Building Microservices with Micronaut

Michael P. Redlich

Who's Mike?

- Bachelor of Science, Computer Science
- "Petrochemical Research Organization"
- Java Queue News Editor, InfoQ
- Leadership Council, Jakarta EE Ambassadors
- Amateur Computer Group of New Jersey

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Objectives

- What is Micronaut?
- Why Micronaut?
- Features
- JVM Language Support
- Getting Started
- Live Demos (yea!)

What is Micronaut?

What is Micronaut?

- A full-featured, full-stack JVM-based lightweight application framework for creating microservice-based, cloud-native and serverless applications that can be written in Java, Groovy and Kotlin
- Created by Graeme Rocher, Principal Software Engineer at Object Computing, Inc.

What is Micronaut?

- First introduced at the Greach Conference in March 2018
- Designed from the ground up for microservices and serverless applications
- Based on Ahead-of-Time (AoT) compilation
- Current version: 1.3.4
    - version 2.0.0-M2 available

Why Micronaut?

First, Let's Travel Back in Time to 2008…

2008

- Grails 1.0 released
- Applications were monoliths
- Before the advent of microservices and technologies such as:
  - Angular
  - React
  - Docker

2008

- Attempt to adapt a monolith-focused framework into the microservices environment
- Spring and Grails were not designed for the microservices environment

Why Micronaut?

"We believe Micronaut is the basis for a framework for the future, by resolving this tension by eliminating all use of reflection and producing all annotation metadata, proxies and and framework infrastructure at compilation time through a set of annotation processors and AST transformations that perform Ahead-of-Time (AoT) compilation…"

Graeme Rocher, Grails & Micronaut Lead at OCI

Why Micronaut?

"…What this allows Micronaut to achieve is blazing fast startup time, low memory consumption and crucially improved compatibility with GraalVM native image."

Graeme Rocher, Grails & Micronaut Lead at OCI

Features

Features

- A JVM-based framework
- Natively cloud-native
- Fast startup time and low memory consumption
- Reactive and non-blocking
- Fast and easy testing

Features

- HTTP Server
- HTTP Client
- Microservice Patterns
  - service discovery
  - distributed tracing
  - circuit breaker

Micronaut Projects

- Micronaut AWS
- Micronaut GCP
- Micronaut Test
- Micronaut RabbitMQ
- Micronaut Data
- Micronaut for Spring
- Micronaut Security
- Micronaut MongoDB
- Micronaut Kafka
- Micronaut Servlet

HTTP Server

- Fully reactive and non-blocking server built on top of Netty
  - Supports Project Reactor and RxJava
- Auto configuration for common databases

HTTP Server

```
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;

@Controller("/hello")
public class HelloController {

    @Get("/")
    public String index() {
        return "Hello World!";
        }
    }
```

HTTP Client

- Declarative, reactive, compile-time client
- Automatic service discovery
- Automatic load balancing

HTTP Client

```
import io.micronaut.http.annotation.Get;
import io.micronaut.http.client.Client;
import io.reactivex.Single;

@Client("/hello")
public interface HelloClient {
    @Get("/")
    Single hello();
    }
```

JVM Language Support

JVM Languages

JVM Build Tools

Let's Get Started…

Install Micronaut

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

```
$ sdk install micronaut
```

```
$ mn
```

```
mn> help
```

Built-In Profiles

- Project templates consisting of skeleton project structures with default configurations, dependencies, etc.
  - service
  - cli
  - configuration
  - etc.

Built-In Commands

- General commands to build various parts of a Micronaut application
  - create-app
  - create-controller
  - create-client
  - create-function
  - etc.

Working with Profiles

```
$ mn list-profiles
```

```
$ mn profile-info service
```

```
$ mn create-app org.redlich.demo --profile cli
```

Generate an Initial Application

```
$ mn create-app org.redlich.demo
```

```
$ mn create-app org.redlich.demo --lang groovy
```

```
$ mn create-app org.redlich.demo --lang kotlin
```

Generate an Initial Application

```
$ mn create-app org.redlich.demo
```

```
$ mn create-app org.redlich.demo --build maven
```

Adding Features

```
$ mn create-app org.redlich.demo --features security-jwt
```

```
$ mn create-app org.redlich.demo --features data-jdbc
```

```
$ mn create-app org.redlich.demo --features jdbc-tomcat
```

```
$ mn create-app org.redlich.demo --feaures rabbitmq
```

Live Demo

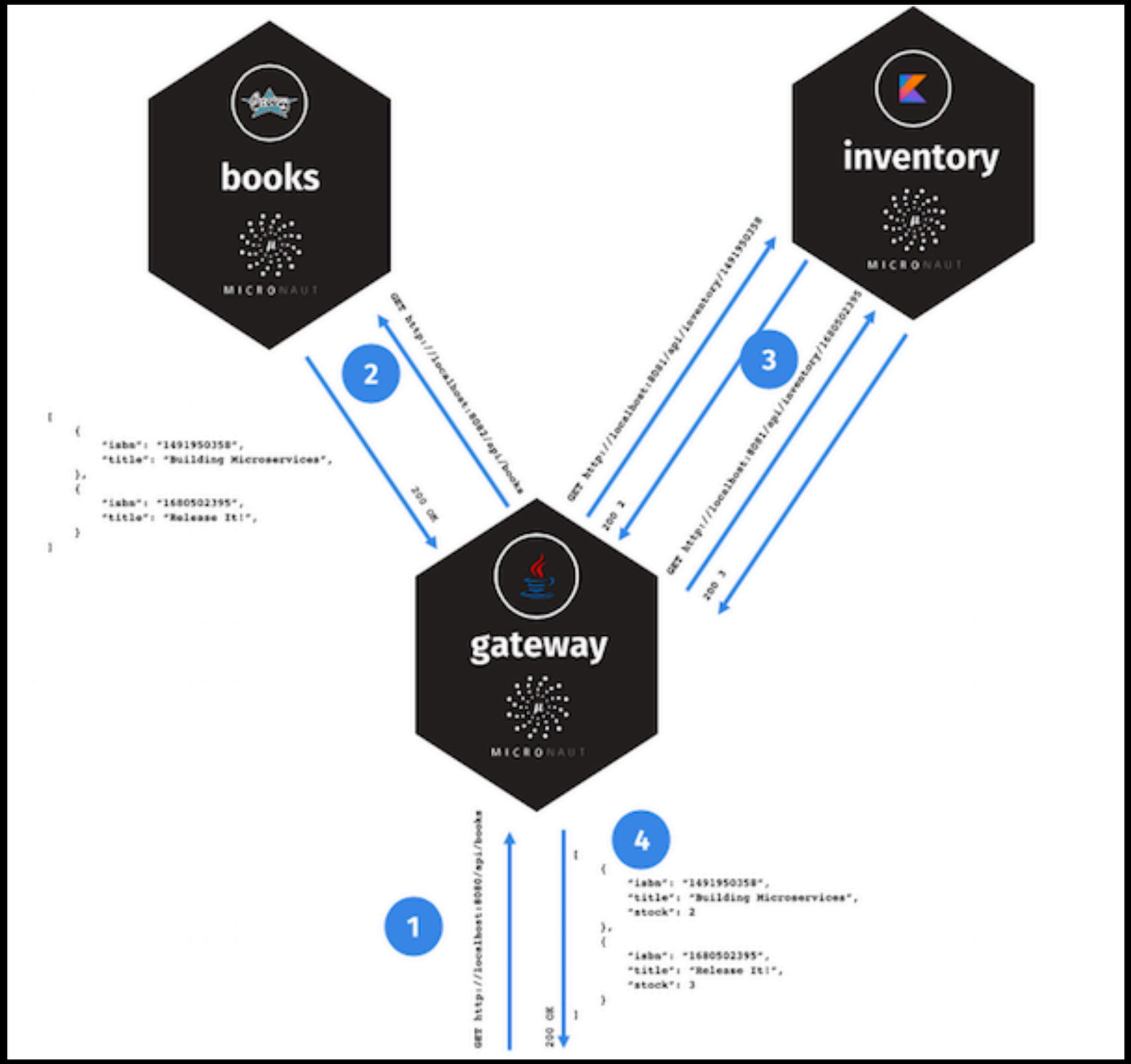
Demo Application

Introduction

- A book inventory application built with three microservices
- Based on a tutorial by Sergio del Ama Caballero, senior software engineer at OCI
- Uses Consul, a distributed service mesh to connect, secure and configure services across any runtime platform or cloud

Microservices

- **`books`** microservice (Groovy)
- **`inventory`** microservice (Kotlin)
- **`gateway`** microservice (Java)


books
MICRONAUT
inventory
MICRONAUT
gateway
MICRONAUT
2
3
1
4
GET http://localhost:8082/api/books
200 OK
GET http://localhost:8081/api/inventory/1491950358
200 3
GET http://localhost:8081/api/inventory/1680502395
200 3
GET http://localhost:8080/api/books
200 OK
[
{
"isbn": "1491950358",
"title": "Building Microservices",
},
{
"isbn": "1680502395",
"title": "Release It!",
}
]
[
{
"isbn": "1491950358",
"title": "Building Microservices",
"stock": 2
},
{
"isbn": "1680502395",
"title": "Release It!",
"stock": 3
}
]

Creating the Application

```
$ mn create-app example.micronaut.books --lang groovy
```

```
$ mn create-app example.micronaut.inventory --lang kotlin
```

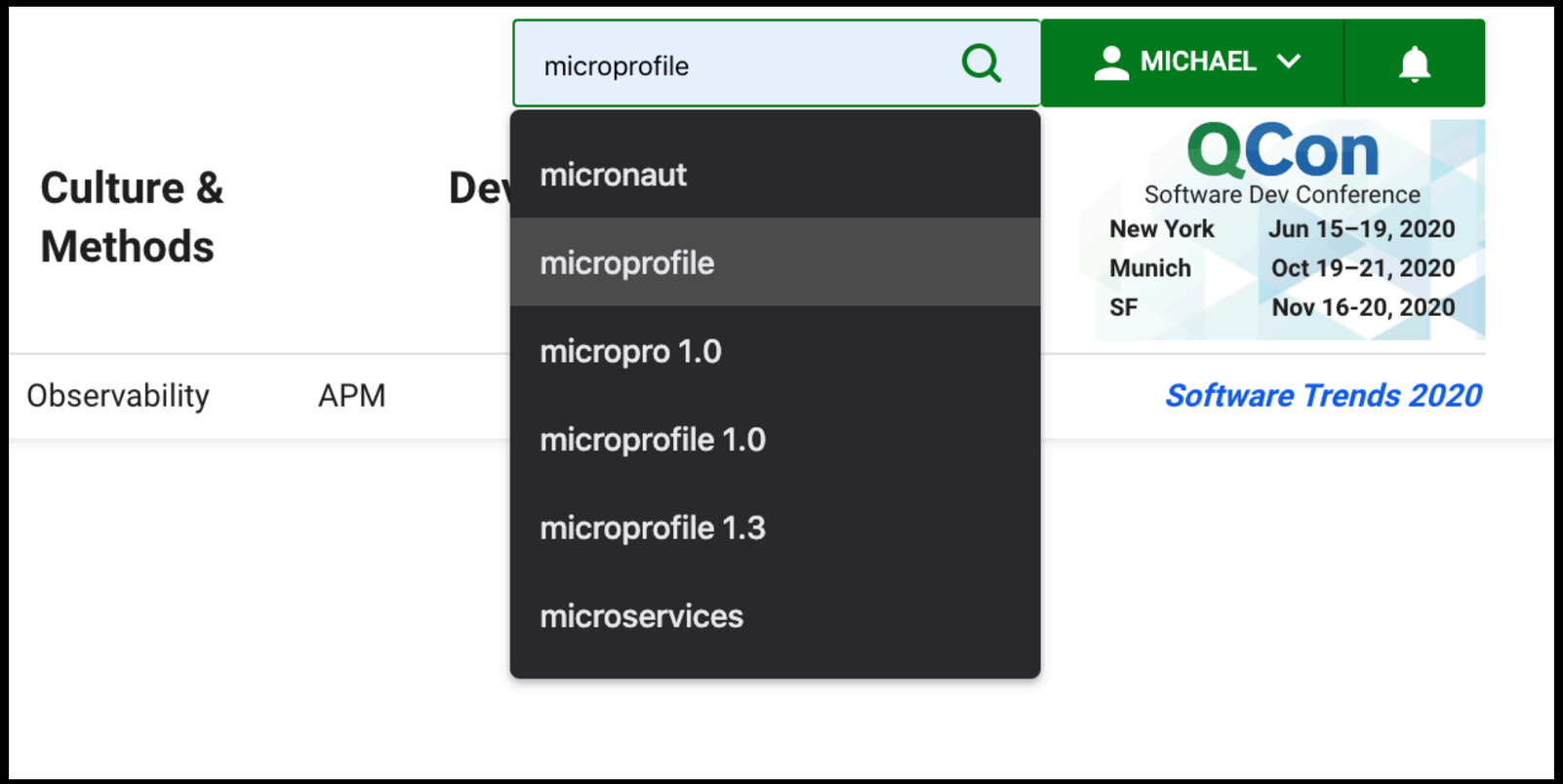
```
$ mn create-app example.micronaut.gateway
```

Live Demo

Micronaut Resources

- https://micronaut.io
- https://guides.micronaut.io
- https://micronaut.io/documentation.html
- objectcomputing.com/news/category/micronaut-blog

Further Reading



Contact Info

mike@redlich.net

@mpredli

redlich.net/portfolio

github.com/mpredli01

Thanks!