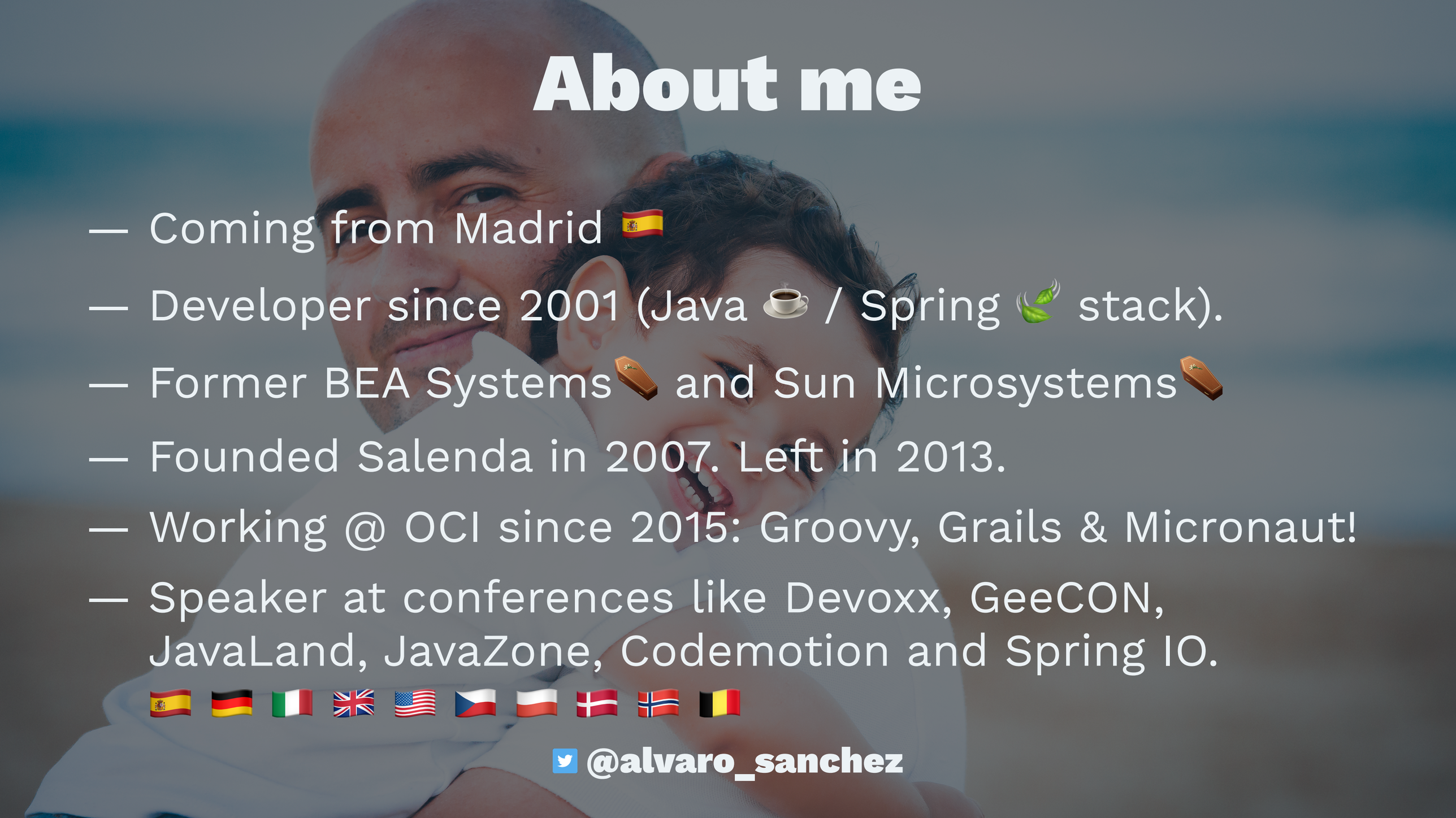
Serverless functions with Micronaut

OCI | WE ARE SOFTWARE ENGINEERS.

MICRONAUT

Álvaro Sánchez-Mariscal
@alvaro_sanchez



About me

- Coming from Madrid 🇪🇸
- Developer since 2001 (Java ☕ / Spring 🌿 stack).
- Former BEA Systems⚰️ and Sun Microsystems⚰️
- Founded Salenda in 2007. Left in 2013.
- Working @ OCI since 2015: Groovy, Grails & Micronaut!
- Speaker at conferences like Devoxx, GeeCON, JavaLand, JavaZone, Codemotion and Spring IO.

🇪🇸 🇩🇪 🇮🇹 🇬🇧 🇺🇸 🇨🇿 🇵🇱 🇩🇰 🇳🇴 🇧🇪

@alvaro_sanchez

Introducing

MICRONAUT

MICRONAUT

@alvaro_sanchez

Introducing Micronaut

The productivity of Spring Boot / Grails with a compile-time, non-blocking performance.

— Designed from the ground up with microservices and the cloud in mind ("Natively Cloud Native").

— Ultra-lightweight and Reactive (based on Netty)

— Integrated AOP and Compile-Time DI for Java, Groovy and Kotlin.

OCI | WE ARE SOFTWARE ENGINEERS.

@alvaro_sanchez

MICRONAUT

Hello, Galaxy! 🧑‍🚀

```
interface HelloApi {
    @Get("/hello") String hello();
}

@Controller("/")
class HelloServer implements HelloApi {
    @Get("/hello")
    public String hello() {
        return "Hello, Galaxy!";
    }
}

@Client("/") // Client Generated at Compile Time
interface HelloClient extends HelloApi {}
```

OCI | WE ARE SOFTWARE ENGINEERS.

@alvaro_sanchez

MICRONAUT

Hello, Galaxy! 🧑‍🚀

```
@MicronautTest
class ApplicationTest {

  @Inject
  HelloClient helloClient;

  @Test
  void testHello() {
    Assertions.assertEquals("Hello, Galaxy!", helloClient.hello());
  }

}
```

OCI | WE ARE SOFTWARE ENGINEERS.

@alvaro_sanchez

MICRONAUT

Natively Cloud Native

— Service Discovery - Consul, Eureka, Route 53 and Kubernetes.

— Configuration Sharing - Consul Supported and Amazon ParameterStore planned.

— Client Side Load Balancing - Integrated or Netflix Ribbon Supported

— Distributed tracing - Zipkin or Jaeger.

— Support for Serverless Computing; AWS Lambda,

Getting started

```
$ curl -s "https://get.sdkman.io" | bash

$ sdk install micronaut

$ mn create-function uppercase
| Generating Java project...
| Function created at /tmp/uppercase
```

Getting started: @FunctionBean

```
import io.micronaut.function.FunctionBean;
import java.util.function.Function;

@FunctionBean("isbn-validator")
public class IsbnValidatorFunction
        implements Function<IsbnValidationRequest, IsbnValidationResponse> {

    @Override
    public IsbnValidationResponse apply(IsbnValidationRequest request) {
        return new IsbnValidationResponse(); //do stuff
    }
}
```

Getting started: @FunctionBean

- Use Java 8's `java.util.function.*`:
  - `Supplier<O>`: no arguments, single result.
  - `Consumer<I>`: single argument, no result.
  - `BiConsumer<I, J>`: two arguments, no result.
  - `Function<I, O>`: single argument, single result.
  - `Function<I, J, O>`: two arguments, single result.

Getting started: @Factory

```
@Factory
public class WordFunctions {

    @FunctionBean("uppercase")
    public Function<String, String> uppercase() {
        return String::toUpperCase;
    }

    @FunctionBean("lowercase")
    public Function<String, String> lowercase() {
        return String::toLowerCase;
    }

}
```

OCI | WE ARE SOFTWARE ENGINEERS.

@alvaro_sanchez

MICRONAUT

Getting started: Groovy

```
//src/main/groovy/com/acme/HelloFunction.groovy

import groovy.transform.Field
import javax.inject.Inject

@Field @Inject HelloService helloService

String hello(String name) {
    helloService.hello(name)
}
```

Testing functions

— Micronaut can expose functions as local REST endpoints.

— `@FunctionClient` can be leveraged in tests:

```
@FunctionClient
public interface IsbnValidatorClient {

    @Named("isbn-validator")
    @Retryable(attempts = "5", delay = "5s")
    Single<IsbnValidationResponse> validate(@Body IsbnValidationRequest req);
}
```

OCI | WE ARE SOFTWARE ENGINEERS.

@alvaro_sanchez

MICRONAUT

Invoking remote functions

— The same `@FunctionClient` can invoke remote functions:

```
# src/main/resources/application.yml
aws:
    lambda:
        functions:
            slack:
                functionName: slack-message
        region: eu-west-1
```

Functions as CLI applications

— The build is configured to produce runnable JARs:

```
$ echo '{value: 3}' | java -jar build/libs/math-function-0.1-all.jar
```

— This allows functions written with Micronaut to be deployed to platforms that process standard input/output, such as OpenFaaS.

GraalVM

— New Polyglot VM from Oracle.
— Runs JS, Java, Ruby, R etc.
— Ability to turn Java code into native images.
— Can run Micronaut applications and functions blazingly fast!

GraalVM™

OCI | WE ARE SOFTWARE ENGINEERS.

@alvaro_sanchez

MICRONAUT

Demo

DEVOXX™

Please Rate Me!

If you enjoyed the talk
Or give feedback

Q & A

OCI | WE ARE SOFTWARE ENGINEERS.

MICRONAUT

Álvaro Sánchez-Mariscal
@alvaro_sanchez