Spring Cloud Function

SpringOne 2021

September 2, 2021

Oleg Zhurakousky

Mark Sailes

Marc DiPasqualle

Agenda

- Functions and Spring Cloud Function (Oleg)
- AWS Lambda integration (Oleg)
  - Routing Function
- AWS Lambda (Mark)
  - AWS CDK
  - Native images of Spring Cloud Function on AWS
- Streaming with Cloud Function, Cloud Stream & Solace (Marc)
  - The Basics
  - Dynamic Publishing

Java Functions - SPEC

- Simplicity
- Portability
- Extensibility
- Consistency

Supplier

Function

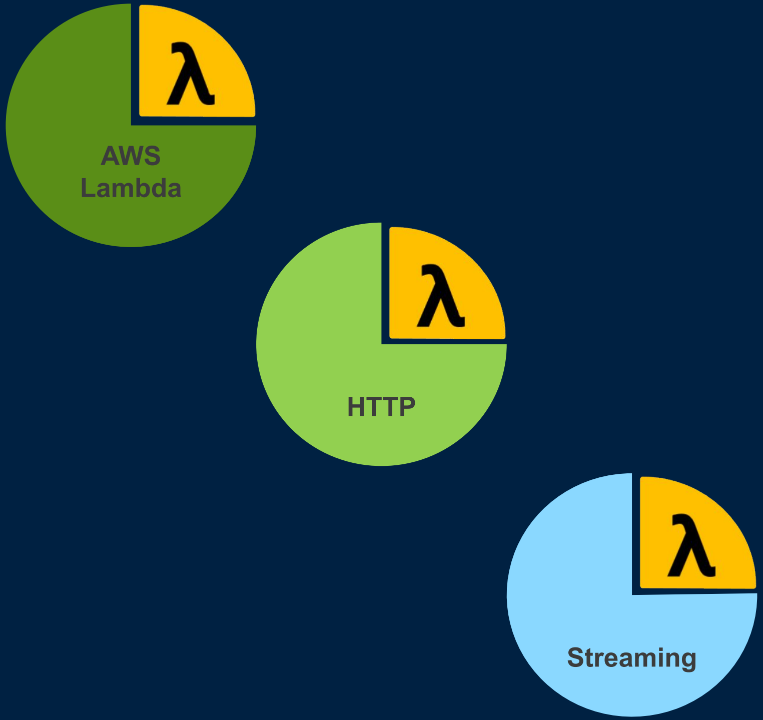
Consumer

Java Functions – Core Tenants

- Contract
- Pattern

Java Functions – Activation/Invocation

- Contract
- Pattern



Spring Cloud Function – goals?

- Promote implementation of business logic via Java Functions
- Uniformed and portable programming model
- Integration with other platforms (i.e., serverless etc.)
    - *AWS Lambda*
    - *Streaming (i.e., Solace, RabbitMQ, Kafka etc.)*
    - *Others. . .*

Spring Cloud Function – core features

- Function Composition (e.g., `func1|func2`)
- Transparent type conversion
- Reactive Support (e.g., `Function<Flux<String>, Flux<Integer>>`)
- POJO Function (*if it looks/smells like a function it's a function*)
- Function Arity (functions with multiple inputs/outputs)

Spring Cloud Function – core features (cont)

- Function Routing
- POJO Function (*if it looks/smells like a function it's a function*)
- Deployment of packaged functions
- Adapters:
    - *Function as an HTTP Endpoint*
    - *Function as an AWS Lambda*
    - *Function as Message handler*
    - *Function as an RSocket listener*
    - *Function as "anything". . .*

Spring Cloud Function – how does it look?

```
@Bean
public Function<String, String> uppercase() {
  return value -> value.toUpperCase();
}

@Bean
public Function<MyPojo, MyOtherPojo> anotherFunction() {
  return pojo -> {
        . . .
        return new MyOtherPojo();
  };
}

@Bean
public Consumer<Message<String>> consumer() {
  return System.out::println;
}
```

DEMO

Streaming with Spring Cloud Function on Solace

- Marc D., your slides begin here

Spring Cloud Function & AWS Lambda

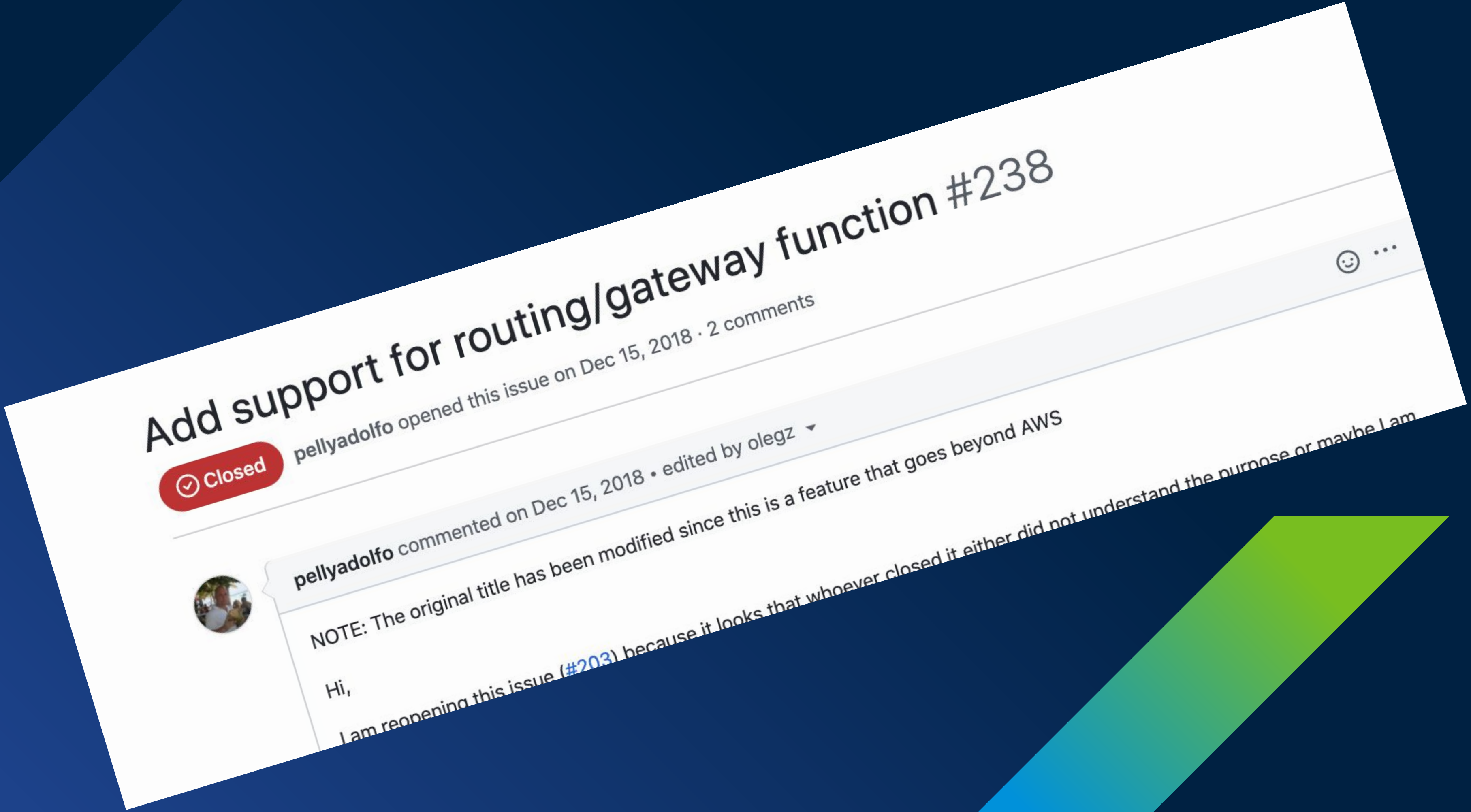
- Goals
  - Ability to extend simple Java Function programming model to AWS Lambda
  - Decouple AWS Lambda specifics from function implementation

    `com.amazonaws.services.lambda.runtime.RequestHandler`

    `com.amazonaws.services.lambda.runtime.RequestStreamHandler`

    `vs.`

    `java.util.function.Function`
  - Integrate with AWS Lambda APIs and tools.
  - Simplify management and maintenance of functions as AWS Lambda functions (e.g., RoutingFunction)



Add support for routing/gateway function #238

Closed pellyadolfo opened this issue on Dec 15, 2018 · 2 comments

pellyadolfo commented on Dec 15, 2018 • edited by olegz

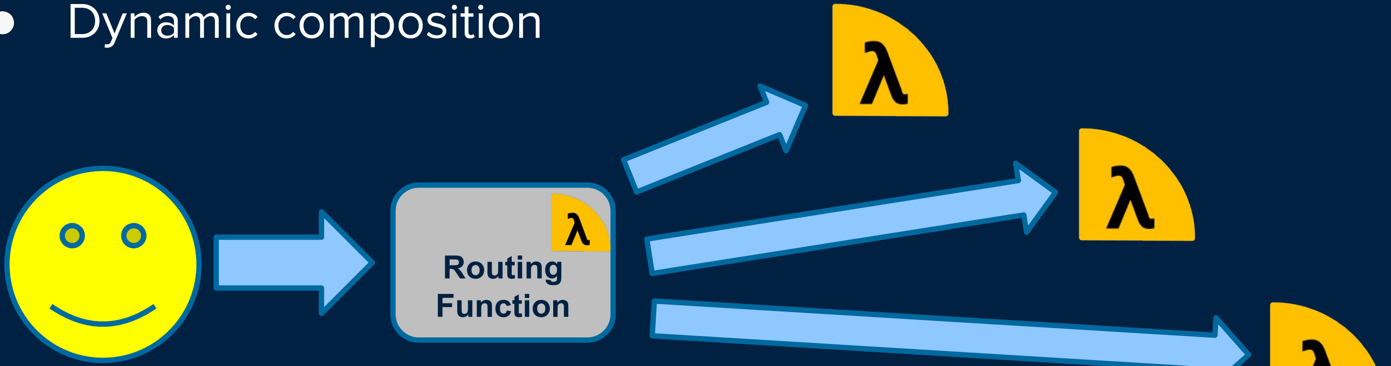
NOTE: The original title has been modified since this is a feature that goes beyond AWS

Hi,

I am reopening this issue (#203) because it looks that whoever closed it either did not understand the purpose or maybe I am

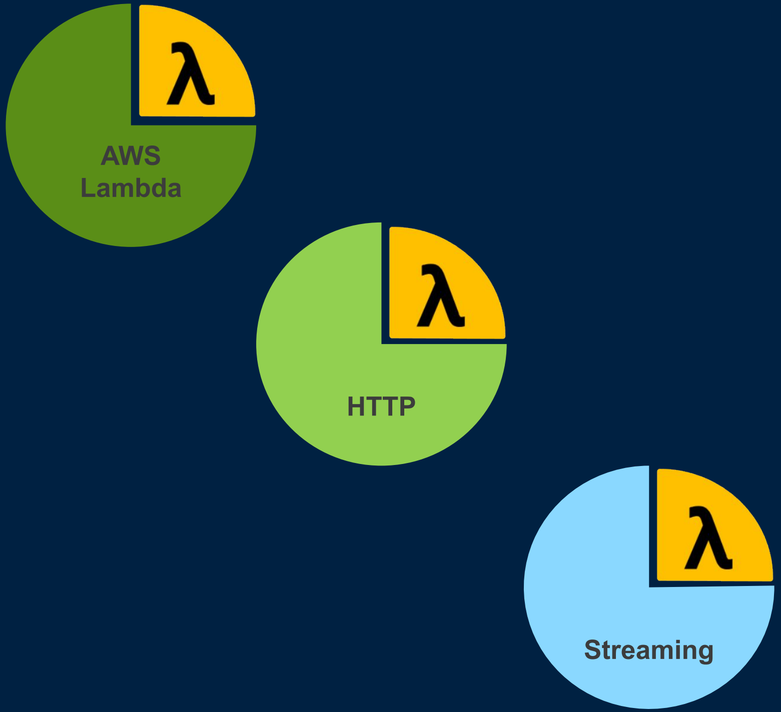
Spring Cloud Function & AWS Lambda - RoutingFunction

- Function that simply routes to other function
- Acts as gateway/firewall
- Single point of maintenance (e.g., single API Gateway)
- Dynamic composition



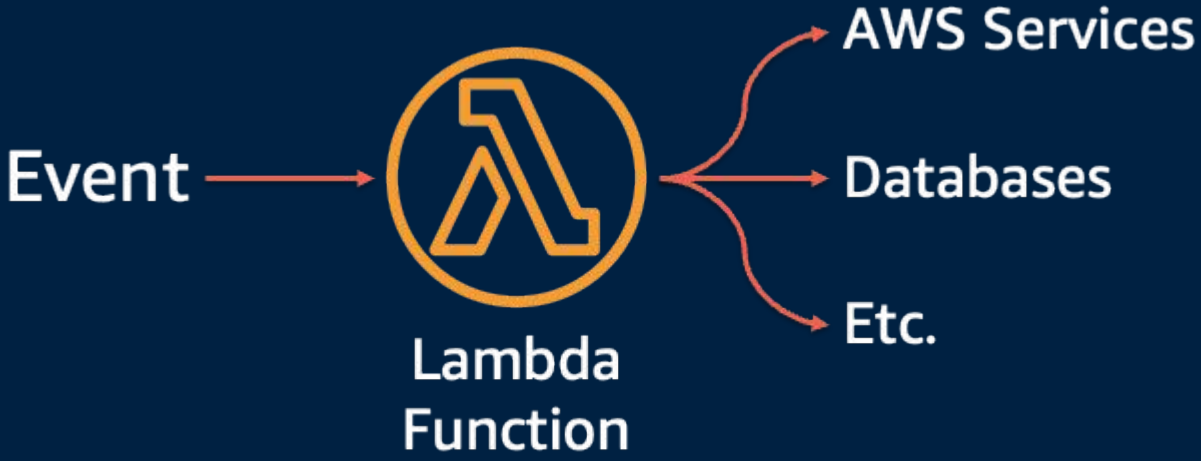
Java Functions – Activation/Invocation

- Contract
- Pattern



AWS Lambda DEMO

The high-level view



Java
Python
Javascript
Golang
C#
BYOL
Container images

What do developers need to drive success?

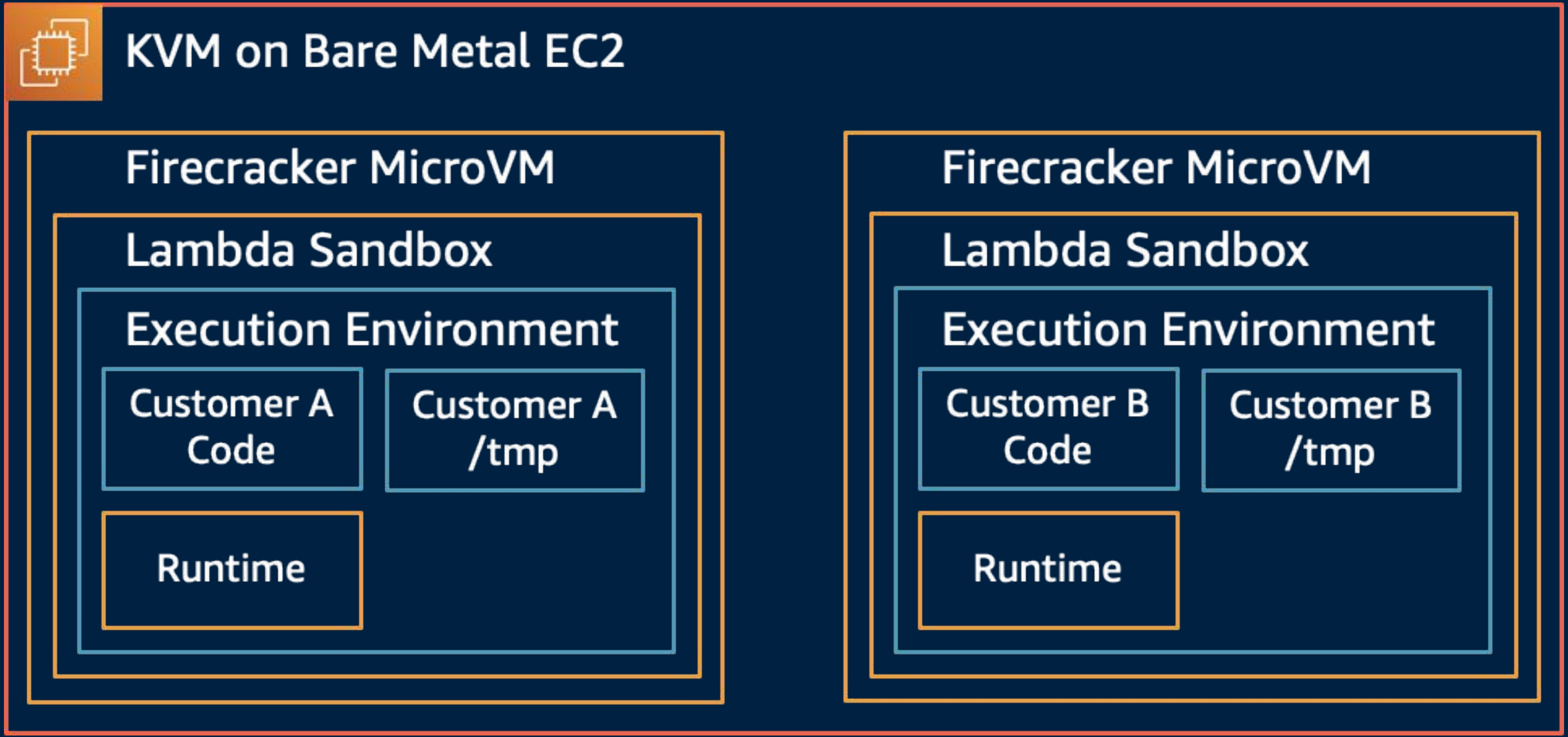
Get to market faster

Lower total cost of ownership

High performance and scalability

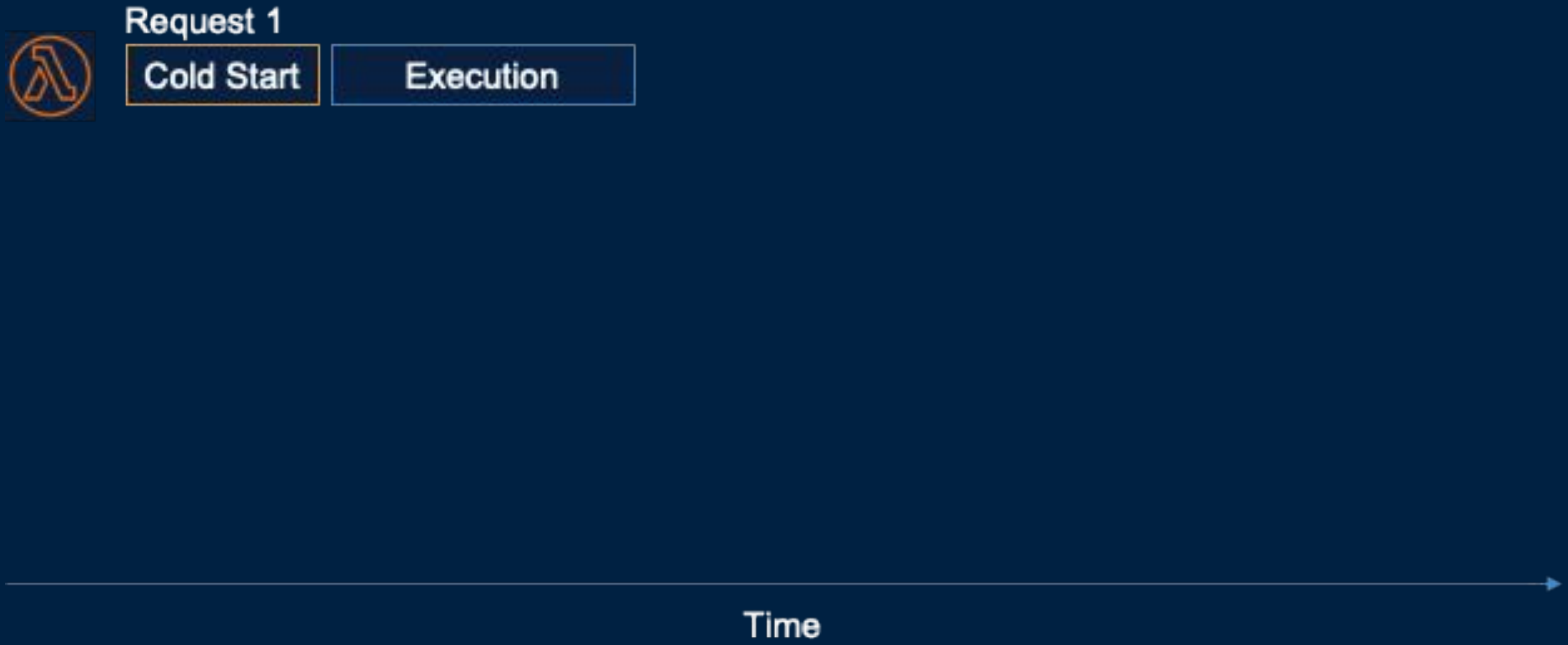
Security and isolation by design

Lambda Security

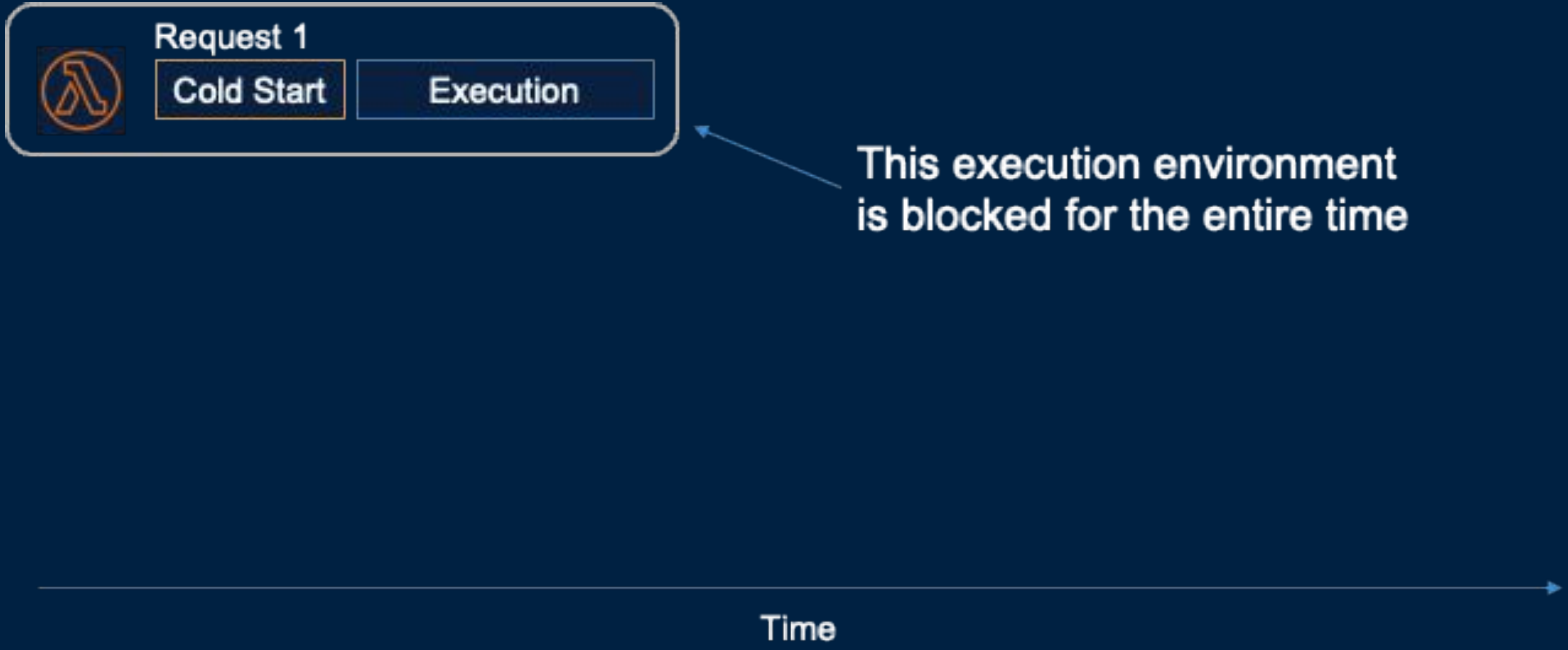


https://docs.aws.amazon.com/whitepapers/latest/security-overview-aws-lambda/

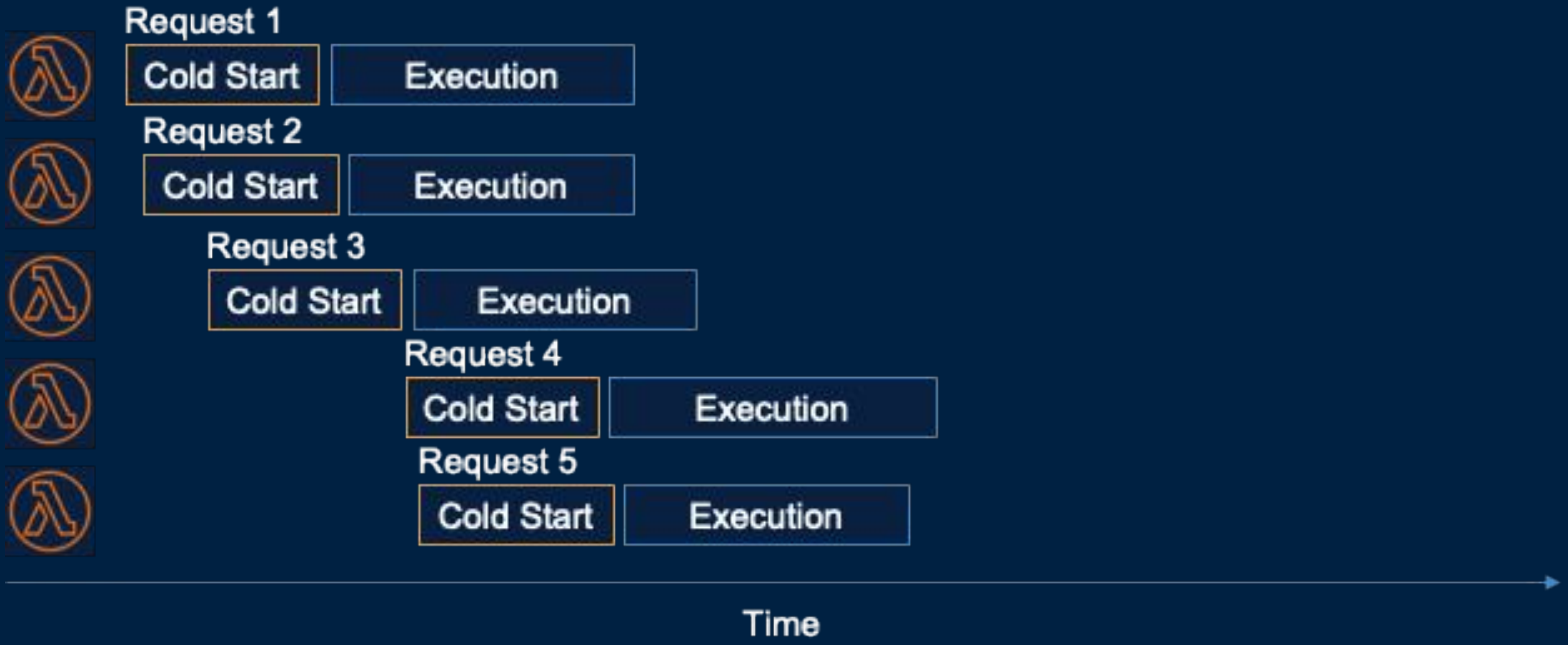
Lambda Scalability



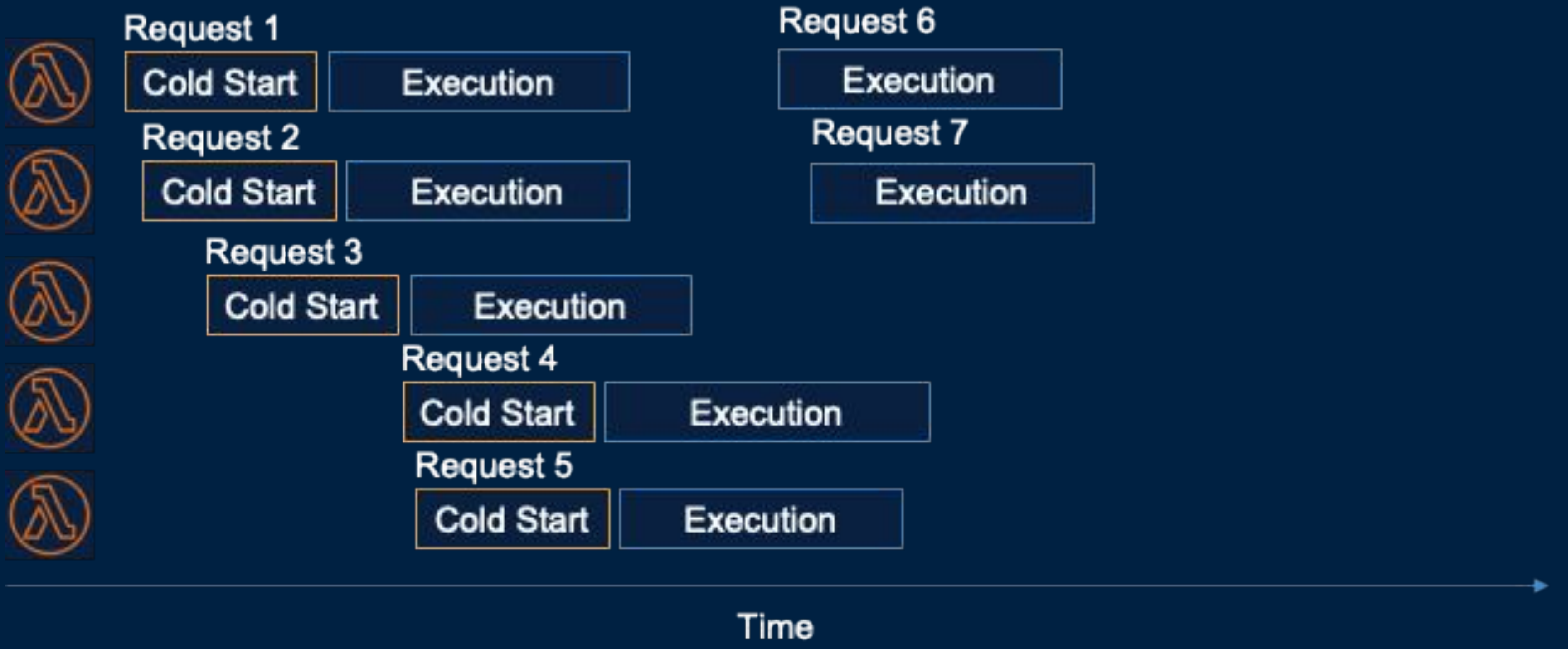
Lambda Scalability



Lambda Scalability



Lambda Scalability



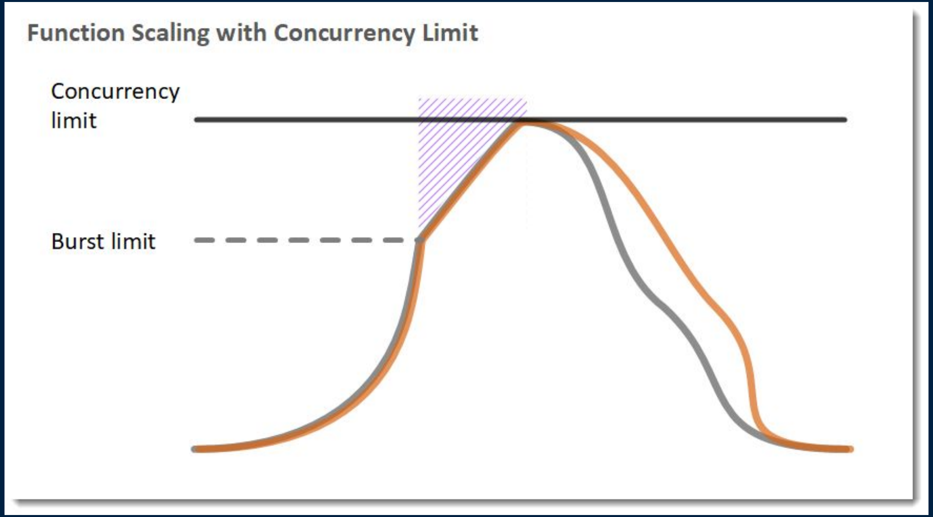
Lambda Scalability

AWS accounts are limited to 1000 concurrency by default.

- This is a soft limit and can be adjust - Raise a support ticket

Burst concurrency varies by region

- 3000 - US West (Oregon), US East (N. Virginia), Europe (Ireland)
- 1000 - Asia Pacific (Tokyo), Europe (Frankfurt)
- 500 - Other Regions



Understanding Cold Starts

- Your code is downloaded from Amazon S3
- A new Firecracker microVM is started
- The JVM is started
- Your application code is loaded
- Your function is invoked

- AWS re:Invent 2020: Ahead of time: Optimize your Java application on AWS Lambda
- https://www.youtube.com/watch?v=sVJOJUD0fhQ

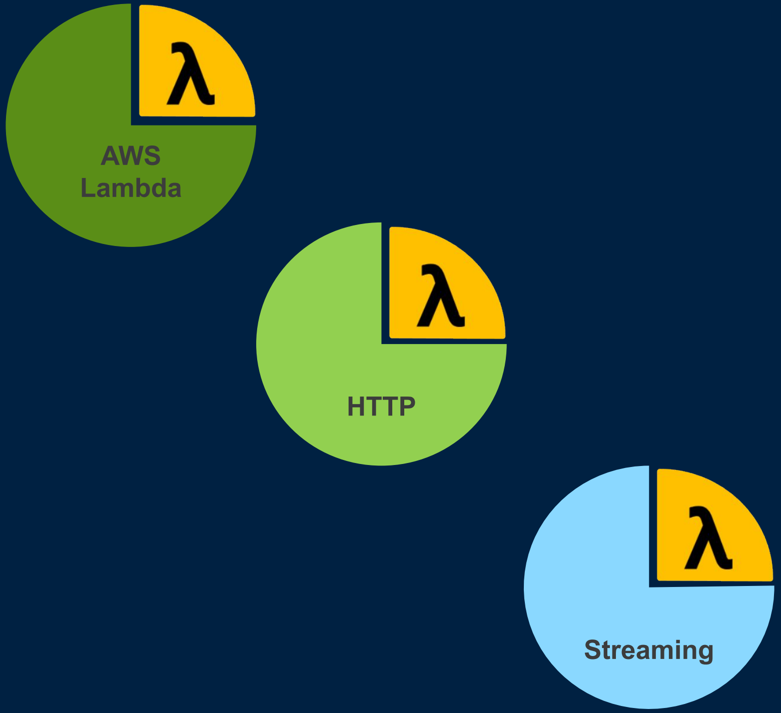
DEMO

Spring Native on AWS Lambda

Streaming w/
Spring Cloud Function,
Spring Cloud Stream
& Solace

Java Functions – Activation/Invocation

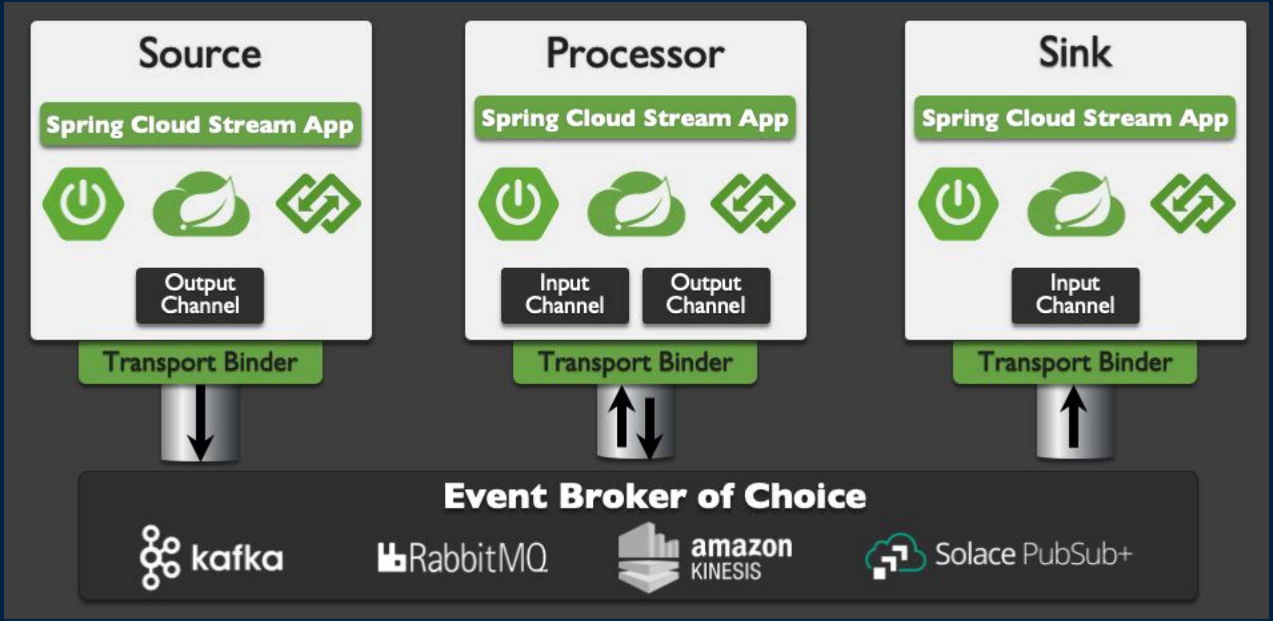
- Contract
- Pattern

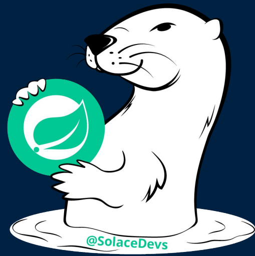


solace

Spring Cloud Stream

- A framework for writing event-driven/stream processing microservices connected to pluggable messaging systems.
- Based on Spring Boot, Spring Integration and Spring Messaging



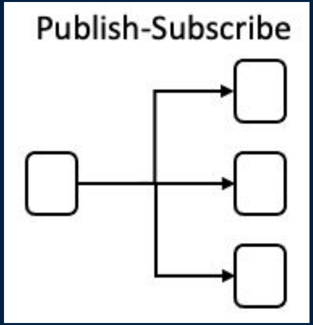


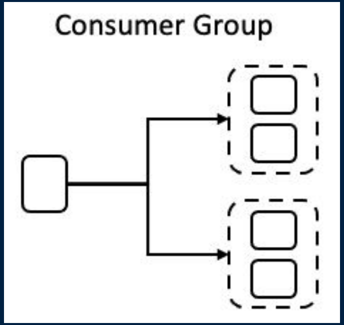
solace

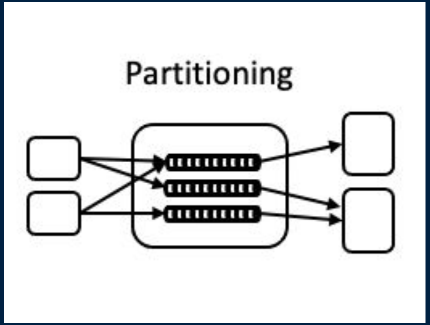
Spring Cloud Stream – Abstraction Framework for Events

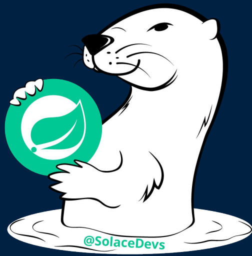
Develop event-driven microservices without having to know messaging APIs

- 3 Communication Models:
  - Persistent Publish-Subscribe
  - Consumer Groups
  - Stateful Partitioning Support







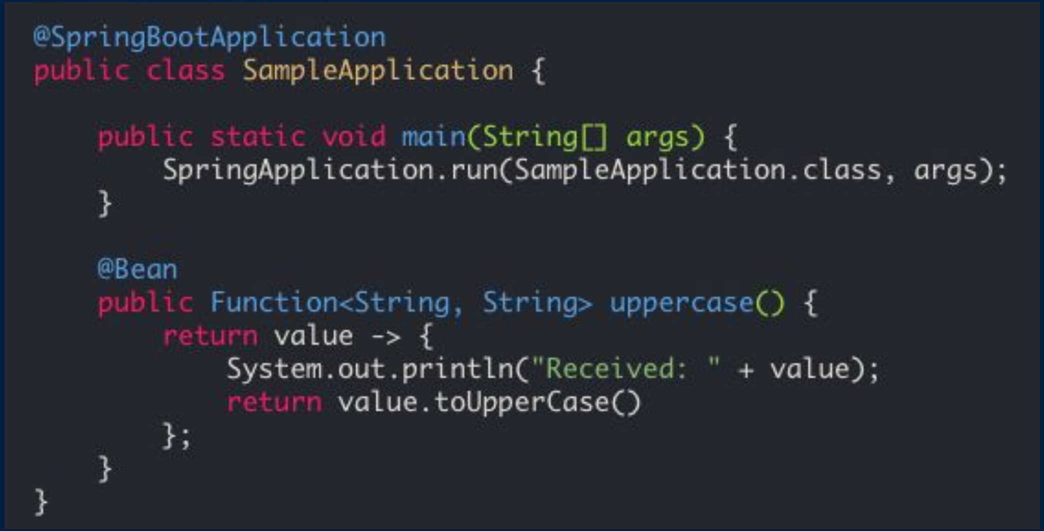


solace

Spring Cloud Stream w/ Spring Cloud Function

Use Spring Cloud Function to write your code!

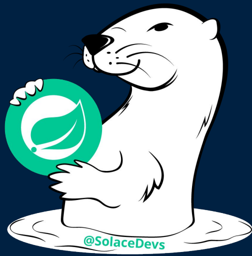
- *java.util.function.Supplier* -> Source
- *java.util.function.Function* -> Processor
- *java.util.function.Consumer* -> Sink



```java
@SpringBootApplication
public class SampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SampleApplication.class, args);
    }

    @Bean
    public Function<String, String> uppercase() {
        return value -> {
            System.out.println("Received: " + value);
            return value.toUpperCase()
        };
    }
}
```



solace.

Demo: Spring Cloud Stream

solace

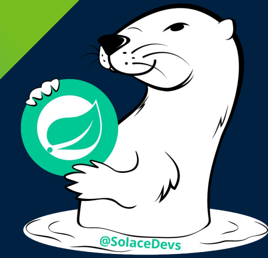
Feature Discussion:

Publishing to Dynamic Destinations (Topics)

solace

Why Dynamic Destinations?

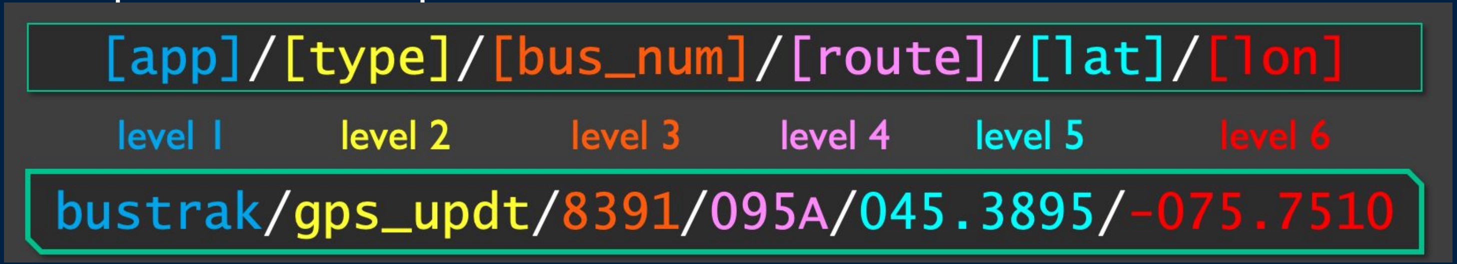
Allows for improved decoupling of producers and consumers in Event-Driven Architecture

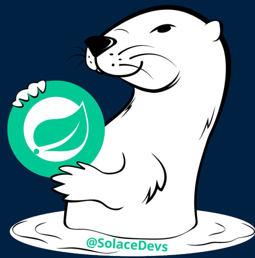


solace

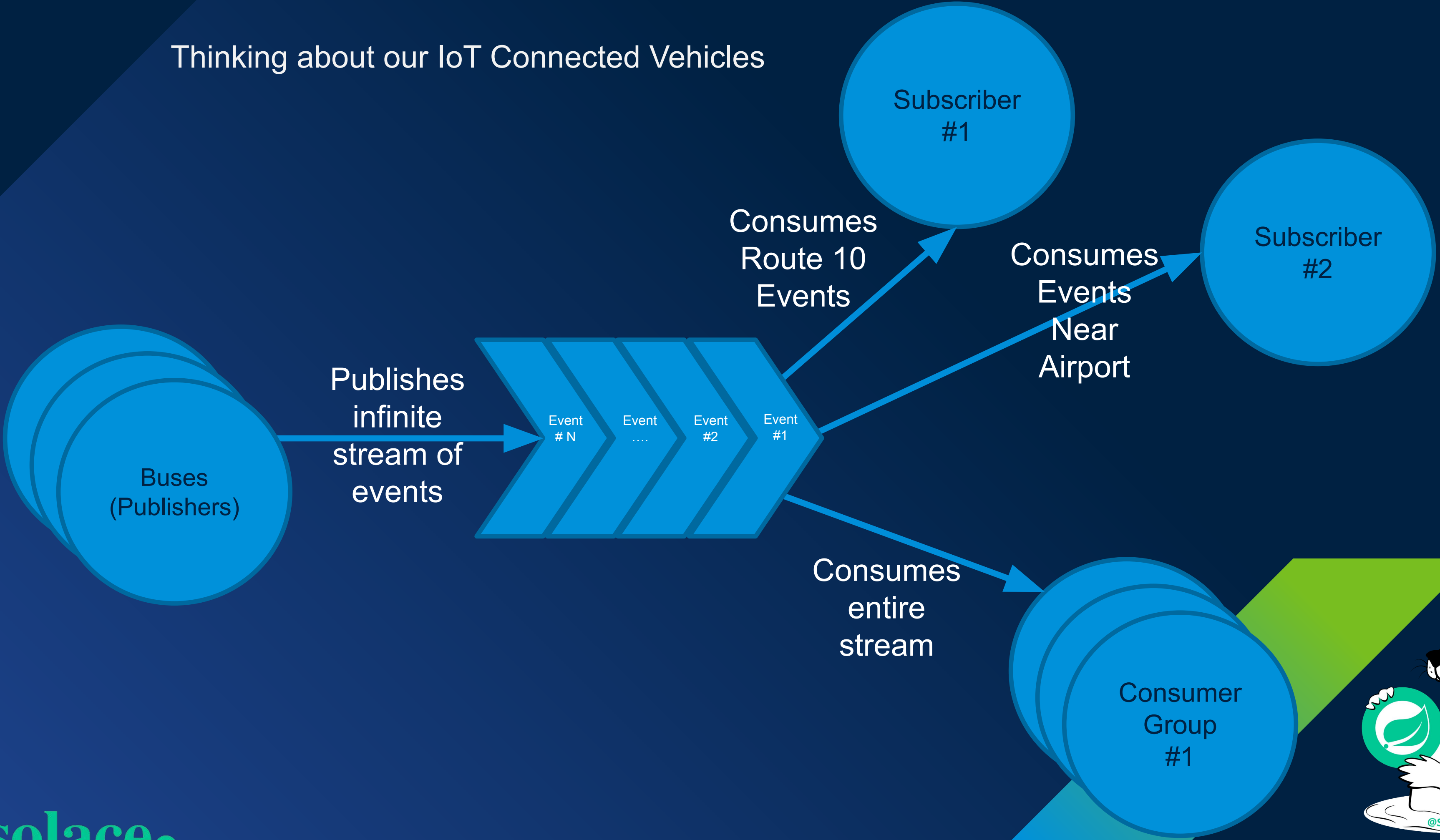
Give Topics Meaning – Describe the Event!

- Producers publish to topics; Consumers Subscribe to Topics
  - Topics are the coupling point of an EDA
- Let's make this coupling point flexible
  - Use a Hierarchical structure allowing for levels (e.g: delimited by "/" forward slash)
  - Each topic level can be a variable or enum, derived from the data
  - Have the topic describe the contents of the message data
- Example Solace topics for an IoT Connected Vehicles architecture


[app]/[type]/[bus_num]/[route]/[lat]/[lon]
level 1
level 2
level 3
level 4
level 5
level 6
bustrak/gps_updt/8391/095A/045.3895/-075.7510

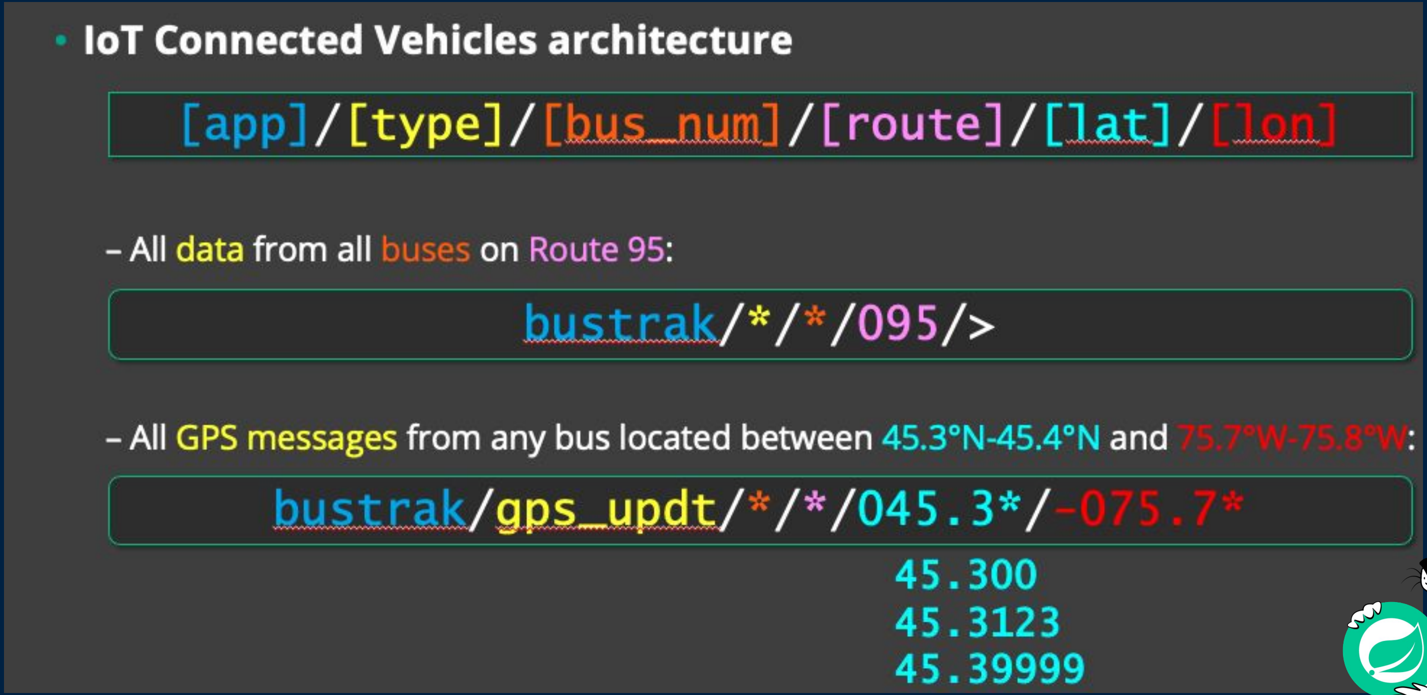


Thinking about our IoT Connected Vehicles



Fine-Grained Filtering for Consumers

- IoT Connected Vehicles architecture



- **IoT Connected Vehicles architecture**

```
[app]/[type]/[bus_num]/[route]/[lat]/[lon]
```

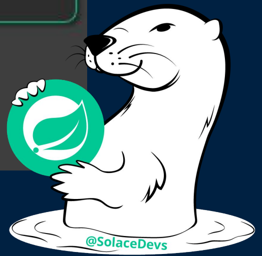
– All data from all buses on Route 95:

```
bustrak/*/*/095/>
```

– All GPS messages from any bus located between 45.3°N-45.4°N and 75.7°W-75.8°W:

```
bustrak/gps_updt/*/*/045.3*/-075.7*
```

```
45.300
45.3123
45.39999
```



solace

Dynamic Topics In Action: IoT Connected Vehicles Demo

solace

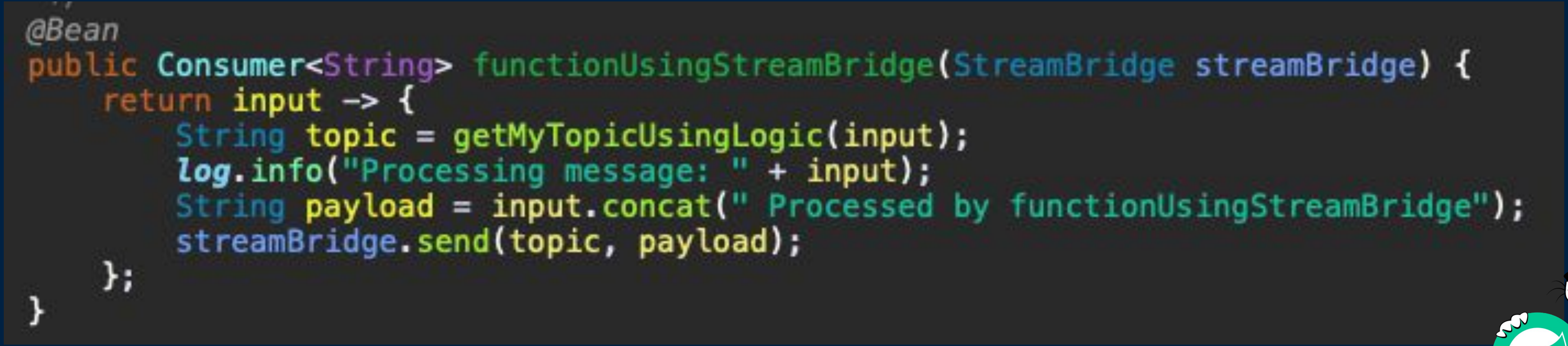
So how do I publish to Dynamic Topics using Spring Cloud Stream?

solace

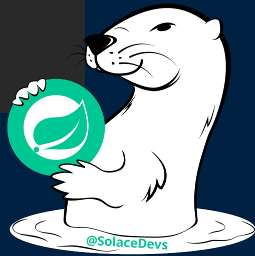
Dynamic Publish Option 1: Using StreamBridge

StreamBridge

- Developer works with only the POJO (no need to specify *Message* object)
- Spring manages each topic as its own Spring Integration Channel (useful for metrics!)
- Caches a configurable number of channels *`spring.cloud.stream.dynamic-destination-cache-size`*



```
@Bean
public Consumer<String> functionUsingStreamBridge(StreamBridge streamBridge) {
    return input -> {
        String topic = getMyTopicUsingLogic(input);
        log.info("Processing message: " + input);
        String payload = input.concat(" Processed by functionUsingStreamBridge");
        streamBridge.send(topic, payload);
    };
}
```

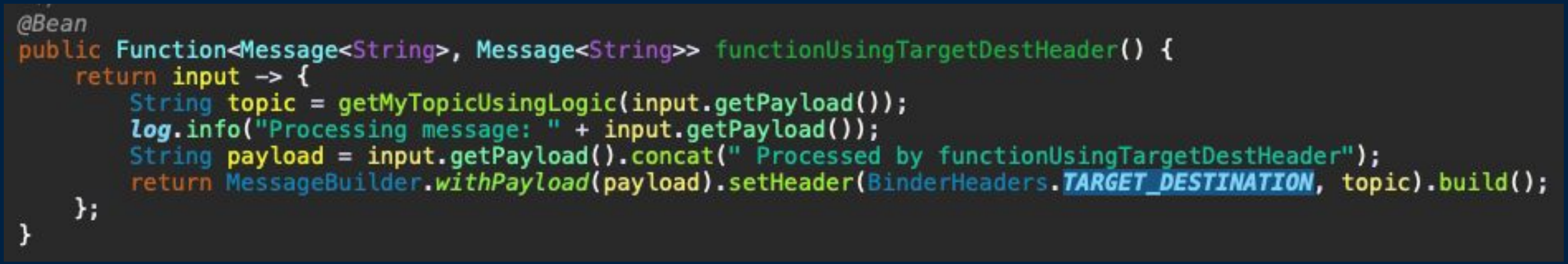


solace

Dynamic Publish Option 2: Using Headers

Add a Special Header to the *Message* object:

1. Framework handles dynamic destination resolution: *spring.cloud.stream.sendto.destination*
2. Binder handles dynamic destination resolution: *BinderHeaders.TARGET_DESTINATION*
   - Only supported by a subset of binders



```java
@Bean
public Function<Message<String>, Message<String>> functionUsingTargetDestHeader() {
    return input -> {
        String topic = getMyTopicUsingLogic(input.getPayload());
        log.info("Processing message: " + input.getPayload());
        String payload = input.getPayload().concat(" Processed by functionUsingTargetDestHeader");
        return MessageBuilder.withPayload(payload).setHeader(BinderHeaders.TARGET_DESTINATION, topic).build();
    };
}
```

Demo: Spring Cloud Stream sending to Dynamic Destinations

solace

Spring Cloud Stream Takeaway

- Spring Cloud Stream allows for invocation and activation of Spring Cloud Function for streaming use cases while providing the necessary functionality to develop event-driven microservices to solve real world challenges
- Want to Learn More?
  - Office Hours at the Solace Booth until 1pm ET
  - Come Ask Questions in #3-solace-sponsor during SpringOne or Join the Solace Community after: https://solace.community

solace

Resources

Demo Repository: https://github.com/olegz/springone2021

Spring Cloud Function GitHub: https://github.com/spring-cloud/spring-cloud-function

Spring Cloud Function: https://spring.io/projects/spring-cloud-function

Spring Cloud Function preso S1-2020 -
https://springone.io/2020/sessions/functions-implement-once-execute-anywhere

Spring Cloud Stream GitHub: https://github.com/spring-cloud/spring-cloud-stream

Spring Cloud Stream: https://spring.io/projects/spring-cloud-stream

AWS CDK - https://aws.amazon.com/cdk/

Hands on CDK Workshop - https://cdkworkshop.com/

Solace w/ Spring Codelabs: https://codelabs.solace.dev/?cat=spring

Solace Spring Samples: https://github.com/SolaceSamples/solace-samples-spring

About...

Oleg Zhurakousky

Spring Cloud Function/Spring Cloud Stream

project lead

Twitter: @z_oleg

Github: github.com/olegz

About...

Marc DiPasquale

Solace

Developer Advocate

Twitter: @Mrc0113

Github: github.com/Mrc0113

About...

Mark Sailes

Amazon Web Services

Specialist SA, Serverless

Twitter: @MarkSailes3

Github: github.com/msailes