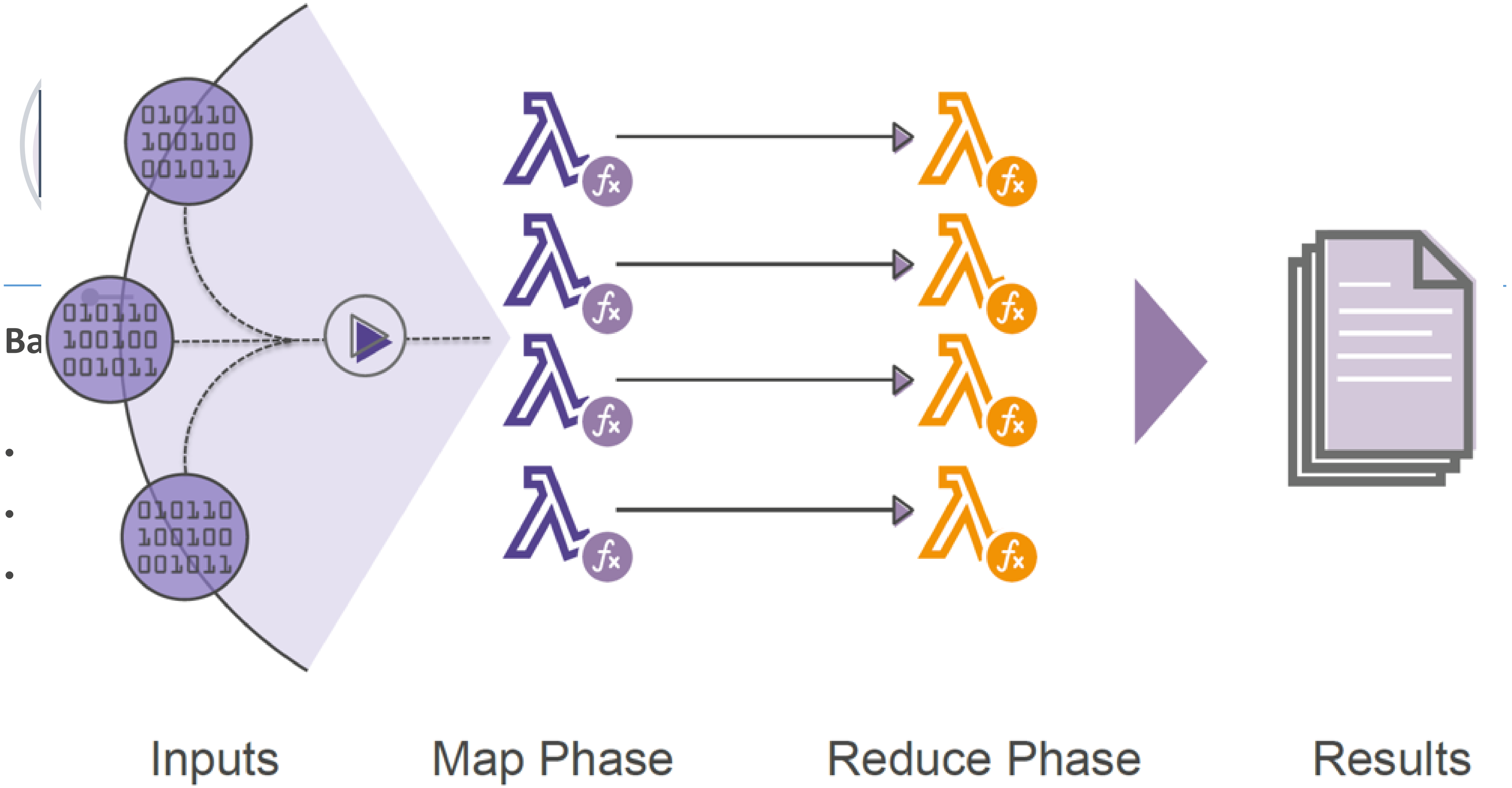
Common Use Cases



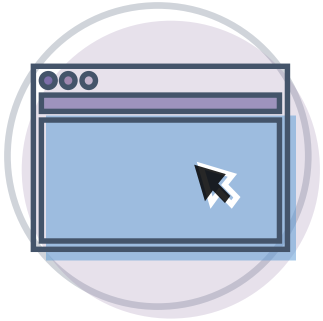
Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Ba

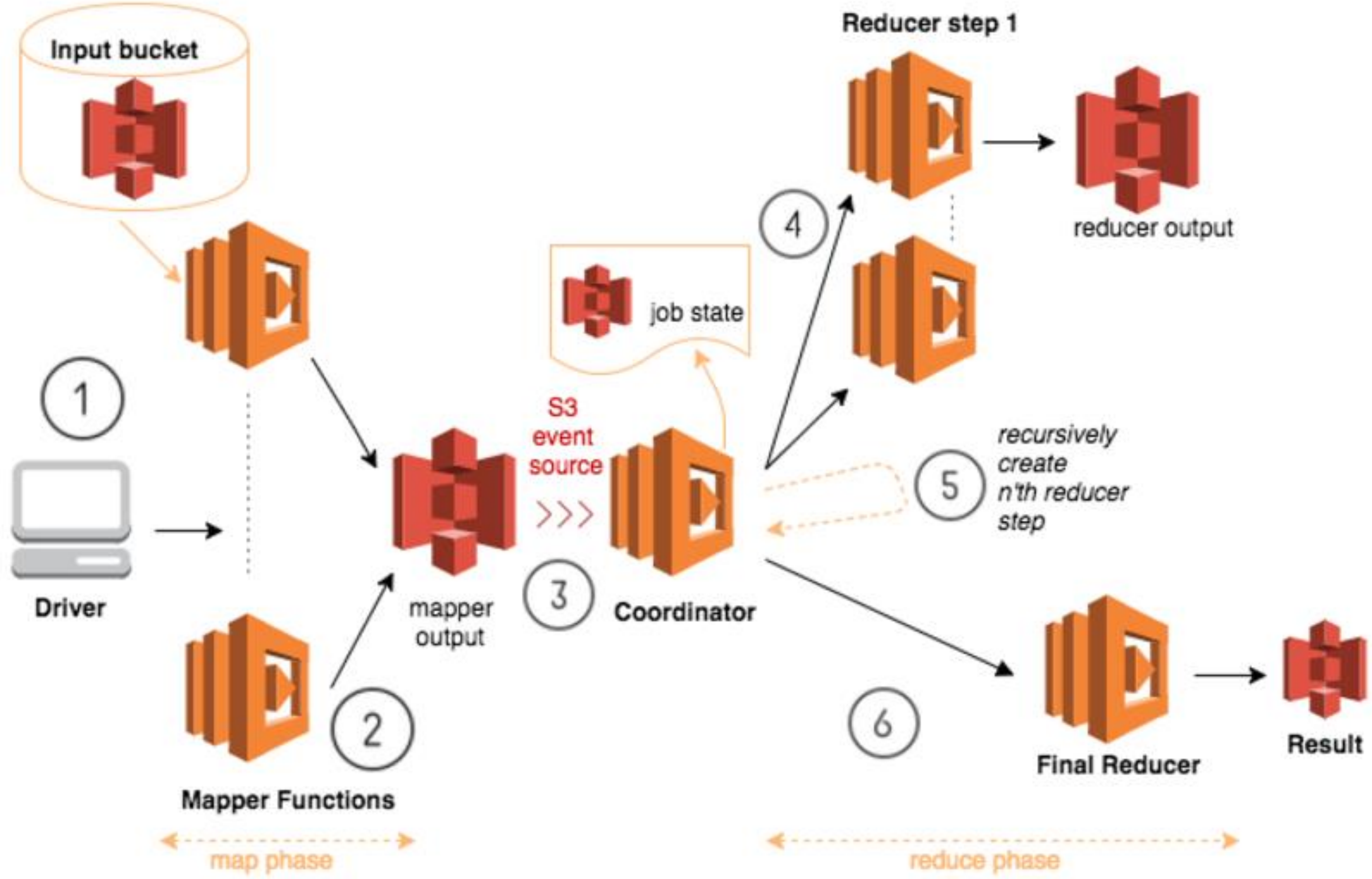


Common Use Cases



Web Applications

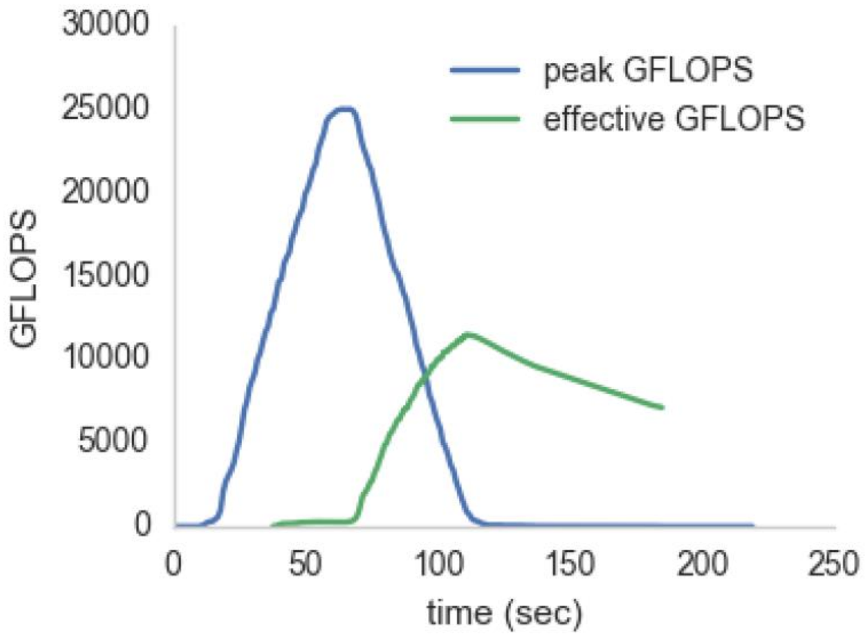
- Static websites
- Complex web apps
- Packages for Flask and Express



https://github.com/awslabs/lambda-refarch-mapreduce

PyWren: a massive data framework for Lambda

- Open source MapReduce framework using Lambda
- 25 TFLOPS performance
- 60 GB/sec read and 50 GB/sec write to S3



https://github.com/pywren/pywren
http://pywren.io/
http://ericjonas.com/

Now run denser workloads with Lambda

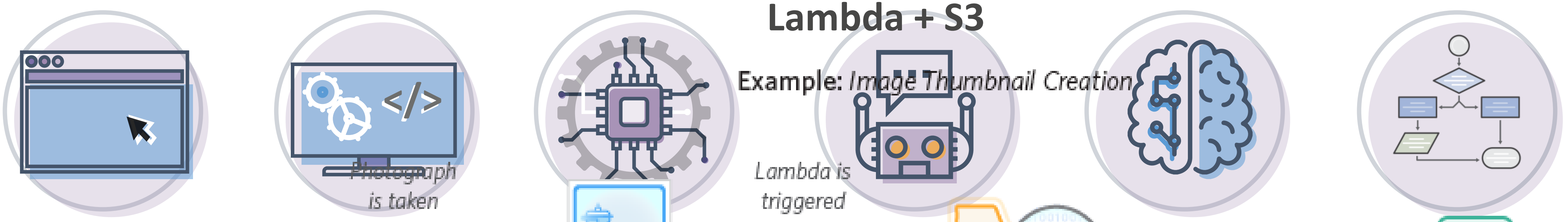
NEW

Default concurrency

=

600 concurrent functions

Common Use Cases



Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps & services
- Mobile
- IoT

Data Processing

- Real time
- MapReduce
- Batch

Chatbots

- Powering chatbot logic

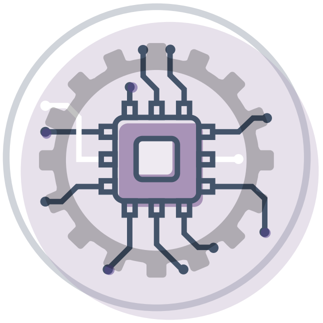
Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit

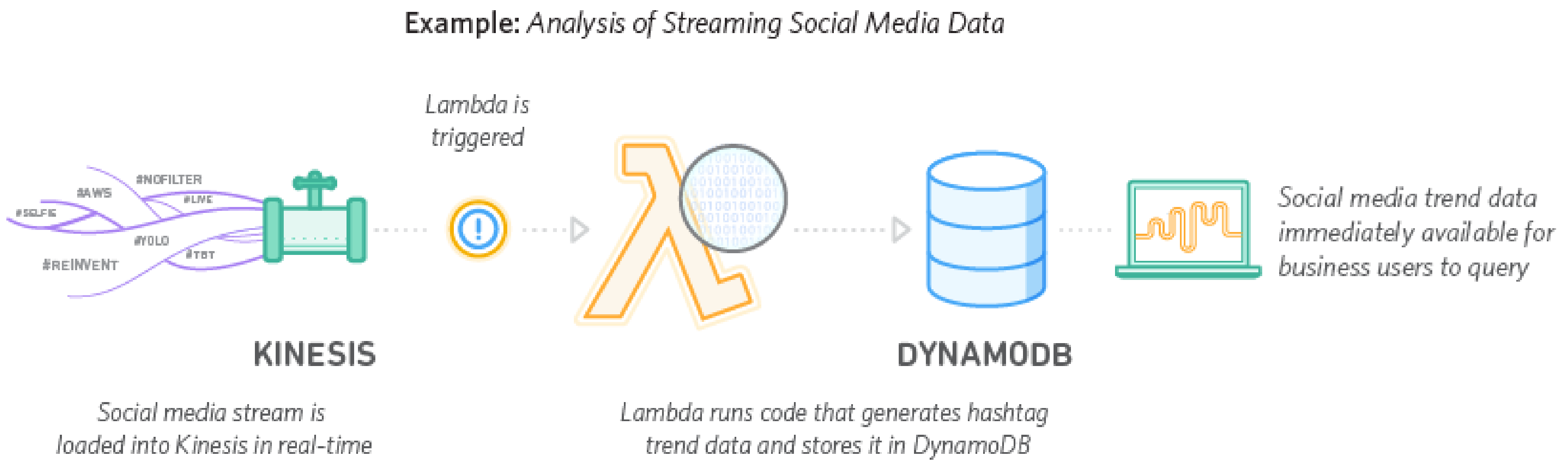
IT Automation

- Policy engines
- Extending AWS services
- Infrastructure management

Common Use Cases



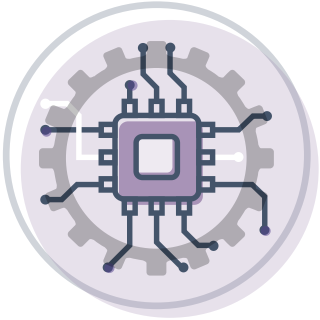
Lambda + Kinesis + DynamoDB



Data Processing

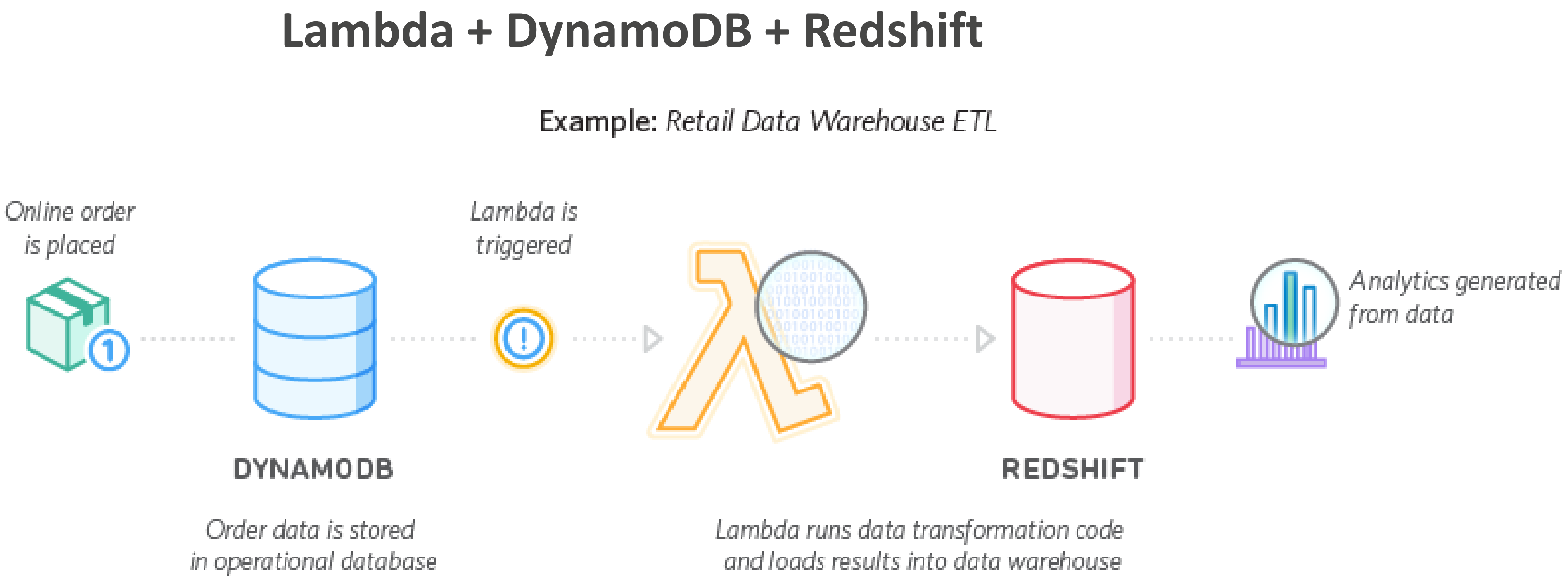
- Real time
- MapReduce
- Batch

Common Use Cases

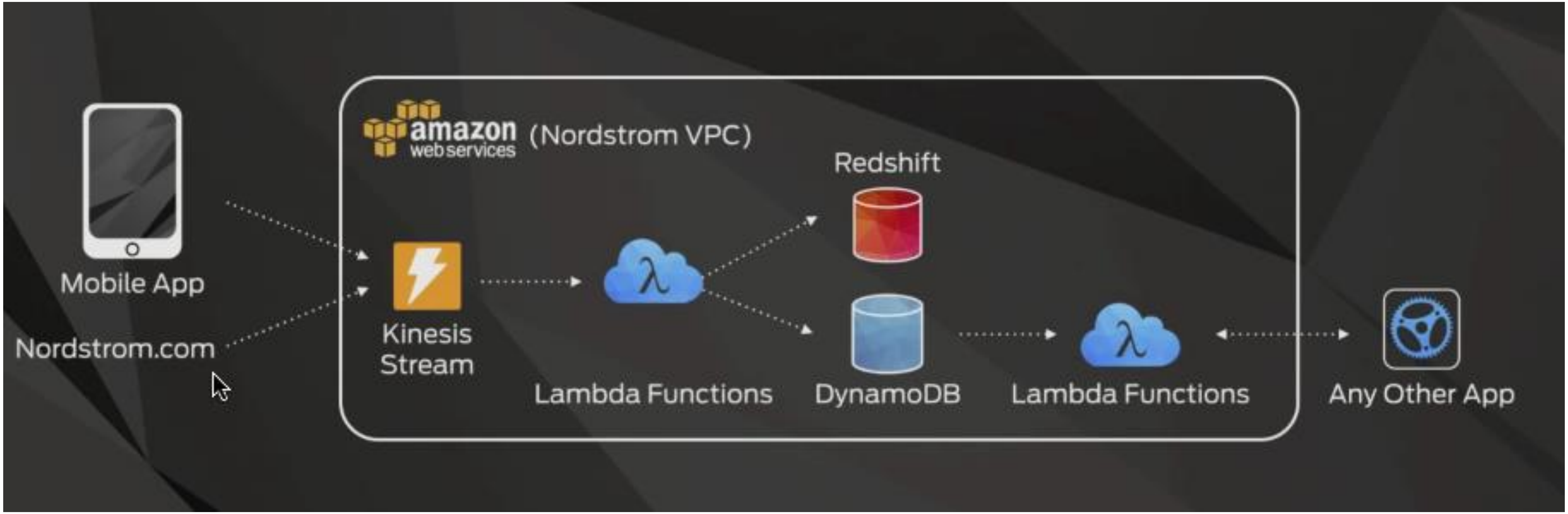


Data Processing

- Real time
- MapReduce
- Batch



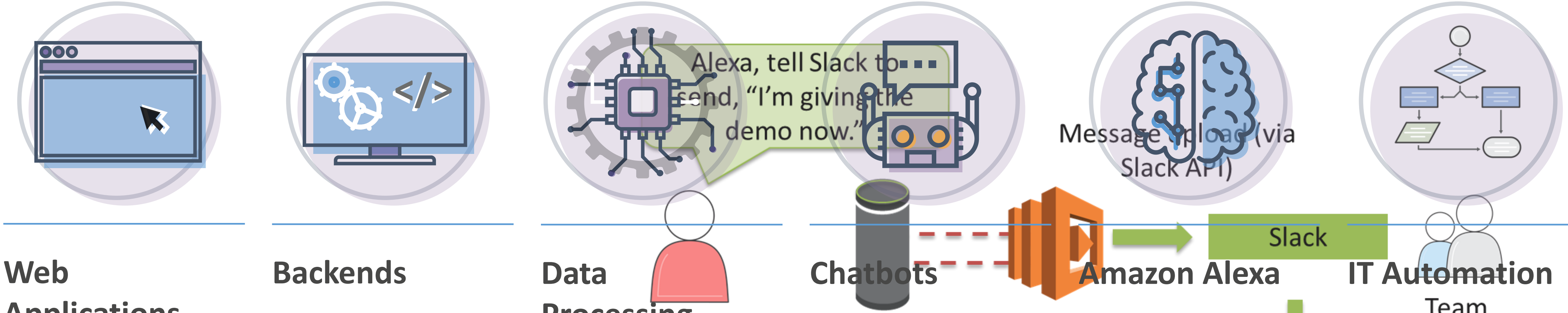
Nordstrom Recommendations



15-20 minutes of processing → now in seconds
2x order of magnitude for cost savings

https://www.youtube.com/watch?v=TXmkj2a0fRE

Common Use Cases



Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps & services
- Mobile
- IoT

Data Processing

- Real time
- MapReduce
- Batch

Chatbots

- Powering chatbot logic

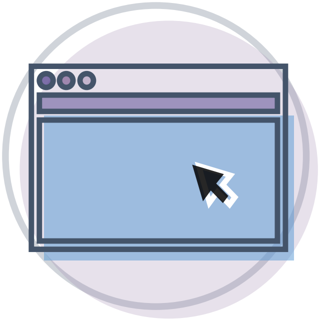
Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit

IT Automation

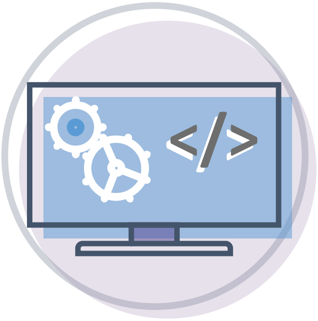
- Policy engines
- Extending AWS services
- Infrastructure management

Common Use Cases



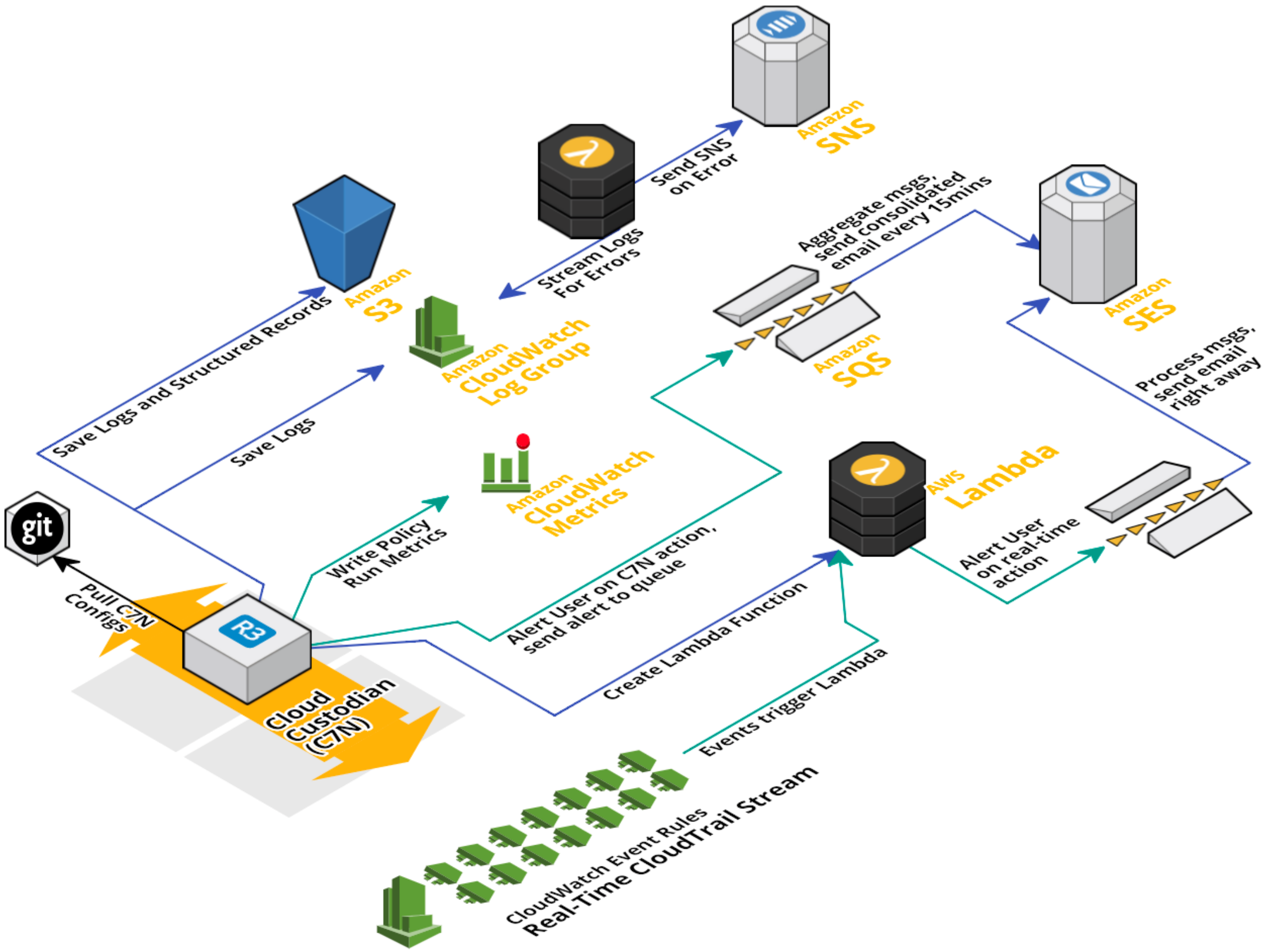
Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express



Backends

- Apps & services
- Mobile
- IoT



nation

ngines

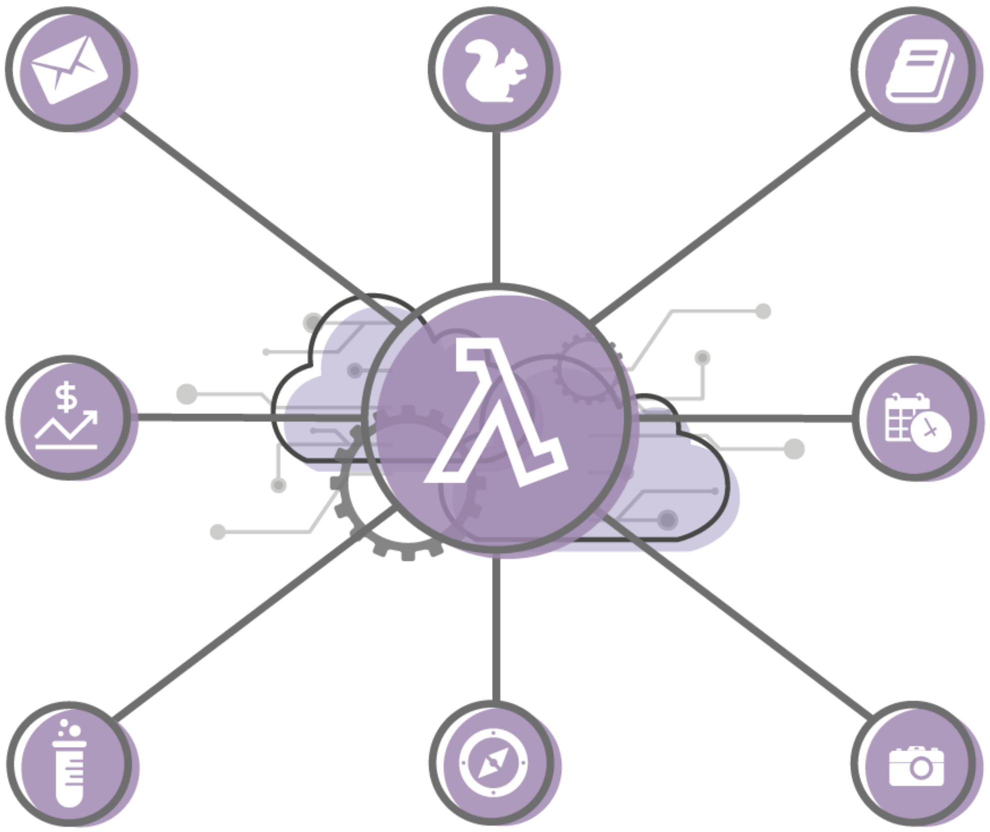
ng AWS

s

ucture

ement

Serverless is a core component of modern apps



Customers innovating with serverless

amazon web services

airbnb
AdRoll
ALT/s
asurion
Atlassian
Benchling
BUSTLE
CITRIX
Coca-Cola
COMCAST
Discovery CHANNEL
Expedia
Experian
FireEye
instacart
LexisNexis
Localytics
Mc Graw Hill
mlbam
New York Public Library
Nextdoor
realtor.com
SCHOLASTIC
The Seattle Times
SQUARE ENIX
THE HONEST CO.
THOMSON REUTERS
vevo
VidRoll
VOGUE
ZAPPROVED
Zillow
23andMe

Enterprises are achieving massive scale with Lambda

- **Thomson Reuters** processes 4,000 requests per second
- **FINRA** processes half a trillion validations of stock trades daily
- **Hearst** reduced the time to ingest and process data for its analytics pipeline by 97%
- **Vevo** can handle spikes of 80x normal traffic
- **Expedia** triggers 1.2 billion Lambda requests each month

Capabilities of a serverless platform

Cloud Logic Layer

Orchestration and State Management

Responsive Data Sources

Application Modeling Framework

Developer Ecosystem

Integrations Library

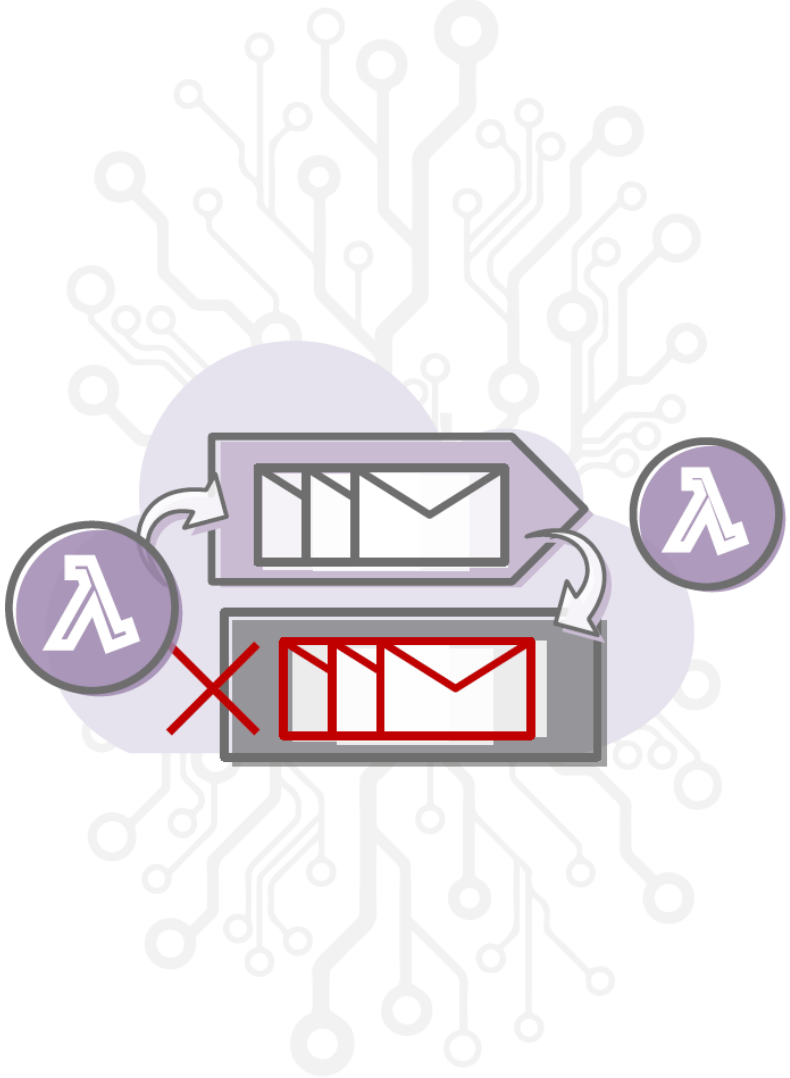
Security and Access Control

Reliability and Performance

Global Scale

amazon web services

Reliability and performance



Dead Letter Queues

- Automatically capture events after exhausting retries
- Build even more reliable event processing applications
- Target Amazon SQS queues or Amazon SNS topics
- Available in all regions

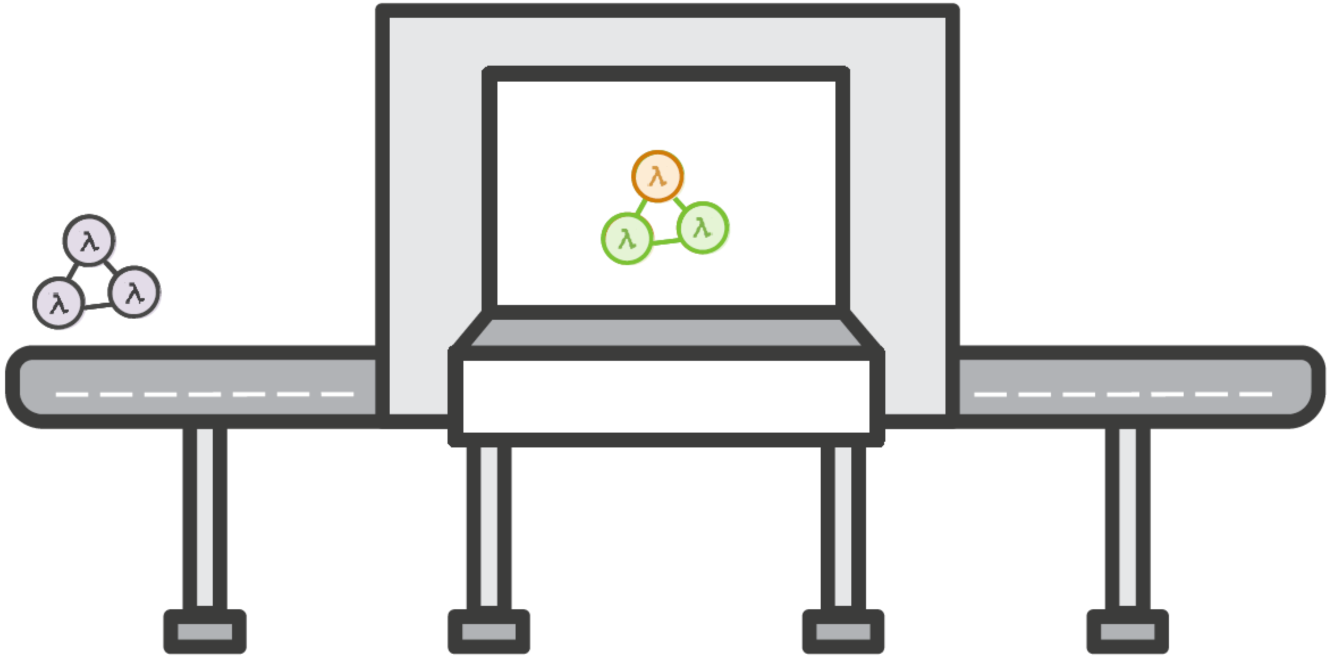
Developer ecosystem — AWS

How do you debug distributed applications made of multiple functions or services?

How do you gain insights into how your functions are performing or behaving?

AWS X-Ray

- Analyze and debug distributed apps in production
- Visualize service call graph of your app
- Identify performance bottlenecks and errors
- Pinpoint service-specific issues
- Identify impact of issues on users of the app
- Trace function executions (preview)



How X-Ray works

Trace Requests

Record Traces

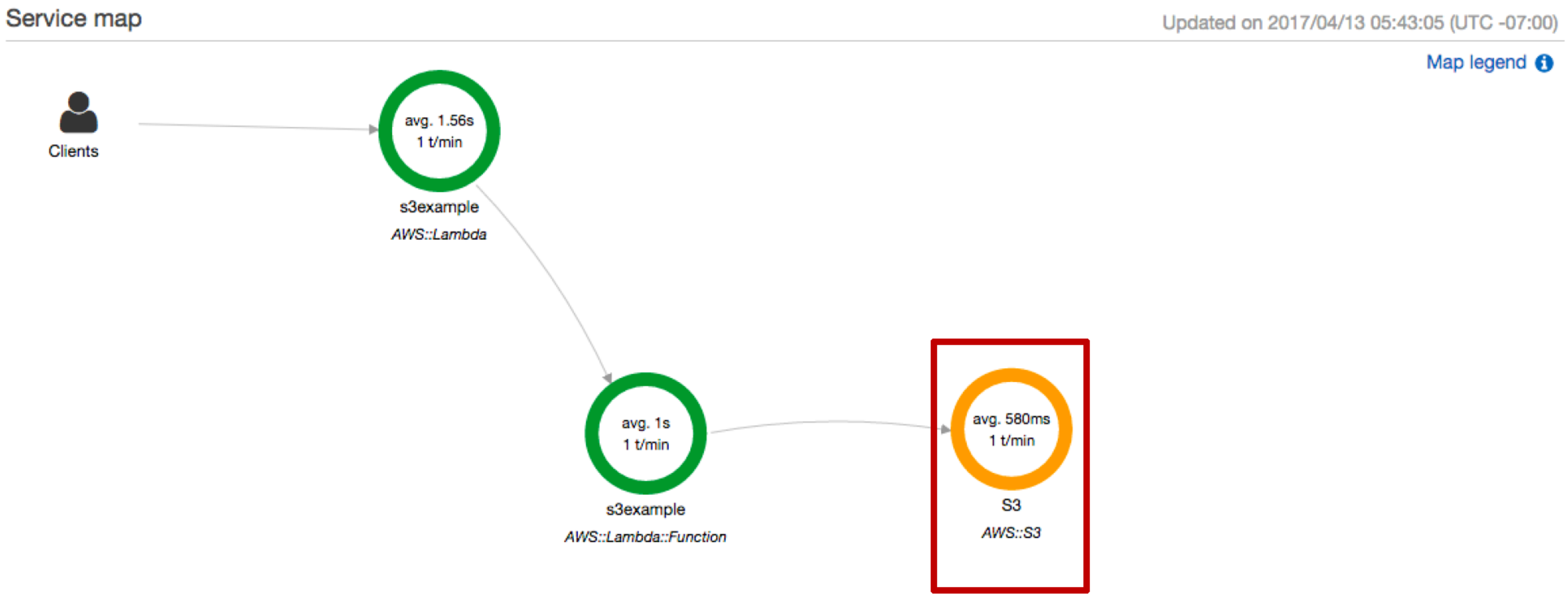
View Service Map

Analyze Issues

X-Ray example

```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));
s3 = new AWS.S3({signatureVersion: 'v4'});

exports.handler = (event, context, callback) => {

    var params = {Bucket: 'tim-example-blucket', Key: 'MyKey', Body: 'Hello!'};

    s3.putObject(params, function(err, data) {});
};
```

X-Ray example

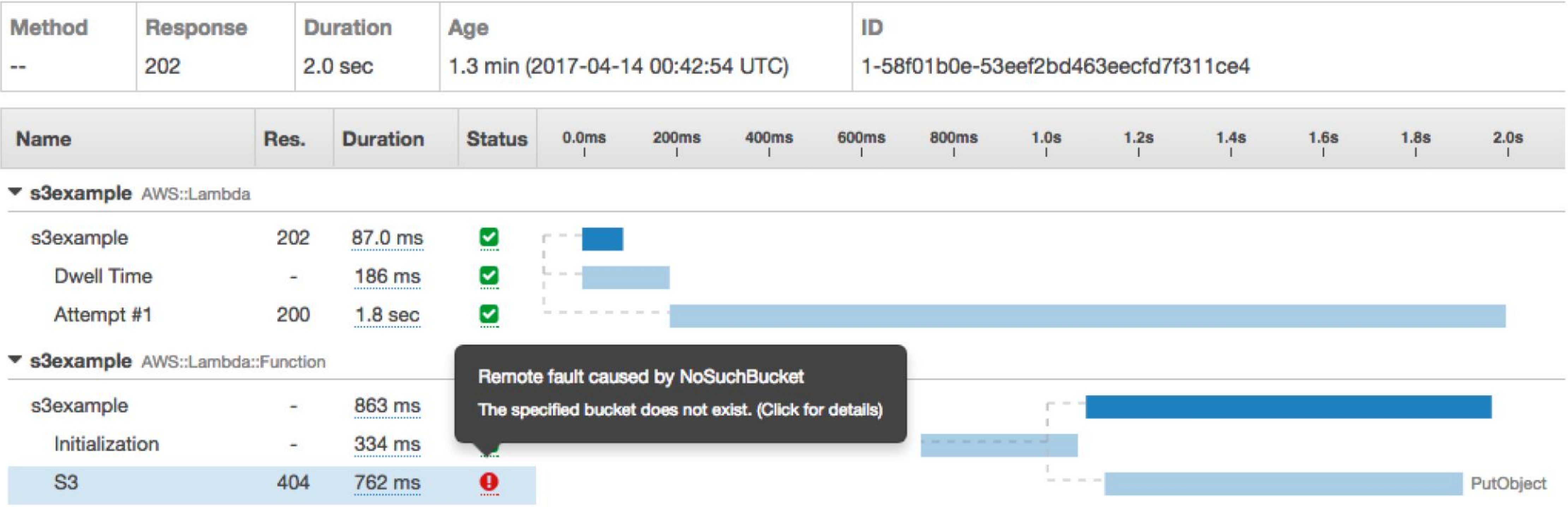


X-Ray example

Method	Response	Duration	Age	ID
--	202	2.0 sec	1.3 min (2017-04-14 00:42:54 UTC)	1-58f01b0e-53eef2bd463eecfd7f311ce4

Name	Res.	Duration	Status	0.0ms 200ms 400ms 600ms 800ms 1.0s 1.2s 1.4s 1.6s 1.8s 2.0s
s3example AWS::Lambda				
s3example	202	87.0 ms	✅	
Dwell Time	-	186 ms	✅	
Attempt #1	200	1.8 sec	✅	
s3example AWS::Lambda::Function				
s3example	-	863 ms	✅	
Initialization	-	334 ms	✅	
S3	404	762 ms	❗	PutObject

X-Ray example



X-Ray example

```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));
s3 = new AWS.S3({signatureVersion: 'v4'});

exports.handler = (event, context, callback) => {

    var params = {Bucket: 'tim-example-blucket', Key: 'MyKey', Body: 'Hello!'};

    s3.putObject(params, function(err, data) {});
};
```

X-Ray example

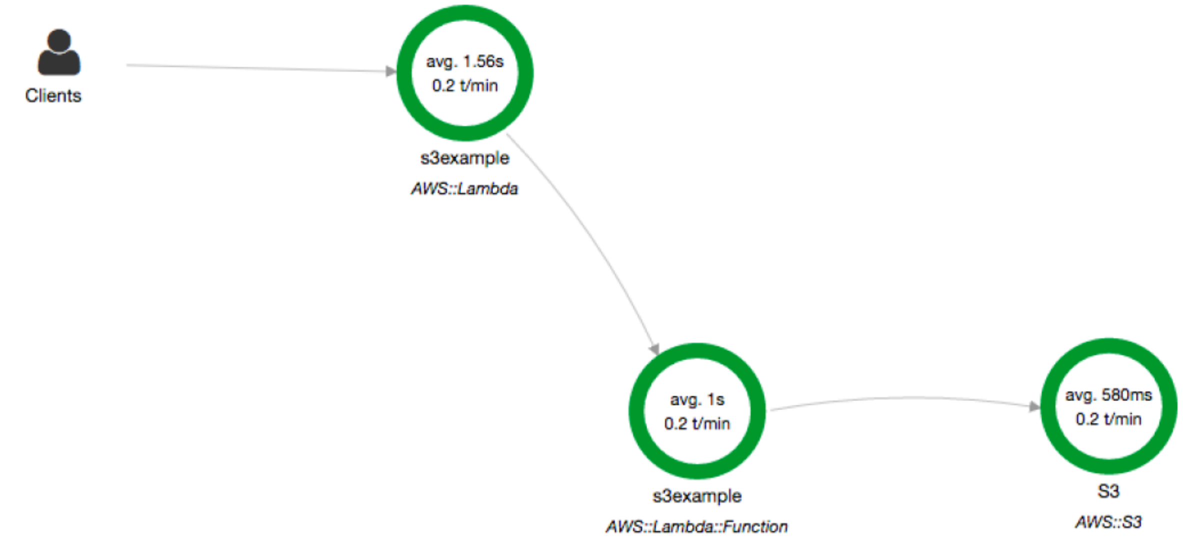
```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));
s3 = new AWS.S3({signatureVersion: 'v4'});

exports.handler = (event, context, callback) => {

    var params = {Bucket: 'tim-example-bucket', Key: 'MyKey', Body: 'Hello!'};

    s3.putObject(params, function(err, data) {});
};
```

X-Ray example

Service map

Updated on 2017/04/13 05:39:27 (UTC -07:00)

Map legend

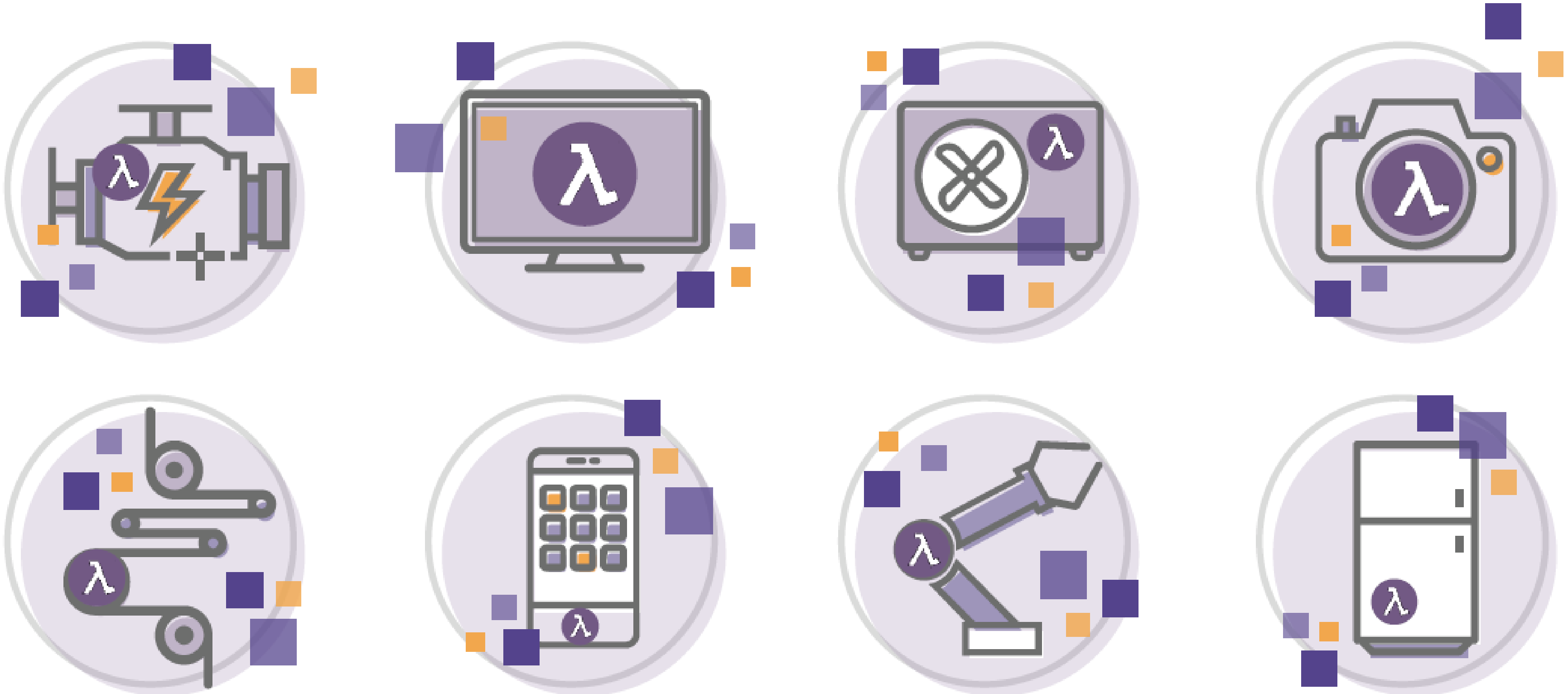


Method	Response	Duration	Age	ID
--	202	1.6 sec	18.1 sec (2017-04-14 00:39:13 UTC)	1-58f01a31-24551f535d0ed5f5a70bdbf2

Name	Res.	Duration	Status	0.0ms 200ms 400ms 600ms 800ms 1.0s 1.2s 1.4s 1.6s
s3example AWS::Lambda				
s3example	202	63.0 ms	✔	
Dwell Time	-	101 ms	✔	
Attempt #1	200	1.5 sec	✔	
s3example AWS::Lambda::Function				
s3example	-	693 ms	✔	
Initialization	-	308 ms	✔	
S3	200	580 ms	✔	PutObject

AWS Greengrass (in preview)

- Extends Lambda functions to devices
- Low latency, near-real time



AWS Snowball Edge

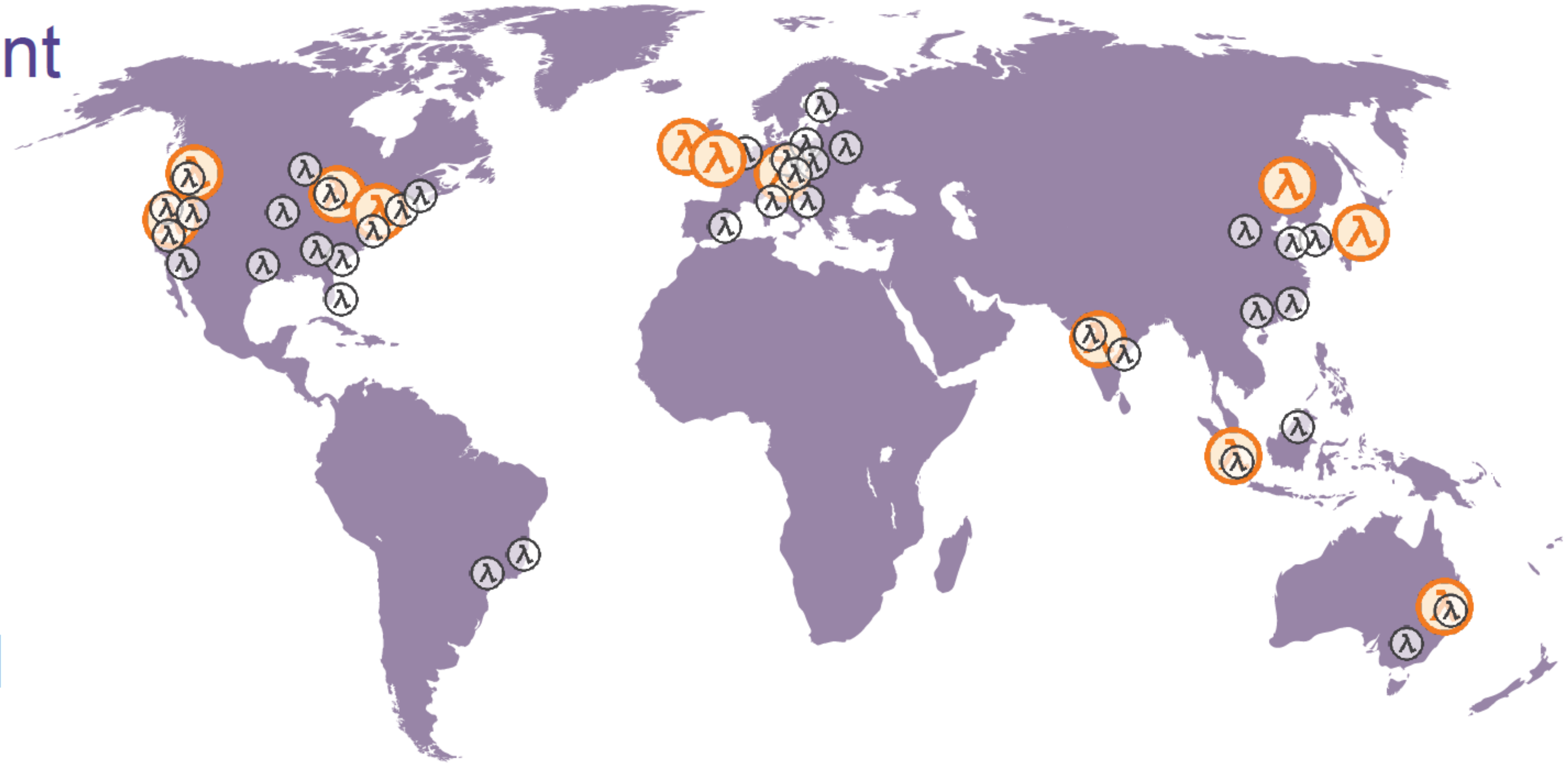
- Petabyte-scale hybrid device with onboard compute and storage

- Deploy AWS Lambda code to Snowball Edge

Lambda@Edge (in preview)

Lambda@Edge is available in all Amazon CloudFront edge locations

- Low-latency request/response customization
- Supports viewer and origin events



Takeaways

Serverless is a Fundamental Component of Modern Applications

- Many enterprise applications can go serverless
- Move to event driven computing

The ecosystem continues to grow

- Tooling, languages, and application capabilities
- But we still have a long ways to go…

Serverless and Edge are technology triggers with the potential to reshape distributed computing and the role of cloud computing