CLOUD DAY 2020

29 OTTOBRE • #CLOUDDAY2020

PATTERN ARCHITETTURALI SERVERLESS PER LO SVILUPPO WEB SU AWS

Alex Casalboni

@alex_casalboni

Kudos

Microsoft

managed/designs

AWS User Group - Italy

AZURE MEETUP MILANO

GDG Cloud Milano

whoami

Software Engineer & Web Developer
Data Science background
Worked in a startup for 4.5 years
(Happy) AWS Customer since 2013

aws

whereami

linkedin.com/in/alexcasalboni
twitter.com/alex_casalboni
dev.to/alexcasalboni
github.com/alexcasalboni
medium.com/@alexcasalboni
twitch.tv/alexcasalboni

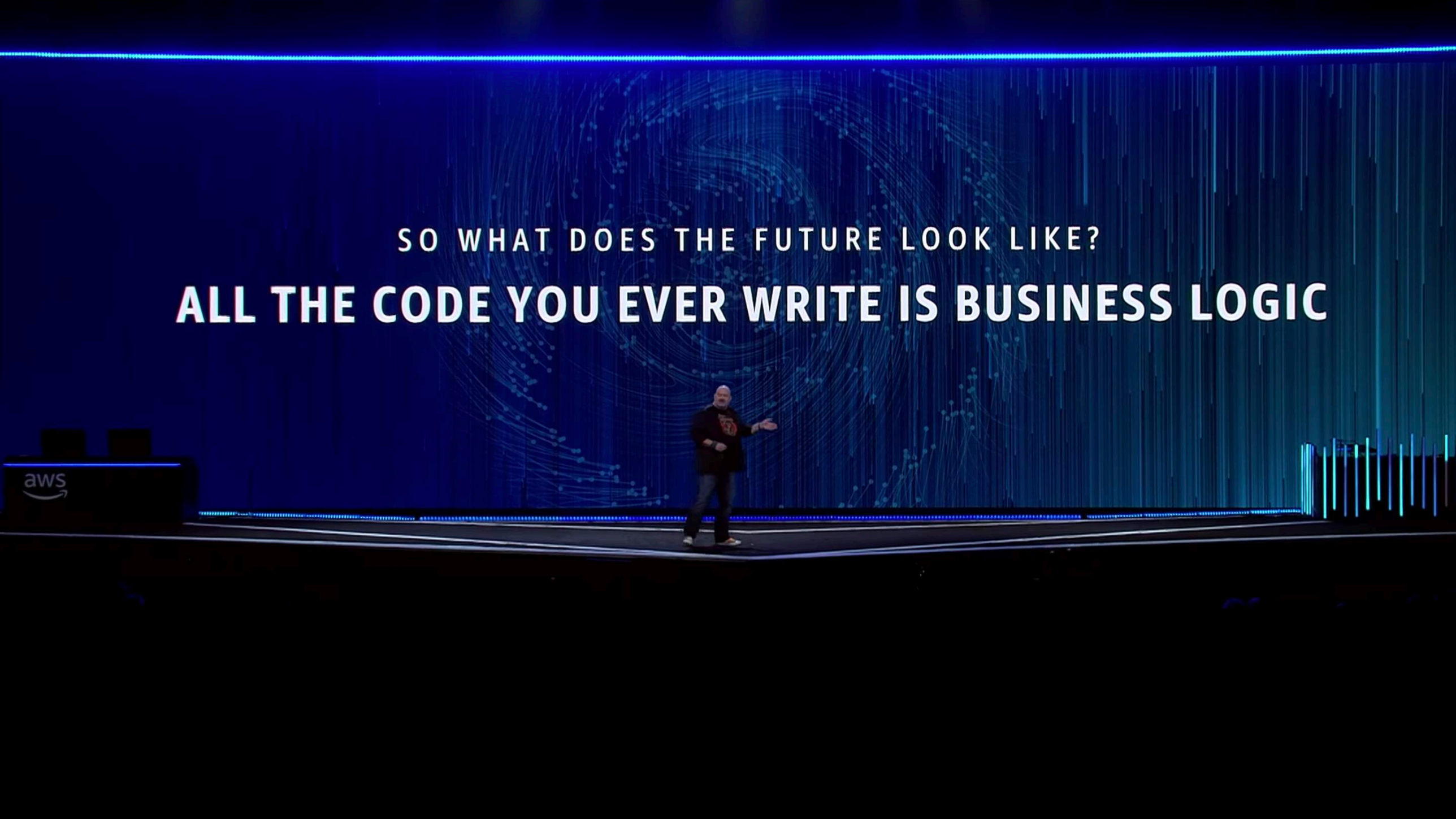
aws

Agenda

Serverless computing fundamentals

Architectural patterns for web apps

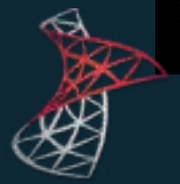
Architectural patterns for data analytics


SO WHAT DOES THE FUTURE LOOK LIKE?
ALL THE CODE YOU EVER WRITE IS BUSINESS LOGIC
aws

Compute Spectrum

“On Amazon EC2”

Amazon **EC2**



Microsoft SQL Server

Managed

Amazon EMR

Amazon RDS

Amazon Elasticsearch

Amazon ElastiCache

Amazon Redshift

Amazon SageMaker

Amazon Neptune

Amazon MQ

AWS Fargate

Amazon DocumentDB

Serverless

AWS Lambda

Amazon Cognito

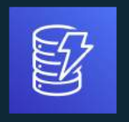
Amazon Kinesis

AWS Step Functions

AWS X-Ray

Amazon Athena

Amazon S3



Amazon DynamoDB

Amazon SQS

Amazon API Gateway

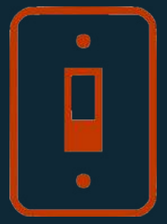
Amazon CloudWatch

AWS IoT

In other words, serverless means…

No provisioning,
no management

Automatic
scaling



Pay for value

Highly available
and secure

Ok, but who's been adopting it?

Growing ecosystem

Anatomy of a Lambda function

Handler() function

Function to be executed upon invocation

Event object

Data sent during Lambda function Invocation

Context object

Methods available to interact with runtime information (request ID, log group, more)

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello World!')
    }
```

aws

```
Import sdk
Import http-lib
Import ham-sandwich

Pre-handler-secret-getter()
Pre-handler-db-connect()
```

Dependencies, configuration information, common helper functions

```
Function myhandler(event, context) {
    <Event handling logic> {
            result = SubfunctionA()
        }else {
                result = SubfunctionB()

    return result;
}
```

Your handler

```
Function Pre-handler-secret-getter() {
}

Function Pre-handler-db-connect(){
}
```

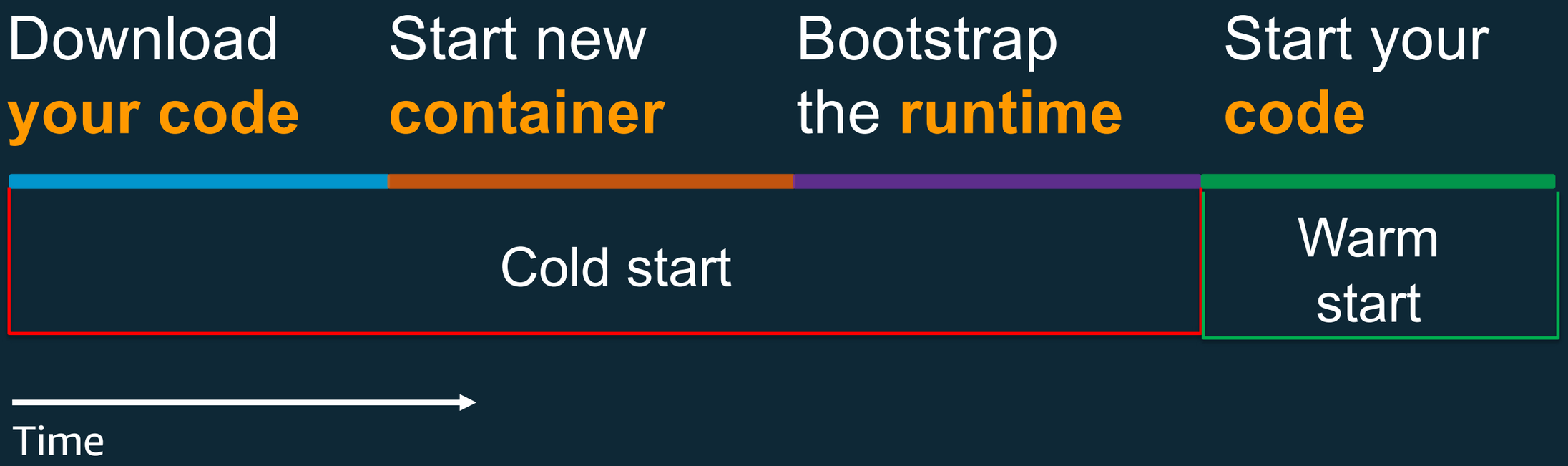
Common helper functions

```
Function subFunctionA(thing){
 ## logic here
}

Function subFunctionB(thing){
 ## logic here
}
```

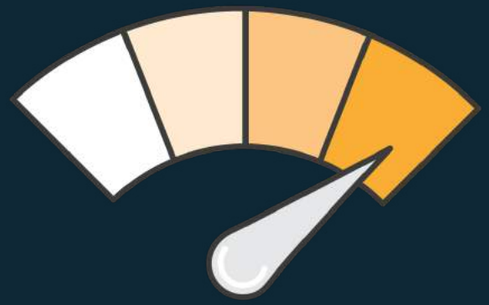
Business logic subfunctions

Serverless Functions - execution lifecycle



Tune your function’s resources (“Power”)

> Memory, > Cores, > Network



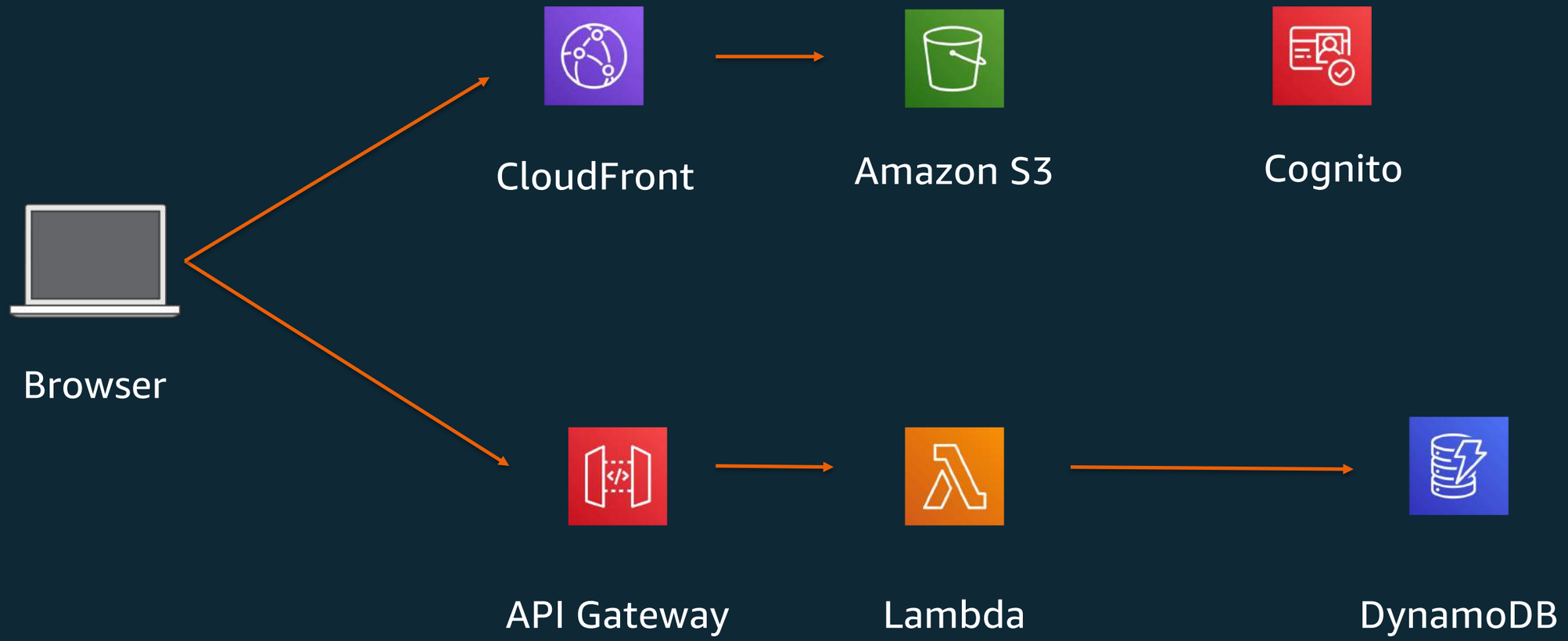
Only a memory control - **% of CPU core and network capacity**

Is your code CPU, network or memory-bound?
If so, it could be **cheaper** to choose more memory

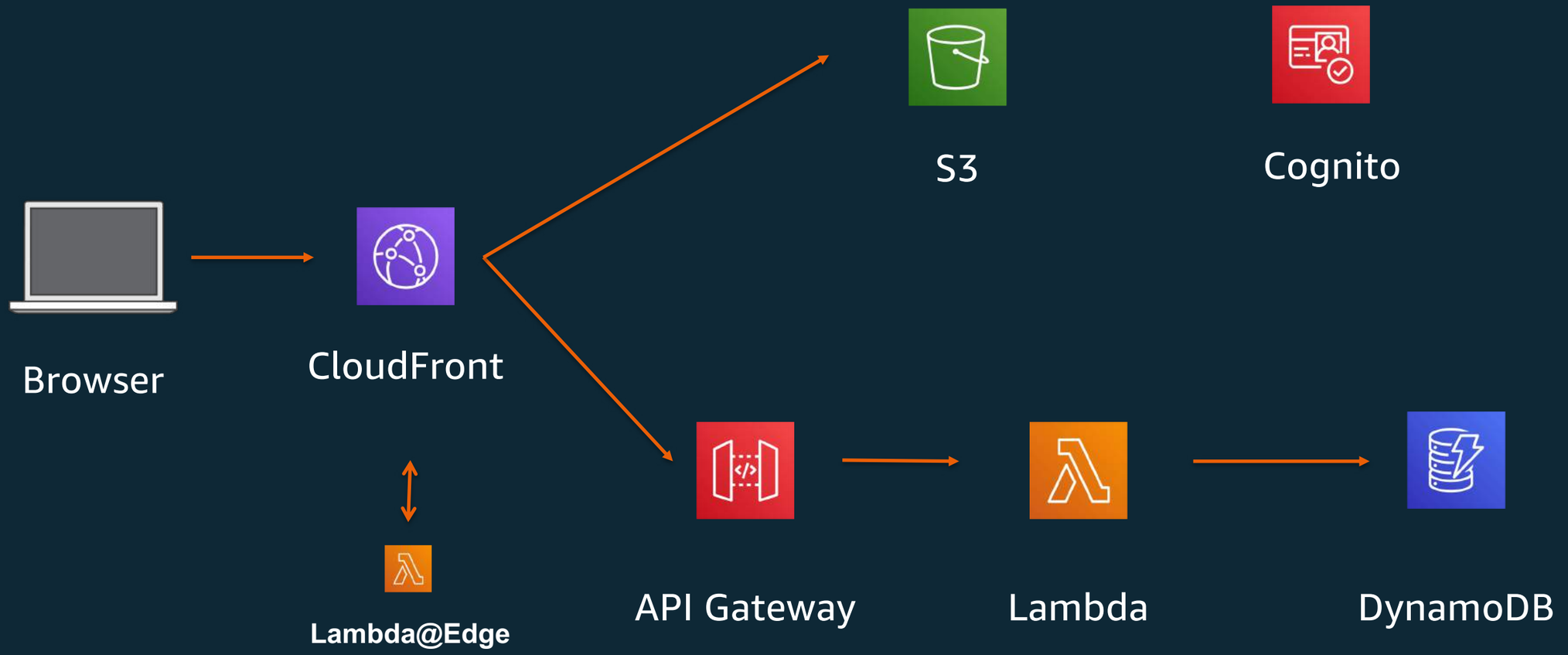
aws

Architectural patterns for Web apps / microservices / APIs

Web application (1)



Web application (2)



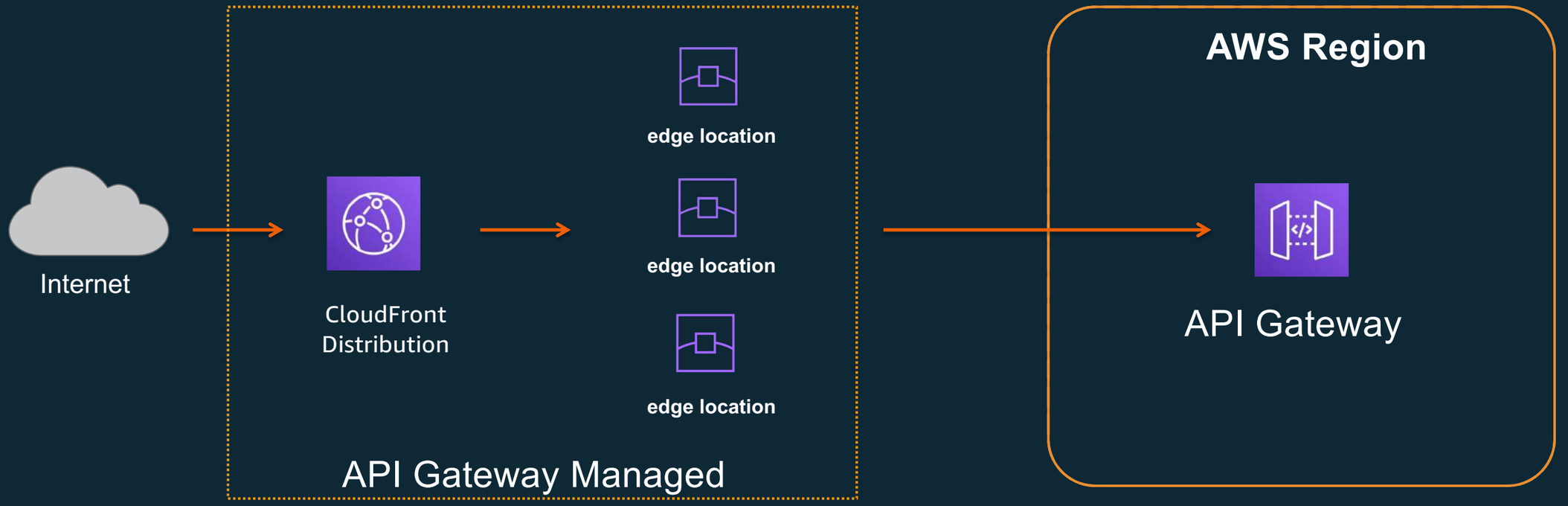
Choose the right API endpoint type

Edge optimized: reduce latency from anywhere on the Internet

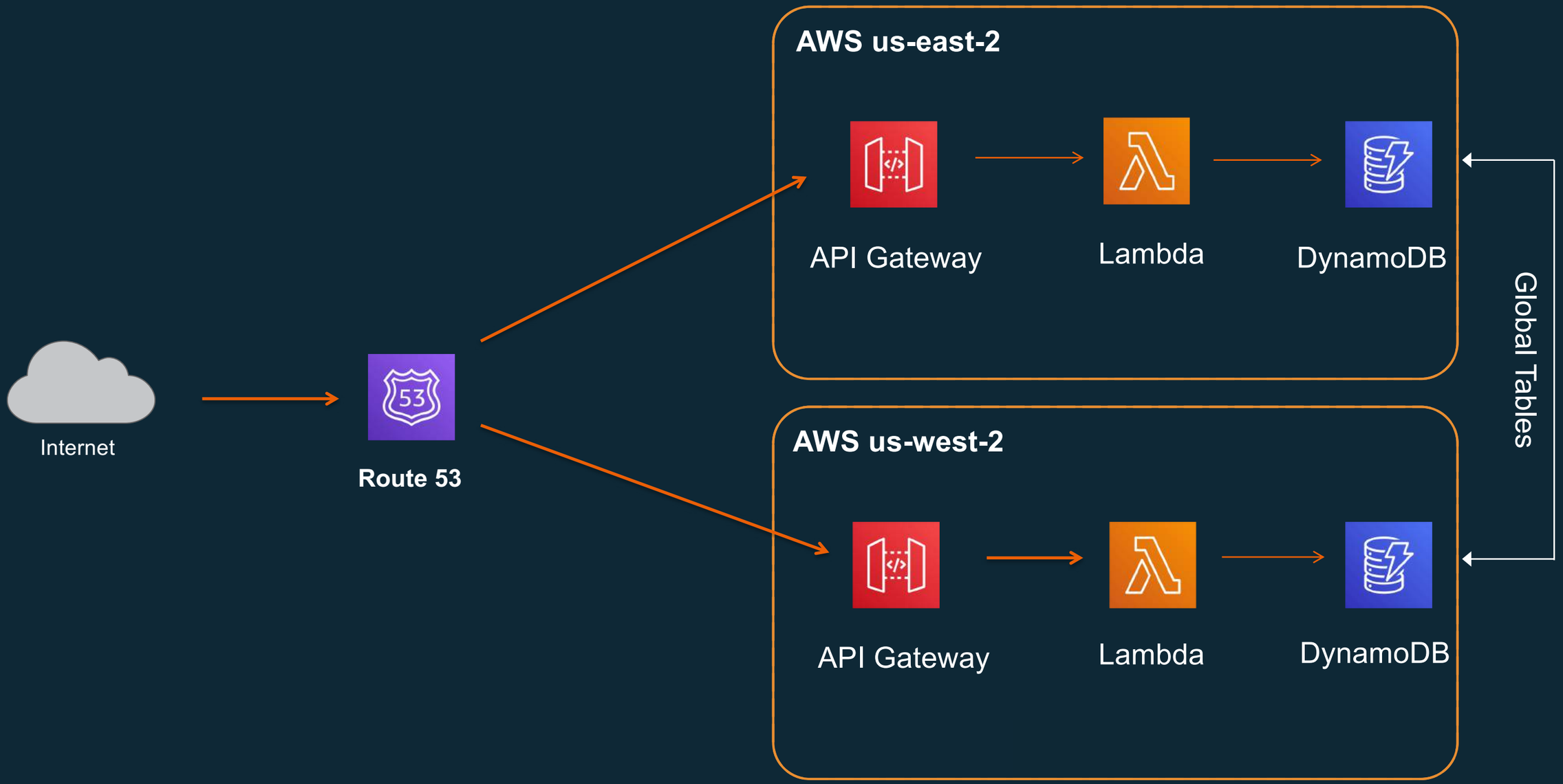
Regional: no CDN, ideal for in-region services

Private: expose APIs only inside your VPC

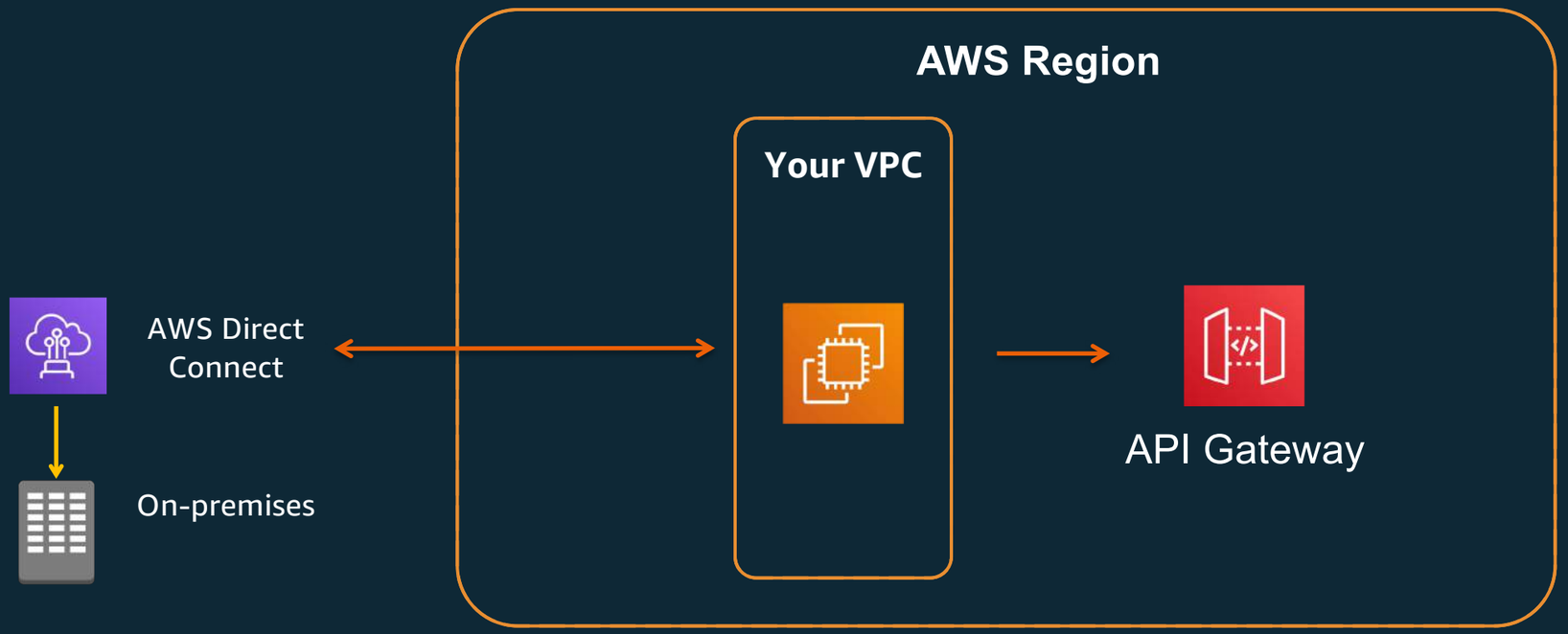
Choose the right API endpoint type (Edge)



Choose the right API endpoint type (Regional)

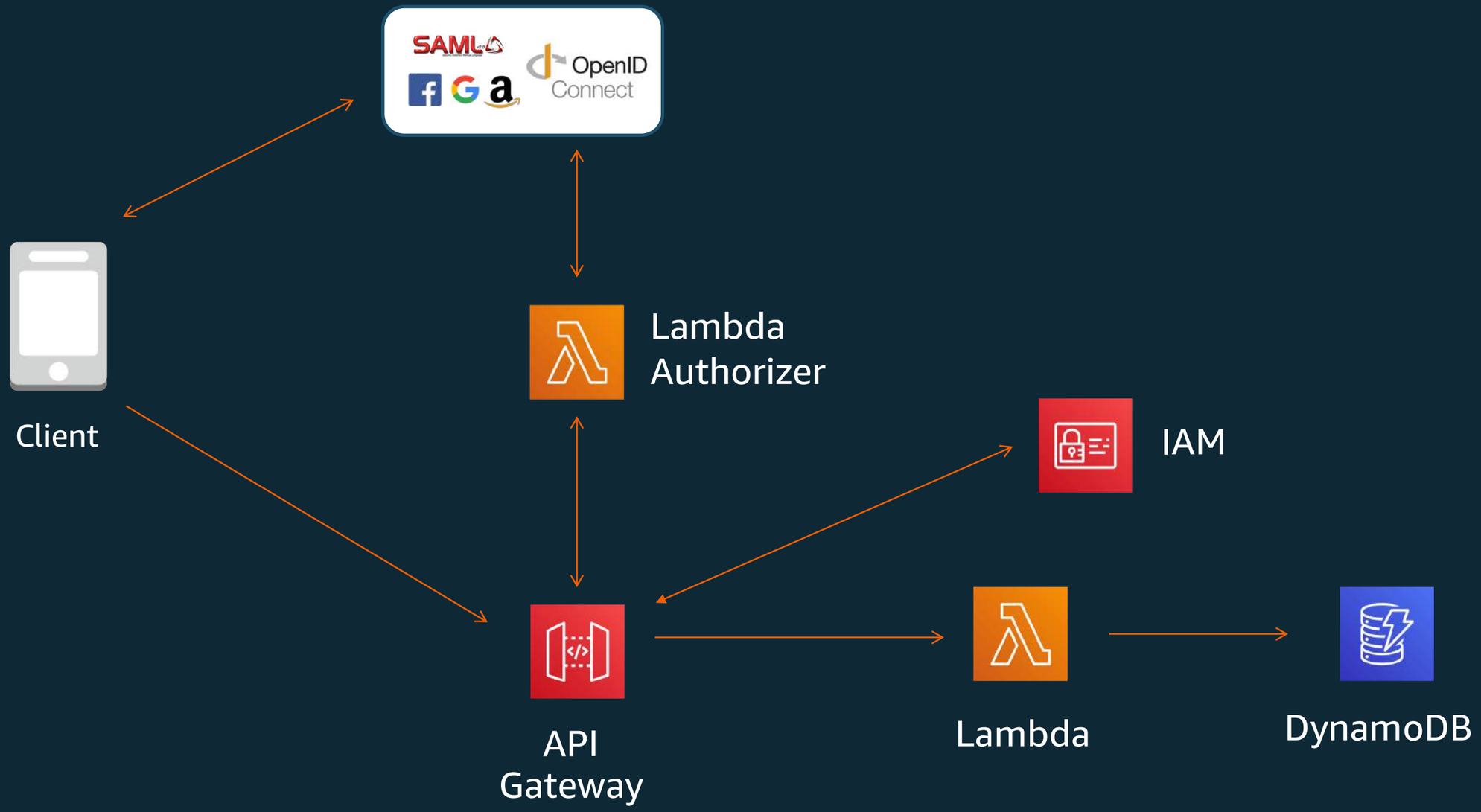


Choose the right API endpoint type (Private)



Serverless web-app security

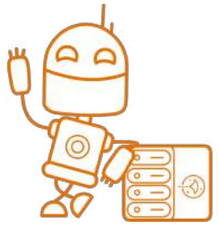
Custom authorizers



What if my requirements are different?

Sometimes you just need a CMS

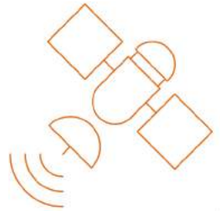
AWS Lightsail



Computing power

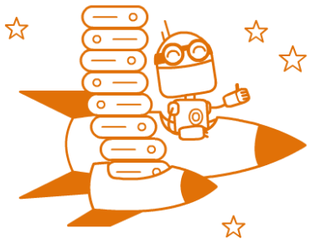
DNS management

One static IP/instance

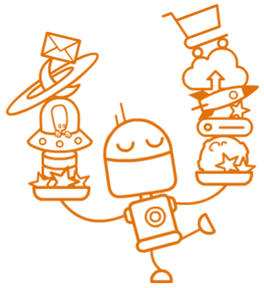


Networking & data transfer

Create larger instances



Add attached block storage

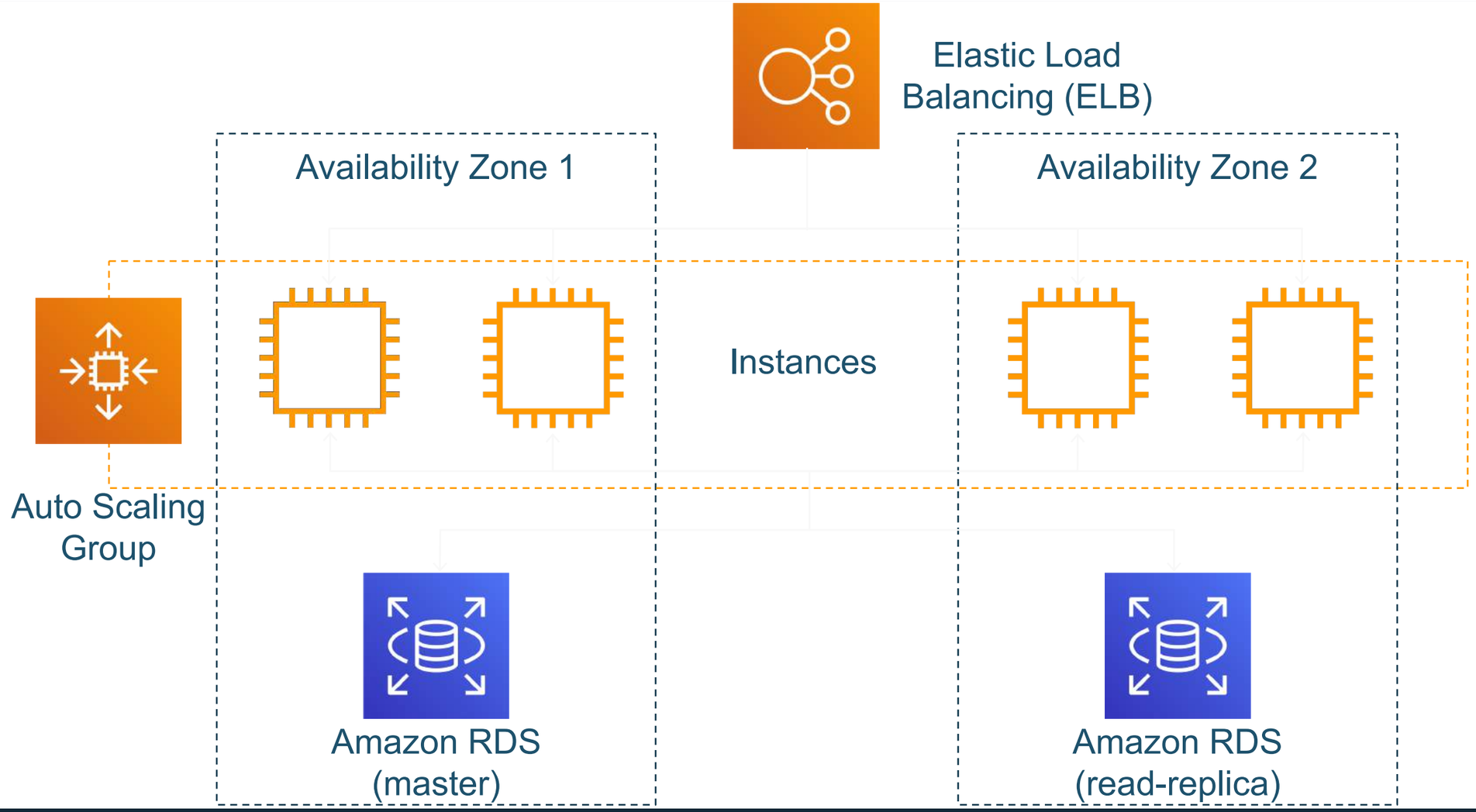


Load balance your application

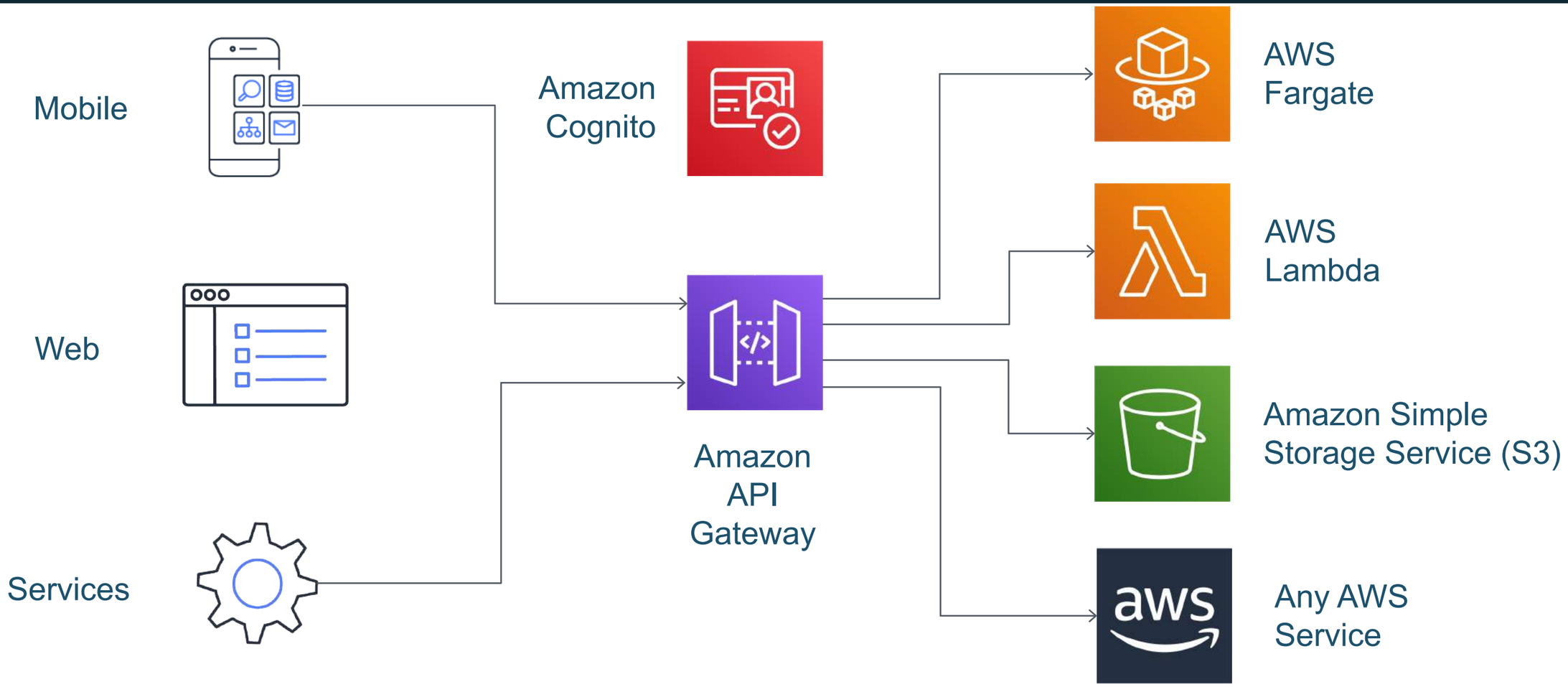
Connect to AWS services

The happy monolith

AWS Elastic Beanstalk

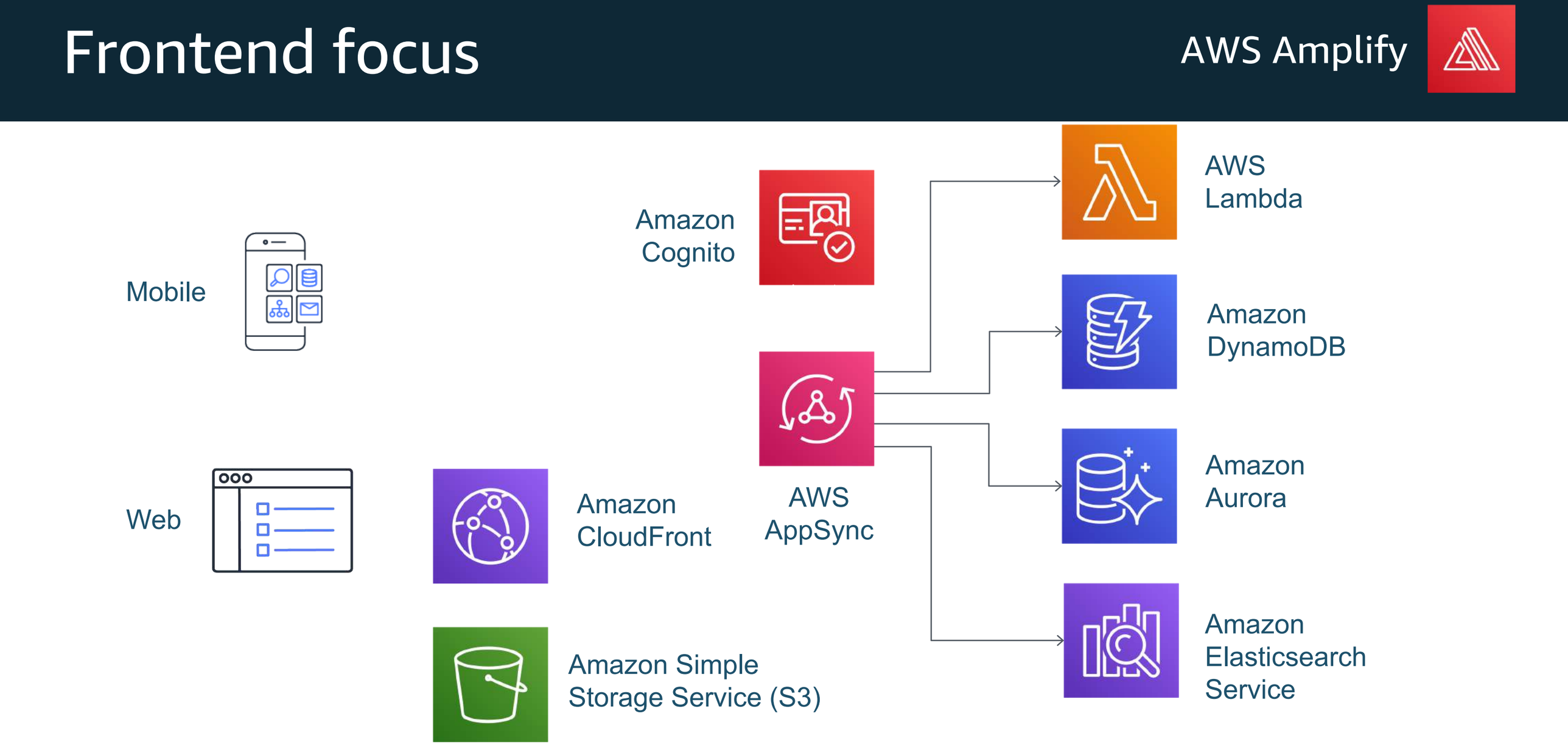


Backend focus

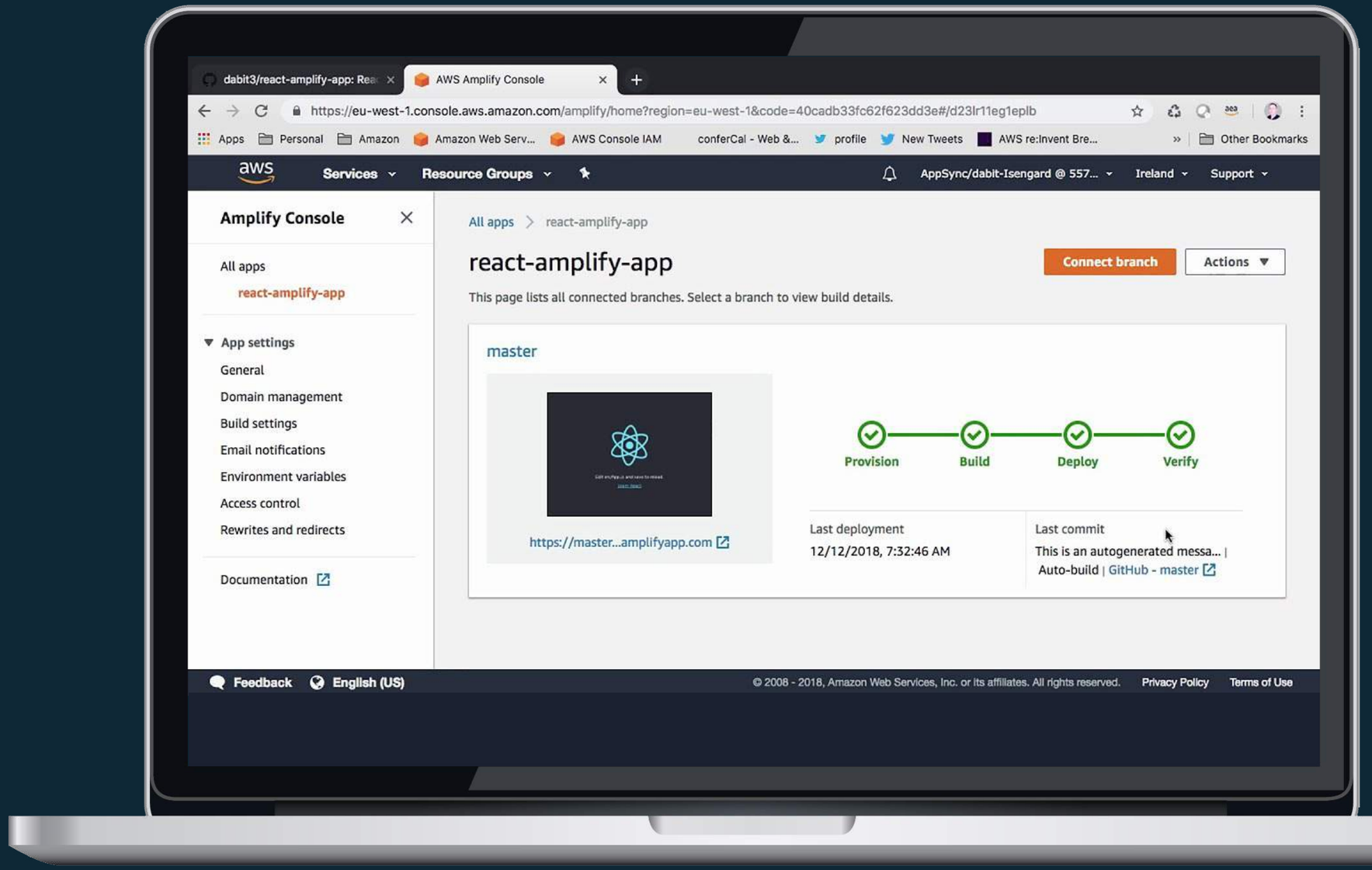


Frontend focus

AWS Amplify



Static / Single-page web apps (Amplify Console)

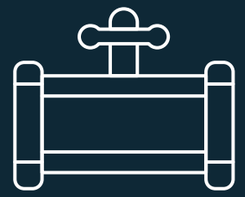


Data ingestion and real-time analytics

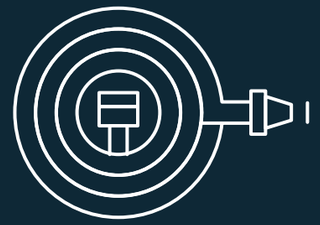
Streaming with Amazon Kinesis

Collect, process, and analyze video and data streams in real time

Kinesis Video Streams



Kinesis Data Streams

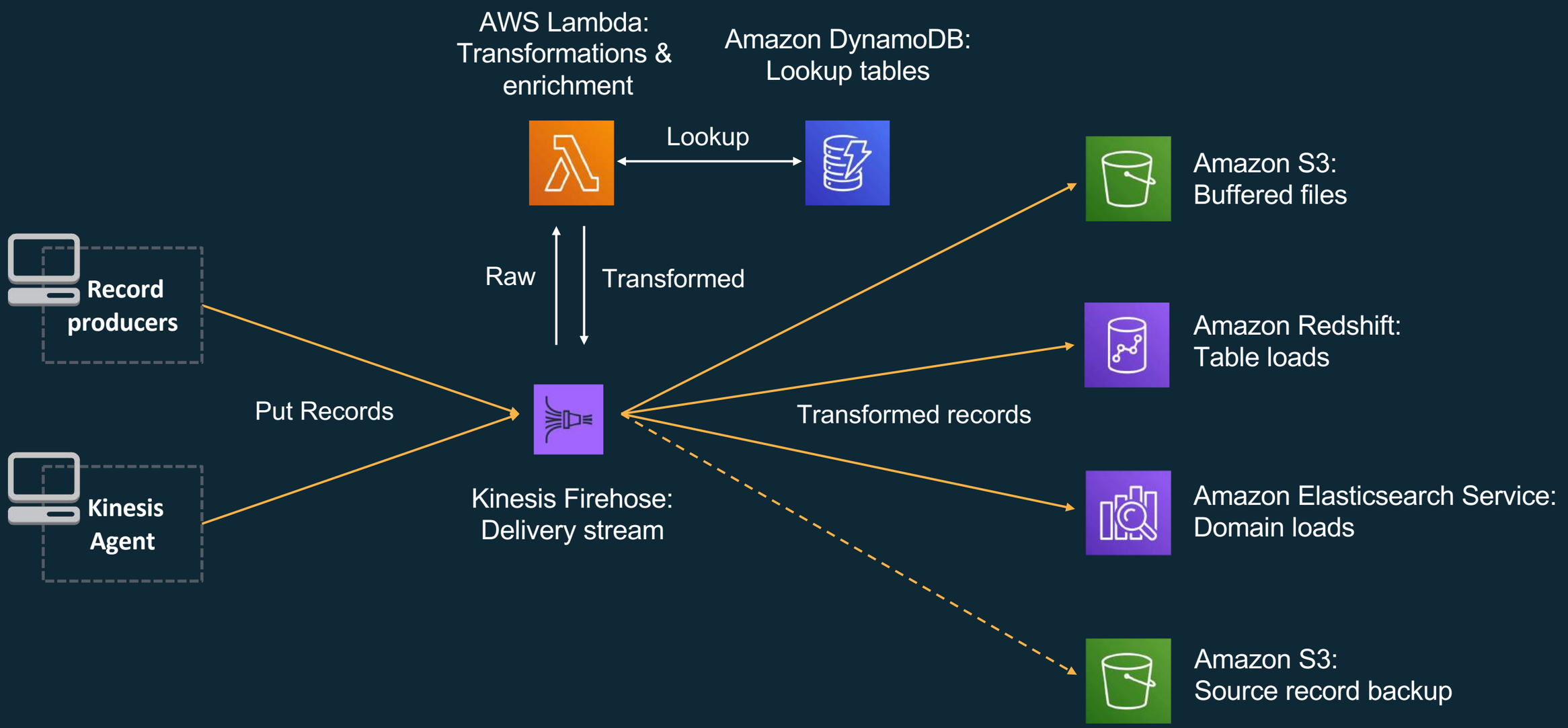


Kinesis Data Firehose

Kinesis Data Analytics

aws

Serverless data ingestion



Kinesis Firehose Best practices

Tune buffer size and buffer interval

- Larger objects = fewer Lambda invocations & Amazon S3 PUTs

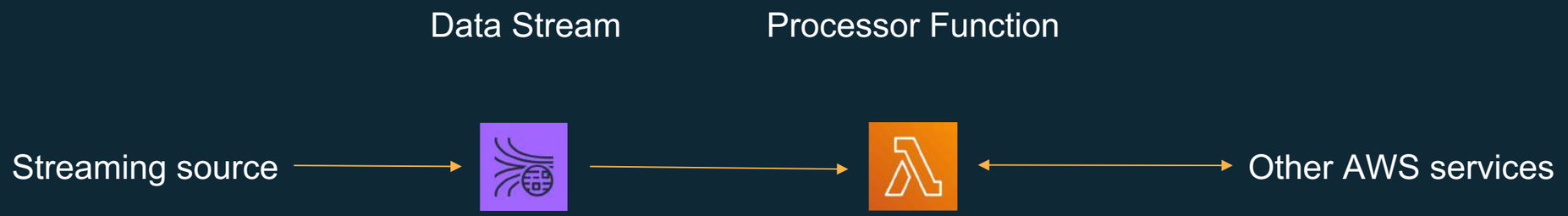
Enable compression to reduce storage costs

Enable Parquet format transformation (columnar)

Enable Source Record Backup for transformations

- Recover from transformation errors

Kinesis Data Streams and Lambda

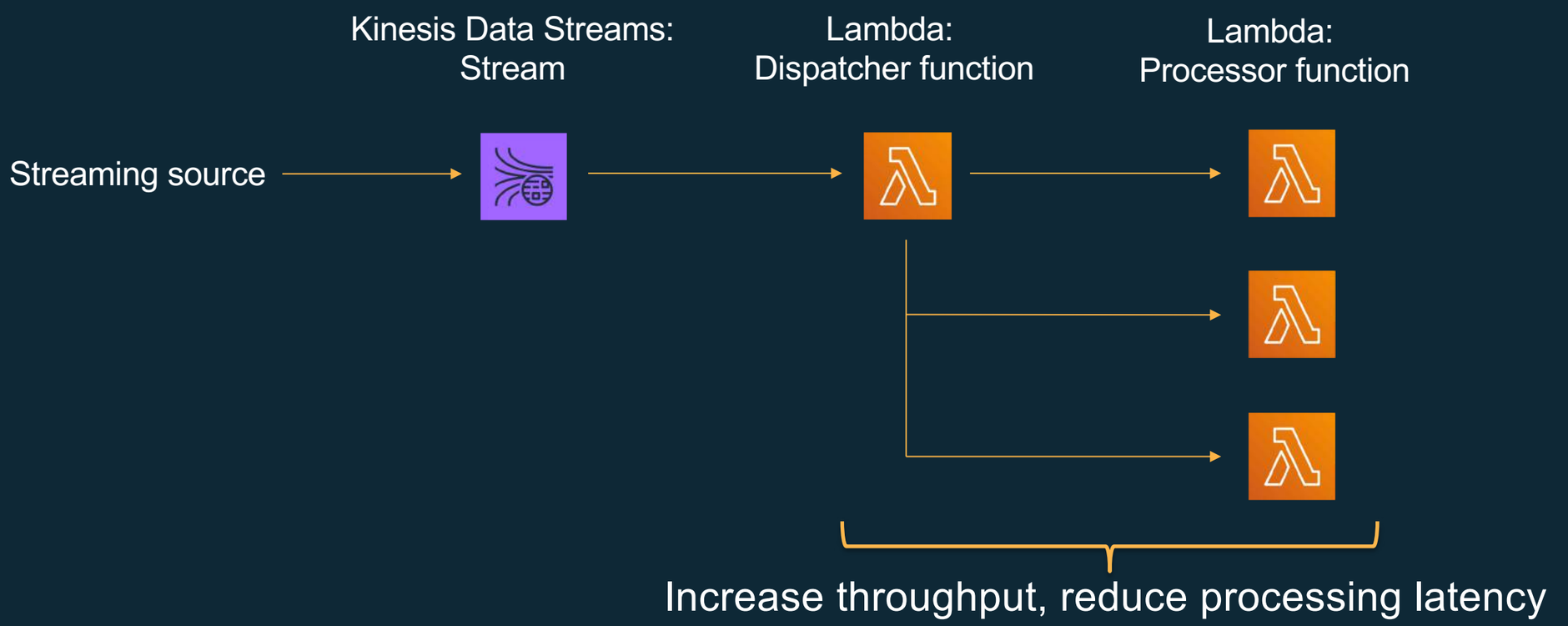


# of shards corresponds to concurrent invocations of Lambda function

Batch size sets maximum # of records per invocation (min 1, max 10K)

Fan-out pattern

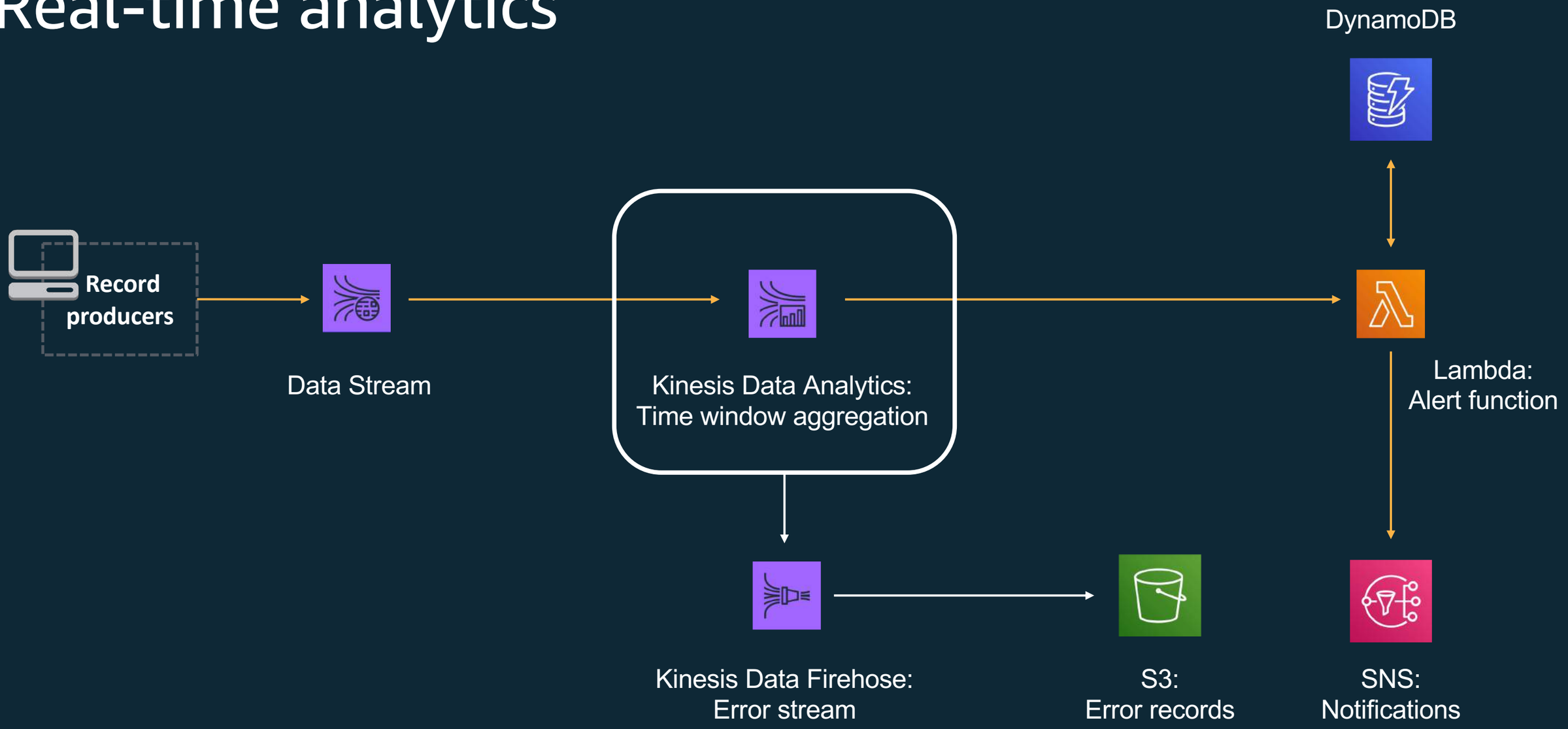
Trade strict message ordering for higher throughput & lower latency



github.com/aws-samples/aws-lambda-fanout

aws

Real-time analytics



Kinesis Data Analytics


```
...
SELECT STREAM
    "device_id",
    STEP(stream.ROWTIME BY INTERVAL '10' MINUTE),
    SUM("measurement") as "sample_sum",
    COUNT(*) AS "sample_count"
FROM "SOURCE_SQL_STREAM_001" as stream
GROUP BY
"device_id"
...
```

Serverless data analysis – meet data lakes!

What’s a data lake?

Collect, store, process, consume, and analyze organizational data

Structured, semi-structured, and unstructured data

Decoupled compute and storage

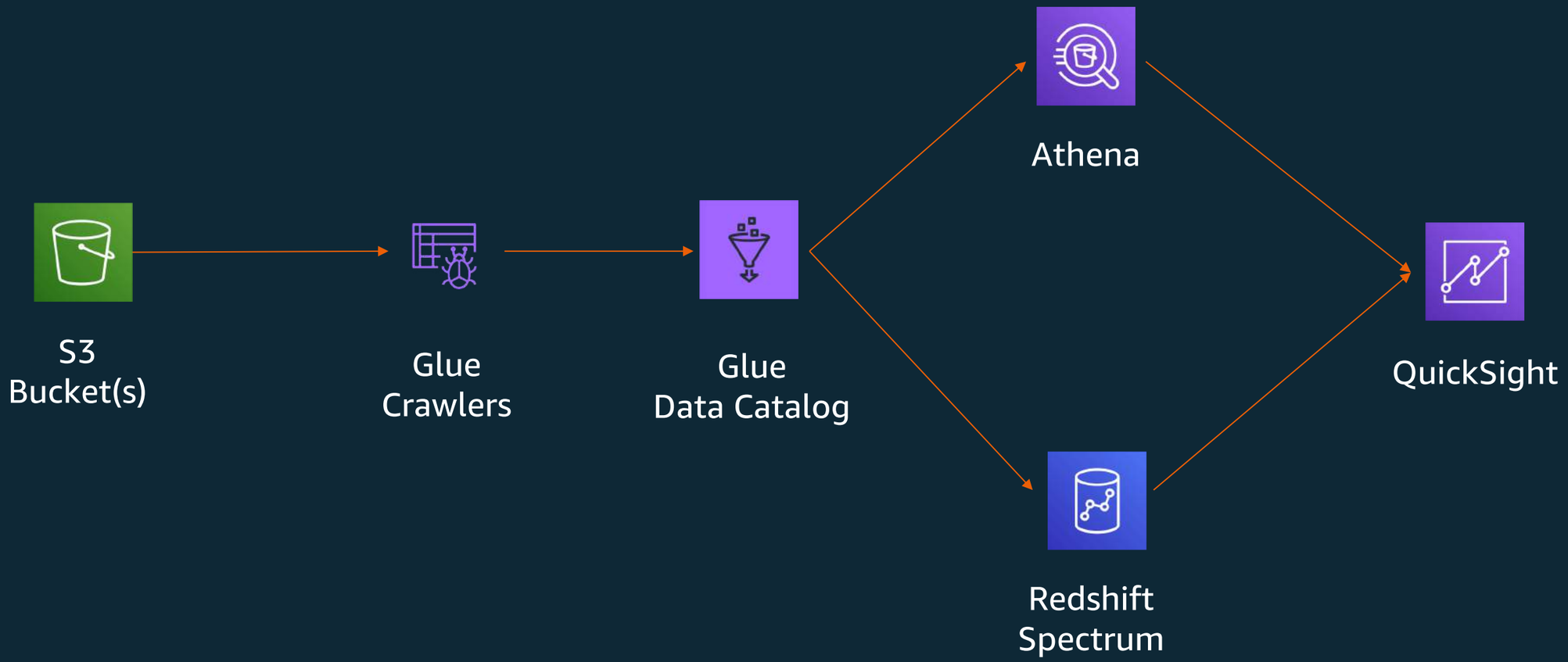
Fast automated ingestion

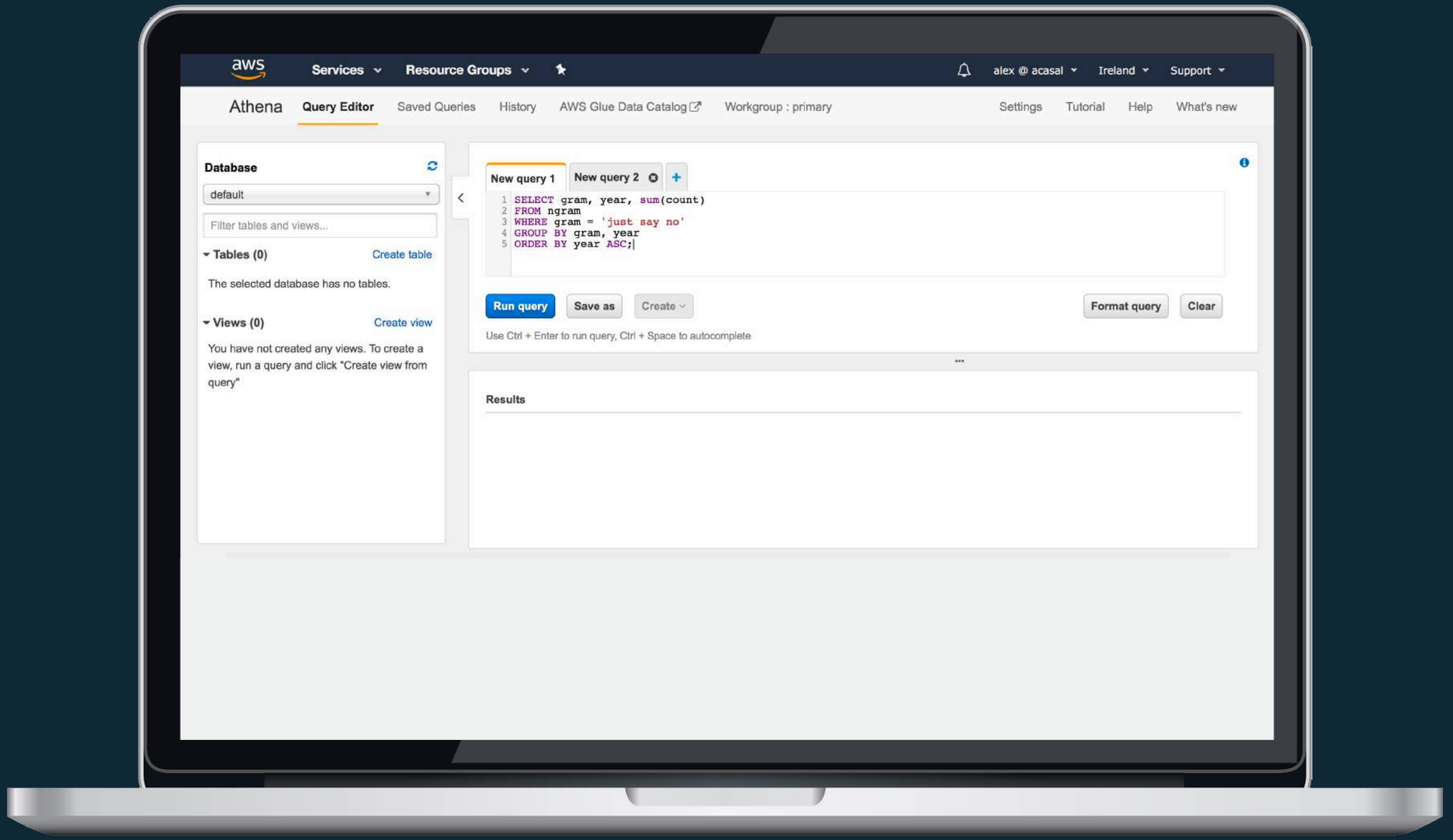
Schema on-read

Complementary to data warehouses

Serverless data lake

How to query all that data on S3?




aws
Services
Resource Groups
alex @ acasal
Ireland
Support
Athena
Query Editor
Saved Queries
History
AWS Glue Data Catalog
Workgroup : primary
Settings
Tutorial
Help
What's new
Database
default
Filter tables and views...
Tables (0)
Create table
The selected database has no tables.
Views (0)
Create view
You have not created any views. To create a view, run a query and click "Create view from query"
New query 1
New query 2
SELECT gram, year, sum(count)
FROM ngram
WHERE gram = 'just say no'
GROUP BY gram, year
ORDER BY year ASC;
Run query
Save as
Create
Format query
Clear
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete
Results

Athena – Just SQL (managed presto)

```
SELECT gram, year, sum(count)
FROM ngram
WHERE gram = 'just say no'
GROUP BY gram, year
ORDER BY year ASC;
```

Query duration: 44.66 seconds

Data scanned: 169.53GB

Cost: $0.85

($5/TB or $0.005/GB)

registry.opendata.aws/google-ngrams

Athena best practices

Partition data

s3://my-bucket/my-data/parquet/year=2018/month=11/day=25/

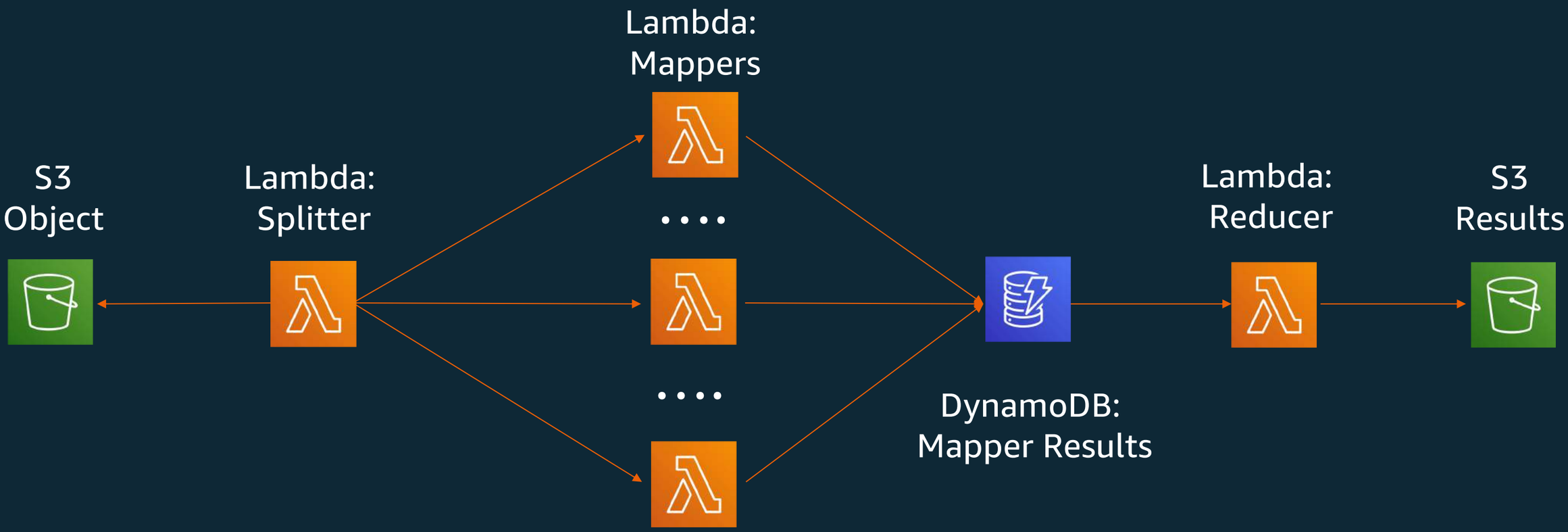
Use columnar formats – Apache Parquet, AVRO, ORC

Compress files with splittable compression (bzip2)

Optimize file sizes

aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena

Serverless MapReduce



Pywren - http://pywren.io

Python library developed by UCI (University of California, Berkeley)

Up to **40 TFLOPS** of peak compute power

Over **700 GB/sec** of read and **500 GB/sec** of write performance using S3

“numpywren: Serverless Linear Algebra”
https://arxiv.org/pdf/1810.09679.pdf

Here to help you build

Go Build!

Grazie!

Alex Casalboni

Developer Advocate, AWS

 @alex_casalboni