Journey of mastering AWS Lambda Java Development

INFINUM

Ivan Miklec & Matej Nedic

neverstop

MATEJ NEDIC

- Lead java Engineer at Infinum
- Spring Cloud AWS Core team Member
- working with

spring

by Pivotal

aws

IVAN MIKLEC

- Java Engineer at Infinum
- working with

spring by Pivotal

aws

AWS Lambda overview

- Event-driven, function as a service
- Allows developers to focus on a code
- Runtime, security, scaling and cost efficiency are all handled by the AWS
- Cost effective since it’s pay-per-use
- Zero server management
- Great for Asynchronous data processing, backends, real time data processing

Limitations

- Non-portable code
- Maximum time lambda can run is 15 minutes, default timeout is 3 seconds
- Cold starts
- Working out the cost can be tricky, especially in a more complex architecture

Building Lambdas with Java

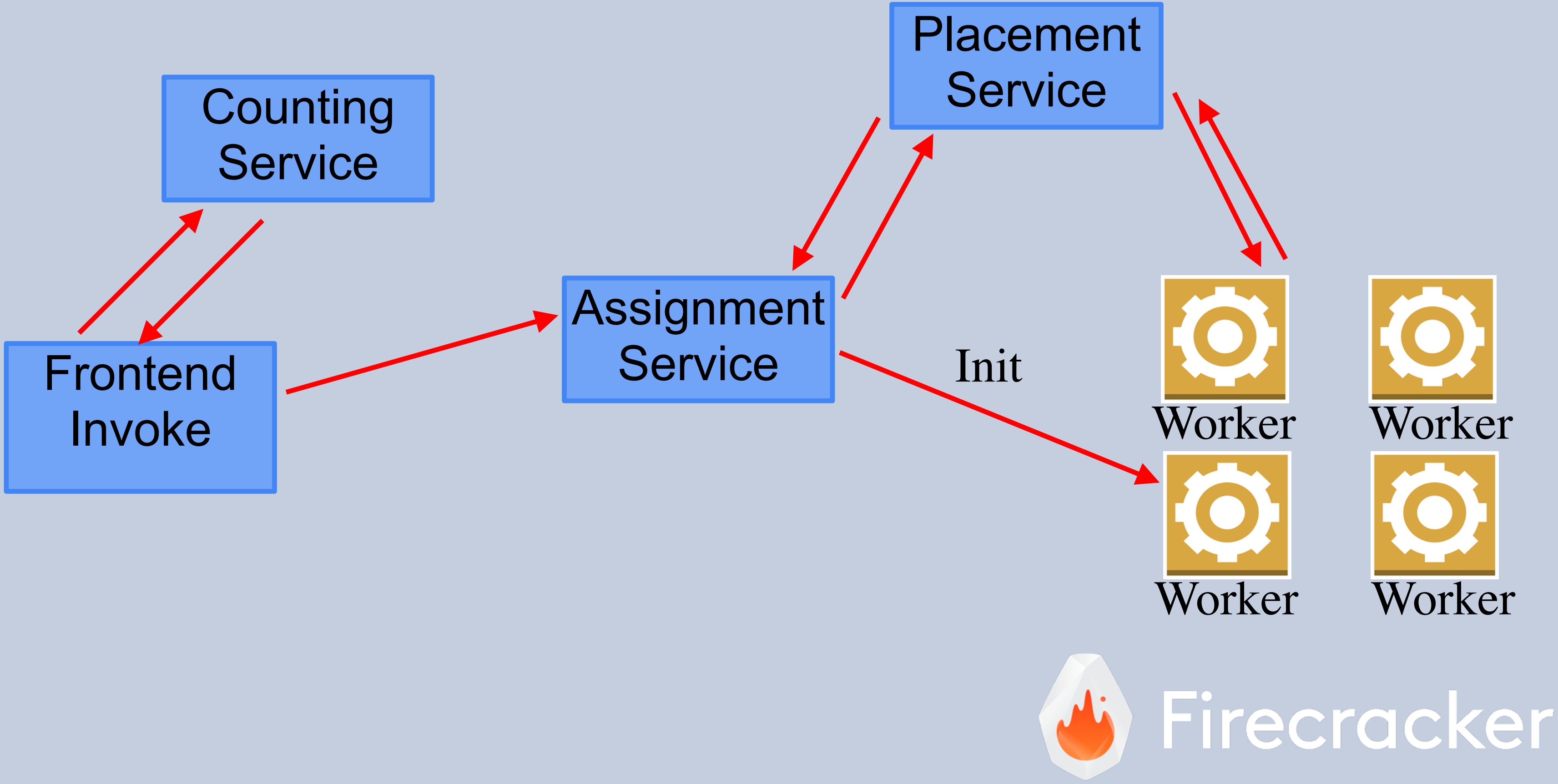
- Java 8, 11, 17 and 21 supported
- RequestHandler and RequestStreamHandler interfaces
- Use layers to package dependencies reusable across multiple functions
- JIT and large file archives result in a long cold start time

Snapstart

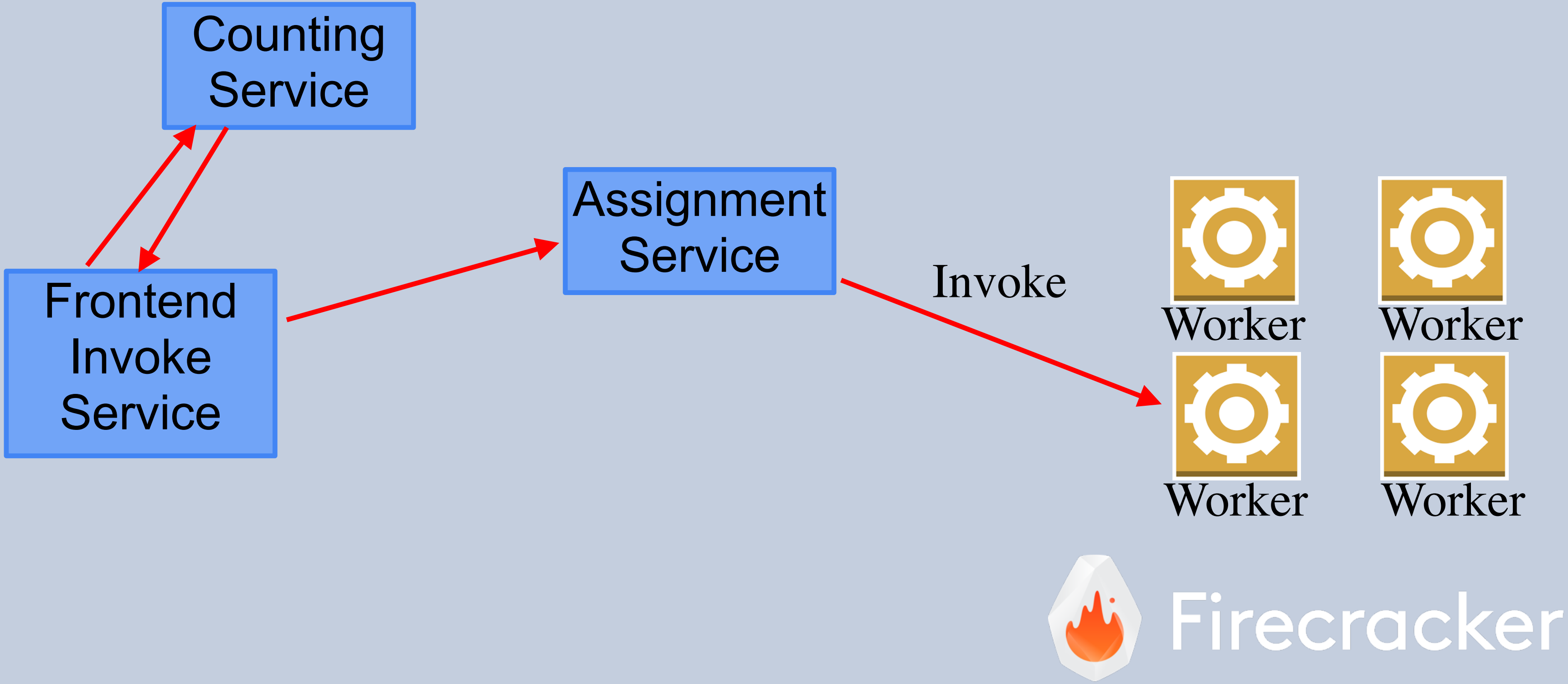
- Little to no code changes needed
- Firecracker memory snapshot is taken on a Function publish
- Ideal for backends used in user flows
- Latency-sensitive data processing
- Avoid saving state during initialisation
- Runtime hooks
- Cheaper alternative to provisioned concurrency

Why should you know how does it work under the hood?

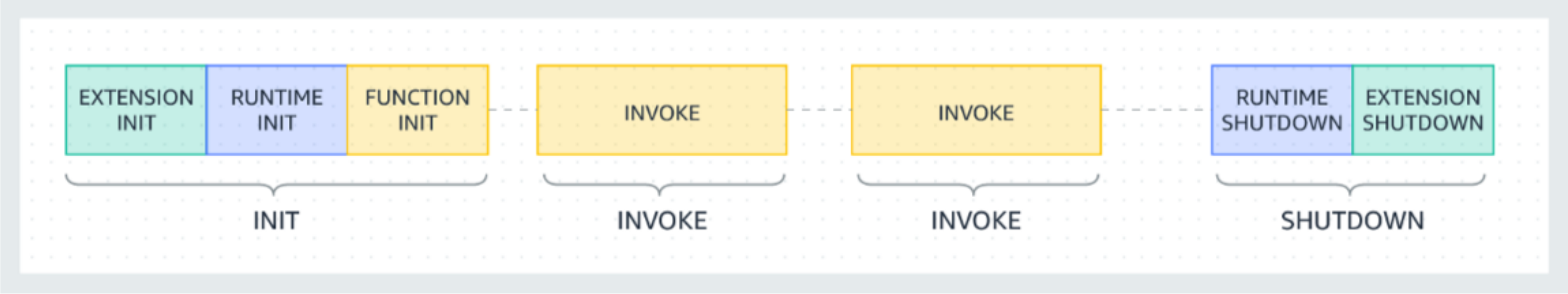
Cold start


Counting Service
Frontend Invoke
Assignment Service
Placement Service
Init
Worker
Worker
Worker
Worker
Firecracker

Warm start

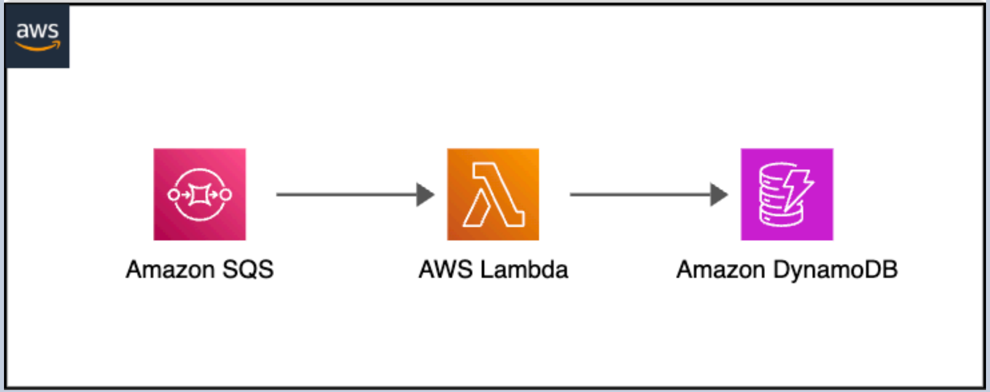

Counting Service
Frontend Invoke Service
Assignment Service
Invoke
Worker
Worker
Worker
Worker
Firecracker

How does this look on diagram



Demo

- Simple Lambda function written in plain java
- Uses SQS and DynamoDB
- CDK infrastructure



Ci/Cd

Git repo

1. Mvn package and upload artifact
2. In Infra Module run mvn install
3. Run CDK deploy

S3 Bucket

Branches are used as Source of truth this way and are versioning AWS Lambdas

Performance fine tuning

C1 vs C2 Compiler

- In environment variable of AWS Lambda pass

- JAVA_OPTIONS -XX:+TieredCompilation -XX:TieredStopAtLevel=1

Garbage Collection

- **SerialGC**
- **G1GC**

ARM64 vs x86_64

- In CDK -> .architecture(ARM_64) that’s is

Adjusting memory changes cpu

- AWS Lambda Power Tunning

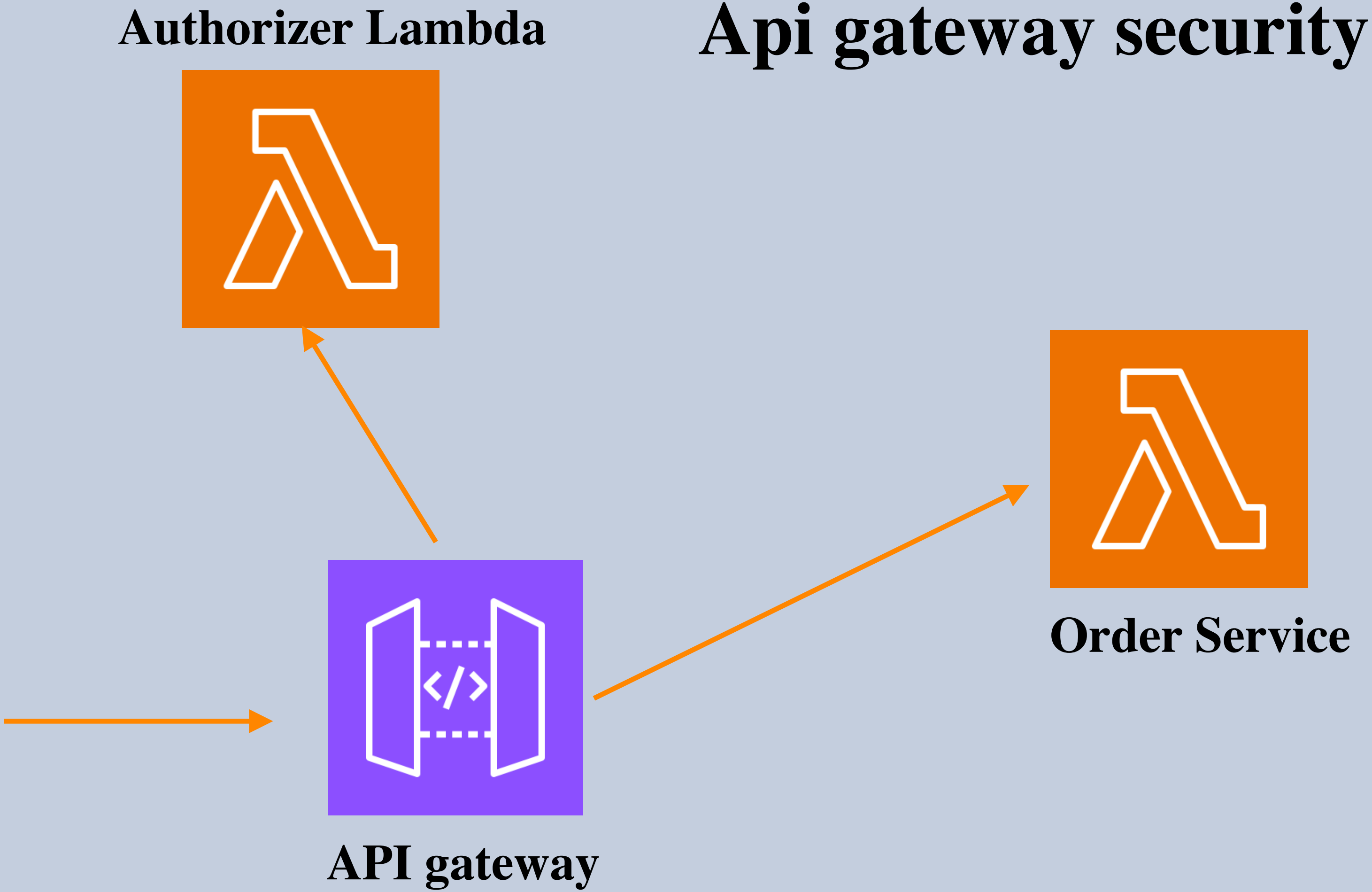
Performance fine tuning

Reduce size of your dependencies

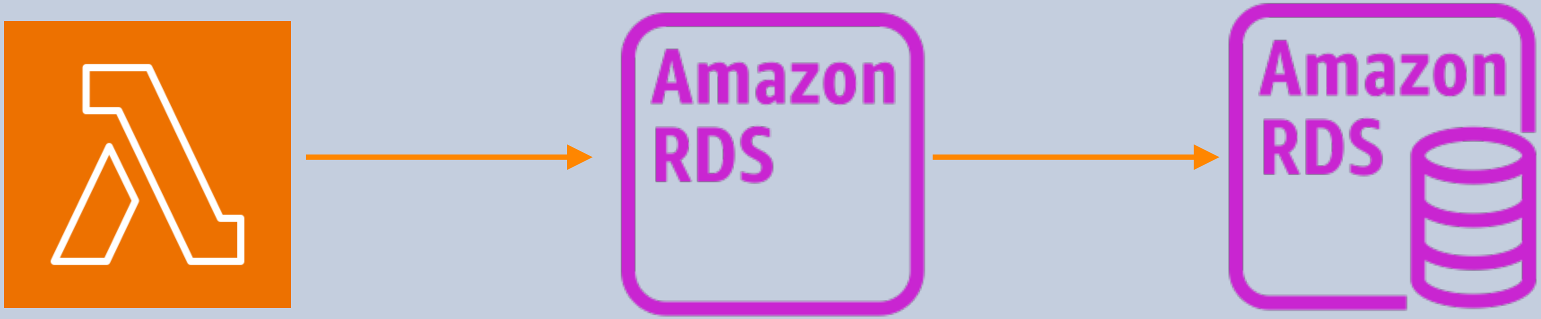
- AWS Lambda Profiler

Use Snapstart

Api gateway security lambda



RDS Proxy

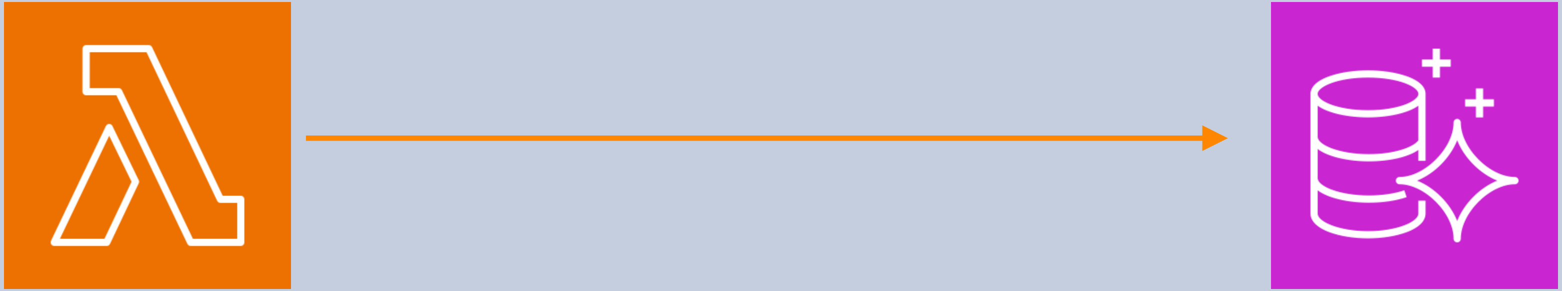


AWS Lambda

RDS Proxy

RDS Instance

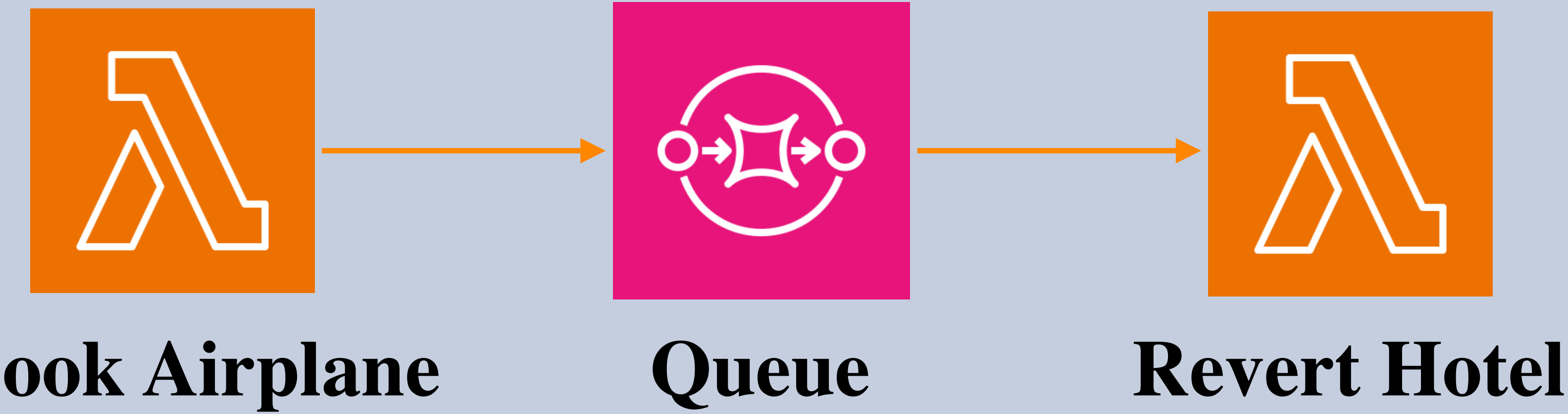
Aurora DSQL



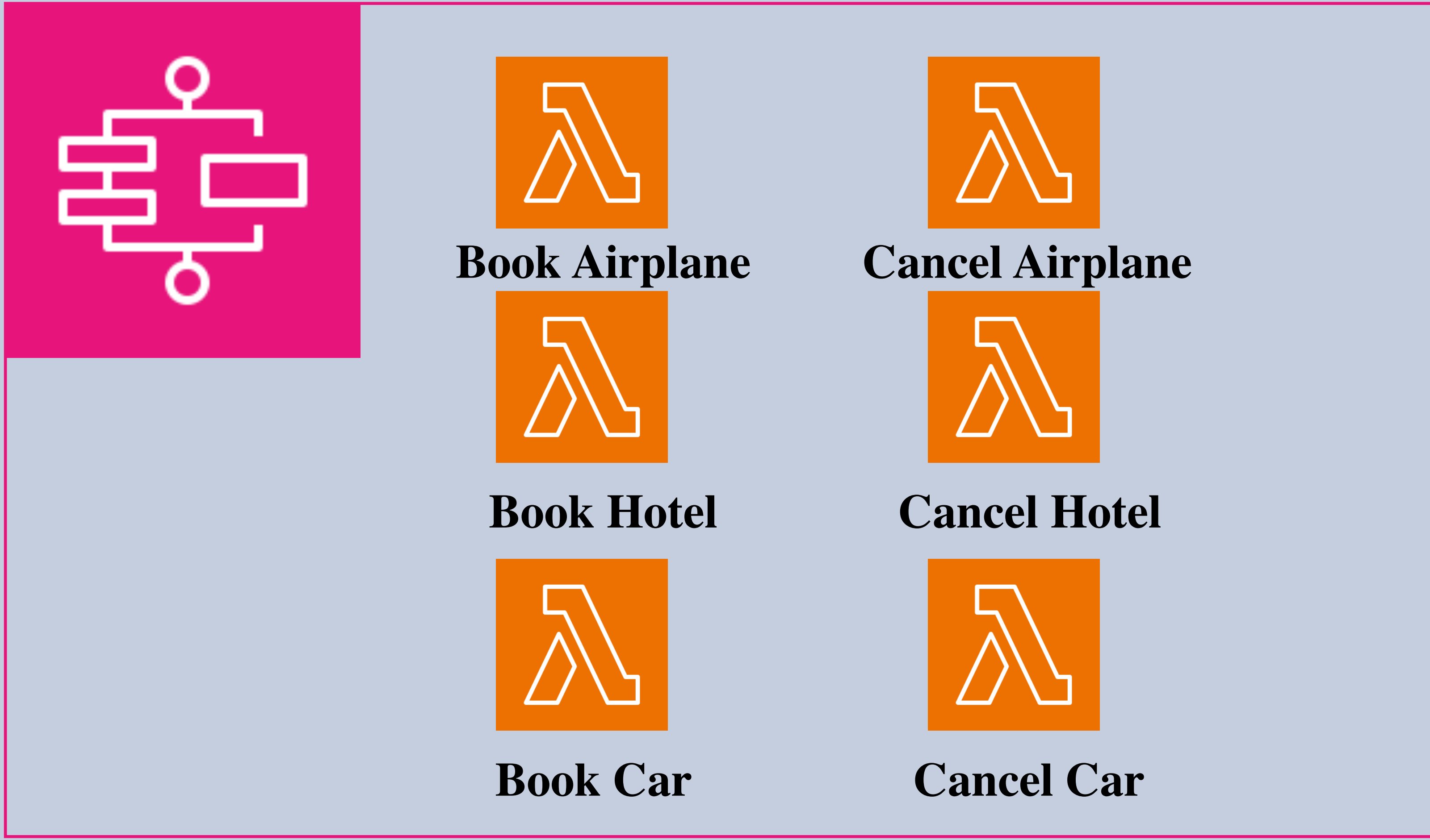
AWS Lambda

Aurora DSQL

Saga pattern



Step function



Book Airplane

Cancel Airplane

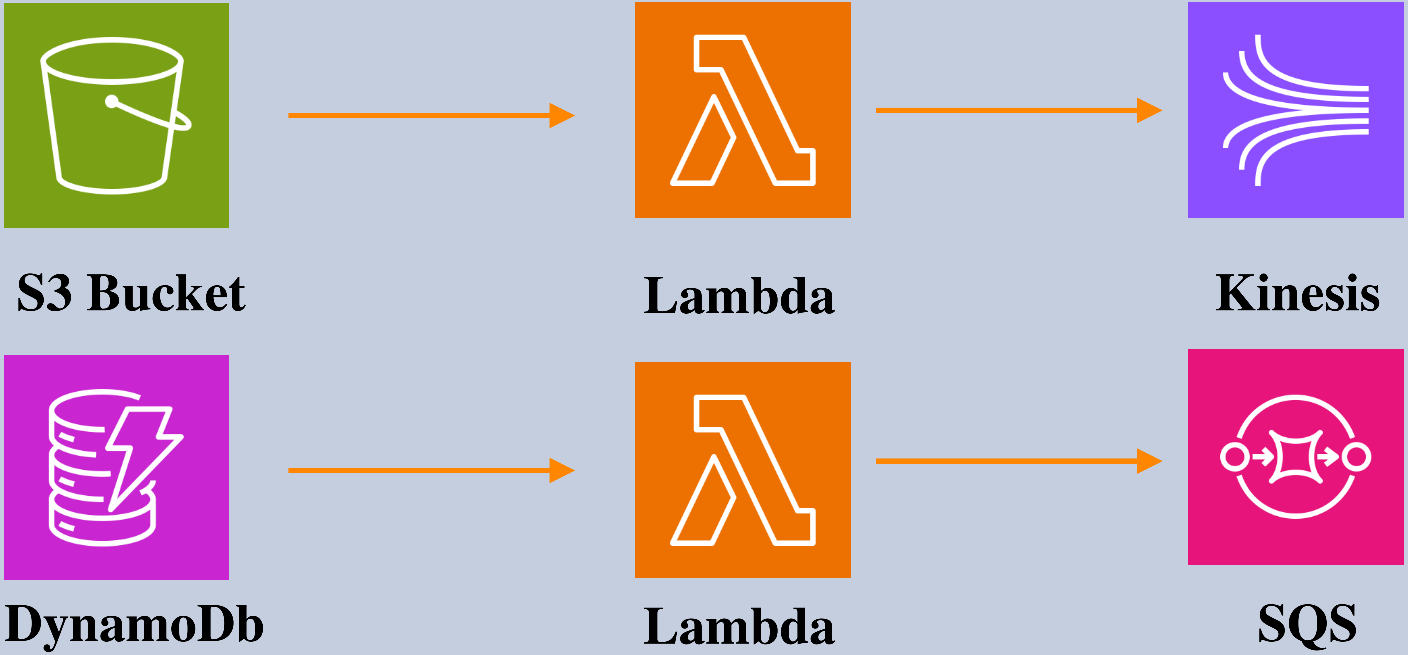
Book Hotel

Cancel Hotel

Book Car

Cancel Car

Reacting on Streams and data processing



Question Time

Thank You!