Money-Saving Tips for the Serverless Developer

Yan Cui

@theburningmonk

Yan Cui

@theburningmonk

http://theburningmonk.com

AWS user since 2010

Yan Cui

@theburningmonk

http://theburningmonk.com

Developer Advocate @ lumigo

Yan Cui

@theburningmonk

http://theburningmonk.com

aws serverless HERO

Independent Consultant

TACO BELL

TOYOTA

HARGREAVES LANSDOWN

CYBERARK

Driver & Vehicle Licensing Agency

Schroders

B.

JustGiving

Billing alarms

Everyone should use billing alarms.

They are not perfect. But they can still save your a$$.

Luc van Donkersgoed
@donkersgood

How a single-line bug cost us $2000 in AWS spend...

We recently refactored a Lambda Function. We extensively tested its functionality and released it into production. And everything still worked as expected. But then the billing alarm went off..

(repost with sanitized images)

Luc van Donkersgoed
@donkersgood

How a single-line bug cost us $2000 in AWS spend...

We recently refactored a Lambda Function. We extensively tested its functionality and released it into production. And everything still worked as expected. But then the billing alarm went off..

(repost with sanitized images)

EBEAlerting APP 18:50

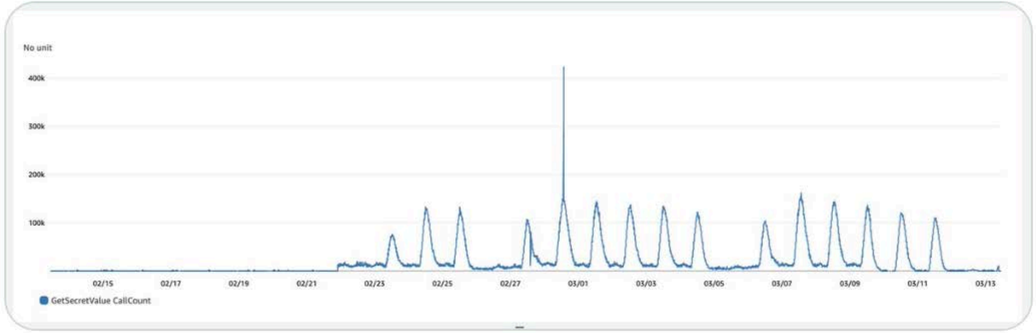
CloudWatch Alarm | ALARM: "prod_EBE_BE_ALM_1007_BILLING_ALARM" in EU (Ireland) 🔴

Account ID: 4 1

Description:
AWS EstimatedCharges (for the current month) exceeds the limit we have set for the EBE Backend account.
- Environment=prod

Luc van Donkersgoed @donkersgood · Mar 13, 2023

The old version cached the credentials outside the event handler, and only retrieved them on cold starts or 401/403 errors. The new version retrieved them on every invocation. Result: 176 million calls to SecretsManager:GetSecretValue 🤯



Luc van Donkersgoed
@donkersgood

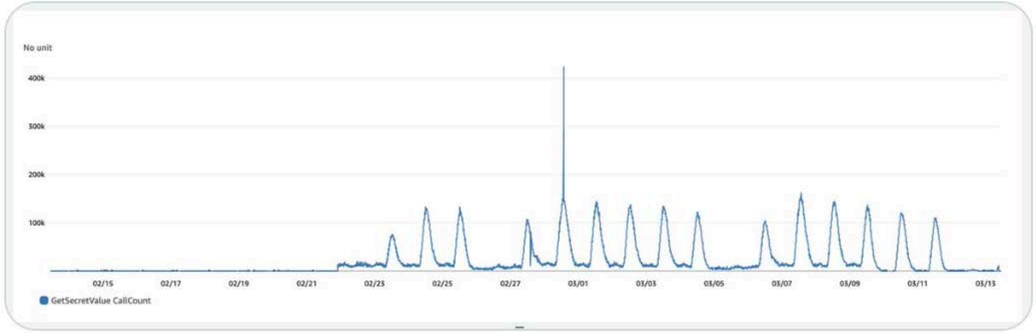
How a single-line bug cost us $2000 in AWS spend...

We recently refactored a Lambda Function. We extensively tested its functionality and released it into production. And everything still worked as expected. But then the billing alarm went off..

(repose with sanitized images)

Luc van Donkersgoed @donkersgood · Mar 13, 2023

The old version cached the credentials outside the event handler, and only retrieved them on cold starts or 401/403 errors. The new version retrieved them on every invocation. Result: 176 million calls to SecretsManager:GetSecretValue 🤯



Luc van Donkersgoed @donkersgood · Mar 13, 2023

So what have we learned:
1. Monitor AWS spend anomalies after deployments
2. Write unit tests for Lambda cache behavior
3. Don't discard symptoms like Lambda timeouts as random behavior
4. Billing alerts work!

/fin

Keeping logging cost under control

CloudWatch often costs much more than your actual application.

As cost goes up, value goes down.

How to keep CloudWatch Logs cost under control

1. Do structured logging.

How to keep CloudWatch Logs cost under control

1. Do structured logging.

DEBUG	Detailed events for debugging application.
INFO	General information that highlights progress of application.
WARN	Potential problems, but doesn’t stop application from working.
ERROR	Issues that require immediate attention.

How to keep CloudWatch Logs cost under control

1. Do structured logging. Log at INFO or above in production.

DEBUG	Detailed events for debugging application.
INFO	General information that highlights progress of application.
WARN	Potential problems, but doesn’t stop application from working.
ERROR	Issues that require immediate attention.

How to keep CloudWatch Logs cost under control

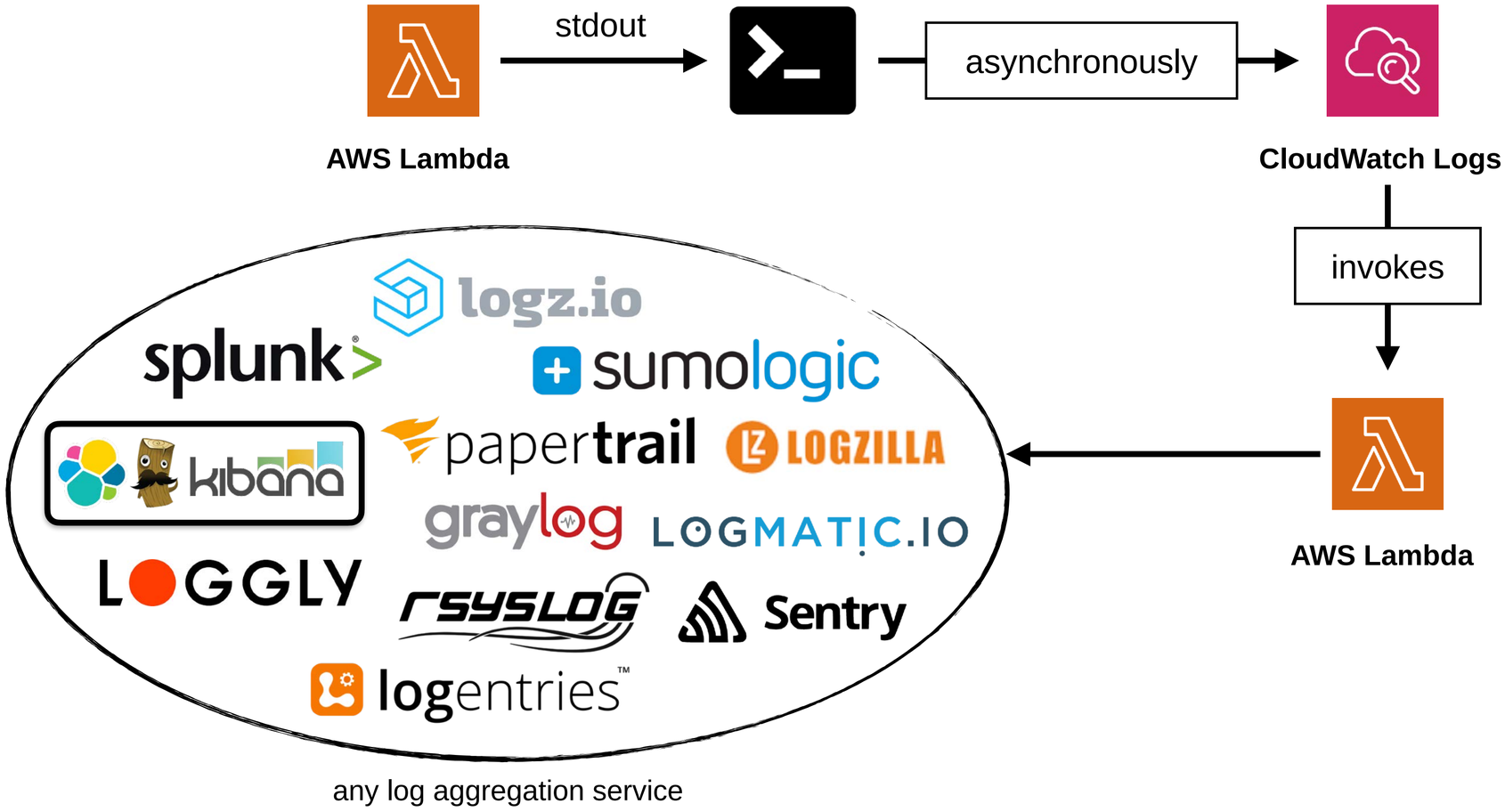
1. Do structured logging. Log at INFO or above in production.

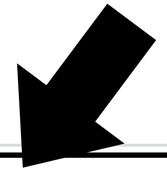
2. Sample DEBUG logs in production. e.g. 5% of invocations.

How to keep CloudWatch Logs cost under control

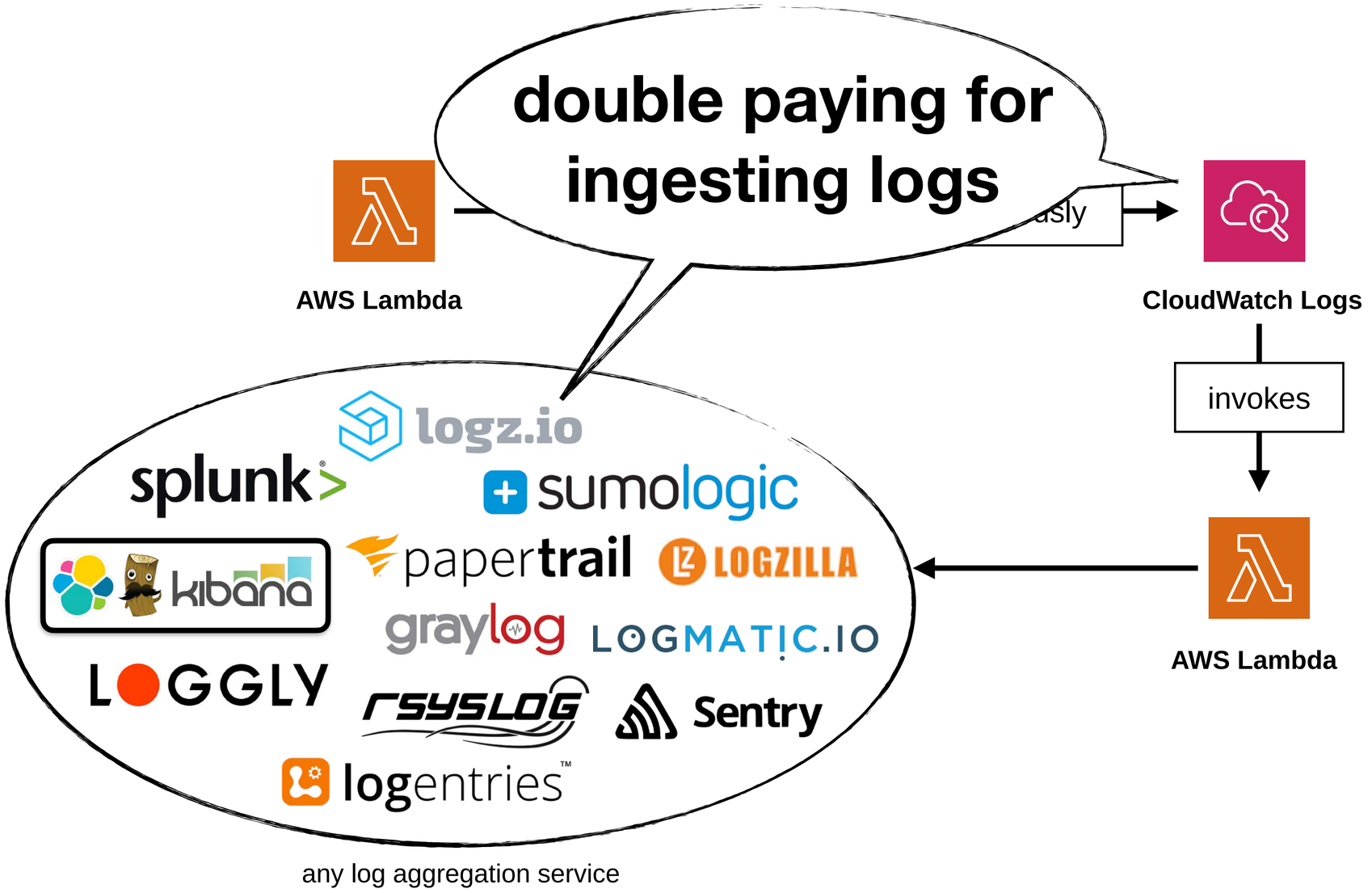
1. Do structured logging. Log at INFO or above in production.
2. Sample DEBUG logs in production. e.g. 5% of invocations.
3. Set log retention to 30 days.

Manage Logs	Price
Collect (Data Ingestion)	
Standard	$0.50 per GB
Infrequent Access	$0.25 per GB
Store (Archival)	$0.03 per GB
Analyze (Logs Insights queries)	$0.005 per GB of data scanned
Detect and Mask (Data Protection)	$0.12 per GB of data scanned
Analyze (Live Tail)	$0.01 per minute

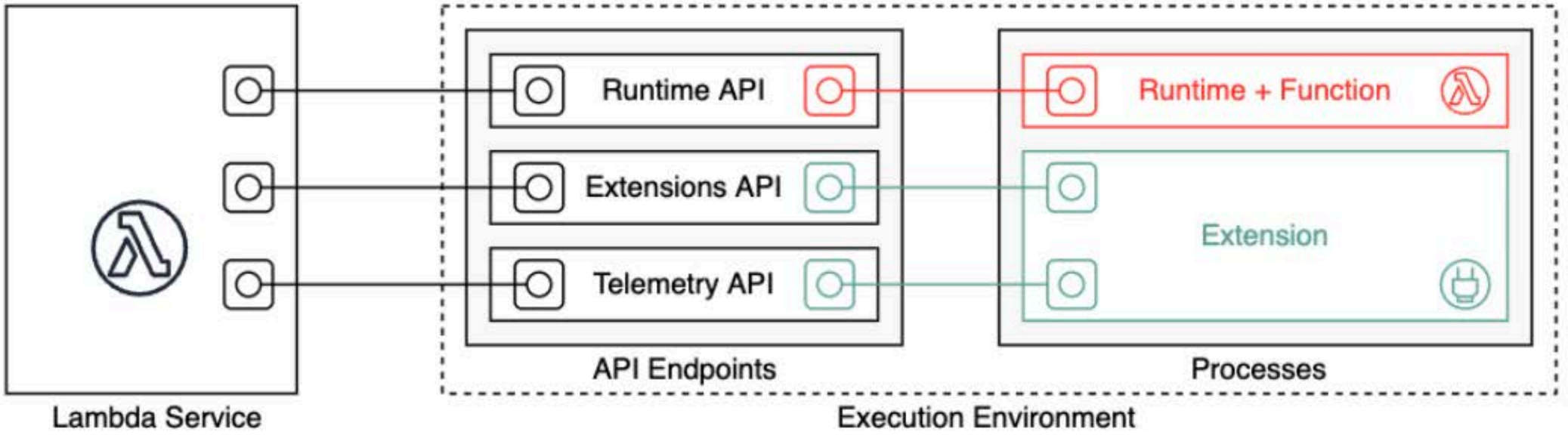

stdout
asynchronously
AWS Lambda
CloudWatch Logs
invokes
AWS Lambda
logz.io
splunk>
sumologic
kibana
papertrail
LOGZILLA
graylog
LOGMATIC.IO
LOGGLY
rsyslog
Sentry
logentries
any log aggregation service



Manage Logs	Price
Collect (Data Ingestion)	
Standard	$0.50 per GB
Infrequent Access	$0.25 per GB
Store (Archival)	$0.03 per GB
Analyze (Logs Insights queries)	$0.005 per GB of data scanned
Detect and Mask (Data Protection)	$0.12 per GB of data scanned
Analyze (Live Tail)	$0.01 per minute


double paying for ingesting logs
AWS Lambda
CloudWatch Logs
invokes
AWS Lambda
logz.io
splunk>
sumologic
papertrail
LOGZILLA
kibana
graylog
LOGMATIC.IO
LOGGLY
rsyslog
Sentry
logentries
any log aggregation service

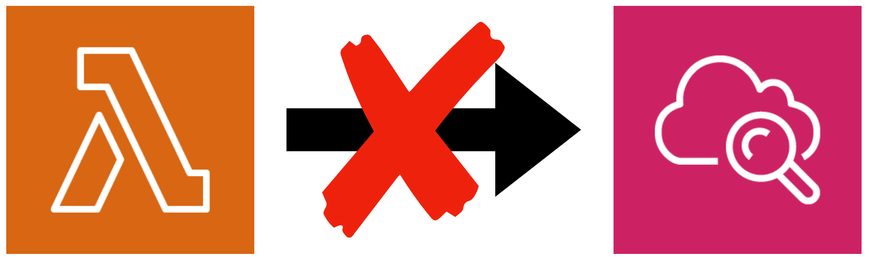
Lambda Extensions + Telemetry API



https://docs.aws.amazon.com/lambda/latest/dg/telemetry-api.html

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}
```



AWS Lambda **CloudWatch Logs**

Lumigo Launches Log Management, Reducing Issue Resolution Time While Slashing Costs Up to 60%

Lumigo has announced the addition of full log management capabilities into its microservices observability and troubleshooting platform, promising customers enormous savings while enabling automatic correlation between log data and distributed traces.

By unifying logs, metrics, and traces into a single interface, Lumigo empowers developers and DevOps teams with comprehensive context for analyzing and resolving issues swiftly. It reduces the time spent on root cause analysis by 80% while dramatically cutting costs. With Lumigo, troubleshooting becomes fast, efficient, and cost-effective, delivering unparalleled visibility across the entire stack. Users can seamlessly search and analyze logs and click directly into the corresponding traces, accelerating resolution times while enjoying significant cost savings.

https://lumigo.io/blog/lumigo-launches-log-management

Remember system messages

```
2024-09-16T21:10:00.599Z                START RequestId: e9ae772a-993d-4a7b-942c-40c150602b9e Version: $LATEST

START RequestId: e9ae772a-993d-4a7b-942c-40c150602b9e Version: $LATEST

2024-09-16T21:10:00.600Z                [redacted]

2024-09-16T21:10:00.602Z                END RequestId: e9ae772a-993d-4a7b-942c-40c150602b9e

END RequestId: e9ae772a-993d-4a7b-942c-40c150602b9e

2024-09-16T21:10:00.602Z                REPORT RequestId: e9ae772a-993d-4a7b-942c-40c150602b9e Duration: 2.73 ms Bil

REPORT RequestId: e9ae772a-993d-4a7b-942c-40c150602b9e  Duration: 2.73 ms       Billed Duration: 3 ms   Memory Si
Memory Used: 158 MB
```

Jack Ellis
@JackEllis

fathom analytics

I can't believe we were paying $1,000/month for this BS.

Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Actions | Start tailing | Create metric filter

Filter events | Clear | 1m | 30m | 1h | 12h | Custom | Local timezone | Display

Timestamp	Message
	No more records within selected time range Retry
2024-01-15T14:06:02.221-06:00	INIT_START Runtime Version: provided:al2.v28 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:c2f9d23694707c1161c0d3fbc0673f4ee14150265f4f3081161948aa71fad0a0
2024-01-15T14:06:03.354-06:00	START RequestId: e97b41c1-85db-5941-8091-1fde17b1f1a4 Version: 121
2024-01-15T14:06:04.562-06:00	END RequestId: e97b41c1-85db-5941-8091-1fde17b1f1a4
2024-01-15T14:06:04.562-06:00	REPORT RequestId: e97b41c1-85db-5941-8091-1fde17b1f1a4 Duration: 1207.97 ms Billed Duration: 2339 ms Memory Size: 512 MB Max Memory Used: 126 MB Init Duration: 1131.00 ms
2024-01-15T14:06:04.570-06:00	START RequestId: 13ece6c8-54ec-5a3e-8e9e-b8d7c680eb5c Version: 121
2024-01-15T14:06:04.610-06:00	END RequestId: 13ece6c8-54ec-5a3e-8e9e-b8d7c680eb5c
2024-01-15T14:06:04.610-06:00	REPORT RequestId: 13ece6c8-54ec-5a3e-8e9e-b8d7c680eb5c Duration: 40.37 ms Billed Duration: 41 ms Memory Size: 512 MB Max Memory Used: 128 MB
2024-01-15T14:06:04.614-06:00	START RequestId: cb11e96a-e184-553c-8e25-b77573e828a3 Version: 121
2024-01-15T14:06:05.066-06:00	END RequestId: cb11e96a-e184-553c-8e25-b77573e828a3
2024-01-15T14:06:05.066-06:00	REPORT RequestId: cb11e96a-e184-553c-8e25-b77573e828a3 Duration: 451.64 ms Billed Duration: 452 ms Memory Size: 512 MB Max Memory Used: 129 MB
2024-01-15T14:06:05.081-06:00	START RequestId: 362772e8-4b0e-5926-ad71-2cf7d9627300 Version: 121
2024-01-15T14:06:05.203-06:00	END RequestId: 362772e8-4b0e-5926-ad71-2cf7d9627300
2024-01-15T14:06:05.203-06:00	REPORT RequestId: 362772e8-4b0e-5926-ad71-2cf7d9627300 Duration: 121.91 ms Billed Duration: 122 ms Memory Size: 512 MB Max Memory Used: 129 MB
2024-01-15T14:06:05.225-06:00	START RequestId: d72de4e2-7fcc-5d80-96e4-bafdeda1a2be Version: 121
2024-01-15T14:06:05.447-06:00	END RequestId: d72de4e2-7fcc-5d80-96e4-bafdeda1a2be
2024-01-15T14:06:05.447-06:00	REPORT RequestId: d72de4e2-7fcc-5d80-96e4-bafdeda1a2be Duration: 221.77 ms Billed Duration: 222 ms Memory Size: 512 MB Max Memory Used: 130 MB
2024-01-15T14:06:05.496-06:00	START RequestId: f7ec79b3-048d-5b9e-b1aa-27ac96e0e8a2 Version: 121

Introducing advanced logging controls for AWS Lambda functions

by David Boyne | on 16 NOV 2023 | in Amazon CloudWatch, AWS Lambda, Serverless | Permalink | Share

This post is written by Nati Goldberg, Senior Solutions Architect and Shridhar Pandey, Senior Product Manager, AWS Lambda

Today, AWS is launching advanced logging controls for AWS Lambda, giving developers and operators greater control over how function logs are captured, processed, and consumed.

This launch introduces three new capabilities to provide a simplified and enhanced default logging experience on Lambda.

First, you can capture Lambda function logs in JSON structured format without having to use your own logging libraries. JSON structured logs make it easier to search, filter, and analyze large volumes of log entries.

Second, you can control the log level granularity of Lambda function logs without making any code changes, enabling more effective debugging and troubleshooting.

Third, you can also set which Amazon CloudWatch log group Lambda sends logs to, making it easier to aggregate and manage logs at scale.

AWS::Lambda::Function LoggingConfig

RSS

Filter View All

The function's Amazon CloudWatch Logs configuration settings.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

JSON

```
{
  "ApplicationLogLevel" : String,
  "LogFormat" : String,
  "LogGroup" : String,
  "SystemLogLevel" : String
}
```

`LogFormat`

The format in which Lambda sends your function's application and system logs to CloudWatch. Select between plain text and structured JSON.

Required: No

Type: String

Allowed values: `Text | JSON`

Update requires: No interruption

`SystemLogLevel`

Set this property to filter the system logs for your function that Lambda sends to CloudWatch. Lambda only sends system logs at the selected level of detail and lower, where `DEBUG` is the highest level and `WARN` is the lowest.

Required: No

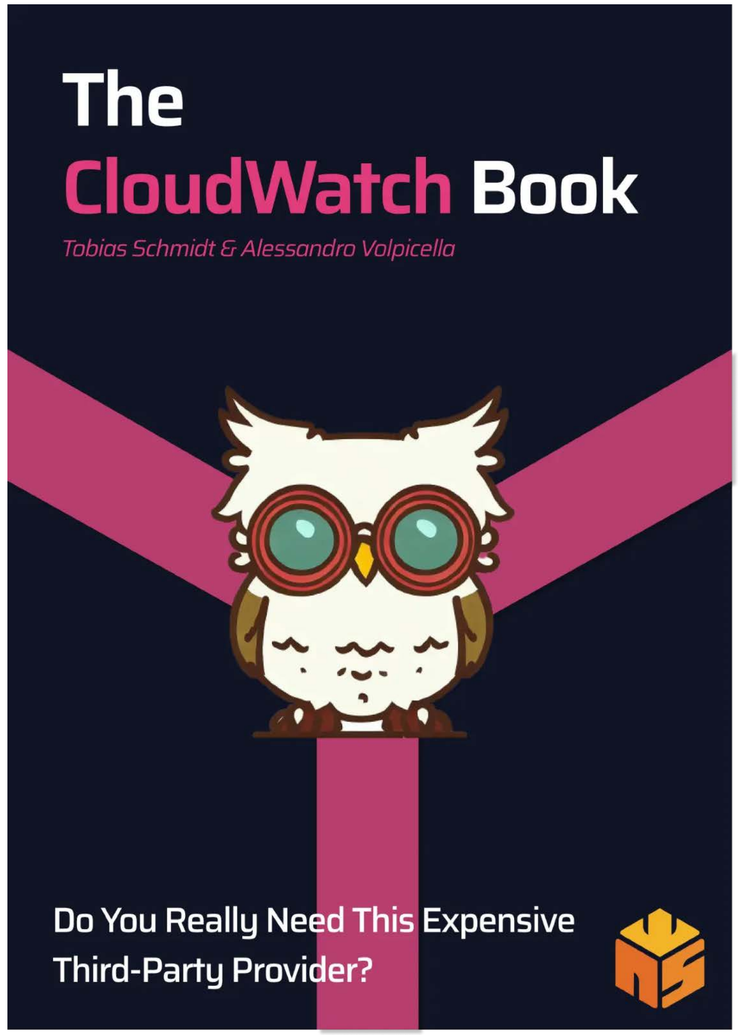
Type: String

Allowed values: `DEBUG | INFO | WARN`

Update requires: No interruption

	DEBUG	INFO	WARN
platform.initStart	✓	✓	✗
platform.initRuntimeDone	✓	✗	✗
platform.initReport	✓	✓	✗
platform.start	✓	✗	✗
platform.runtimeDone	✓	✗	✗
platform.report	✓	✓	✗
unhandled exception	✓	✓	✓

	DEBUG	INFO	WARN
platform.initStart	✓	✓	✗
platform.initRuntimeDone	✓	✗	✗
platform.initReport	✓	✓	✗
platform.start	✓	✗	✗
platform.runtimeDone	✓	✗	✗
platform.report	✓	✓	✗
unhandled exception	✓	✓	✓



cloudwatchbook.com

“definitive guide for learning CloudWatch”

- me

Sandro Volpicella

Tobias Schmidt

Right-sizing Lambda memory

More memory = more CPU = more network bandwidth

Memory (MB)	Price per 1ms
128	$0.0000000021
512	$0.0000000083
1024	$0.0000000167
1536	$0.0000000250
2048	$0.0000000333
3072	$0.0000000500
4096	$0.0000000667
5120	$0.0000000833
6144	$0.0000001000
7168	$0.0000001167
8192	$0.0000001333
9216	$0.0000001500
10240	$0.0000001667

Easy to be wrong by an order of magnitude.

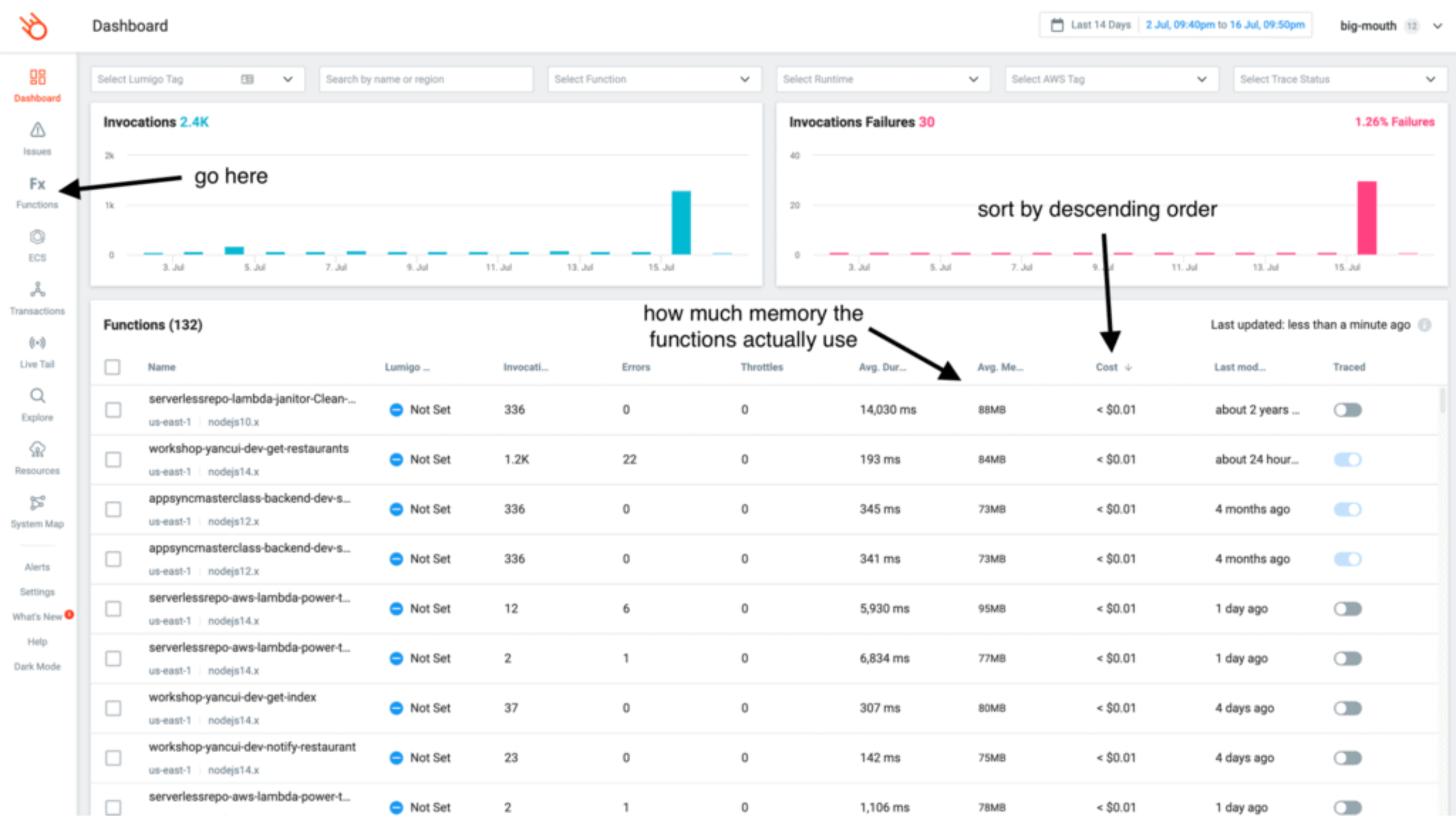

STORY TIME

We should forget about small efficiencies, say about 97% of the time.

Premature optimization is the root of all evil.

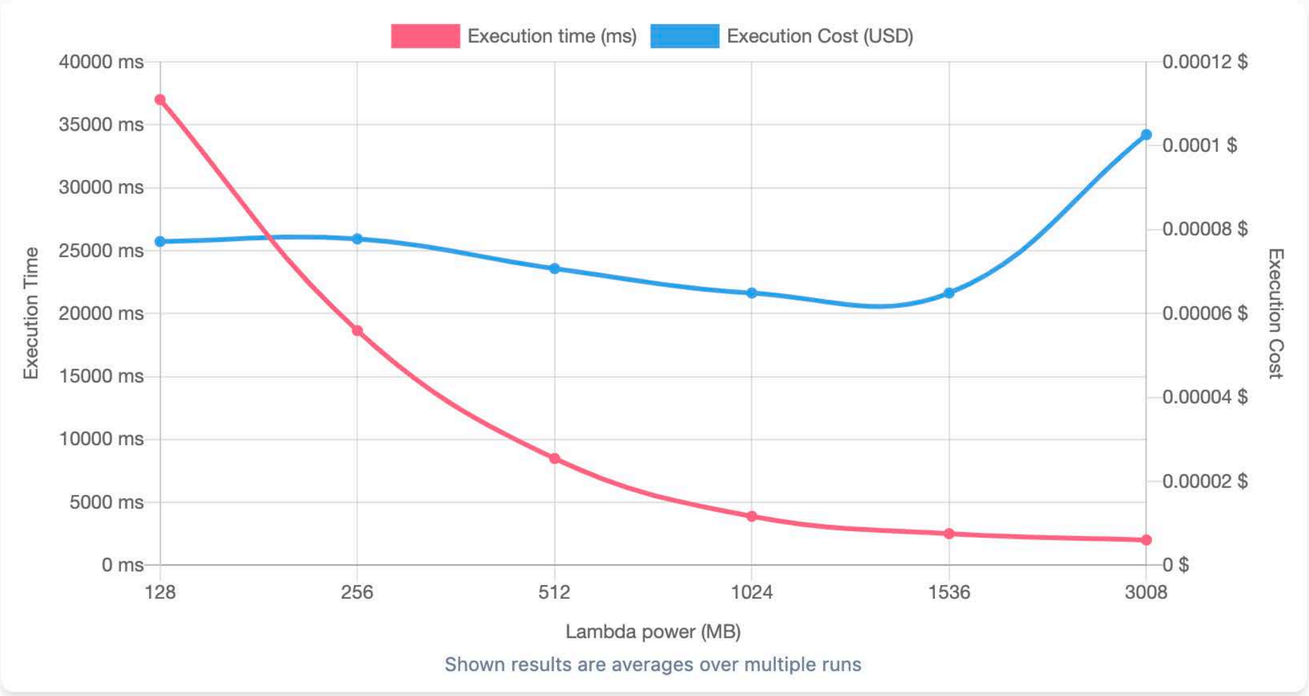
Yet we should not pass up our opportunities in that critical 3%.

Donald Knuth


Dashboard
Last 14 Days
2 Jul, 09:40pm to 16 Jul, 09:50pm
big-mouth
Dashboard
Issues
Functions
ECS
Transactions
Live Tail
Explore
Resources
System Map
Alerts
Settings
What's New
Help
Dark Mode
Select Lumigo Tag
Search by name or region
Select Function
Select Runtime
Select AWS Tag
Select Trace Status
Invocations 2.4K
Invocations Failures 30
1.26% Failures
go here
sort by descending order
how much memory the functions actually use
Functions (132)
Last updated: less than a minute ago
Name
Lumigo ...
Invocati...
Errors
Throttles
Avg. Dur...
Avg. Me...
Cost
Last mod...
Traced
serverlessrepo-lambda-janitor-Clean-...
us-east-1 | nodejs10.x
Not Set
336
0
0
14,030 ms
88MB
< $0.01
about 2 years ...
workshop-yancui-dev-get-restaurants
us-east-1 | nodejs14.x
Not Set
1.2K
22
0
193 ms
84MB
< $0.01
about 24 hour...
appsyncmasterclass-backend-dev-s...
us-east-1 | nodejs12.x
Not Set
336
0
0
345 ms
73MB
< $0.01
4 months ago
appsyncmasterclass-backend-dev-s...
us-east-1 | nodejs12.x
Not Set
336
0
0
341 ms
73MB
< $0.01
4 months ago
serverlessrepo-aws-lambda-power-t...
us-east-1 | nodejs14.x
Not Set
12
6
0
5,930 ms
95MB
< $0.01
1 day ago
serverlessrepo-aws-lambda-power-t...
us-east-1 | nodejs14.x
Not Set
2
1
0
6,834 ms
77MB
< $0.01
1 day ago
workshop-yancui-dev-get-index
us-east-1 | nodejs14.x
Not Set
37
0
0
307 ms
80MB
< $0.01
4 days ago
workshop-yancui-dev-notify-restaurant
us-east-1 | nodejs14.x
Not Set
23
0
0
142 ms
75MB
< $0.01
4 days ago
serverlessrepo-aws-lambda-power-t...
Not Set
2
1
0
1,106 ms
78MB
< $0.01
1 day ago

Right-sizing Lambda functions

AWS Lambda Power Tuning Results



Best Cost
1536MB

Best Time
3008MB

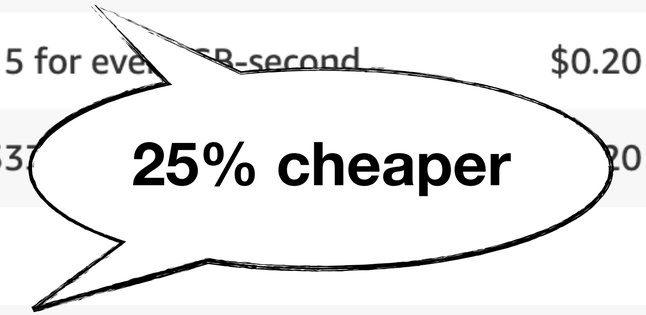
Worst Cost
3008MB

Worst Time
128MB

https://github.com/alexcasalboni/aws-lambda-power-tuning

Use ARM architecture

Architecture	Duration	Requests
x86 Price		
First 6 Billion GB-seconds / month	$0.0000166667 for every GB-second	$0.20 per 1M requests
Next 9 Billion GB-seconds / month	$0.000015 for eve[illegible]B-second	$0.20 per 1M requests
Over 15 Billion GB-seconds / month	$0.00001333[illegible]	[illegible]0 per 1M requests
Arm Price		
First 7.5 Billion GB-seconds / month	$0.0000133334 for every GB-second	$0.20 per 1M requests
Next 11.25 Billion GB-seconds / month	$0.0000120001 for every GB-second	$0.20 per 1M requests
Over 18.75 Billion GB-seconds / month	$0.0000106667 for every GB-second	$0.20 per 1M requests



Performance may vary…

Best for functions with a lot of IO wait time.

No lambda-to-lambda invocations

Invoke

PDF | RSS

Invokes a Lambda function. You can invoke a function synchronously (and wait for the response), or asynchronously. By default, Lambda invokes your function synchronously (i.e. the `InvocationType` is `RequestResponse`). To invoke a function asynchronously, set `InvocationType` to `Event`. Lambda passes the `ClientContext` object to your function for synchronous invocations only.

For synchronous invocation, details about the function response, including errors, are included in the response body and headers. For either invocation type, you can find more information in the execution log and trace.

When an error occurs, your function may be invoked multiple times. Retry behavior varies by error type, client, event source, and invocation type. For example, if you invoke a function asynchronously and it returns an error, Lambda executes the function up to two more times. For more information, see Error handling and automatic retries in Lambda.

For asynchronous invocation, Lambda adds events to a queue before sending them to your function. If your function does not have enough capacity to keep up with the queue, events may be lost. Occasionally, your function may receive the same event multiple times, even if no error occurs. To retain events that were not processed, configure your function with a dead-letter queue.

InvocationType

Choose from the following options.

- `RequestResponse` (default) – Invoke the function synchronously. Keep the connection open until the function returns a response or times out. The API response includes the function response and additional data.
- `Event` – Invoke the function asynchronously. Send events that fail multiple times to the function's dead-letter queue (if one is configured). The API response only includes a status code.
- `DryRun` – Validate parameter values and verify that the user or role has permission to invoke the function.

Valid Values: `Event | RequestResponse | DryRun`

InvocationType

Choose from the following options.

- `RequestResponse` (default) – Invoke the function synchronously. Keep the connection open until the function returns a response or times out. The API response includes the function response and additional data.
- `Event` – Invoke the function asynchronously. Send events that fail multiple times to the function's dead-letter queue (if one is configured). The API response only includes a status code.
- `DryRun` – Validate parameter values and verify that the user or role has permission to invoke the function.

Valid Values: `Event | RequestResponse | DryRun`

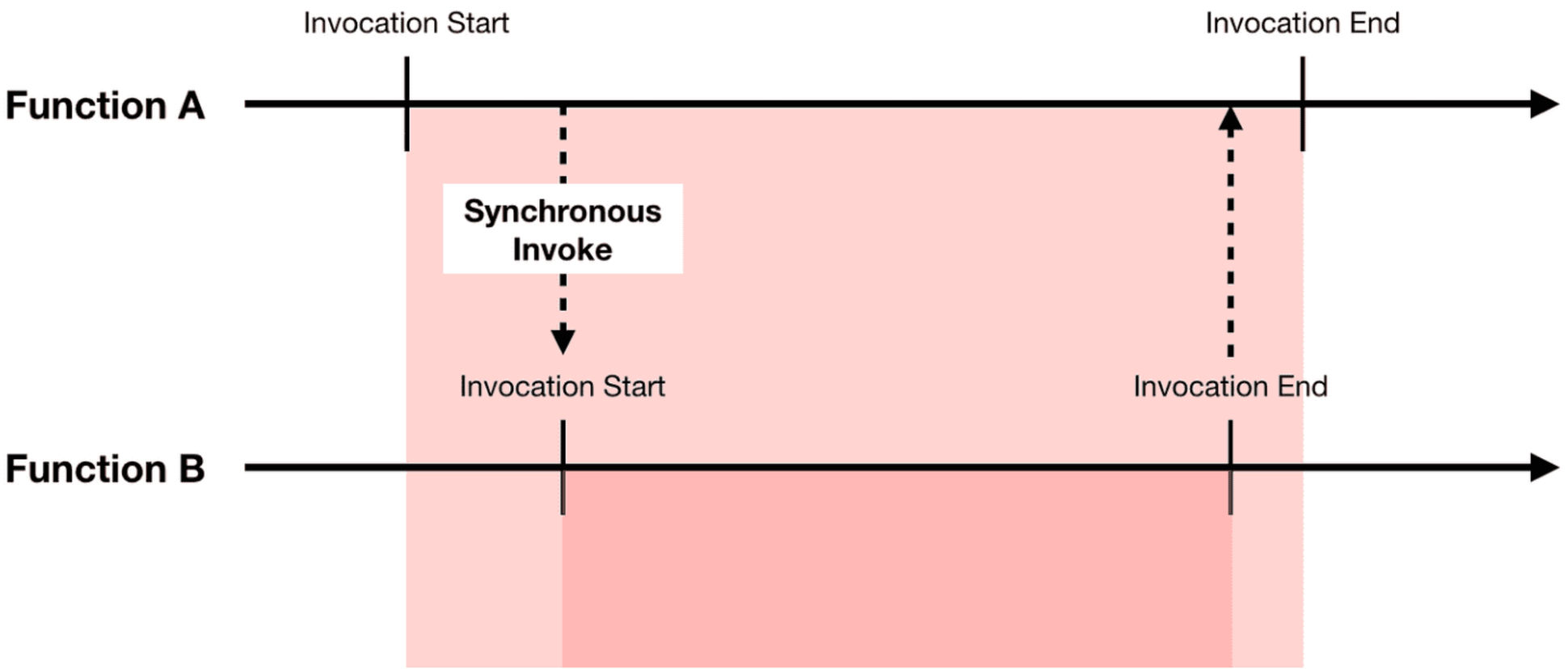
InvocationType

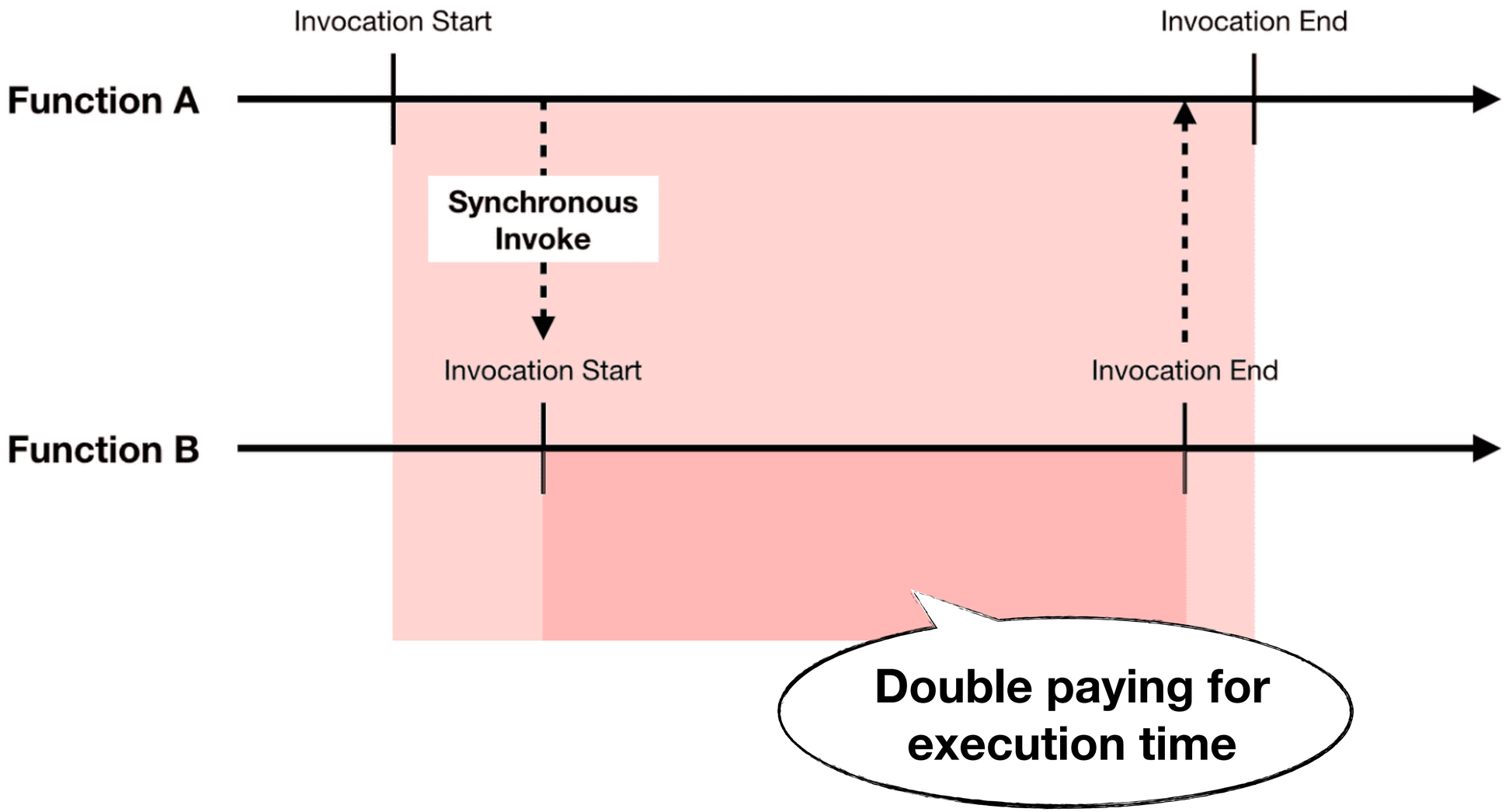
Choose from the following options.

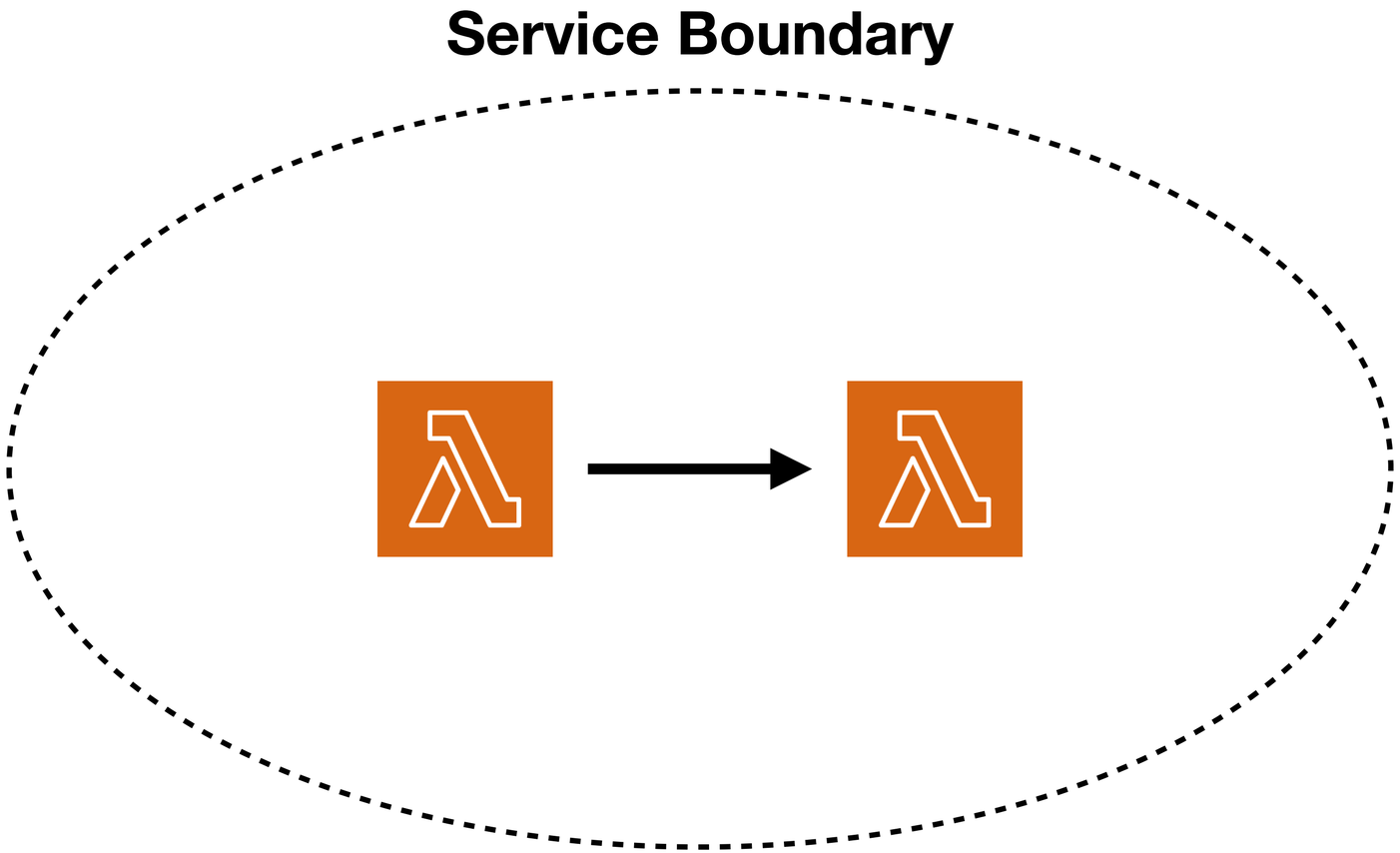
- `RequestResponse` (default) – Invoke the function synchronously. Keep the connection open until the function returns a response or times out. The API response includes the function response and additional data.
- `Event` – Invoke the function asynchronously. Send events that fail multiple times to the function's dead-letter queue (if one is configured). The API response only includes a status code.
- `DryRun` – Validate parameter values and verify that the user or role has permission to invoke the function.

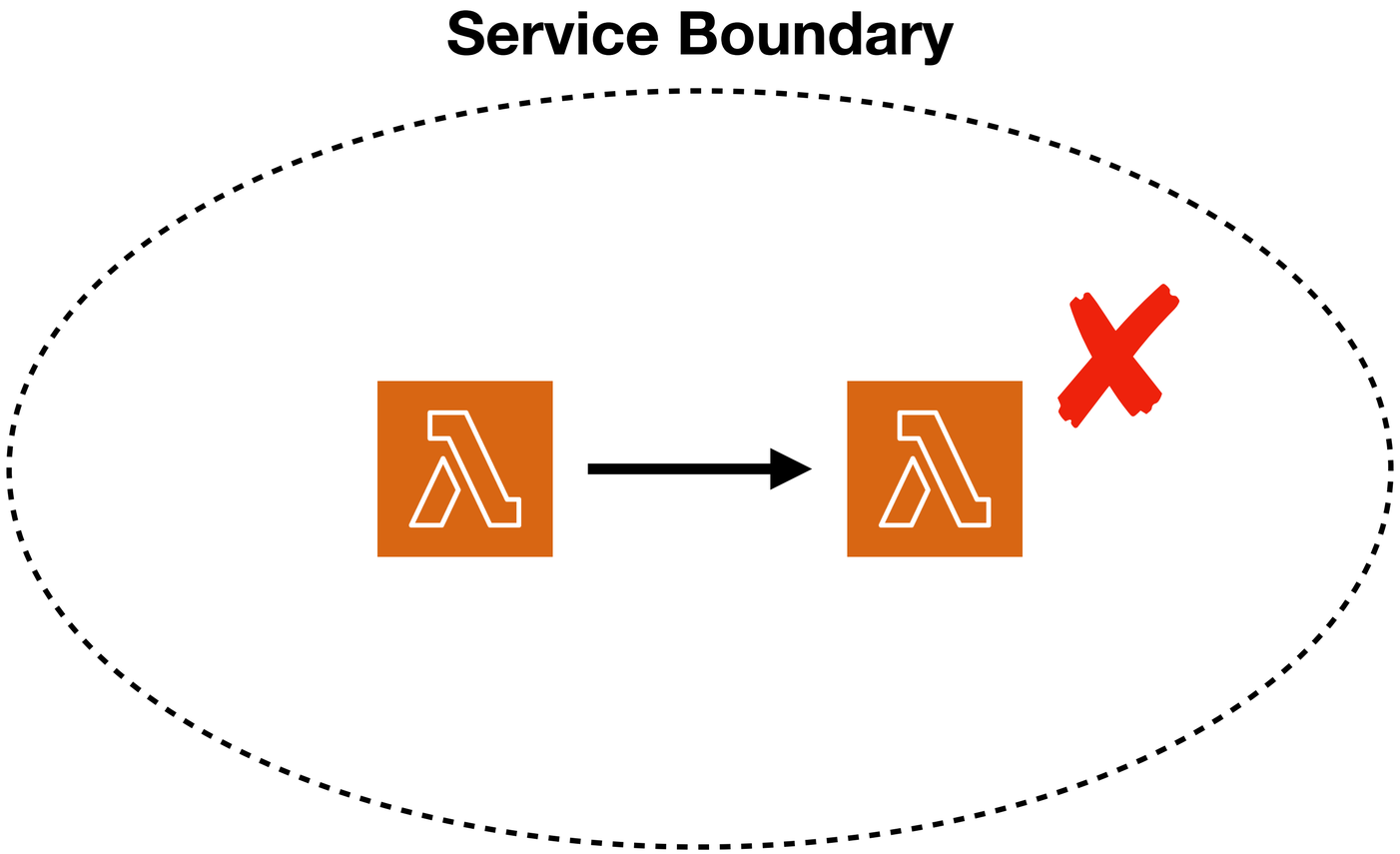
Valid Values: `Event | RequestResponse | DryRun`

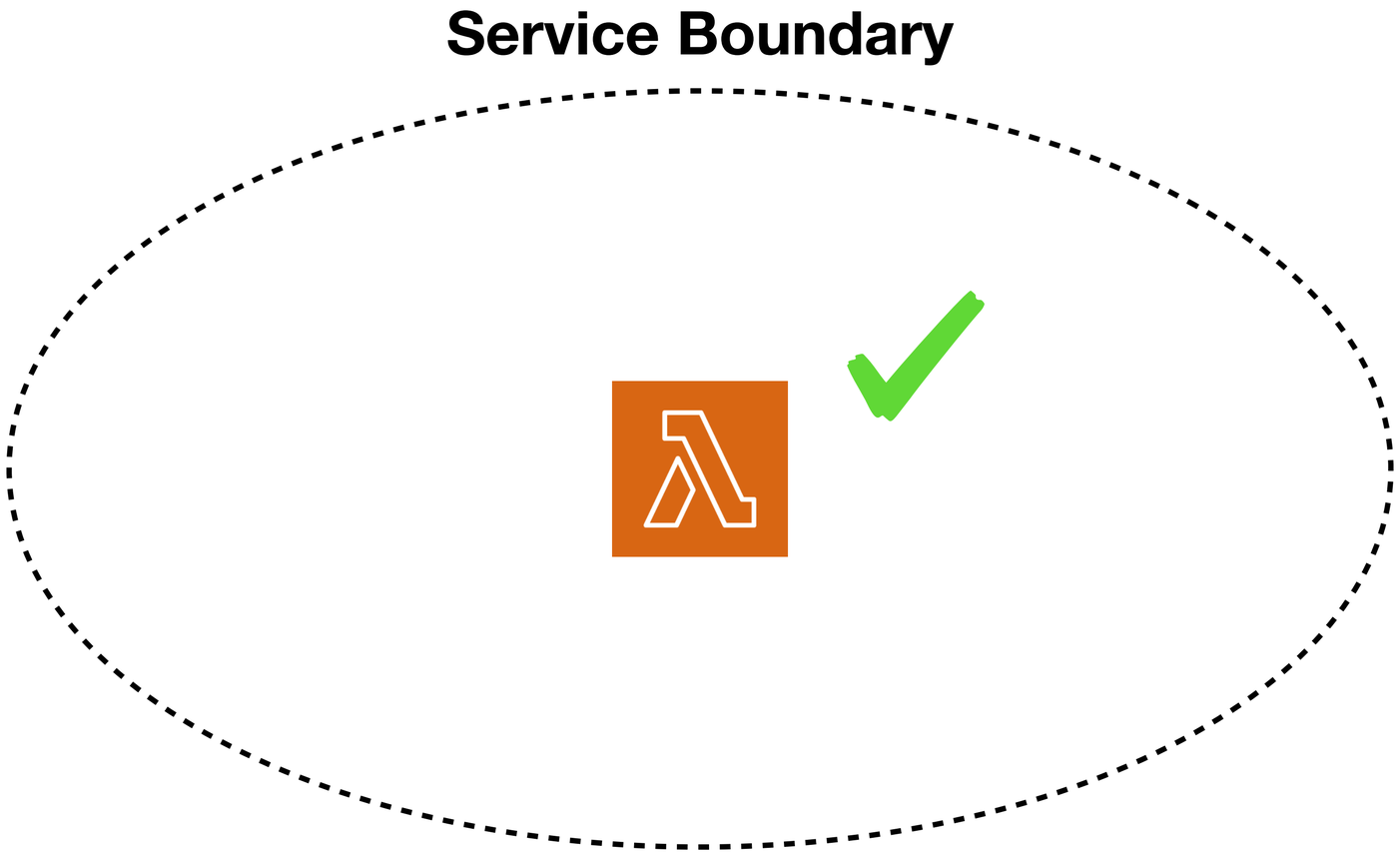
SYNCHRONOUS Lambda-to-Lambda are *almost* always a sign of bad design.


Invocation Start
Invocation End
Function A
Synchronous Invoke
Invocation Start
Invocation End
Function B

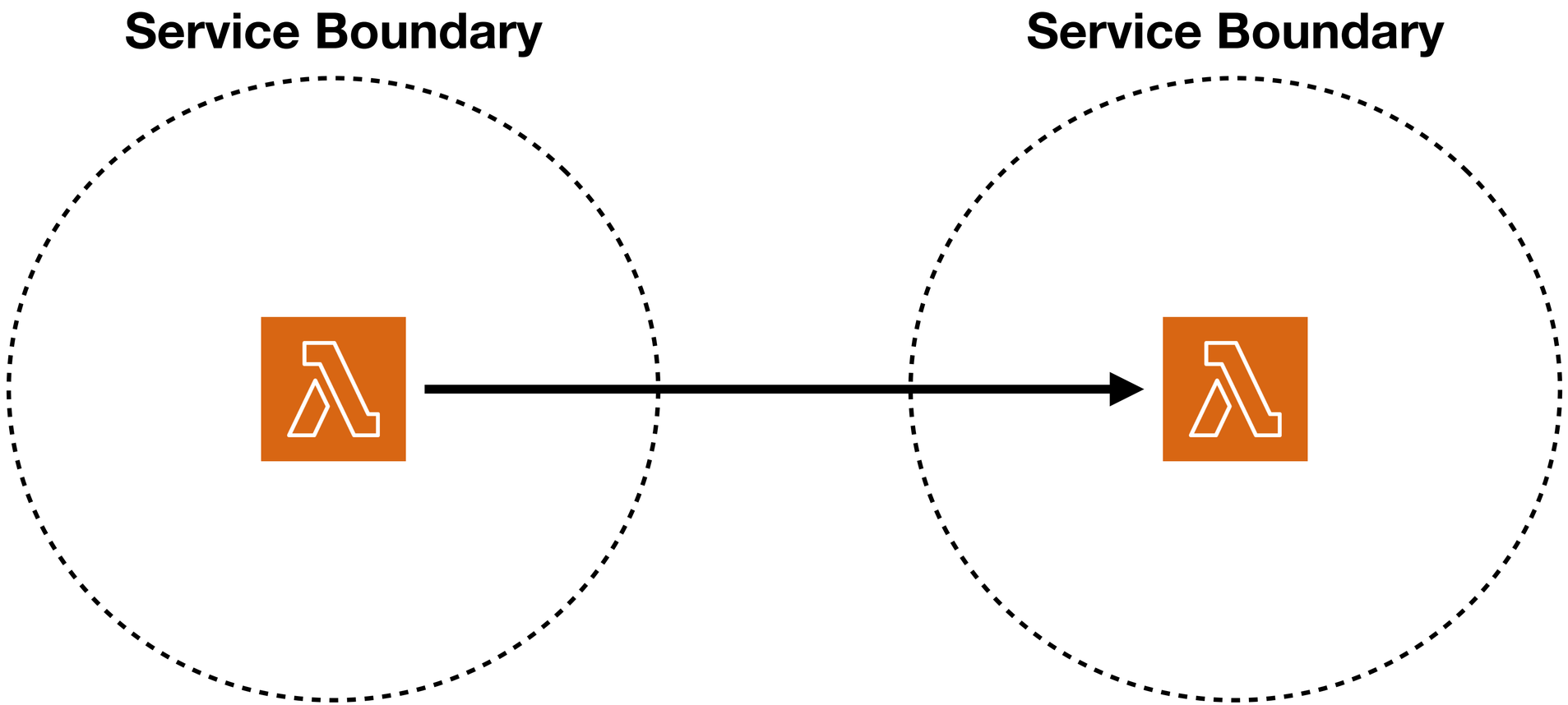

Invocation Start
Invocation End
Function A
Synchronous Invoke
Invocation Start
Invocation End
Function B
Double paying for execution time

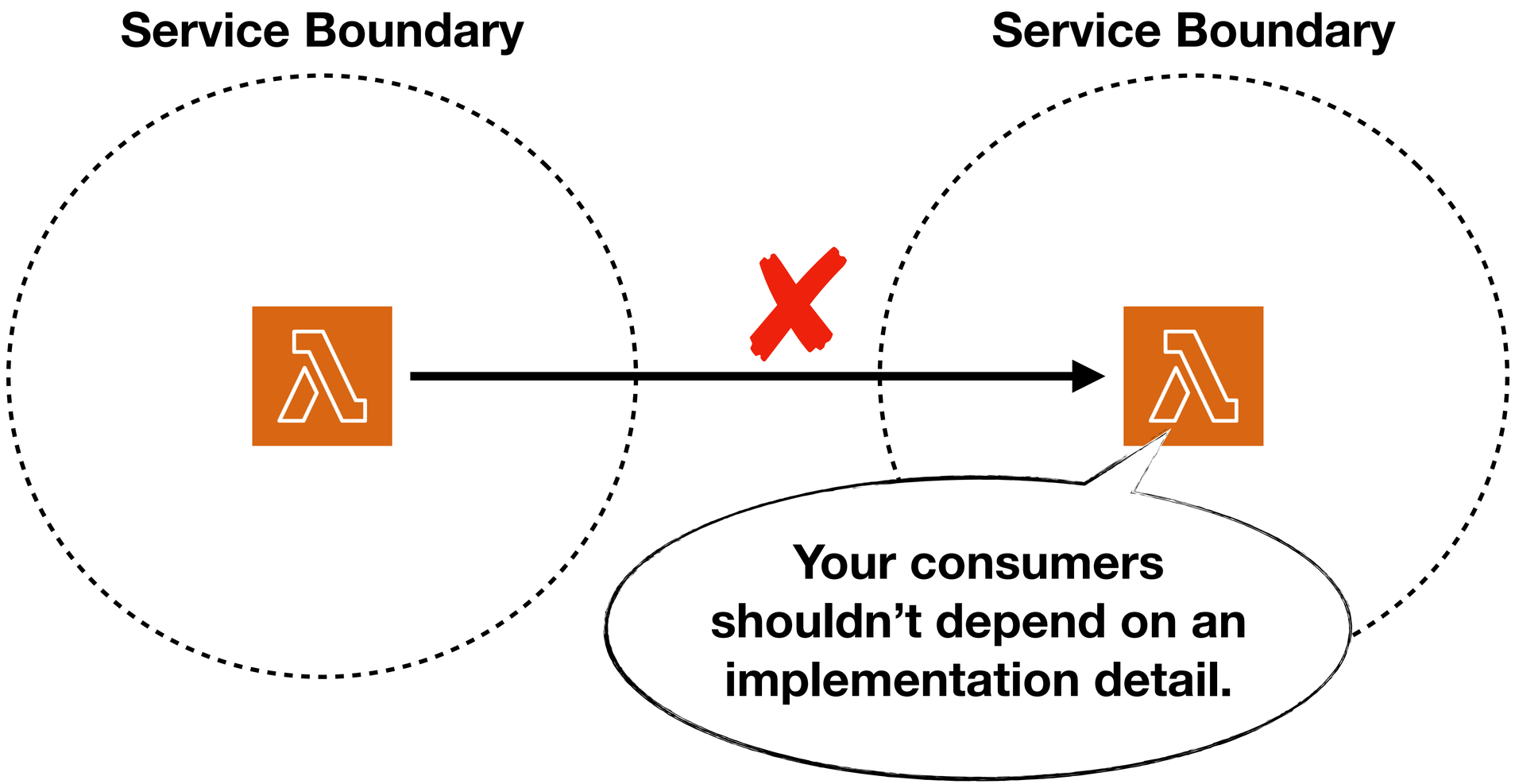

Service Boundary

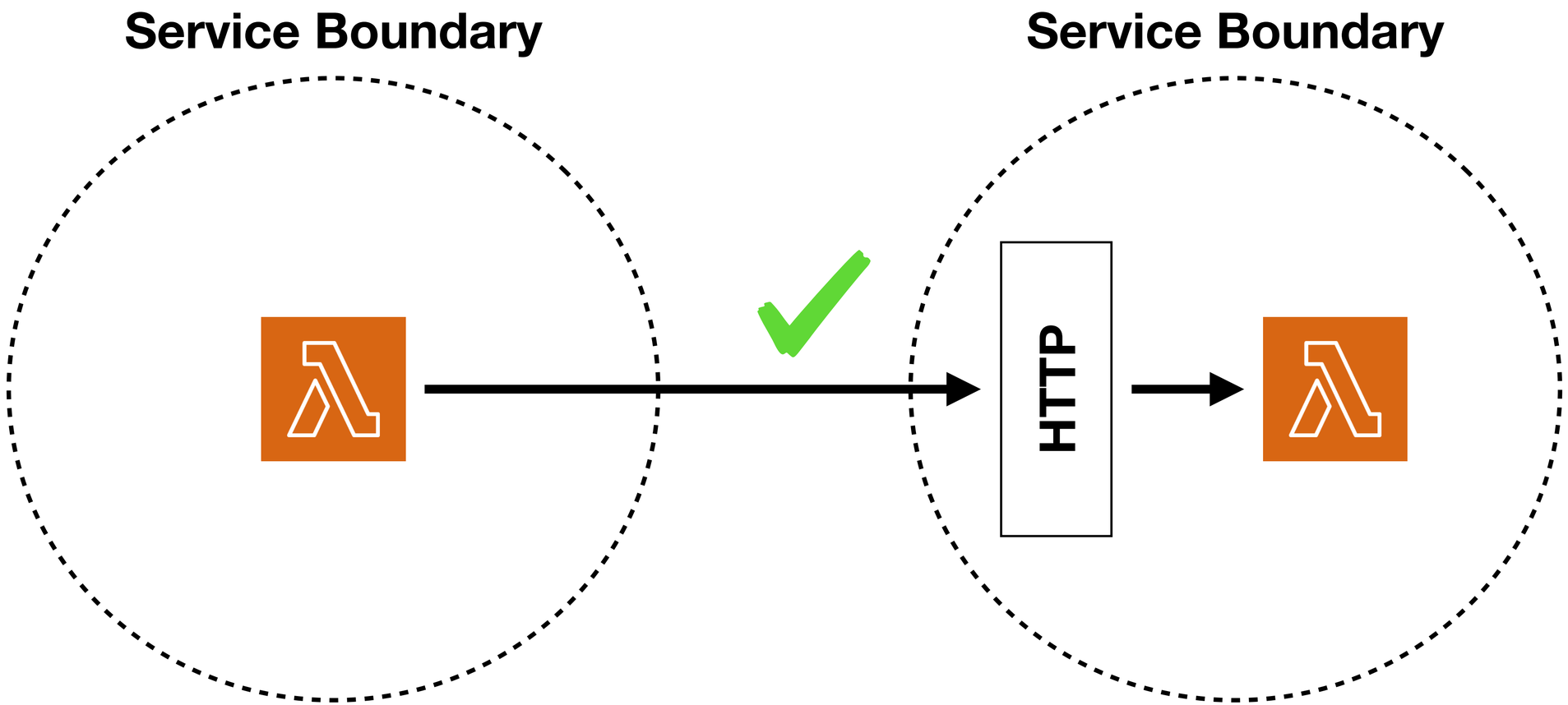

Service Boundary

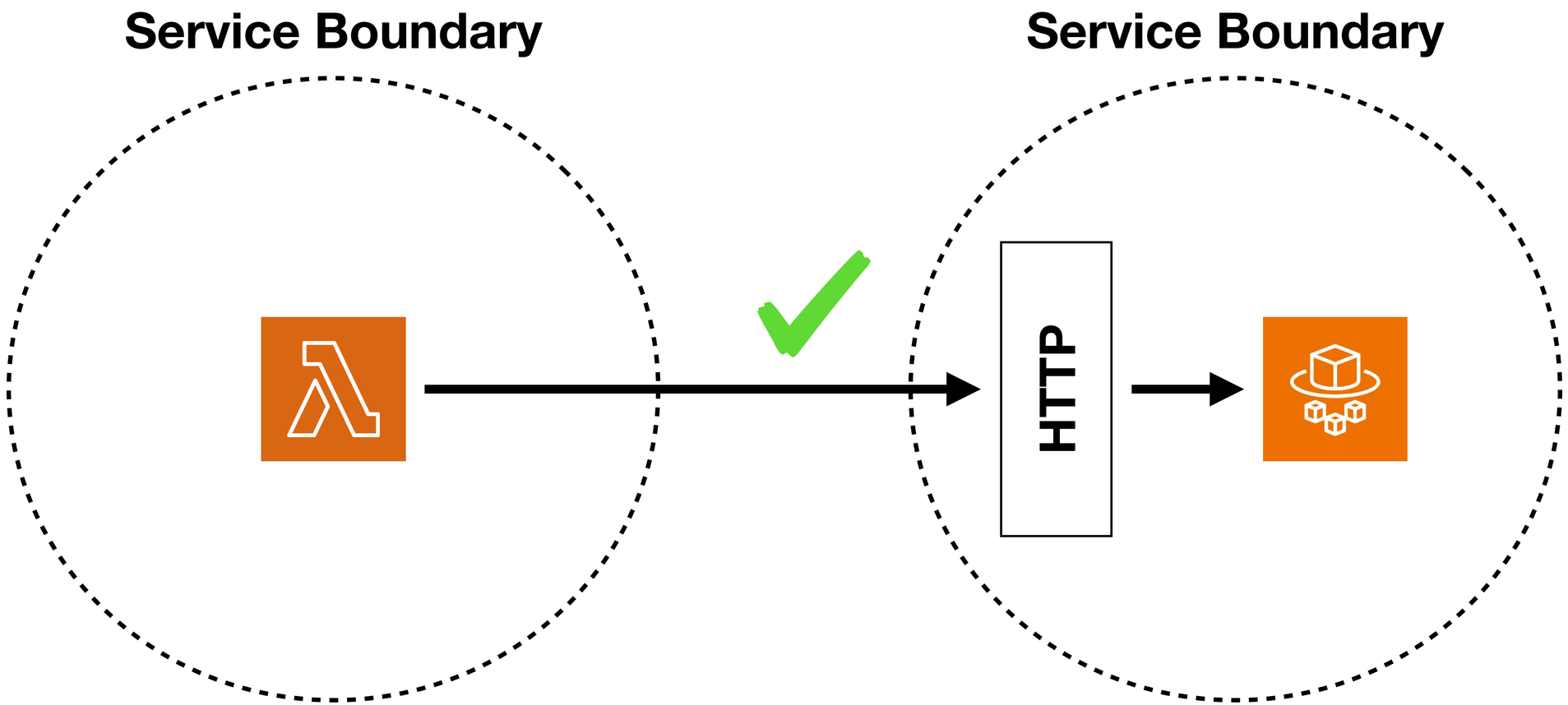

Service Boundary

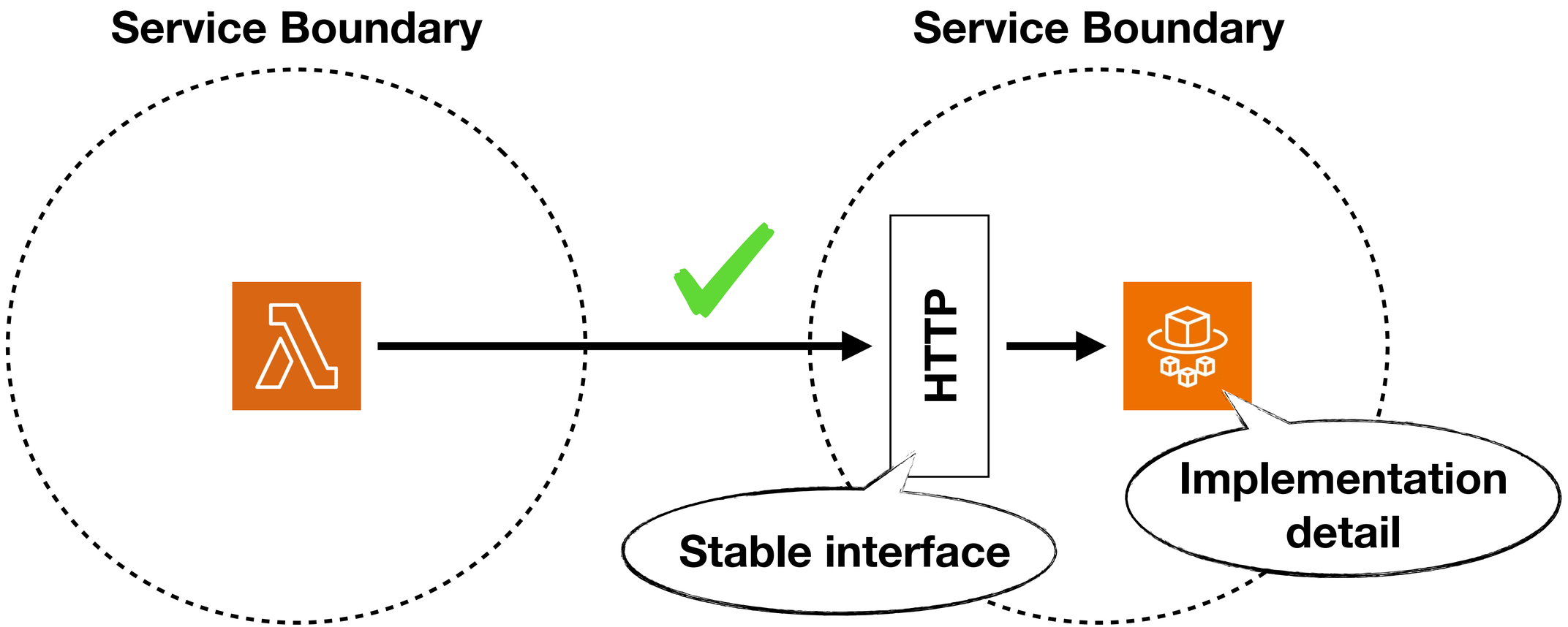
AWS Lambda function !== lambda function in programming


Service Boundary
Service Boundary

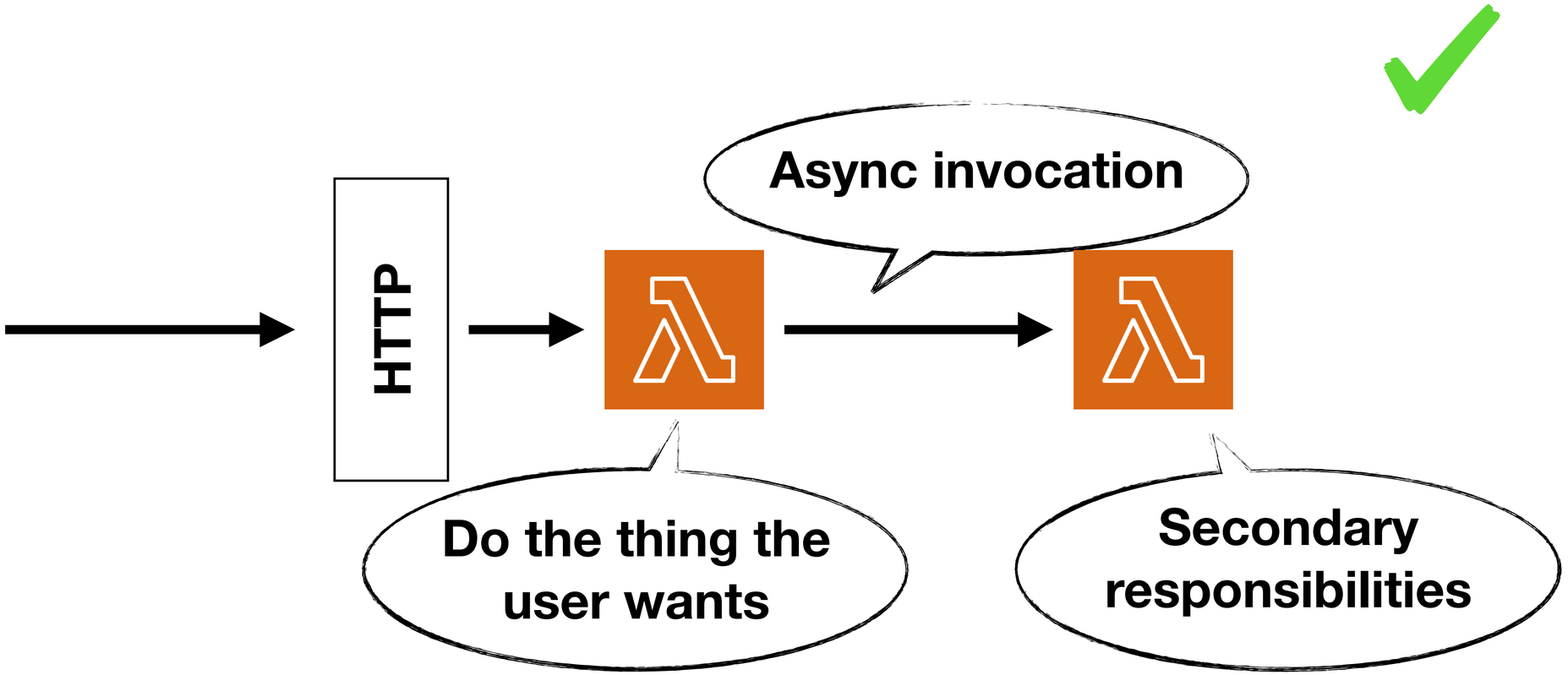

Service Boundary
Service Boundary
Your consumers shouldn’t depend on an implementation detail.

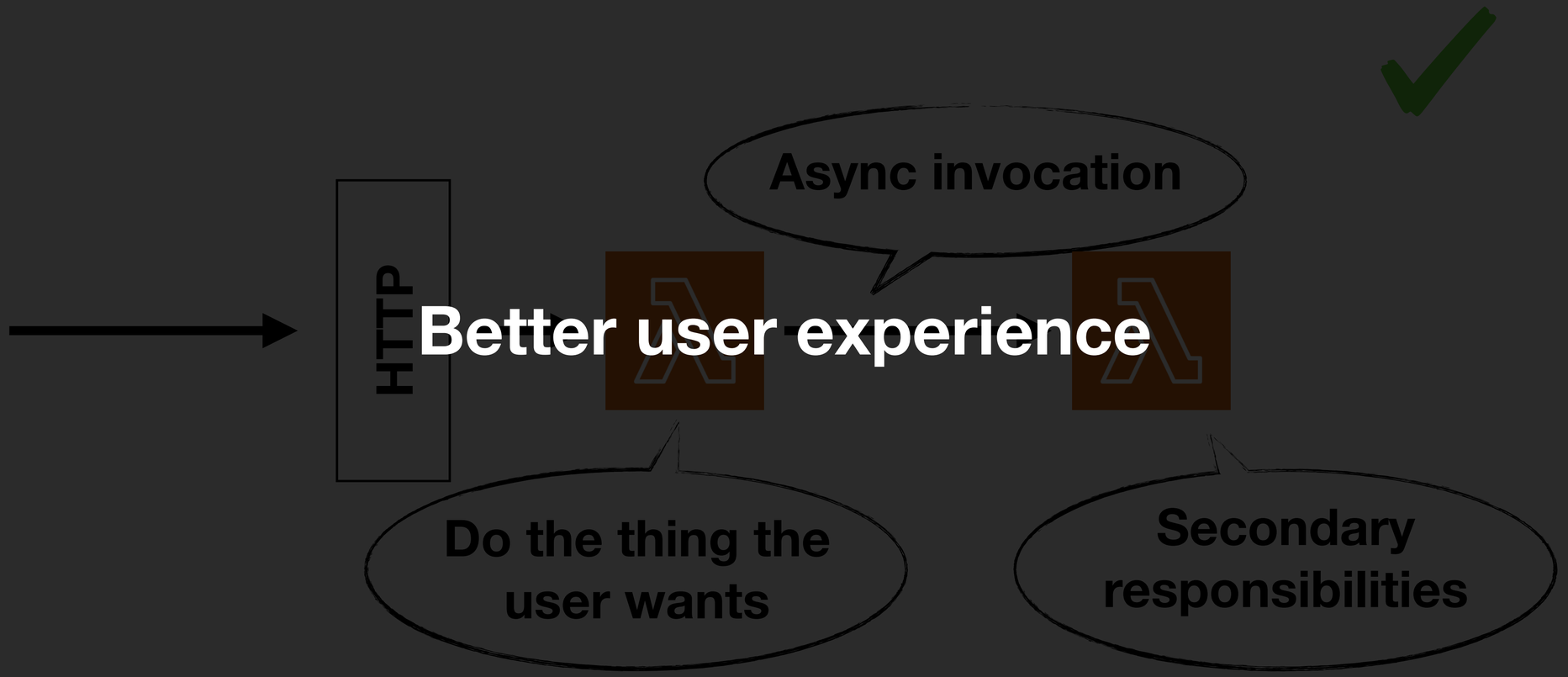

Service Boundary
Service Boundary
HTTP

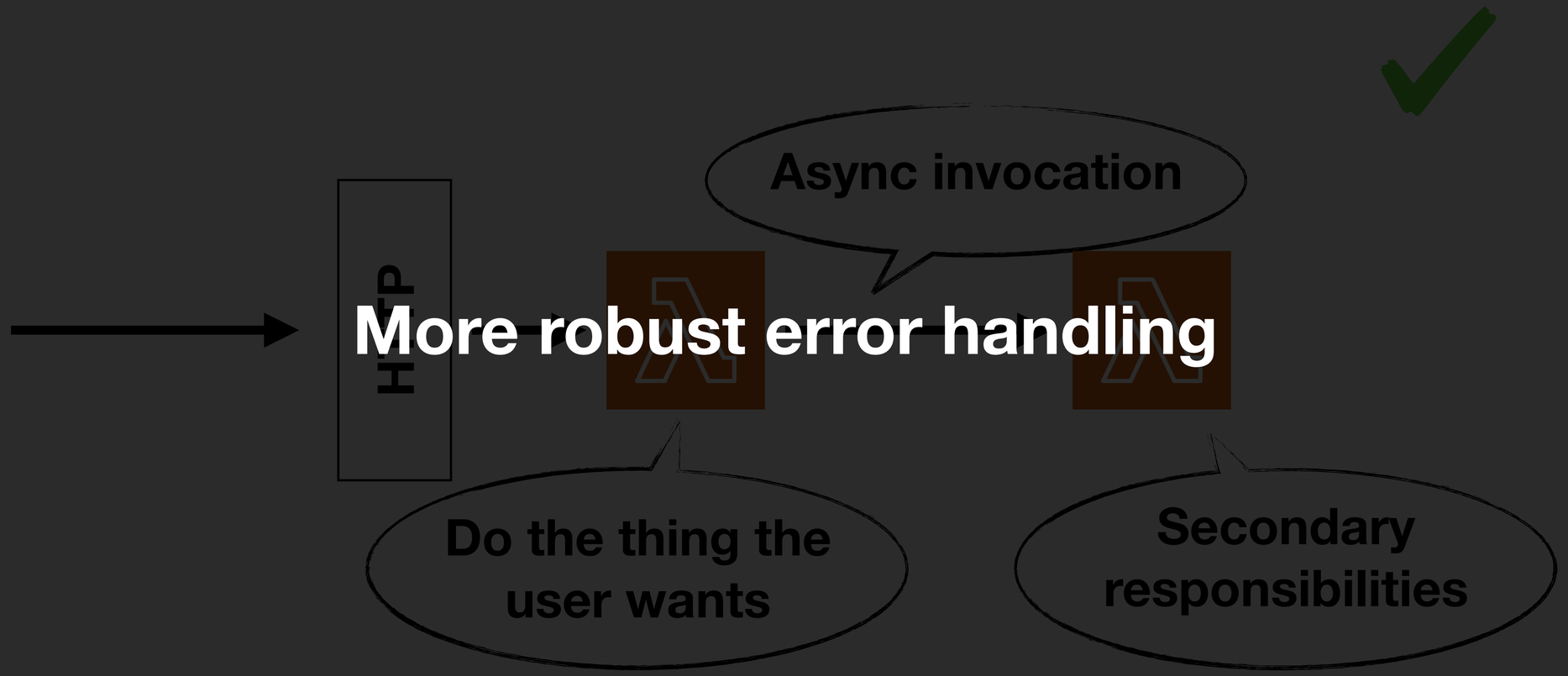

Service Boundary
Service Boundary
HTTP


Service Boundary
Service Boundary
HTTP
Stable interface
Implementation detail

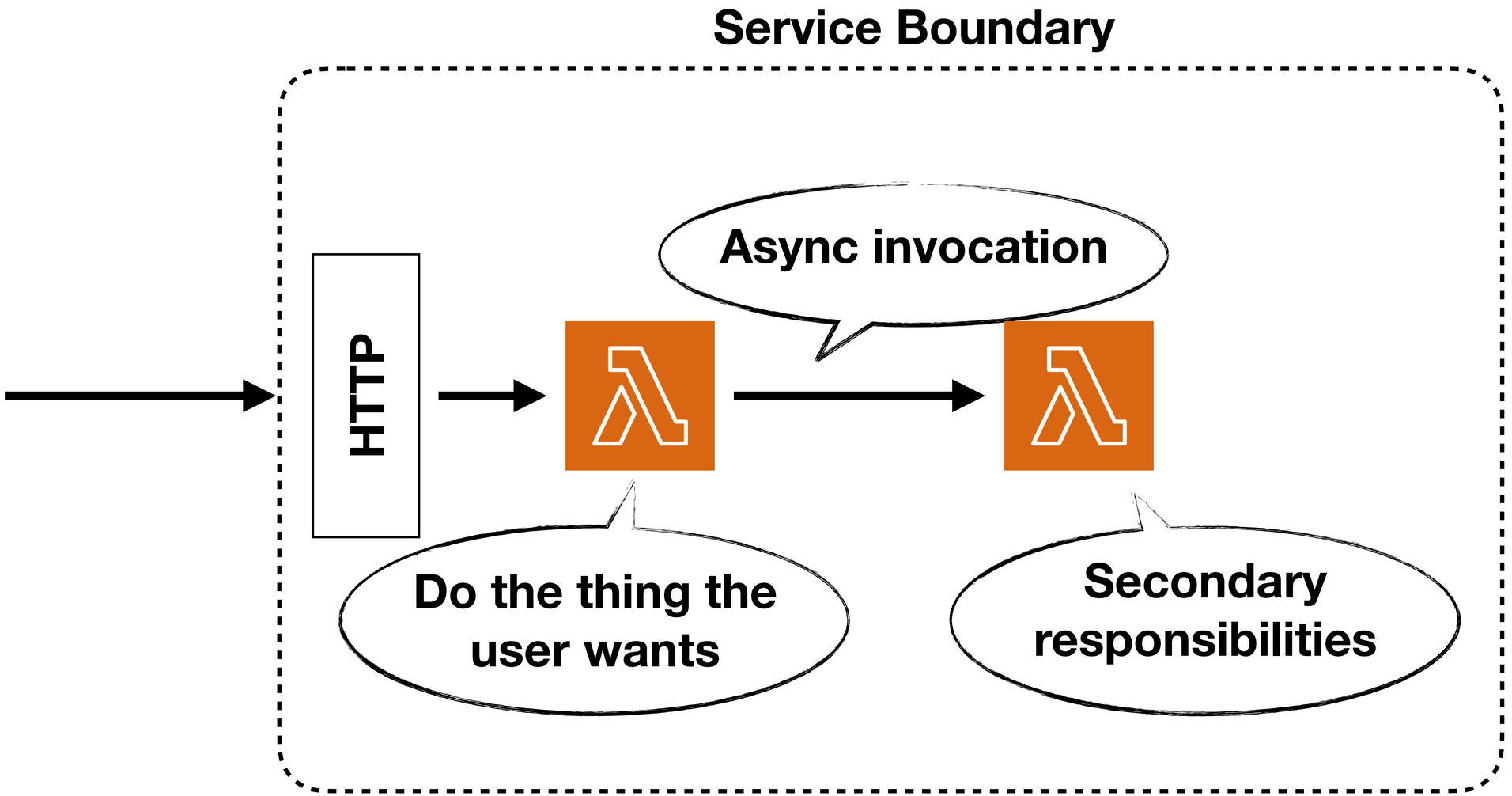
What about async invocations?

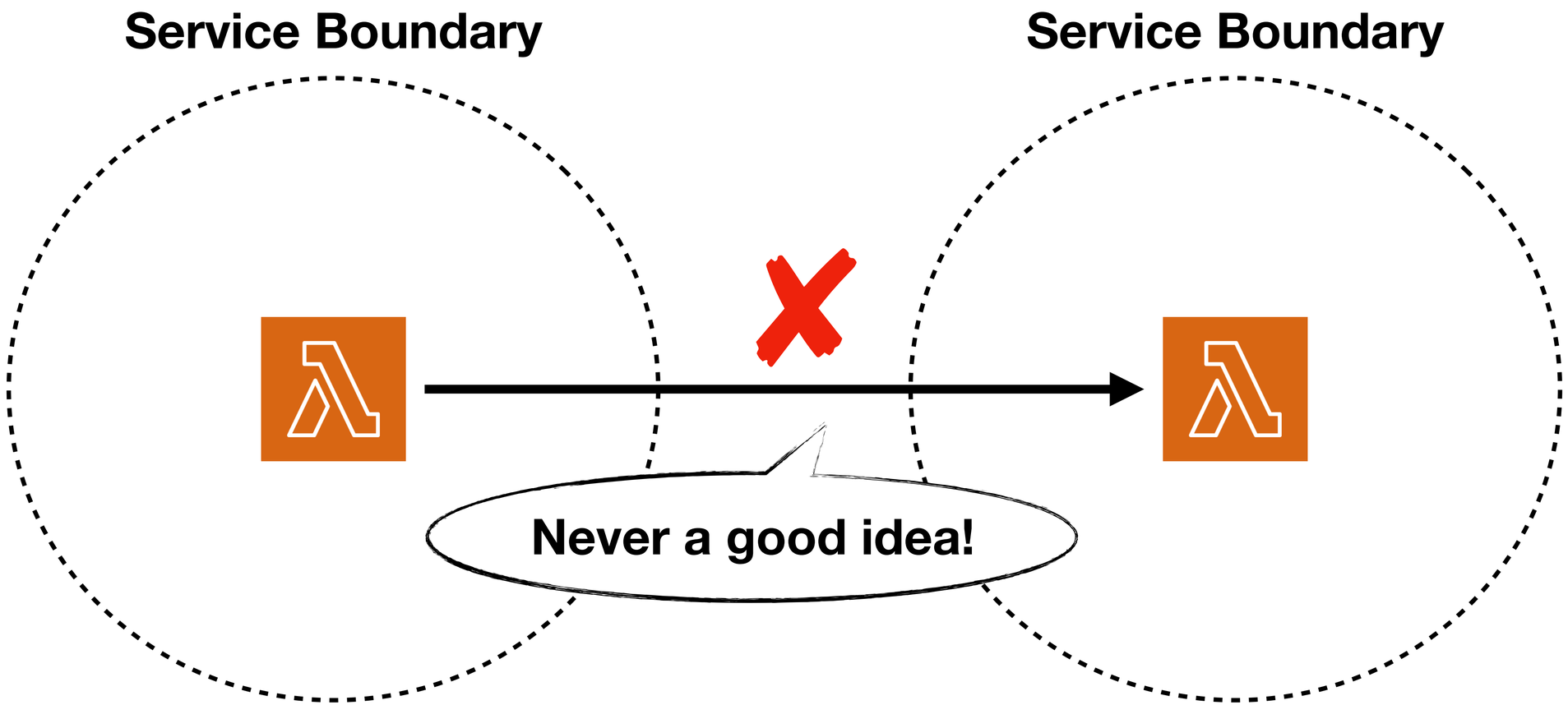

HTTP
Async invocation
Do the thing the user wants
Secondary responsibilities


HTTP
Async invocation
Better user experience
Do the thing the user wants
Secondary responsibilities



More robust error handling


Service Boundary
HTTP
Async invocation
Do the thing the user wants
Secondary responsibilities


Service Boundary
Service Boundary
Never a good idea!

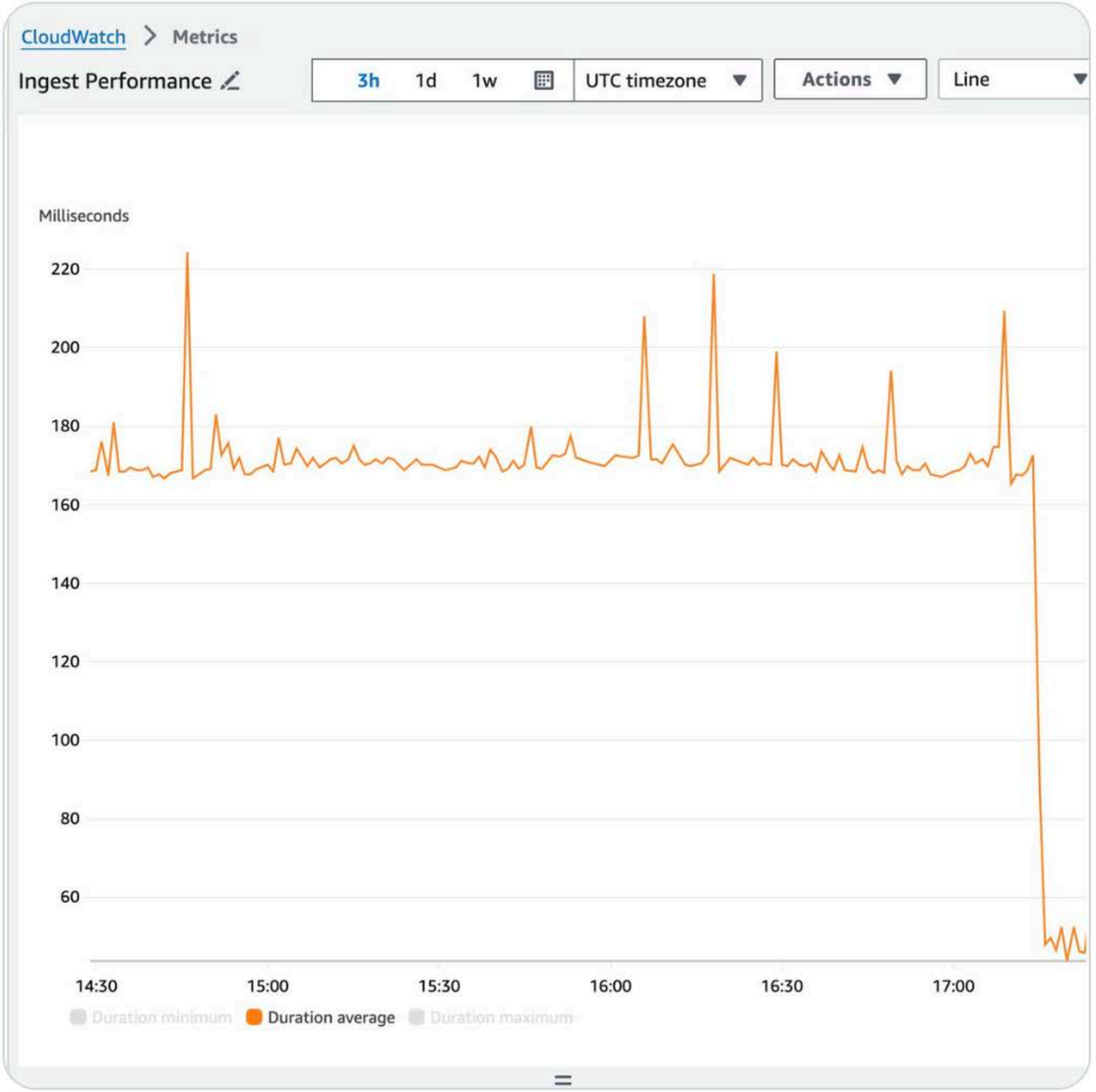
Are async Lambda-to-Lambda invocations OK?

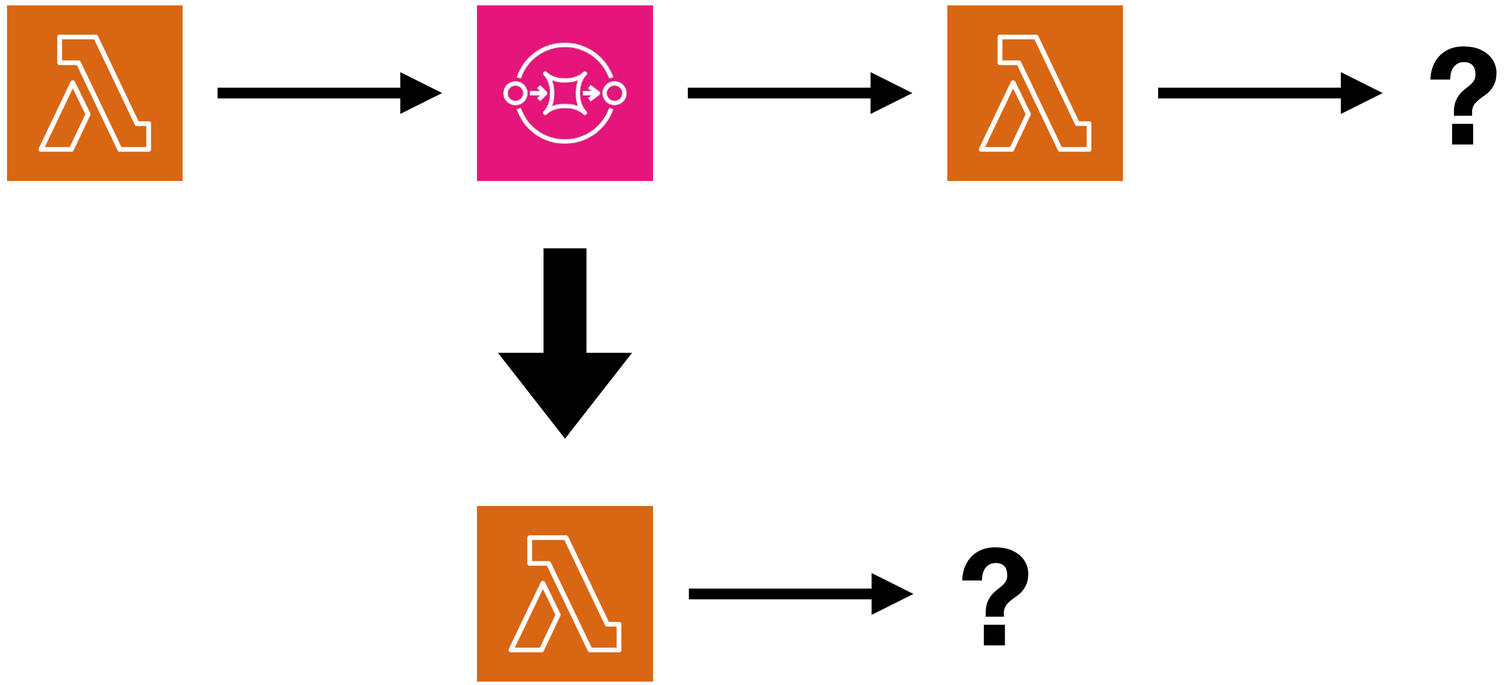
It depends…

Every component in your architecture should serve a purpose and provide a ROI.

Jack Ellis @JackEllis · Jan 17

Does the Golden Globes have a category for web performance? I'd like to nominate this chart.





Lambda & SQS

Lambda savings: $20,862 per year SQS savings: $23,989 per year

Now, let's get into where we've really cut costs. Up until recently, we were doing Lambda -> SQS -> Lambda, and this felt pretty good. After all, we wanted resilience, and, when making this decision initially, our database was in a single AZ, so we had to use SQS in-between because it was a multi-AZ, infinitely scalable service.

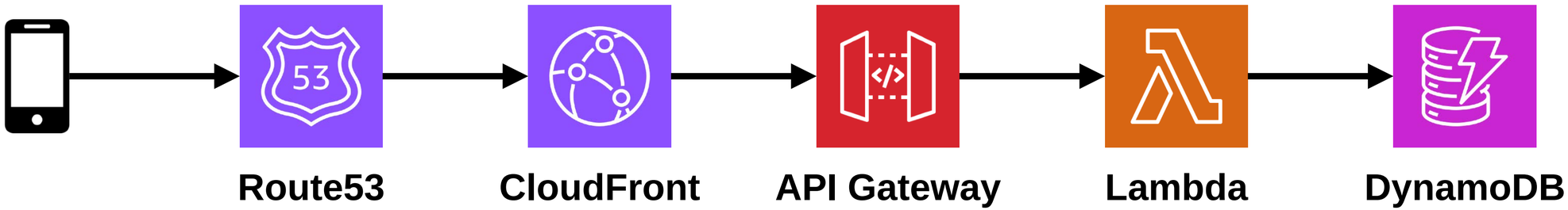
But now we've built our infrastructure where we have our databases in multiple availability zones, so we just don't need SQS, and it's instantly dropped our Lambda cost.

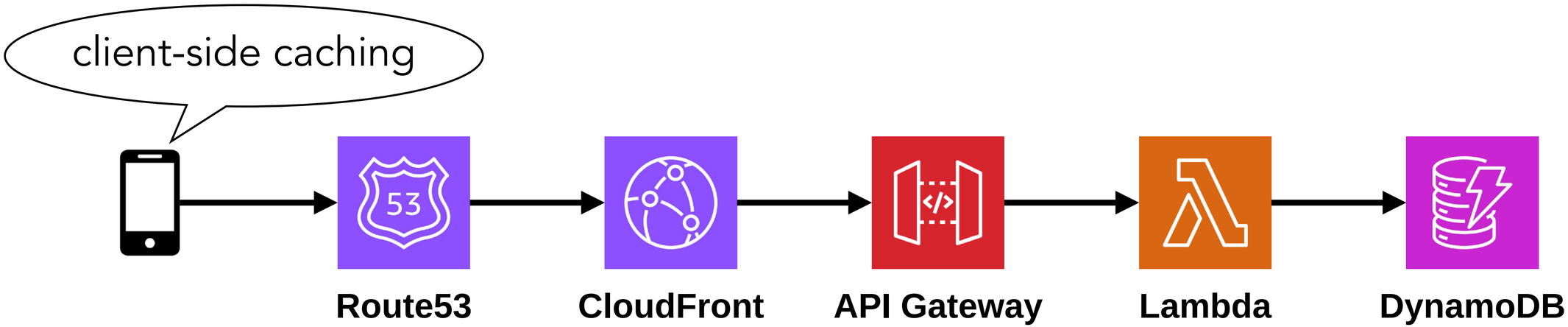
This is happening because we've cut the Lambda requests in half and introduced the following changes:

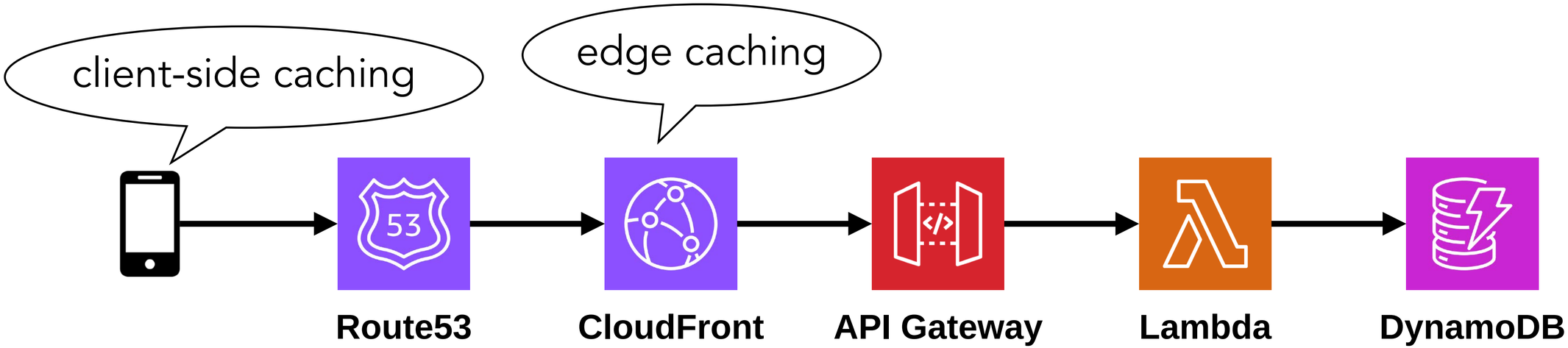
1. There is now only one Lambda request per pageview instead of two
2. The average Lambda duration on the HTTP endpoint has decreased significantly since we're no longer putting a job into SQS, we're simply hitting Redis and running a database insert (each of these operations takes 1ms or less typically)
3. We are still using SQS as a fallback (e.g. if our database is offline), but we're not using it for every request.
4. We are no longer running additional requests to SQS for each pageview/event

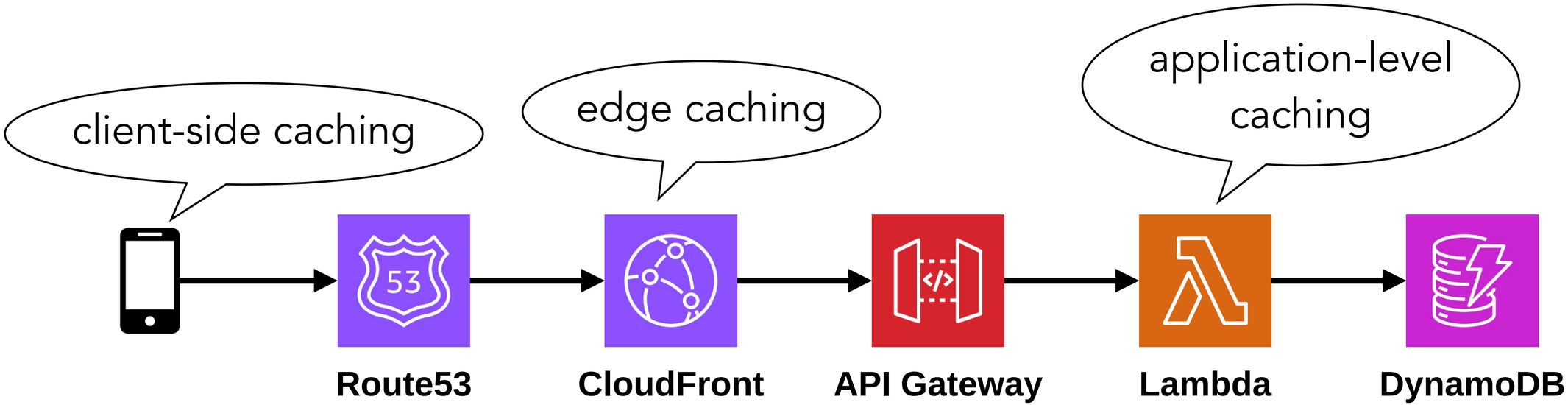
Caching

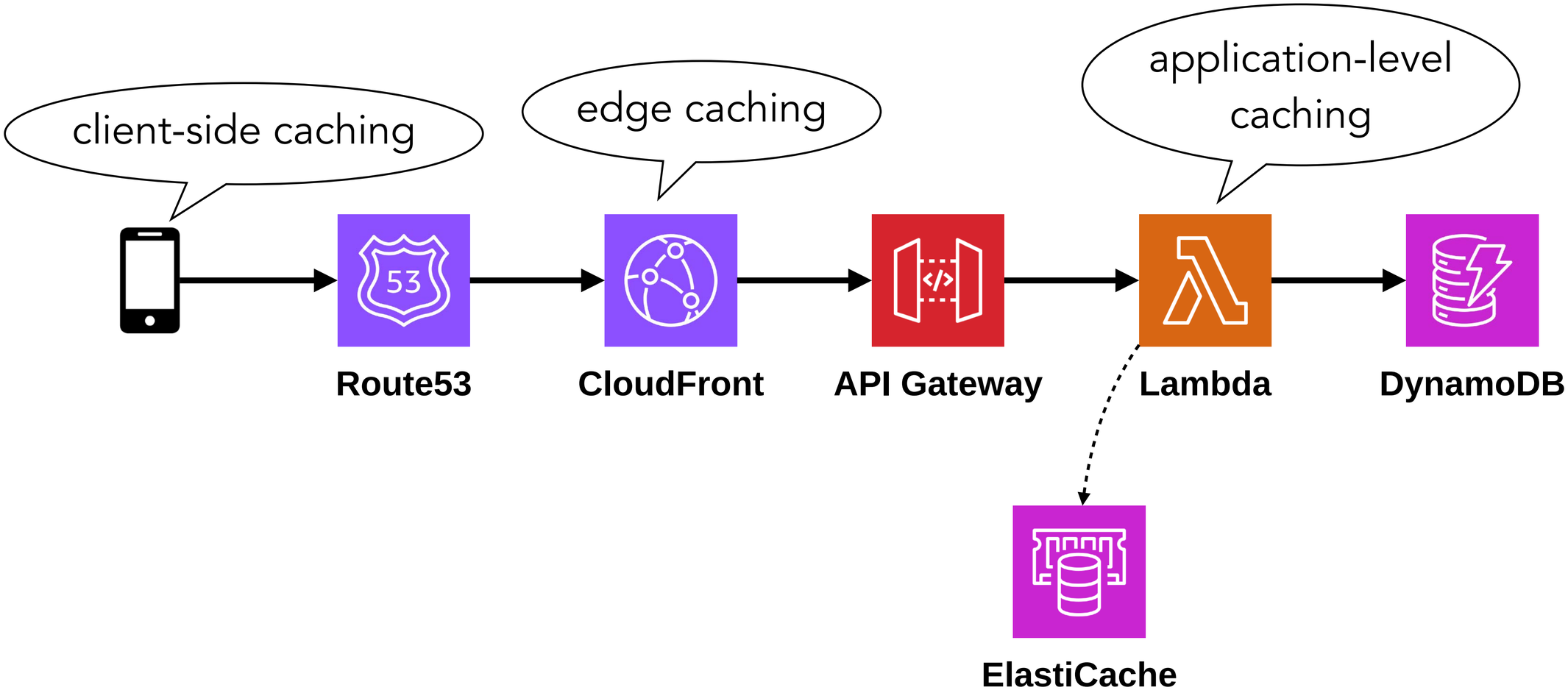
Caching is a cheat code for building performant & scalable applications.

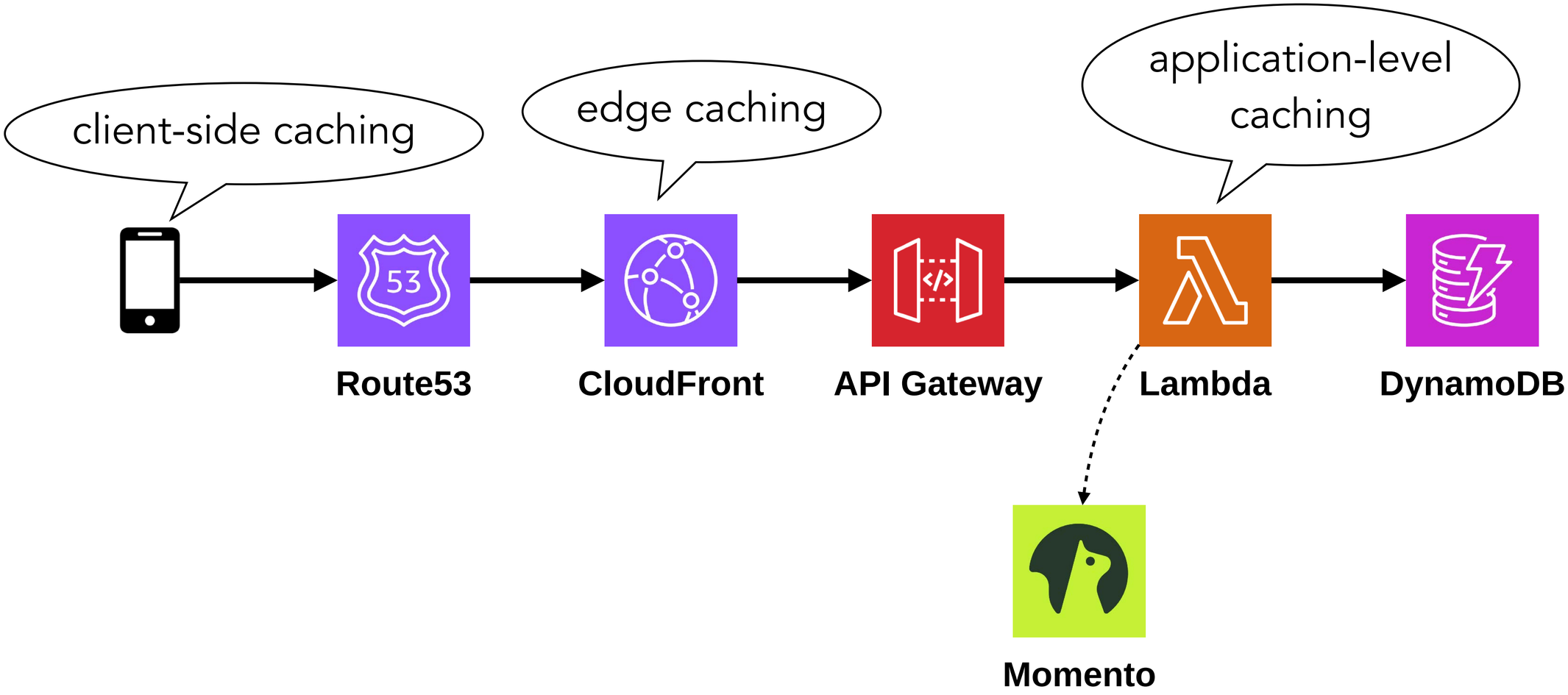

Route53
CloudFront
API Gateway
Lambda
DynamoDB


client-side caching
53
Route53
CloudFront
API Gateway
Lambda
DynamoDB


client-side caching
edge caching
Route53
CloudFront
API Gateway
Lambda
DynamoDB


client-side caching
edge caching
application-level caching
53
Route53
CloudFront
API Gateway
Lambda
DynamoDB


client-side caching
edge caching
application-level caching
53
Route53
CloudFront
API Gateway
Lambda
DynamoDB
ElastiCache


client-side caching
edge caching
application-level caching
Route53
CloudFront
API Gateway
Lambda
DynamoDB
Momento

Route53 TTL

DNS Queries

The following query prices are prorated; for example, a public hosted zone with 100,000 standard queries per month would be charged $0.04, and a public hosted zone with 100,000 latency-based routing queries per month would be charged $0.06. Route 53 does not charge for queries on private hosted zones.

Standard Queries	◦ $0.40 per million queries (first 1 billion queries per month) ◦ $0.20 per million queries (over 1 billion queries per month)
Latency-Based Routing Queries	◦ $0.60 per million queries (first 1 billion queries per month) ◦ $0.30 per million queries (over 1 billion queries per month)
Geolocation and Geoproximity Queries	◦ $0.70 per million queries (first 1 billion queries per month) ◦ $0.35 per million queries (over 1 billion queries per month)
IP-Based Routing Queries*	◦ $0.80 per million queries (first 1 billion queries per month) ◦ $0.40 per million queries (over 1 billion queries per month)

Use longer TTL for stable domains

Avoid CORS

Enabling CORS for API Gateway is easy

Enabling CORS for API Gateway is easy

But you still pay for those CORS requests!

Brett Andrews
@AWSbrett

If you're using the authorization header (which you likely are), you MUST specify that header in `Access-Control-Allow-Headers` if you want `Access-Control-Max-Age` to work (see twitter.com/annevk/status/... for more details)

Anne van Kesteren @annevk · Aug 4, 2021

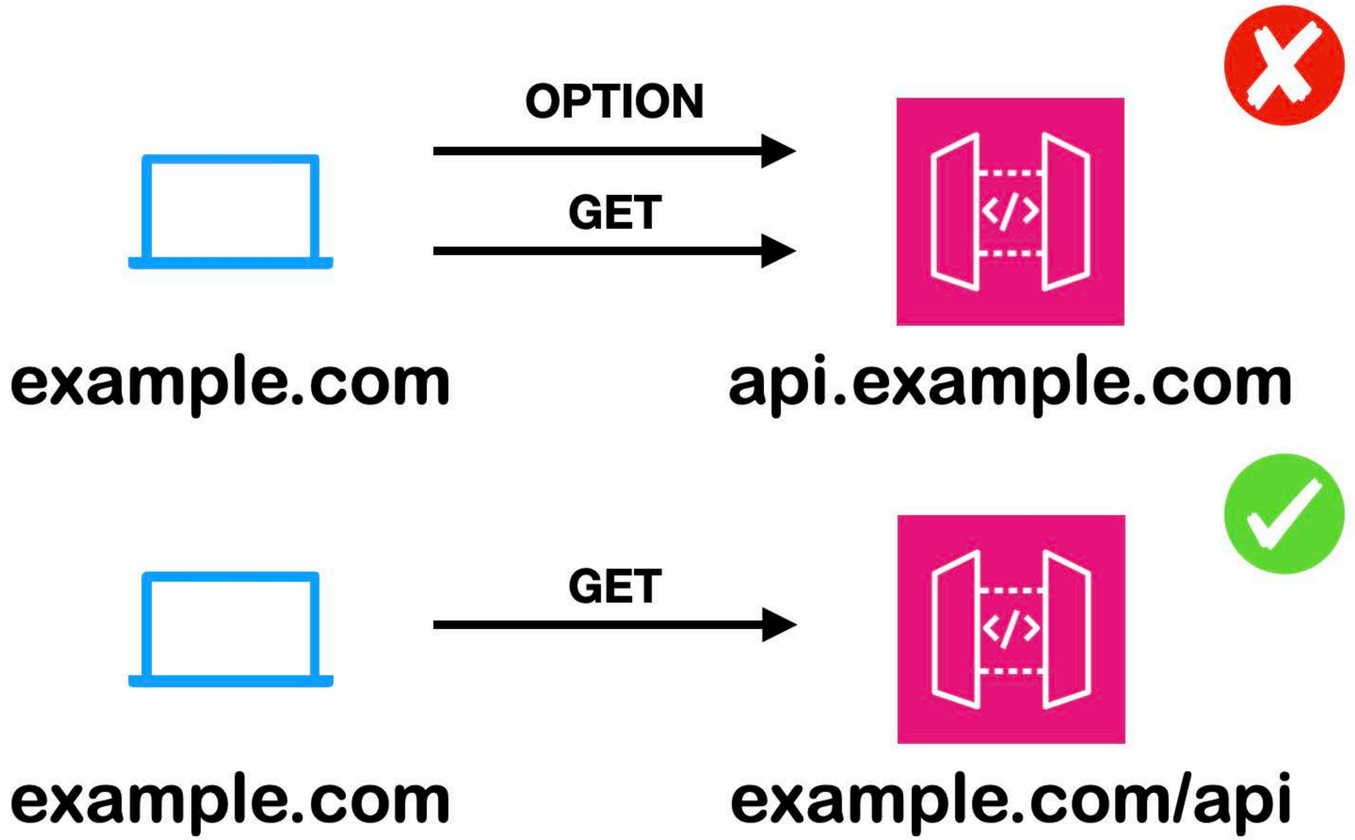
Replying to @da_adler @jaffathecake and @hirano_y_aa

No, the preflight response needs Access-Control-Allow-Headers: Authorization, *. Assuming the request is with credentials set to "same-origin" or "omit". (When it is "include" a wildcard does not work, but either way you need to list Authorization as currently defined.)

You might be double paying for every user request to your API…

Solution: roll your own OPTIONS methods

or…


OPTION
GET
example.com
api.example.com
GET
example.com
example.com/api

Choose the right service

Every architectural decision is a buying decision.

Using the wrong service can be very costly.

	1 msg/s
 **SNS**	$1.296
 **SQS**	$1.037
 **EventBridge**	$2.592
 **Kinesis Provisioned**	$10.836
 **Kinesis On-Demand**	$28.998

Estimated cost per month, assuming each message is 1KB in size.

	1 msg/s	1,000 msg/s
 **SNS**	$1.296	$1296.00
 **SQS**	$1.037	$1036.80
 **EventBridge**	$2.592	$2592.00
 **Kinesis Provisioned**	$10.836	$47.088
 **Kinesis On-Demand**	$28.998	$226.55

Estimated cost per month, assuming each message is 1KB in size.

		1 TPS
 API Gateway	REST	$9.072
	HTTP	$2.592
 ALB		$21.96

Assuming 1KB per request

		1 TPS	1,000 TPS
 API Gateway	REST	$9.072	$9072
	HTTP	$2.592	$2592
 ALB		$21.96	$246.6

Assuming 1KB per request

Services that charge by uptime are order(s) of magnitude cheaper at scale.

Services that charge by uptime are order(s) of magnitude cheaper at scale.

But, you must understand the cost **dimensions** of individual services.

		1 TPS	1,000 TPS
 API Gateway	REST	$9.072	$9072
	HTTP	$2.592	$2592
 ALB		?	?

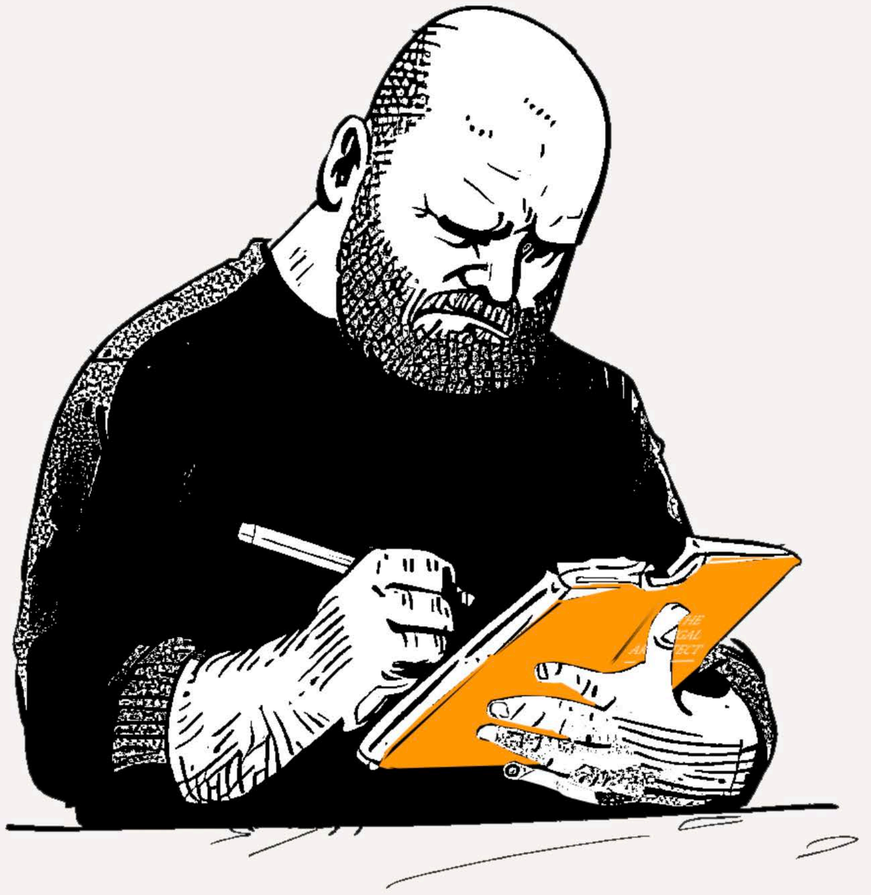
Assuming **1MB** per request

		1 TPS	1,000 TPS
API Gateway	REST	$9.072	$9072
	HTTP	$2.592	$2592
ALB		**$68.99**	**$53,837.87**

Assuming **1MB** per request

THE FRUGAL ARCHITECT

Simple laws for building cost-aware, sustainable, and modern architectures.



www.thefrugalarchitect.com

Law I.

Make Cost a Non-functional Requirement.

Law II.

Systems that Last Align Cost to Business.

Law III.

Architecting is a Series of Trade-offs.

Law IV.

Unobserved Systems Lead to Unknown Costs.

Law V.

Cost Aware Architectures Implement Cost Controls.

Law VI.

Cost Optimization is Incremental.

Law VII.

Unchallenged Success Leads to Assumptions.

Law I.

Make Cost a Non-functional Requirement.

Law II.

Systems that Last Align Cost to Business.

Law III.

Architecting is a Series of Trade-offs.

Law IV.

Unobserved Systems Lead to Unknown Costs.

Law V.

Cost Aware Architectures Implement Cost Controls.

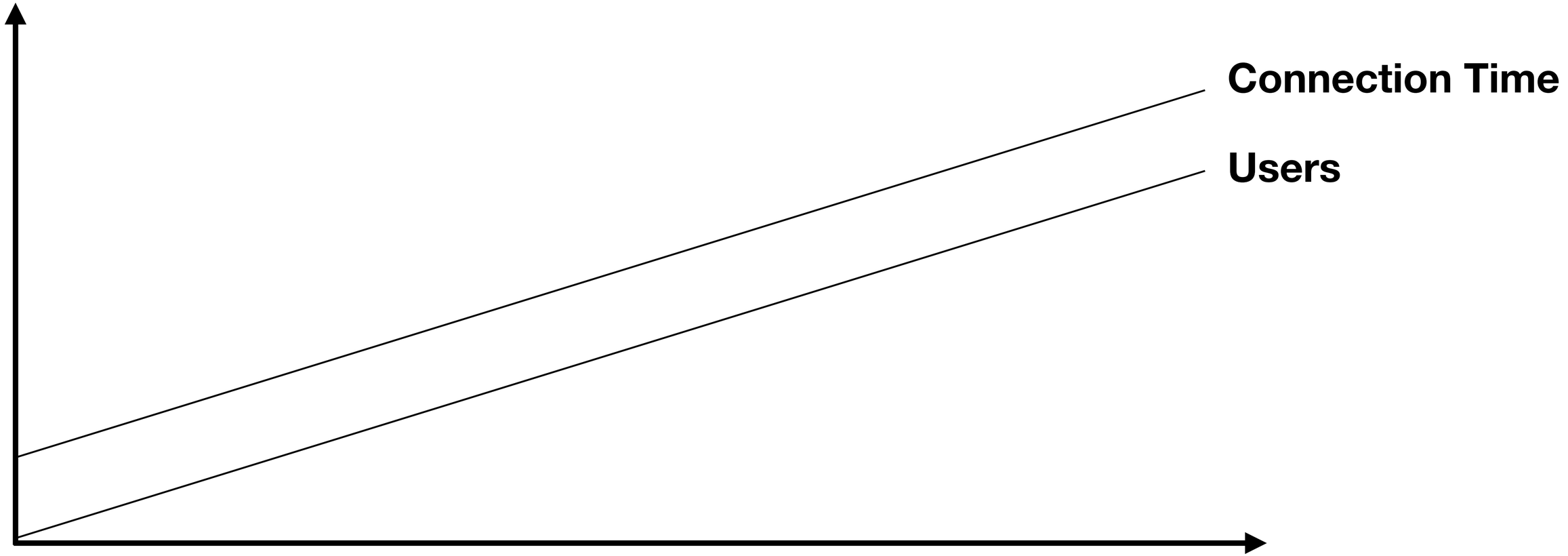
Law VI.

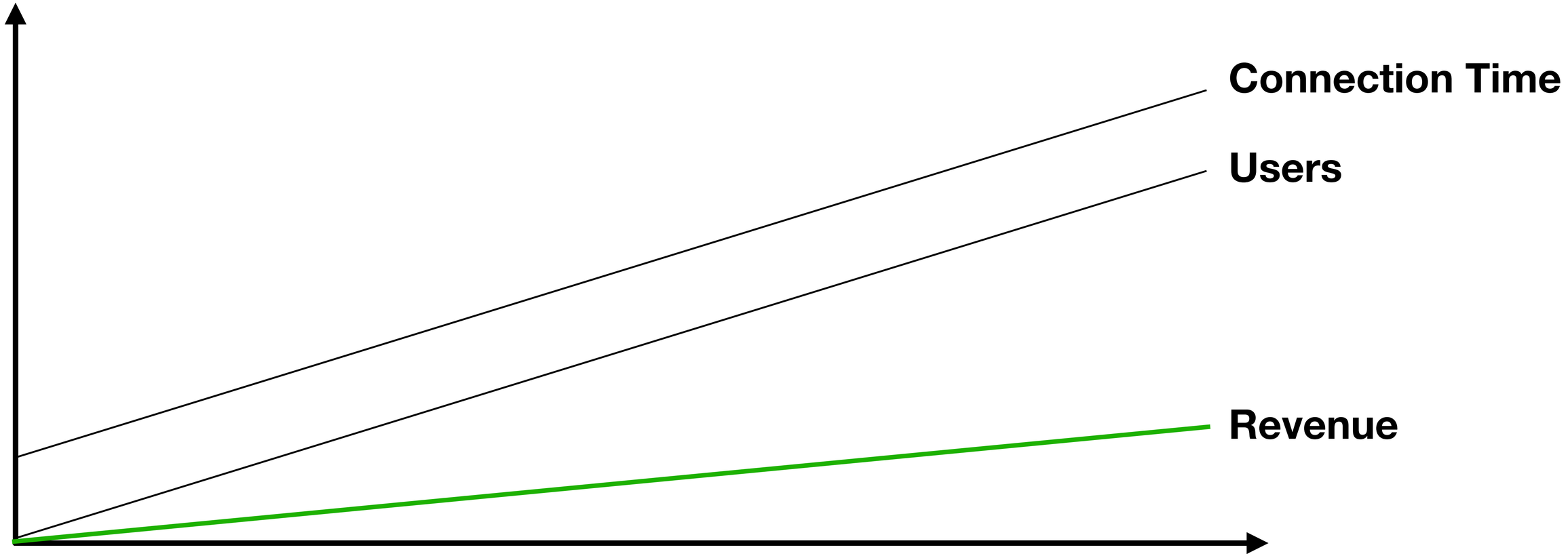
Cost Optimization is Incremental.

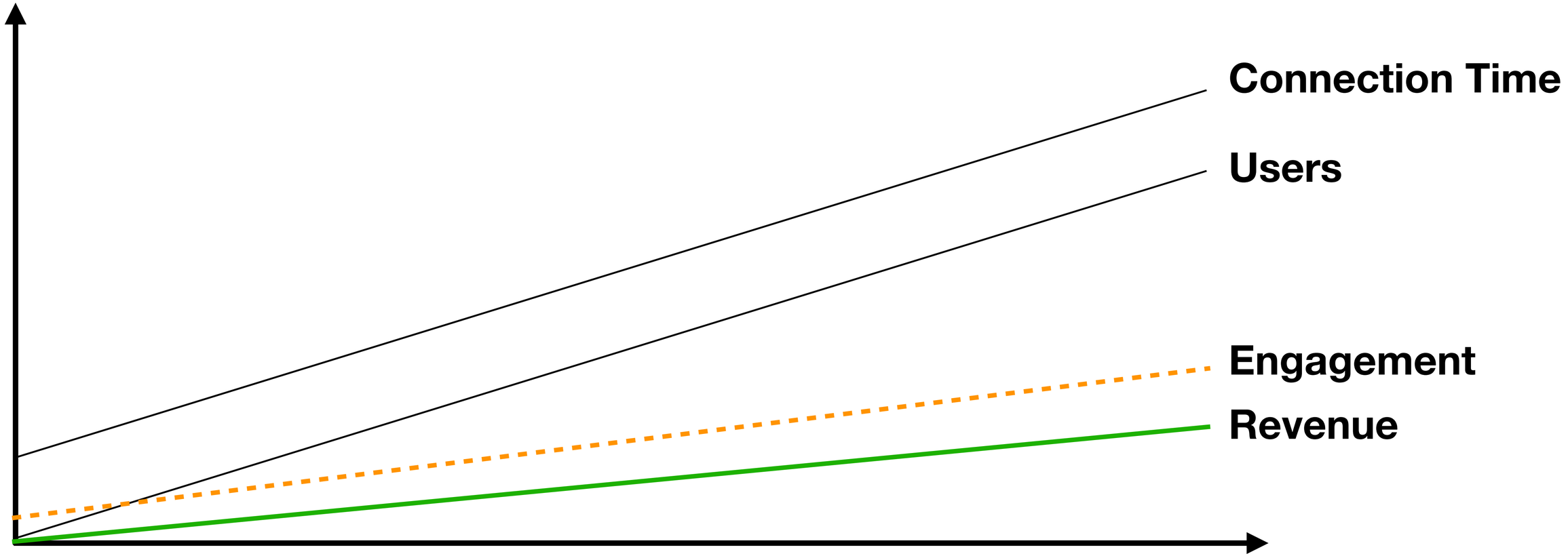
Law VII.

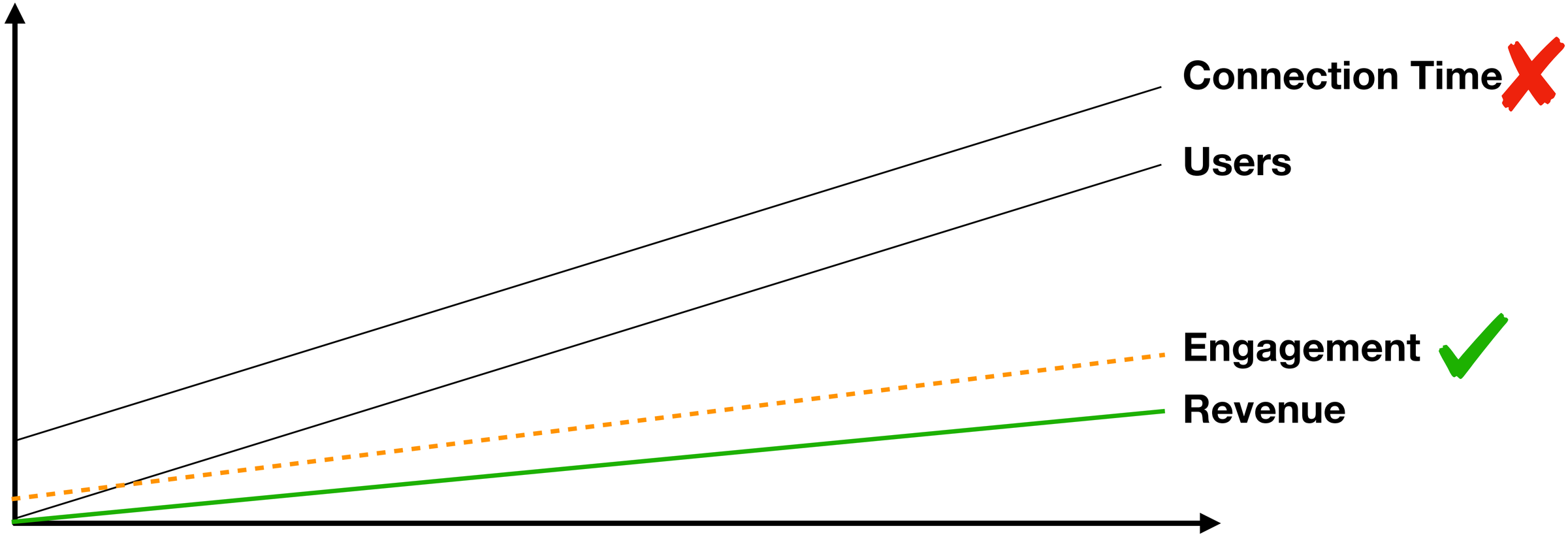
Unchallenged Success Leads to Assumptions.

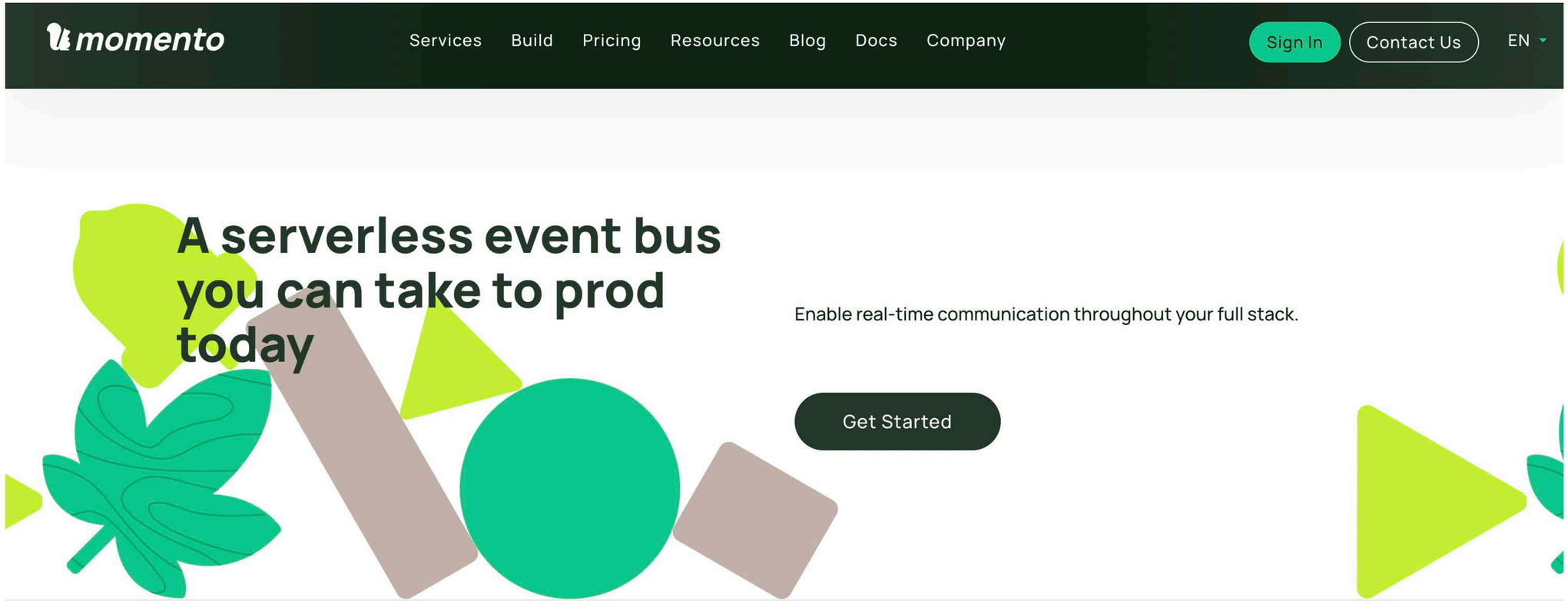
	API Gateway	AppSync	IOT core
Messages	$1 per million	$2 per million	$1 per million
Connection Time	$0.25 per million mins	$0.08 per million mins	$0.08 per million mins


Connection Time
Users


Connection Time
Users
Revenue


Connection Time
Users
Engagement
Revenue


Connection Time
Users
Engagement
Revenue



www.gomomento.com/services/topics

BEST VALUE

On-Demand

Simple pay-as-you-go pricing! Best for moving fast with smaller workloads.

✓ Fully-managed platform
✓ Built-in auth, security, and more!

Plus:
✓ Pay only for what you use
✓ Generous free tier

No Connection Time cost!

$1.00

per million operations

Get Started

Provisioned

Procure a cost-effective pool of capacity for workloads at any scale.

✓ Fully-managed platform
✓ Built-in auth, security, and more!

Plus:
✓ Flexibly reallocate capacity
✓ Build out to unlimited scale
✓ 24/7 On-call support

starting at

$5,000

100,000 ops/sec

Create Your Plan

Enterprise Edition

Build your own plan

Momento offers volume and reservation discounts.

✓ Volume discounts
✓ Custom service limits
✓ Private connectivity
✓ SOC 2 Type II + HIPAA
✓ 24/7 On-call support with higher SLA standards

Contact Us

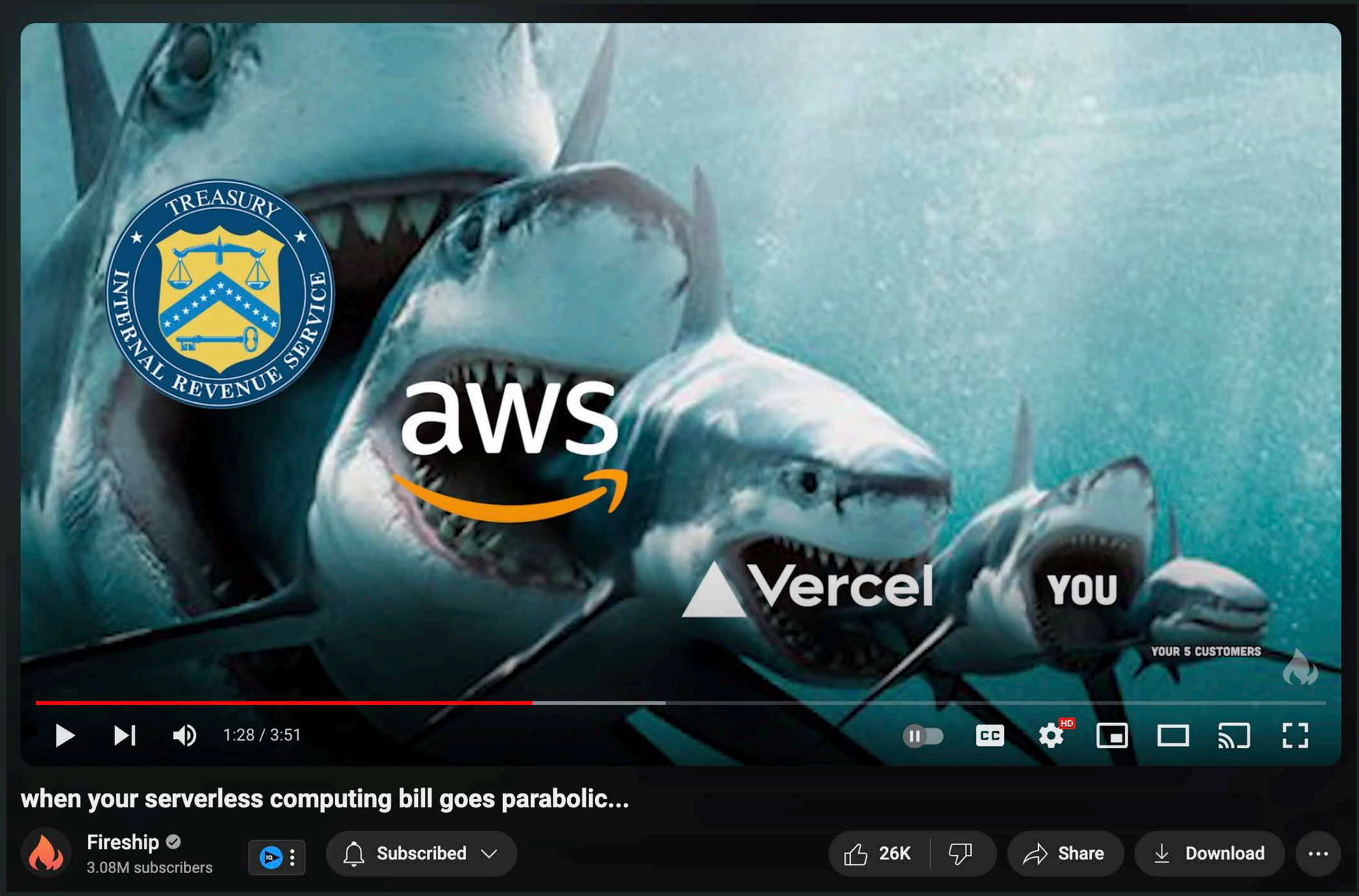
	API Gateway	AppSync	IOT core	Momento
Messages	$1 per million	$2 per million	$1 per million	$1 per million
Connection Time	$0.25 per million mins	$0.08 per million mins	$0.08 per million mins	-
Data Transfer	EC2 rates	EC2 rates	EC2 rates	-

Speaking of picking cost-efficient services…

Jingna Zhang @ cara.app/zemotion
@zemotion

So freaking speechless right now. Seen many @vercel functions stories but first time experiencing such discrepancy vs request logs like, this is cannot be real??

> This is your daily notification that your team [redacted] has used **24166% of your monthly included Serverless Function Execution amount** which has added **$96,280** to your bill thus far. You'll continue to be charged **$40 per 100 GB Hrs**.



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

Jingna Zhang @ cara.app/zemotion
@zemotion

So freaking speechless right now. Seen many @vercel functions stories but first time experiencing such discrepancy vs request logs like, this is cannot be real??

> This is your daily notification that your team [redacted] has used **24166% of your monthly included Serverless Function Execution amount** which has added **$96,280** to your bill thus far. You'll continue to be charged **$40 per 100 GB Hrs.**

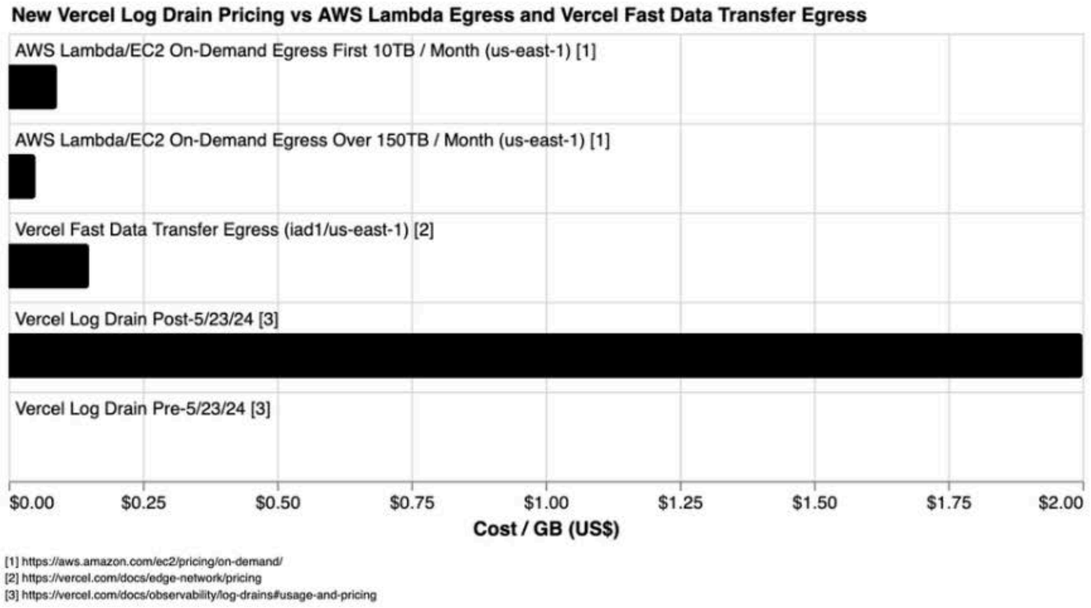
~7x markup!!!

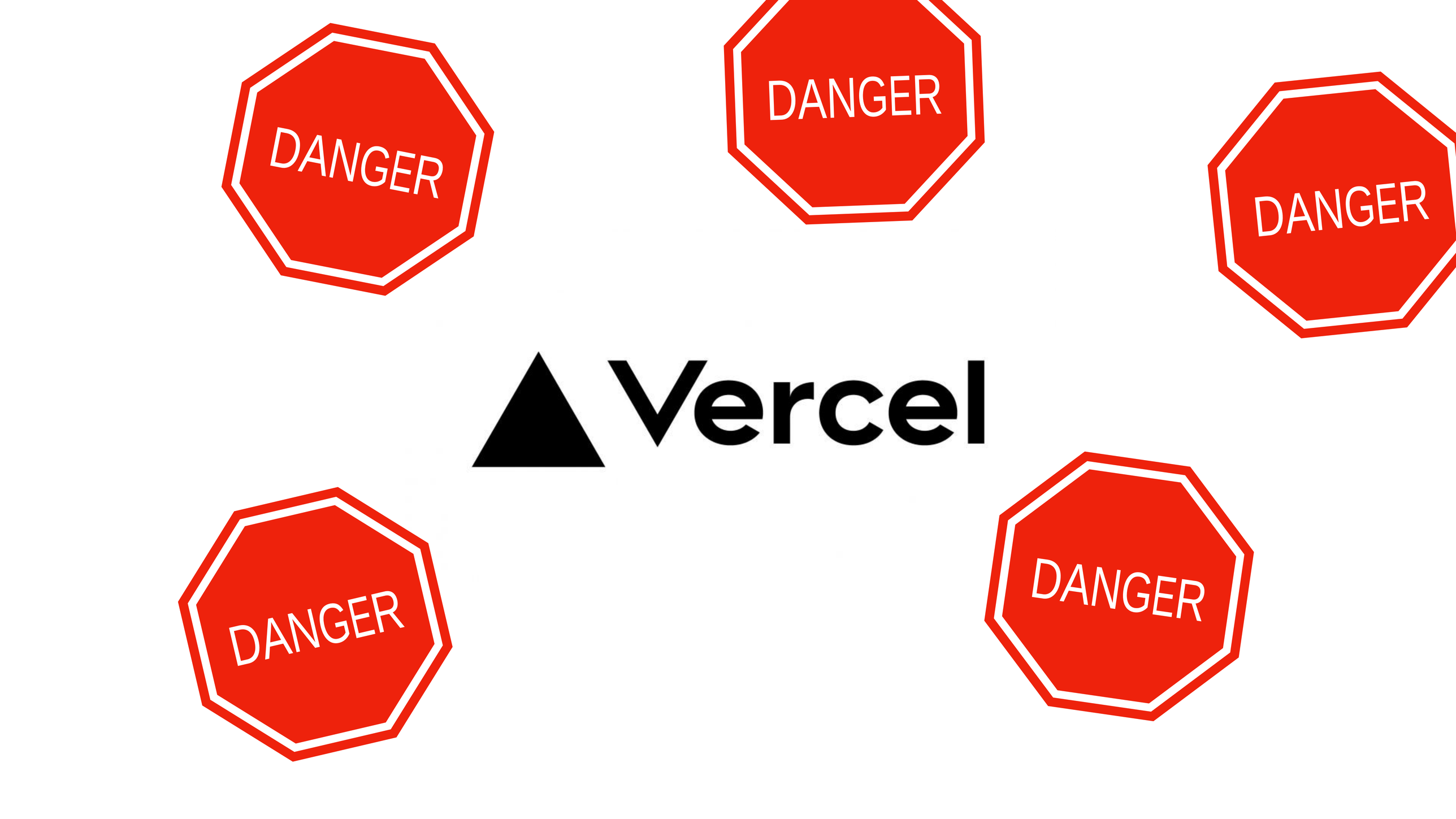
Lambda: $6 per 100 GB Hrs

AJ Stuyvenberg
@astuyve

Vercel just announced that Log Drains, a previously *free* capability, will now incur a $2/GB charge.

If that seems negligible, here's a chart showing how this compares with their regular egress charges:

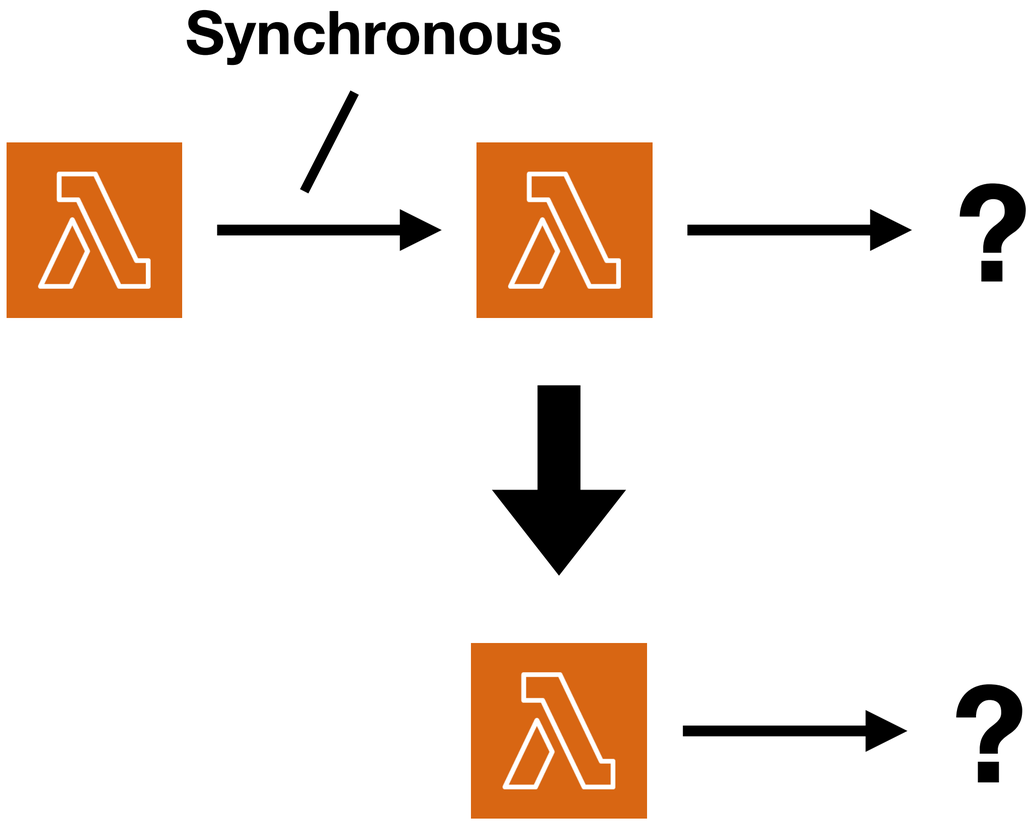


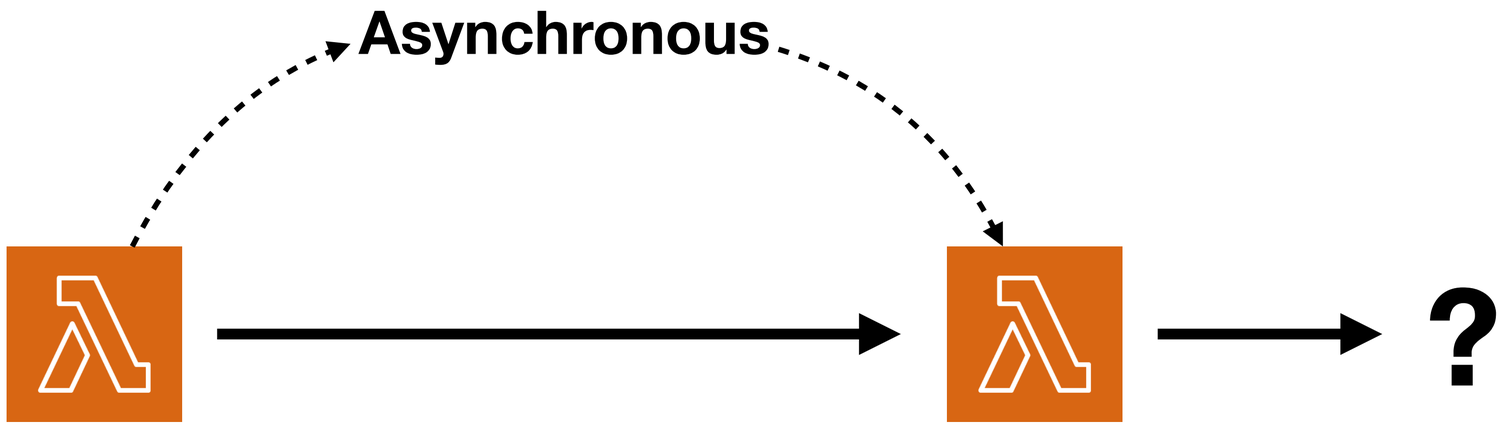

DANGER
DANGER
DANGER
DANGER
DANGER
Vercel

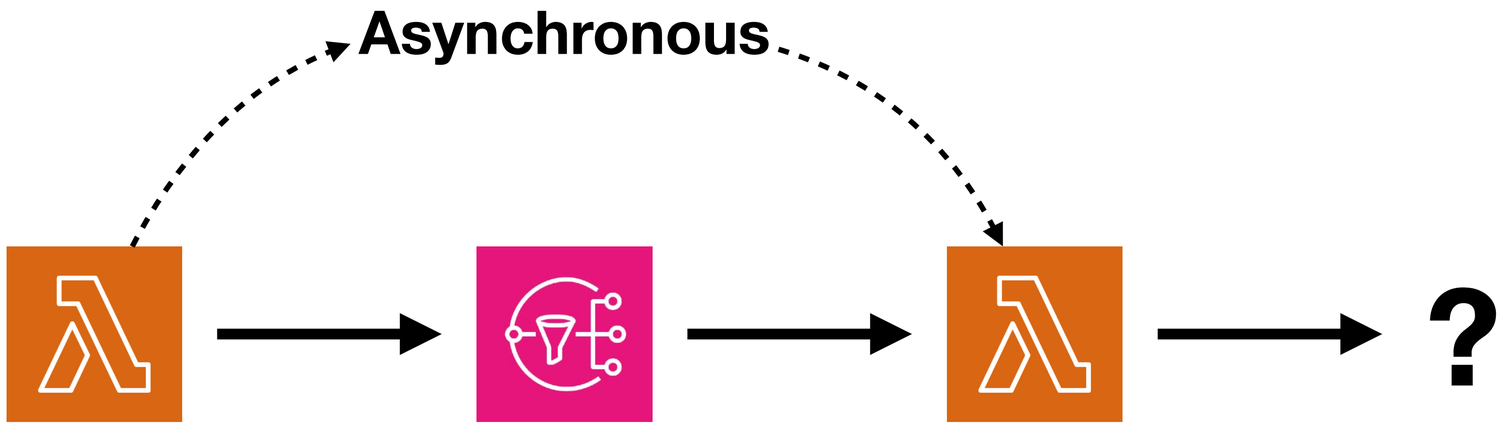
Simplify your architecture

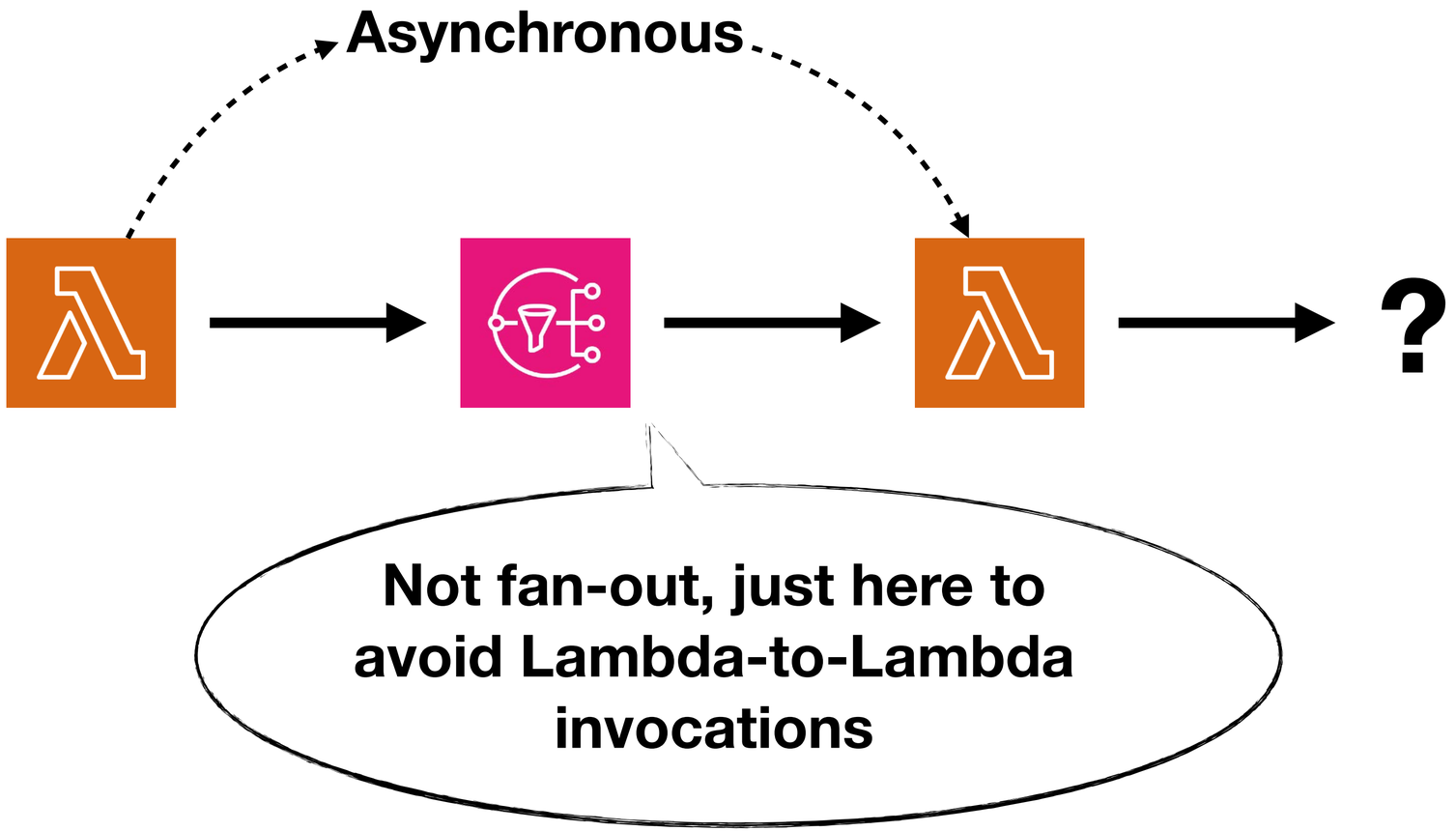
Avoid unnecessary moving parts to your architecture.

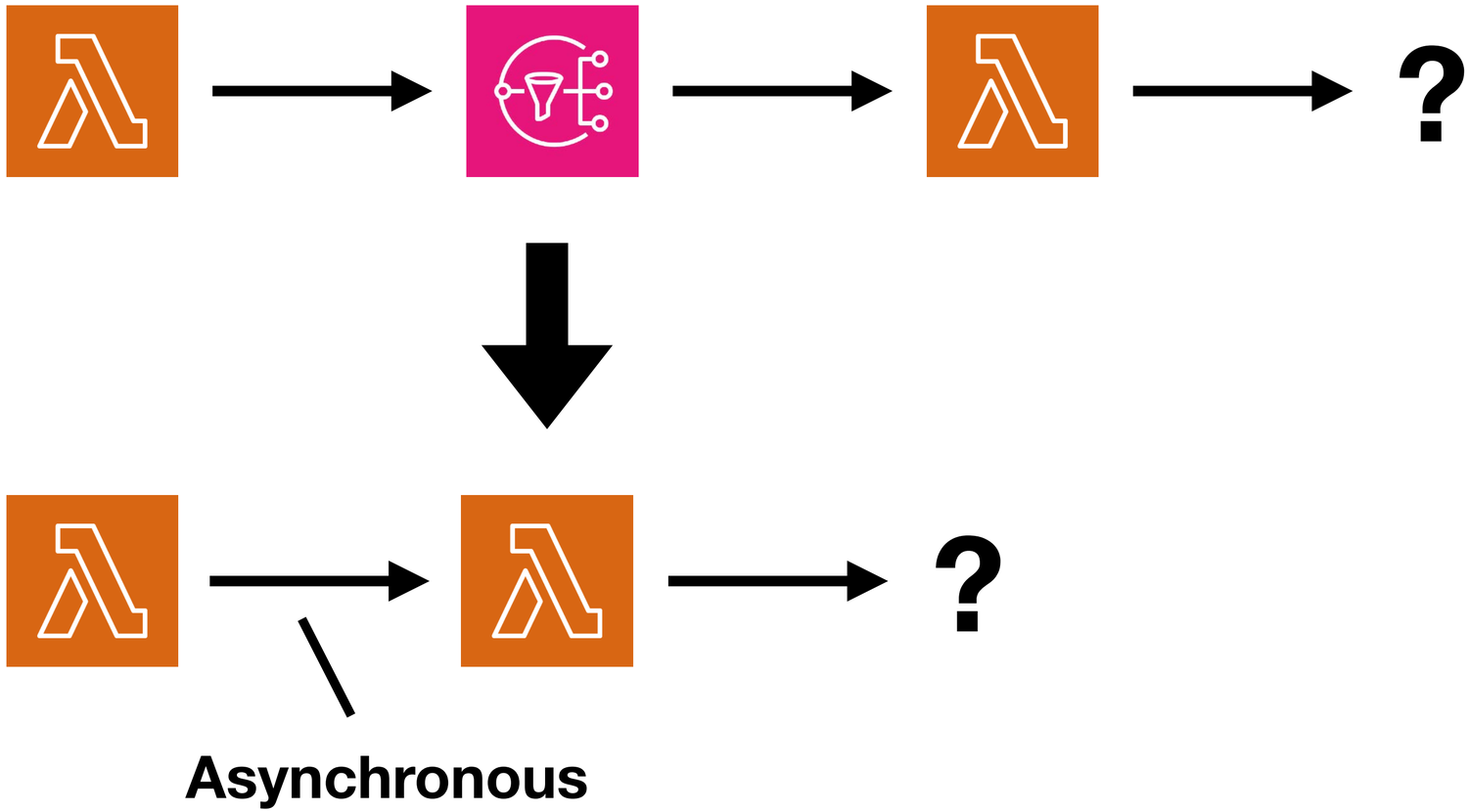

Synchronous
?


Synchronous
?
?


Asynchronous
?


Asynchronous
?


Asynchronous
?
Not fan-out, just here to avoid Lambda-to-Lambda invocations

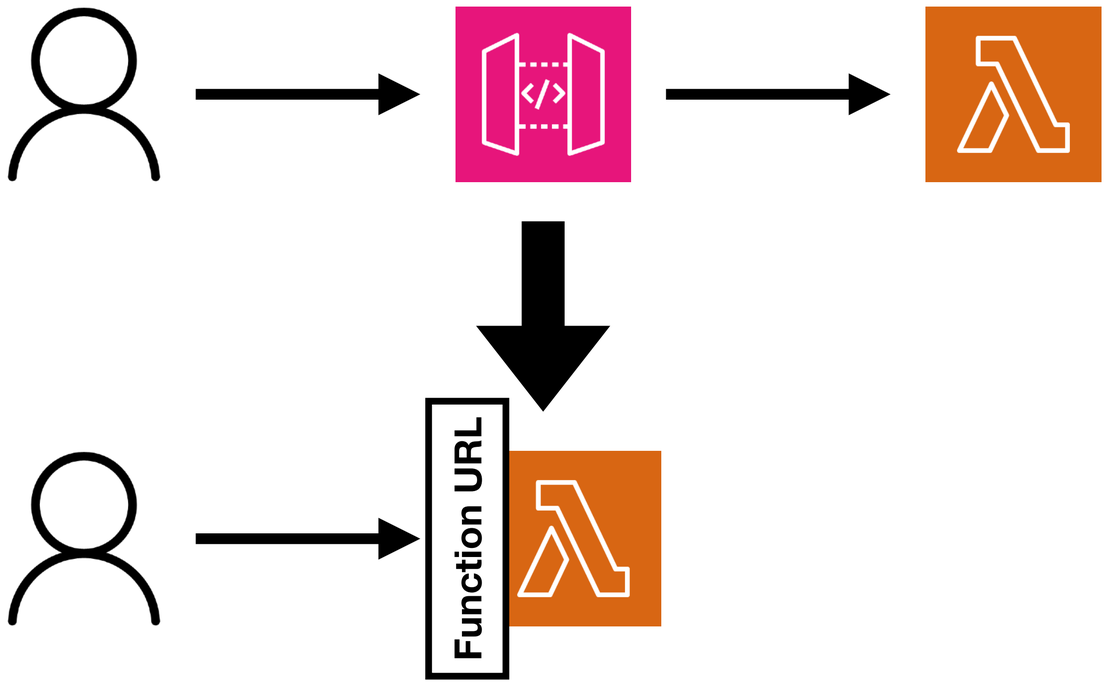

?
?
Asynchronous

Every component in your architecture should serve a purpose and provide a ROI.

The most dangerous phrase in the language is "we've always done it this way".

- Grace Hopper

Function URLs


Function URL

If you’re not using API Gateway features

(e.g. Cognito authoriser, request models, direct integration)

Or, if you’re hitting API Gateway limits

(e.g. 29s timeout, no response streaming)

Have to write Lambdaliths

Have to write Lambdaliths

(No per-endpoint metrics & alerts, no fine-grained access control, no per-endpoint auth)

Have to write Lambdaliths

(No per-endpoint metrics & alerts, no fine-grained access control, no per-endpoint auth)

(Large frameworks affect cold start performance)

Best for public or internal APIs

Brett Andrews
@AWSbrett

Okay it's not a terrible idea! If you bundle JWKS with your Lambda Function it costs ~4ms to validate first time per container. Once validated and cached it's ~0.3ms.

This is using aws-jwt-verify. May be able to find something more performant if you care about 4ms

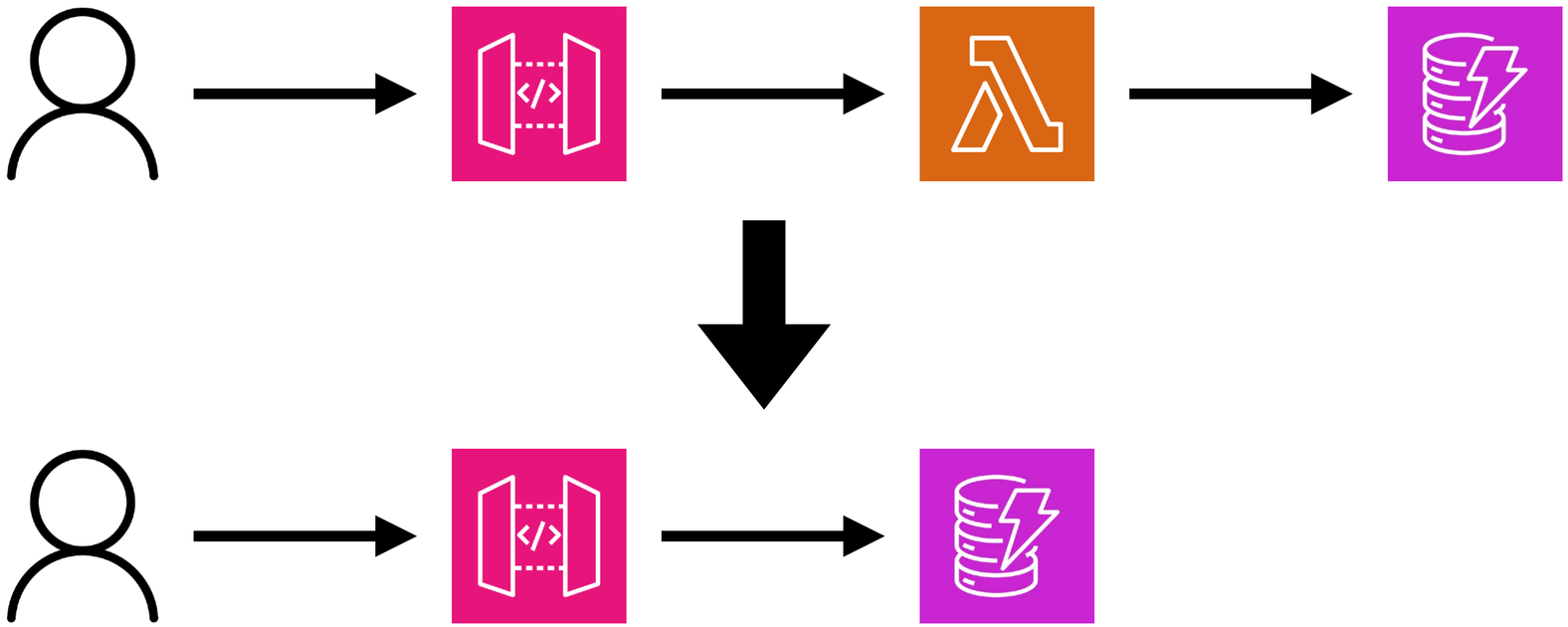
David Behroozi @rooToTheZ · Apr 7

Replying to @AWSbrett @heitor_lessa and @vinii_joga10

The keys don't rotate (yet), so you can just package the key with your lambda. My guess is it's easier to use APIG, but more performant to use Lambda if you include the key in your lambda.

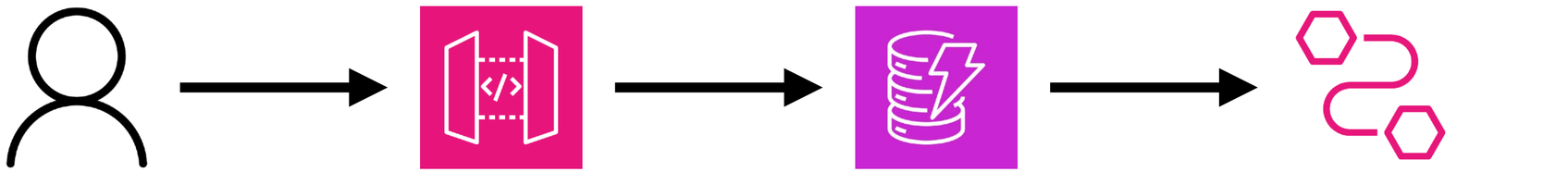
10:13 AM · Apr 14, 2024 · **1,048** Views

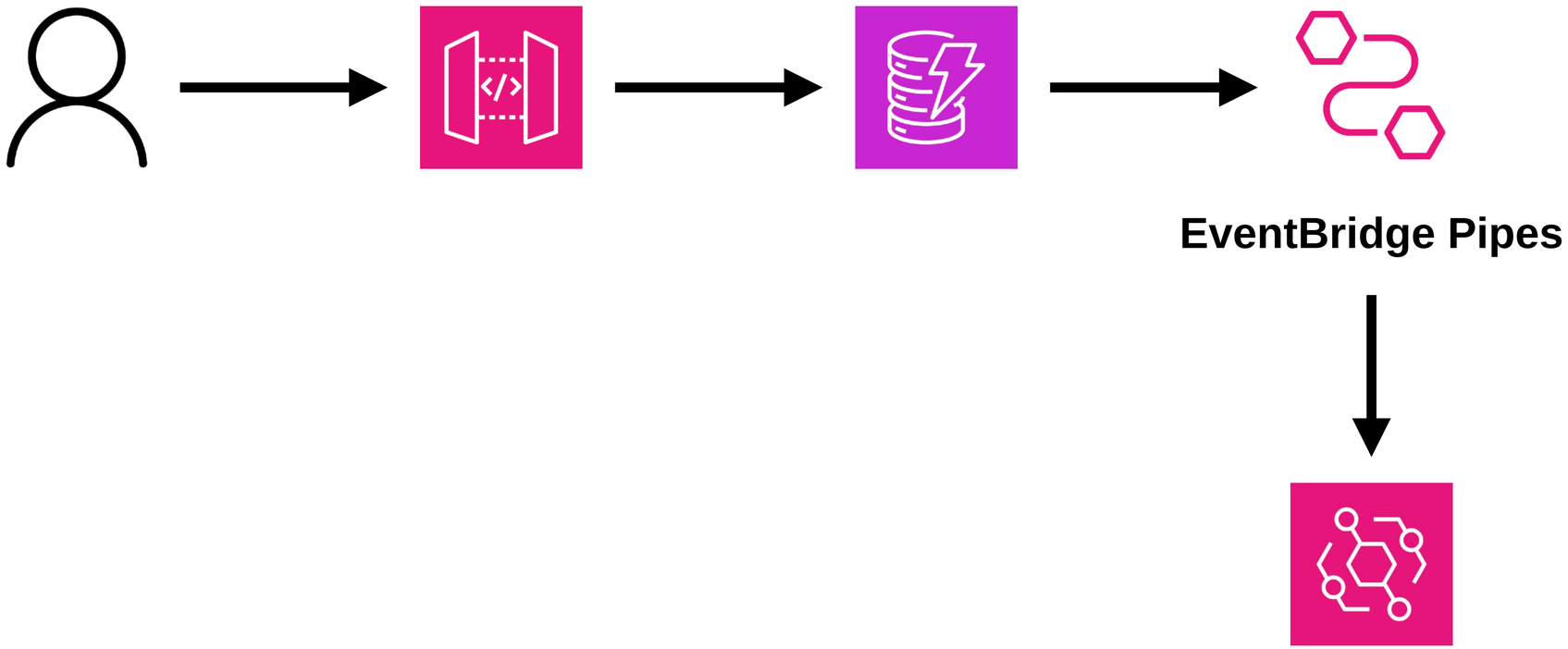
Functionless

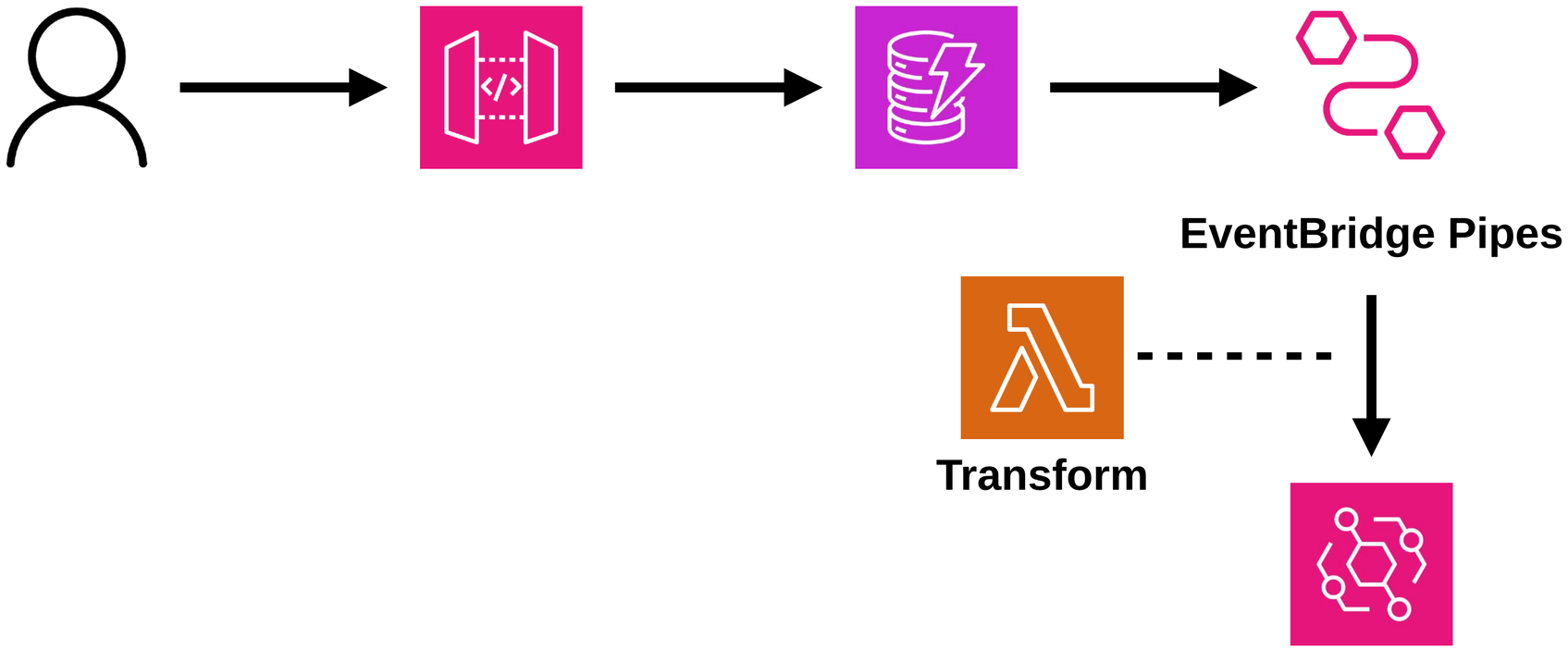


No Lambda = no cold starts

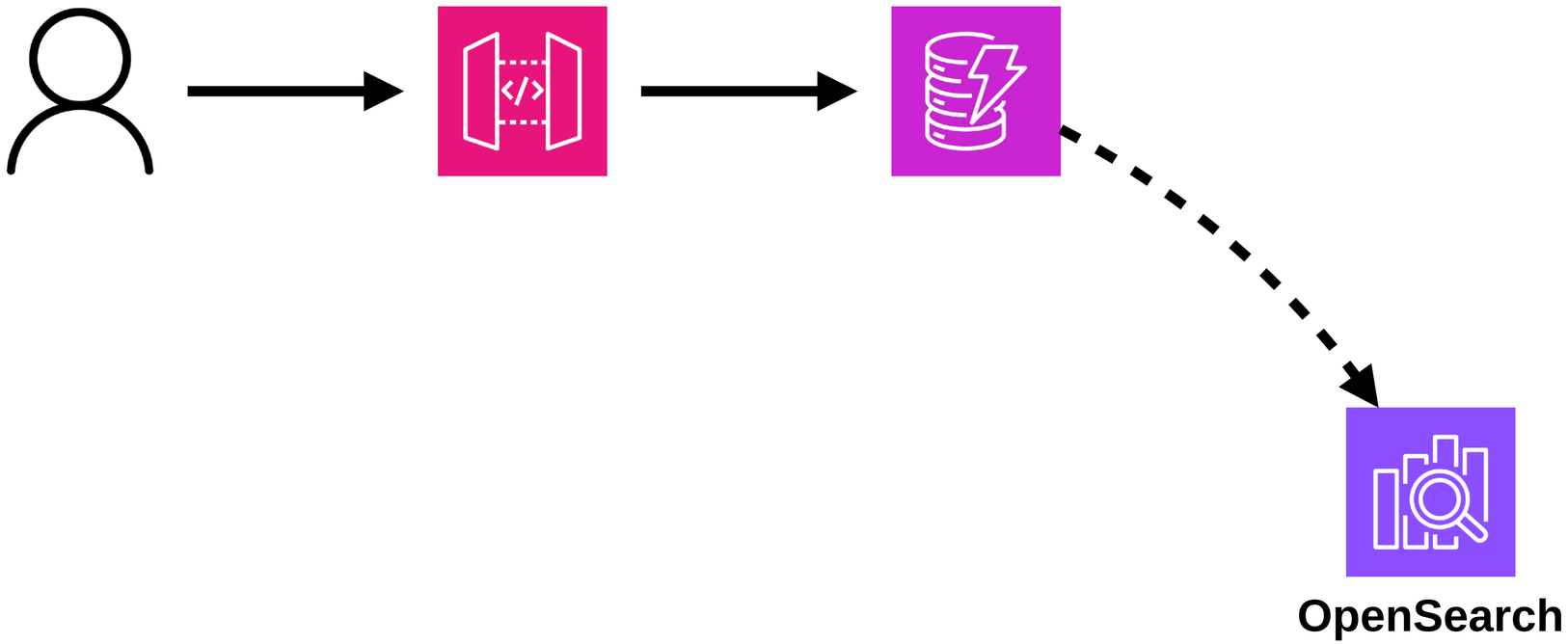
No Lambda = no Lambda costs

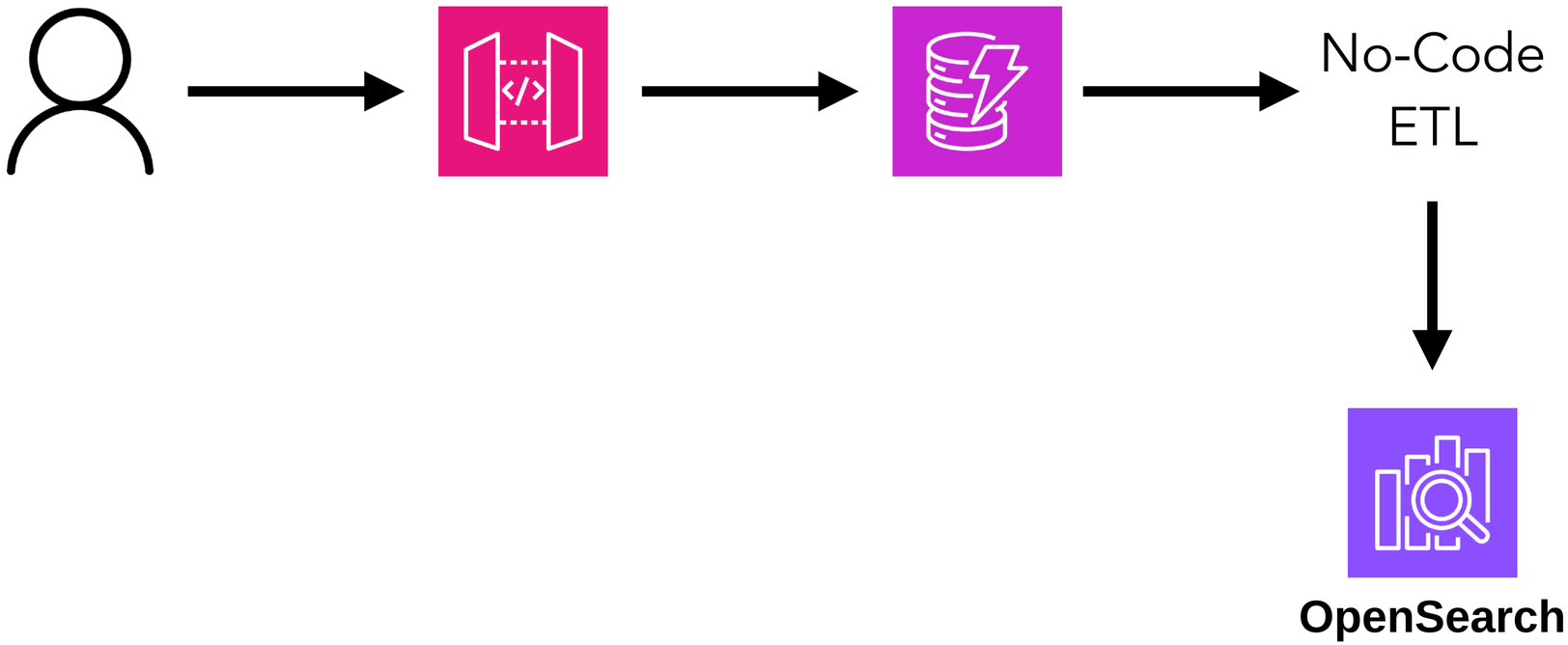

EventBridge Pipes

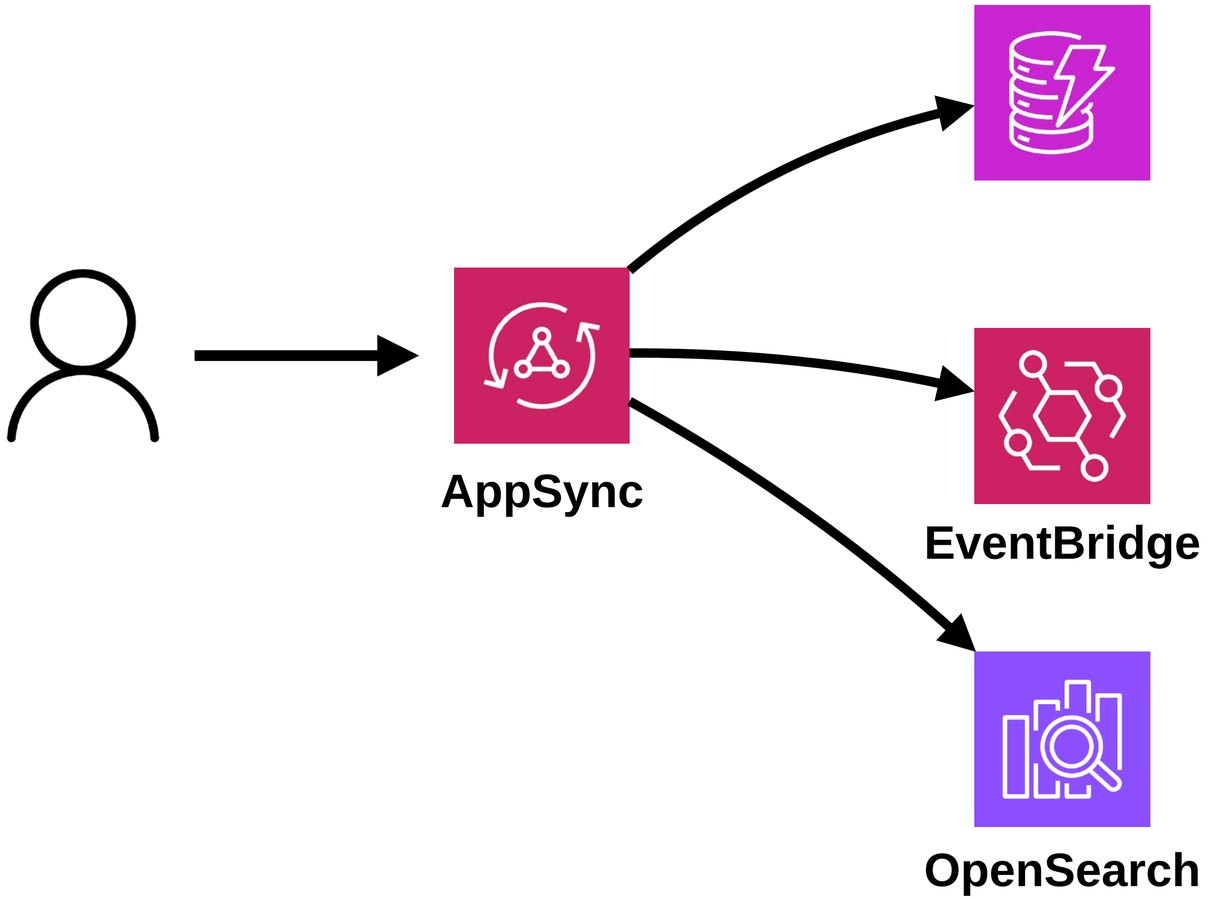

EventBridge Pipes

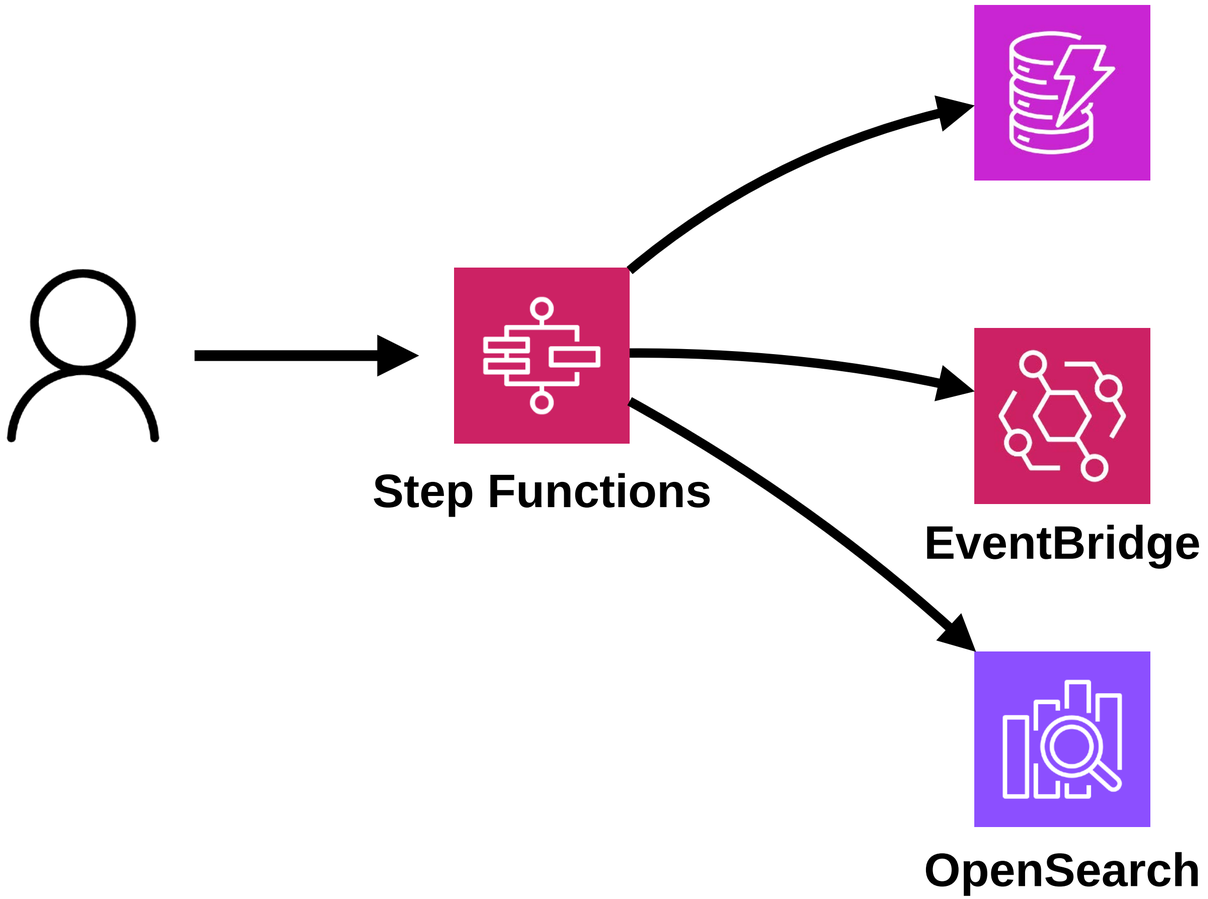

EventBridge Pipes
Transform

Use Lambda functions to transform data,
NOT transport data


OpenSearch

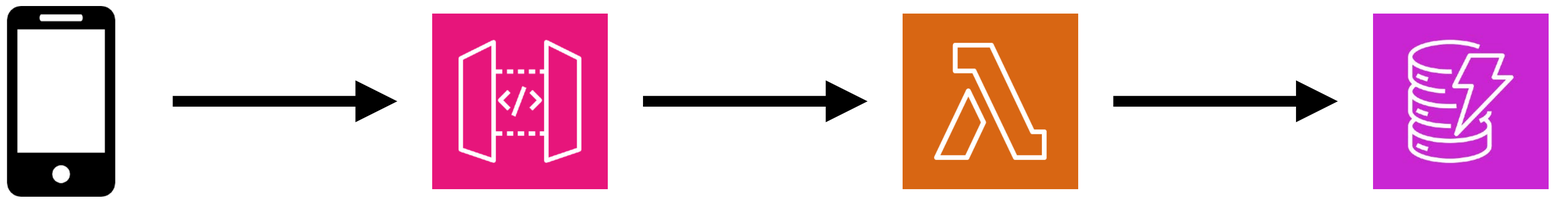

No-Code
ETL
OpenSearch

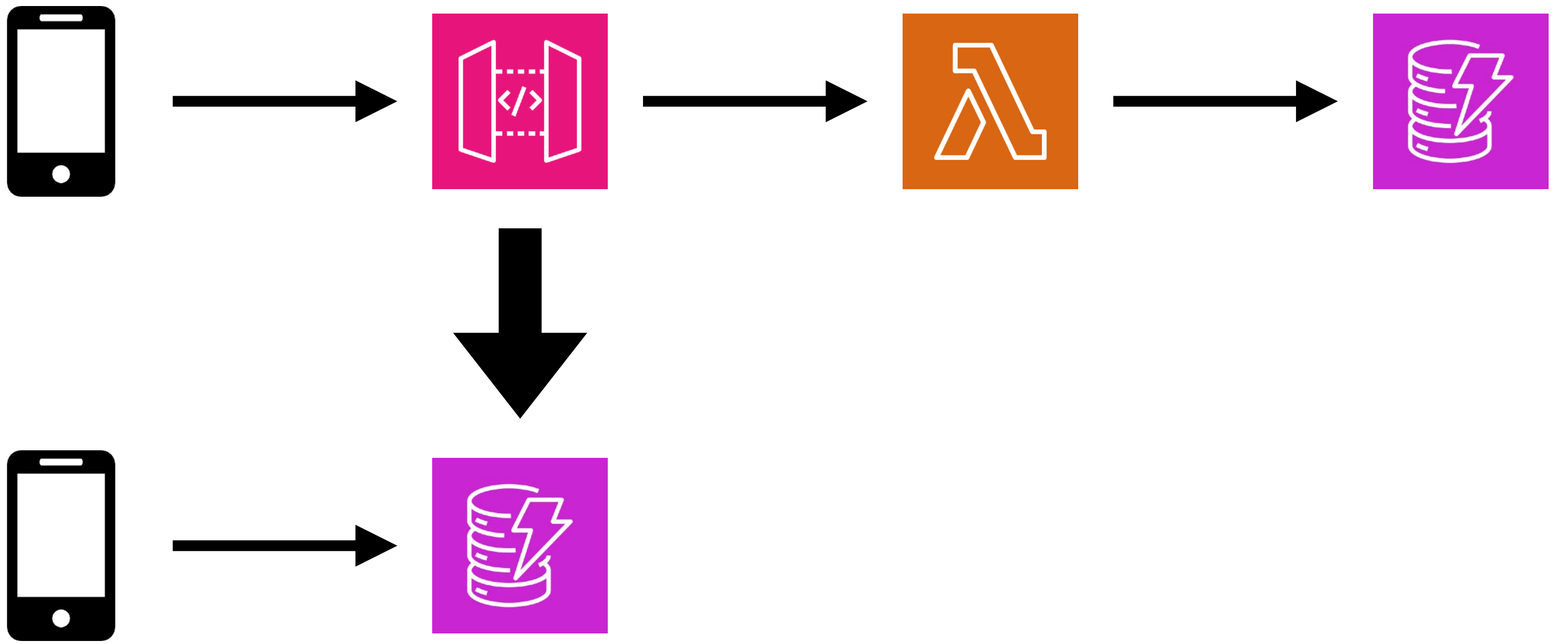

AppSync
EventBridge
OpenSearch

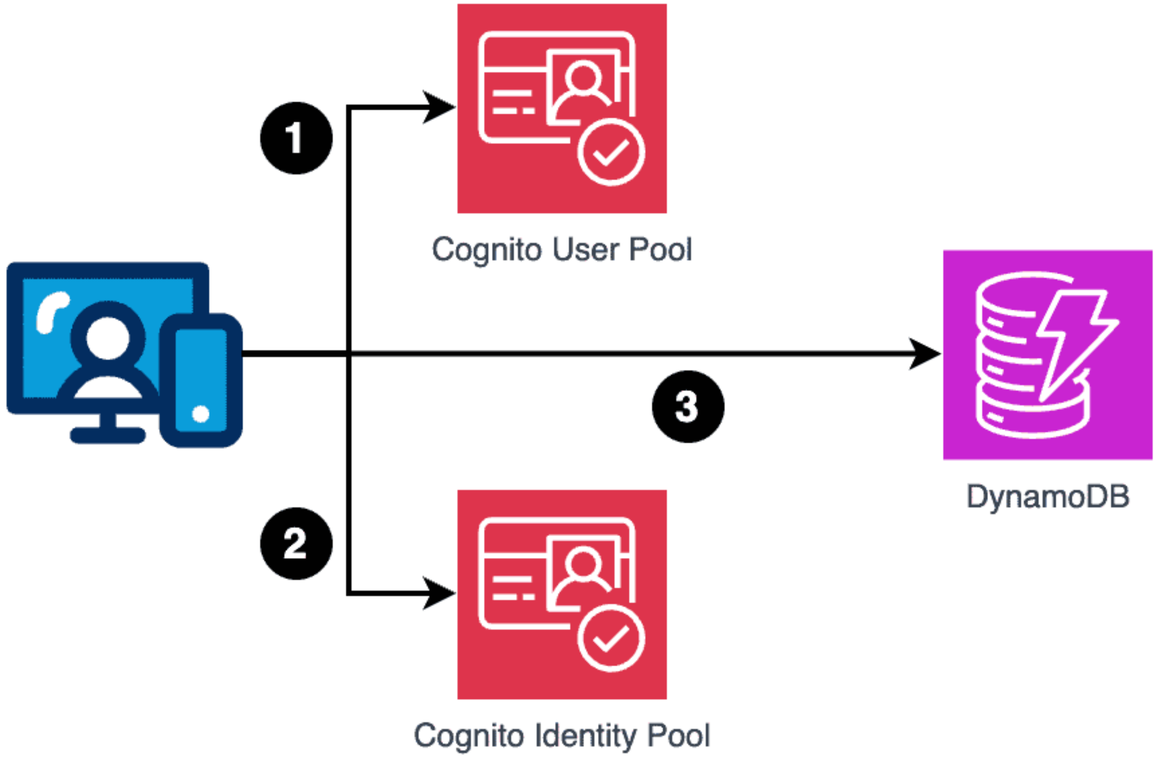

Step Functions
EventBridge
OpenSearch

Every component in your architecture should serve a purpose and provide a ROI.

Direct client access to AWS



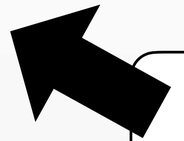



1
Cognito User Pool
3
DynamoDB
2
Cognito Identity Pool

```yaml
CognitoIdentityPoolRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: FeToDDBDemoAuthenticatedRole
    AssumeRolePolicyDocument:
      ...
    Policies:
      - PolicyName: FeToDDBDemoAuthenticatedRolePolicy
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - dynamodb:PutItem
                - dynamodb:GetItem
                - dynamodb:UpdateItem
                - dynamodb:DeleteItem
                - dynamodb:Query
              Resource: !GetAtt DynamoDBTable.Arn
              Condition:
                ForAllValues:StringEquals:
                  dynamodb:LeadingKeys:
                    - "${cognito-identity.amazonaws.com:sub}"
```

```
    - dynamodb:UpdateItem
    - dynamodb:DeleteItem
    - dynamodb:Query
  Resource: !GetAtt DynamoDBTable.Arn
  Condition:
    ForAllValues:StringEquals:
      dynamodb:LeadingKeys:
        - "${cognito-identity.amazonaws.com:sub}"
```

```
    - dynamodb:UpdateItem
    - dynamodb:DeleteItem
    - dynamodb:Query
  Resource: !GetAtt DynamoDBTable.Arn
  Condition:
    ForAllValues:StringEquals:
      dynamodb:LeadingKeys:
        - "${cognito-identity.amazonaws.com:sub}"
```



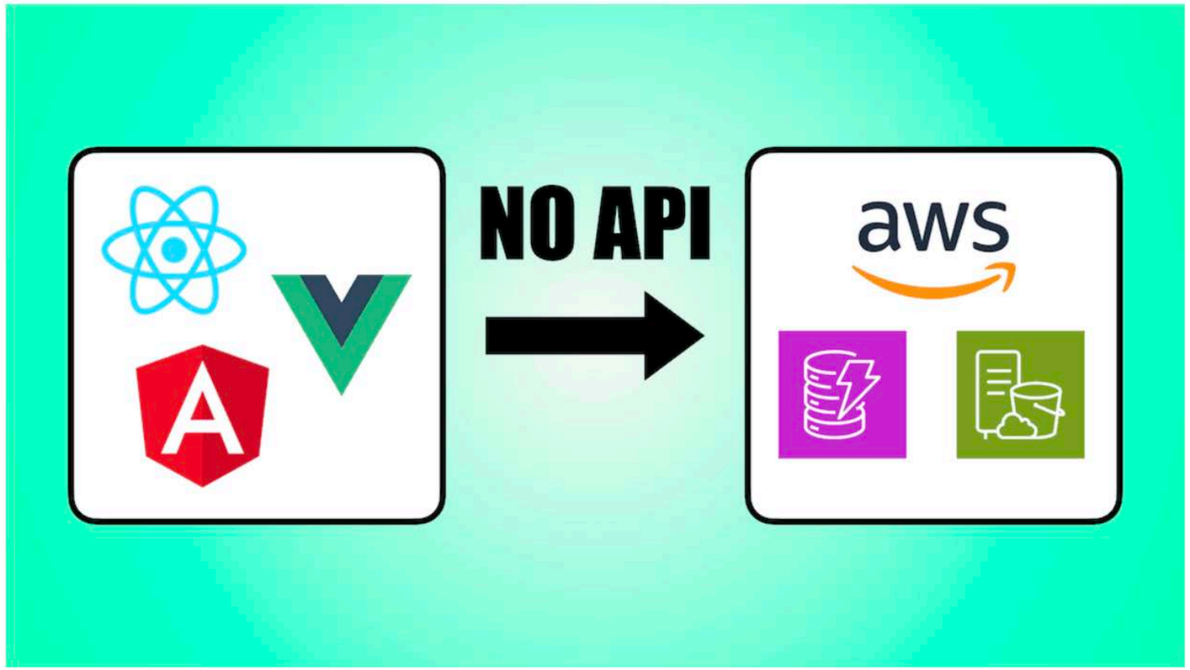
Only allow access if hash key matches cognito sub

Not for the feint hearted…

When not to use this:

- **Regulatory requirements**. Many regulations (e.g. HIPAA) require additional layers of logging, auditing, or data handling (e.g. encryption) that are impossible to enforce with this approach.
- **Complex authorization requirements**. If your application requires complex authorization logic that can't be easily mapped to IAM roles and policies, then this approach is also not suitable.
- **Rate limiting and Throttling**. This approach doesn't provide rate limiting or throttling. Which are often crucial in preventing abuse or simply protecting your application from denial-of-service attacks.
- **Sensitive business logic**. With this approach, all of your business logic would need to be implemented in the frontend application. For many organizations, this business logic is a trade secret and needs to be closely guarded.
- **Global audience**. Applications serving a global audience might face variable latencies when accessing AWS services directly from different regions. A well-configured CDN or intermediary API can offer more consistent performance globally.
- **Complex transactions**. If your application requires orchestrating multiple AWS services in a single request, managing this complexity on the client side can be challenging and might lead to inefficient code. An intermediary layer can abstract away this complexity.

High risk, high reward!



How to Securely let Frontend Apps to Directly Access AWS services

AWS, DynamoDB, S3, Serverless

In this post, let's discuss a radical idea – if the API layer is not adding any value besides authentication and calling the AWS SDK, then why not just remove it and let the frontend talk to your AWS resources directly? It will be the cheapest way to build a full-stack application, and there are similar precedents in the IoT space already.

It's not the way that I'd recommend for most of you. But it's possible to do it safely so that a user can only access his/her data. All you need is a little bit of IAM policy and a Cognito Identity Pool.

https://theburningmonk.com/2023/12/direct-access-for-frontend-apps-to-aws-services

1. Billing alarms
2. Keeping logging cost under control
3. Right-size Lambda functions
4. No Lambda-to-Lambda calls
5. Caching
6. Route53 TTL
7. Avoid CORS
8. Choosing the right service
9. Simplify your architecture
10. Function URLs
11. Functionless
12. Direct client access to AWS

Join 20+ AWS Heroes & Community Builders and 1000+ happy students in levelling up your serverless game.

productionreadyserverless.com

Questions?