Serverless Architecture Patterns And Best Practices

Arun Gupta
Principal Technologist
argu@amazon.com
@arungupta
arun-gupta

Adrian Hornsby
Cloud Architecture Evangelist
adhorn@amazon.com
@adhorn
adhorn

Agenda

1. Serverless Key Concepts
2. Lambda Basics
3. Lambda Best Practices
4. Serverless Application Model
5. CI/CD using CodeStar
6. Monitoring
7. Event Processing
8. Real-time Streaming

aws

Serverless Key Concepts

Serverless means…

No servers to provision or manage

Scales with usage

Never pay for idle

Availability and fault tolerance built in

Spectrum of AWS offerings

"On EC2"

Amazon EC2

Microsoft SQL Server

docker

Managed

Amazon EMR

Amazon ES

Amazon ElastiCache

Amazon Redshift

Amazon RDS

Serverless

AWS Lambda

Amazon Cognito

Amazon Kinesis

Amazon S3

Amazon DynamoDB

Amazon SQS

Amazon API Gateway

Amazon CloudWatch

AWS IoT

aws

Lambda Basics

Using AWS Lambda

Bring your own code

- Node.js, Java, Python, C#, Go
- Bring your own libraries (even native ones)

Simple resource model

- Select power rating from 128 MB to 3 GB
- CPU and network allocated proportionately

Flexible use

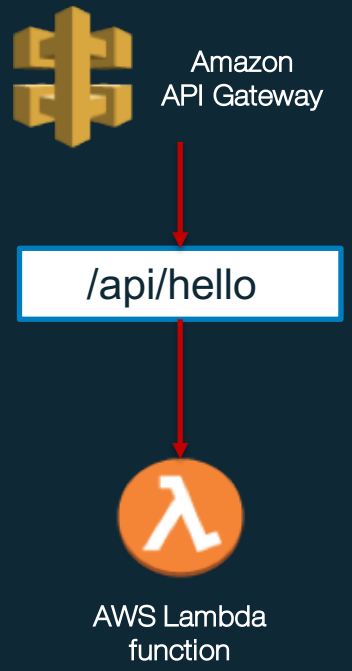
- Synchronous or asynchronous
- Integrated with other AWS services

Flexible authorization

- Securely grant access to resources and VPCs
- Fine-grained control for invoking your functions

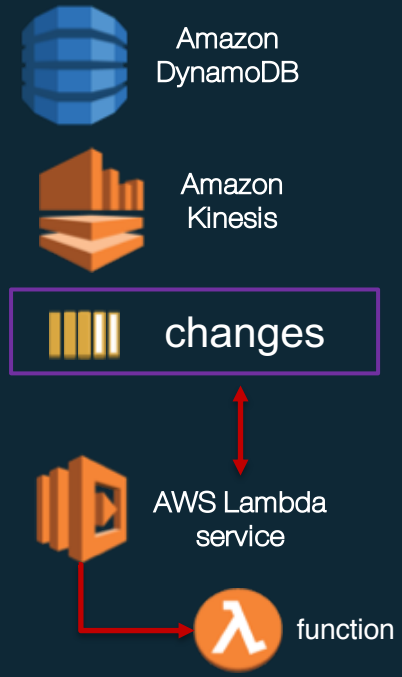
Lambda execution model

Synchronous (push)



Asynchronous (event)

Stream-based



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, etc.)

```
public String handleRequest(Book book, Context context) {
    saveBook(book);

    return book.getName() + " saved!";
}
```

Lambda Best Practices

Lambda Best Practices

- Separate the Lambda handler from core logic

```
public class BookPostHandler implements RequestHandler<Book, String> {

    static DynamoDBMapper mapper = DDBUtil.getMapper();

    public String handleRequest(Book book, Context context) {
        System.out.println("Adding book: " + book);
        saveBook(book);

        return book.getName() + " saved!";
    }

    private void saveBook(Book book) {
        mapper.save(book);
    }
}
```

Lambda Best Practices

- Minimize package size to necessities

```
<dependencies>
   <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-lambda-java-core -->
   <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-lambda-java-core</artifactId>
      <version>1.1.0</version>
   </dependency>

   <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-dynamodb -->
   <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-dynamodb</artifactId>
      <version>1.11.127</version>
   </dependency>
</dependencies>
```

Lambda Best Practices

- Use Environment Variables to modify operational behavior

```
String region = System.getenv("AWS_REGION");
. . .
String bucket = System.getenv("S3_BUCKET");
```

Lambda Best Practices

- Self-contain dependencies in your function package

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-shade-plugin</artifactId>
   <version>3.1.0</version>
   <executions>
      <execution>
         <phase>package</phase>
         <goals>
            <goal>shade</goal>
         </goals>
      </execution>
   </executions>
</plugin>
```

Lambda Best Practices

- Leverage "Max Memory Used" to right-size your functions
  - Calculate 1000x all prime numbers < 1m

Memory	Compute time	Cost
128 MB	11.722965 secs	$0.024628
256 MB	6.678945 secs	$0.028035
512 MB	3.194954 secs	$0.026830
1024 MB	1.465984sec	$0.024638

https://github.com/jconning/lambda-cpu-cost

aws

Lambda Best Practices

- Delete large unused functions (75GB limit per region)

AWS Lambda Resource Limits per Invocation

Resource	Limits
Memory allocation range	Minimum = 128 MB / Maximum = 3008 MB (with 64 MB increments). is exceeded, function invocation will be terminated.
Ephemeral disk capacity ("/tmp" space)	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds
Invoke request body payload size (RequestResponse/synchronous invocation)	6 MB
Invoke request body payload size (Event/asynchronous invocation)	128 K

Meet SAM!

Serverless Application Model

CloudFormation extension optimized for serverless

New serverless resource types: functions, APIs, and tables

Supports anything CloudFormation supports

Open specification (Apache 2.0)

https://github.com/awslabs/serverless-application-model

aws

SAM Template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Simple CRUD webservice.
Resources:
  GetFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: org.sample.aws.samlocal.BookGetHandler
      Runtime: java8
      CodeUri: ./target/sam-local-java-1.0-SNAPSHOT.jar
      Policies: AmazonDynamoDBReadOnlyAccess
      Timeout: 30
      Environment:
        Variables:
          TABLE_NAME: !Ref Table
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /books
            Method: get

  Table:
    Type: AWS::Serverless::SimpleTable
```

SAM Commands

Package

- Creates a deployment package (.zip file)
- Uploads deployment package to an S3 bucket
- Adds a `CodeUri` property with S3 URI

Deploy

- Creates CloudFormation resources

SAM Local

- CLI for local testing of Serverless apps
- Works with Lambda functions and “proxy style” APIs
- Response object and function logs available on your local machine
- Currently supports Java, Node.js and Python
- Accepting PRs

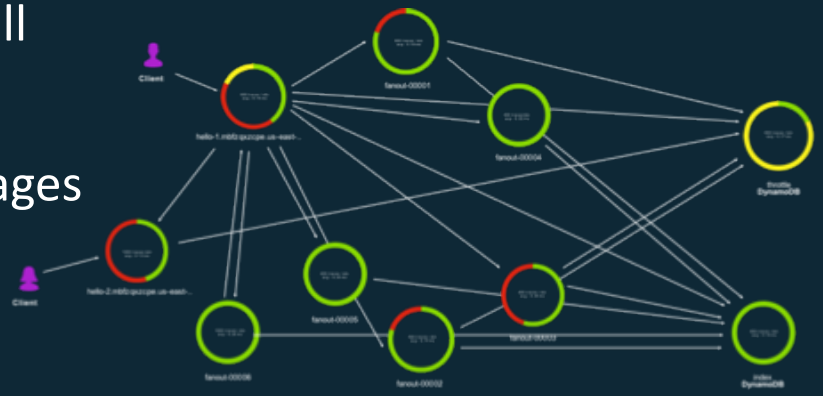
https://github.com/awslabs/aws-sam-local

CI/CD using AWS CodeStar, AWS CodeBuild and AWS CodePipeline

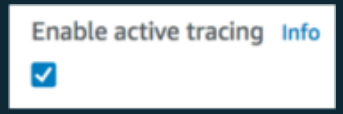
Monitoring

AWS X-Ray Integration with Serverless

- Lambda instruments incoming requests for all supported languages
- Lambda runs the X-Ray daemon on all languages with an SDK



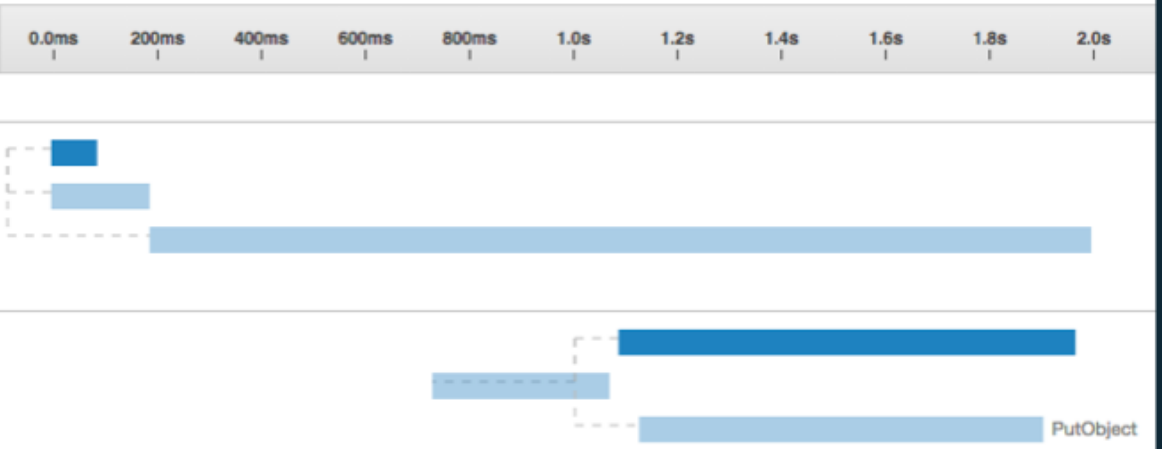

```
var AWSXRay = require('aws-xray-sdk-core');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));
S3Client = AWS.S3();
```



X-Ray Trace 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
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	!



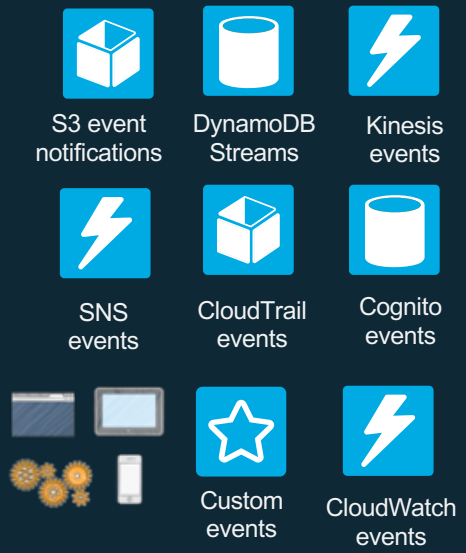
Event Processing

Event driven

Event A on B triggers C

Event-driven platform

Invoked in response to events
- **Changes in data**
- **Changes in state**



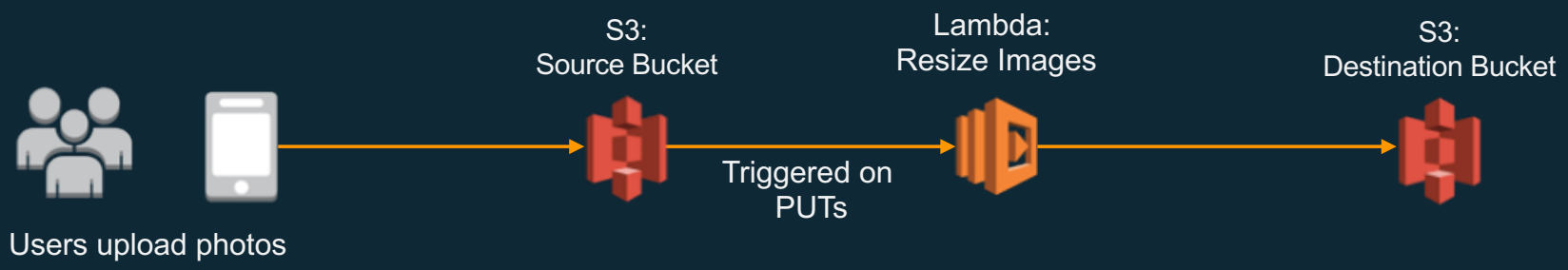
Lambda functions

Access any service, including your own

Any custom

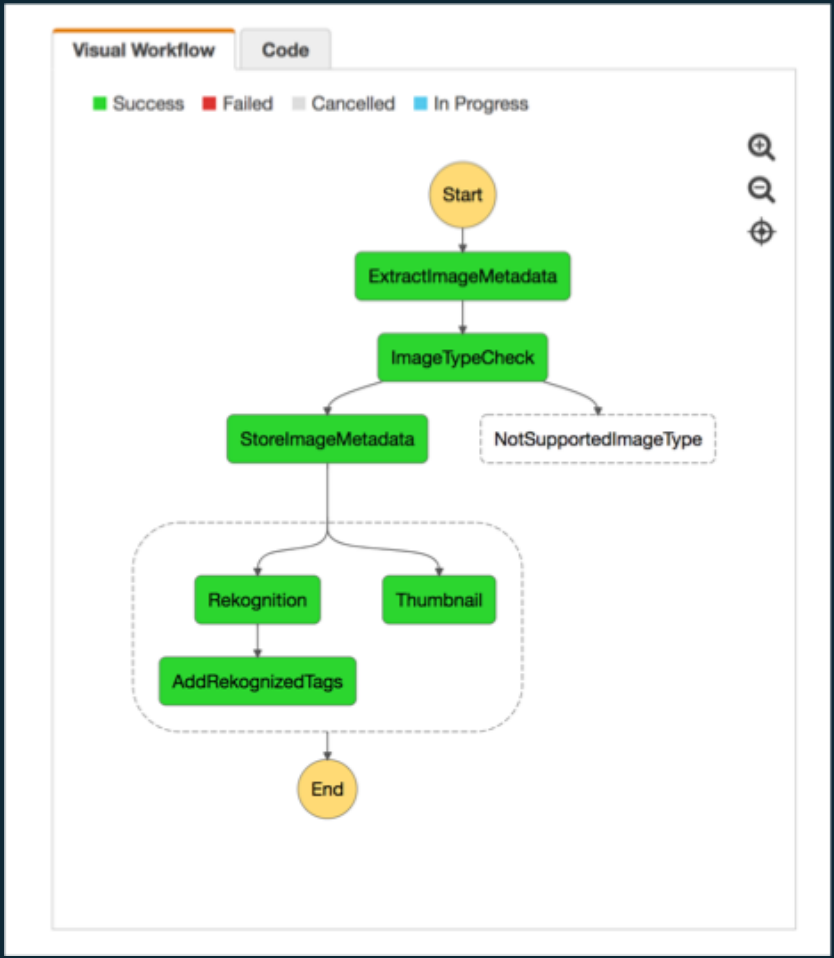
Such as…

Event-driven actions



AWS Step Functions:

Orchestrate a Serverless processing workflow using AWS Lambda



Real-time Streaming

supercell.com/en/games/clashofclans/

SUPERCELL

OUR STORY CAREERS GAMES SUPPORT CONTACT US SUPERCELL SHOP

Supercell Case Study

SUP
ERC
ELL

> The world of gaming never sleeps ... We owe every player a great experience, and AWS is our platform to make that happen.
>
> Sami Yliharju
>
> Services Lead

About Supercell

Finland-based Supercell, founded in 2010 by six game-industry veterans, is one of the fastest-growing social game companies in the world. With more than 100 employees, its three games are massively successful, attracting tens of millions of players on iOS and Android devices every day. These games are Hay Day, a social farming game, and Clash of Clans and Boom Beach, which combine social resource management and strategic combat elements.

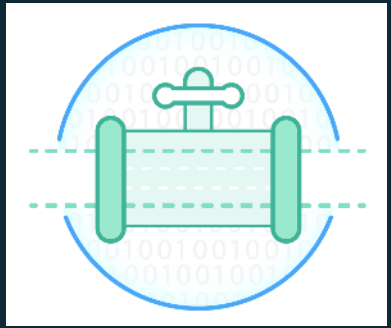
The Challenge

"When Supercell launched," says Sami Yliharju, services lead at Supercell, "the founders wanted to create a company where the focus would be on the best people making the best games." Designing specifically for mobile lets the developers concentrate on creating the best experience for gamers—and working in small development teams helps, too. Each game team is unique, but usually includes a lead, a game designer, a game tester, an artist, a server engineer, and a game programmer. Supporting technical teams are of similar size, but have a different structure depending on their responsibilities.

https://aws.amazon.com/solutions/case-studies/supercell/

aws

Amazon Kinesis makes it easy to work with real-time streaming data

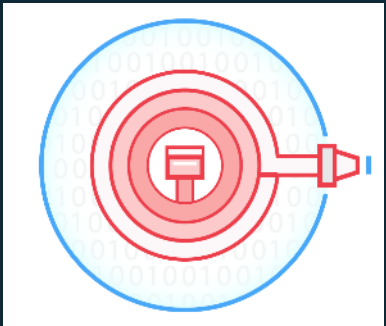


Amazon Kinesis Streams

- For Technical Developers
- Collect and stream data for ordered, replay-able, real-time processing

Amazon Kinesis Analytics

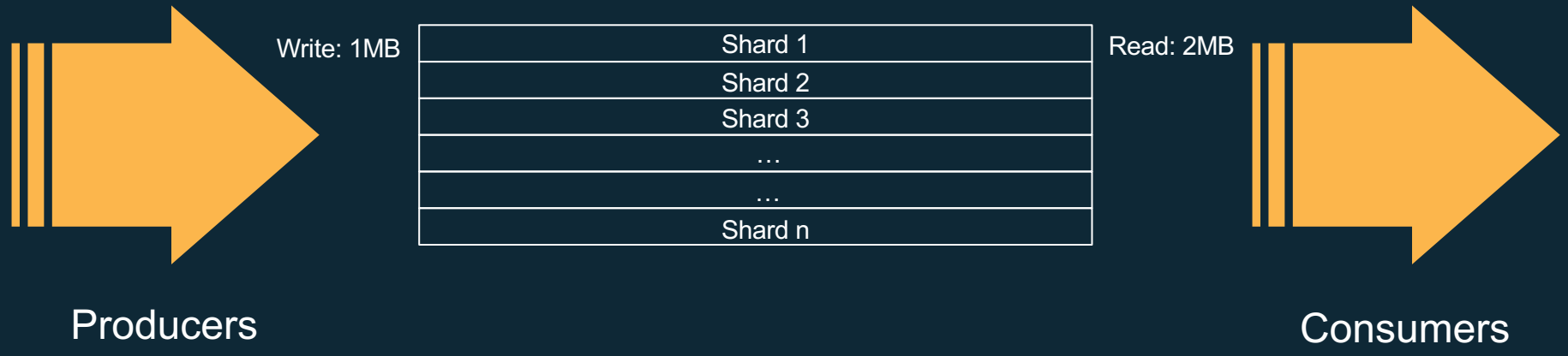
- For all developers, data scientists
- Easily analyze data streams using standard SQL queries



Amazon Kinesis Firehose

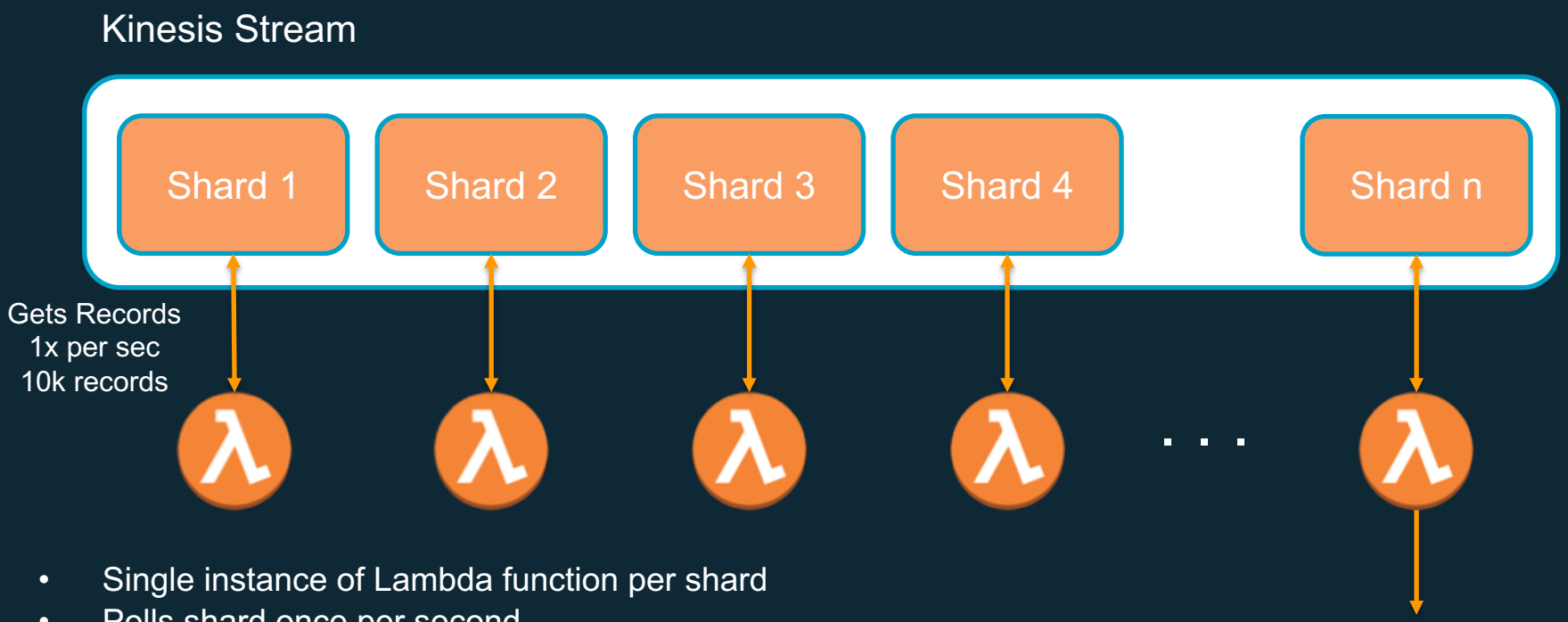
- For all developers, data scientists
- Easily load massive volumes of streaming data into Amazon S3, Redshift, ElasticSearch

Amazon Kinesis



** A *shard* is a group of data records in a stream

Processing a Kinesis Streams with AWS Lambda



- Single instance of Lambda function per shard
- Polls shard once per second
- Lambda function instances created and removed automatically as stream is scaled

Kinesis Analytics

Use SQL to build real-time applications

Connect to streaming source

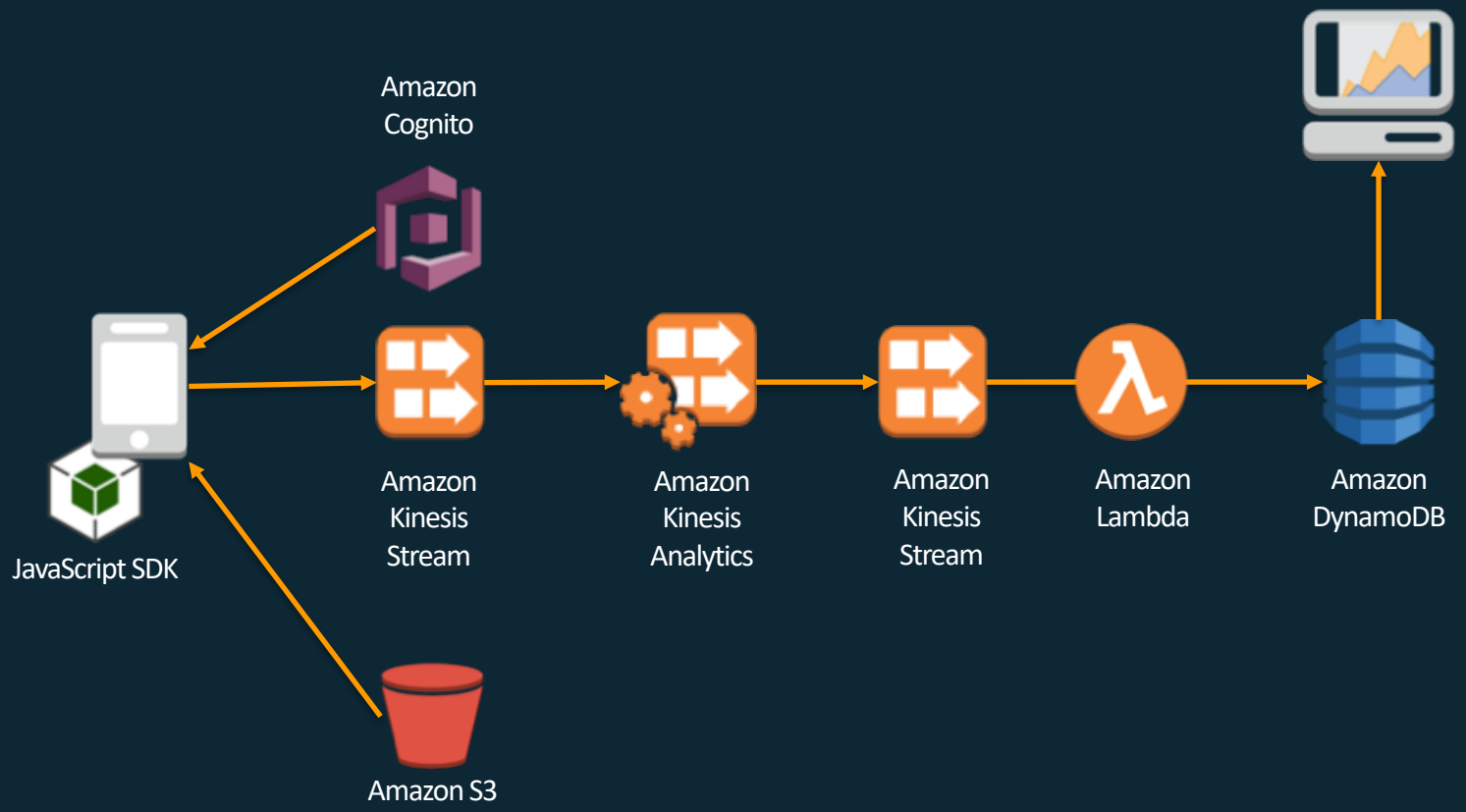
Easily write SQL code to process streaming data

Continuously deliver SQL results

Real-time Analytics Demo

http://quad.adhorn.me

Real-time analytics



Further Reading

Serverless Architectures with AWS Lambda
https://d1.awsstatic.com/whitepapers/serverless-architectures-with-aws-lambda.pdf

Optimizing Enterprise Economics with Serverless Architectures
https://d0.awsstatic.com/whitepapers/optimizing-enterprise-economics-serverless-architectures.pdf

Serverless Applications Lens - AWS Well-Architected Framework
https://d1.awsstatic.com/whitepapers/architecture/AWS-Serverless-Applications-Lens.pdf

Streaming Data Solutions on AWS with Amazon Kinesis
https://d1.awsstatic.com/whitepapers/whitepaper-streaming-data-solutions-on-aws-with-amazon-kinesis.pdf

AWS Serverless Multi-Tier Architectures
https://d1.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Archiectures.pdf

More info:

https://aws.amazon.com/serverless/

Thank you!