GraalVM™

ConFoo.ca
DEVELOPER CONFERENCE

Unleash the power of your applications with Micronaut® and GraalVM

Álvaro Sánchez-Mariscal
Principal Member of Technical Staff
Oracle

@alvaro_sanchez

Álvaro Sánchez-Mariscal

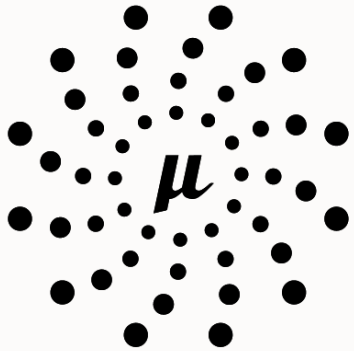
Principal Member of Technical Staff, Oracle

- Coming from Madrid 🇪🇸
- Developer since 2001 (Java ☕ stack)
- Micronaut core developer since its inception (2017).
  - Author: Maven Plugin, Object Storage, Kubernetes, Cache, Control Panel.
  - Others: Core, Gradle, AWS, GCP, Azure, Security, Test, etc.
- Currently at Oracle Labs:
  - Micronaut and Graal Cloud Native (GCN).
  - GraalVM Native Build Tools.

- Speaker at conferences: Devoxx, GeeCON, JavaLand, JavaZone, jPrime, Codemotion, Commit Conf, etc.

ORACLE Labs

Slides available on Speaker Deck



MICRONAUT®

Introduction to Micronaut

Modern Java Framework

Micronaut has been designed from scratch in 2017, focused on modern architectures like microservices and serverless, and with the cloud in mind.

All application types

Micronaut is a complete solution for any type of application: microservices, message-driven producers or consumers, command-line apps, serverless functions, etc.

Highly optimised

Micronaut leverages Java annotation processors and other optimisations to compute the framework infrastructure at compile-time, drastically reducing startup time and memory consumption.

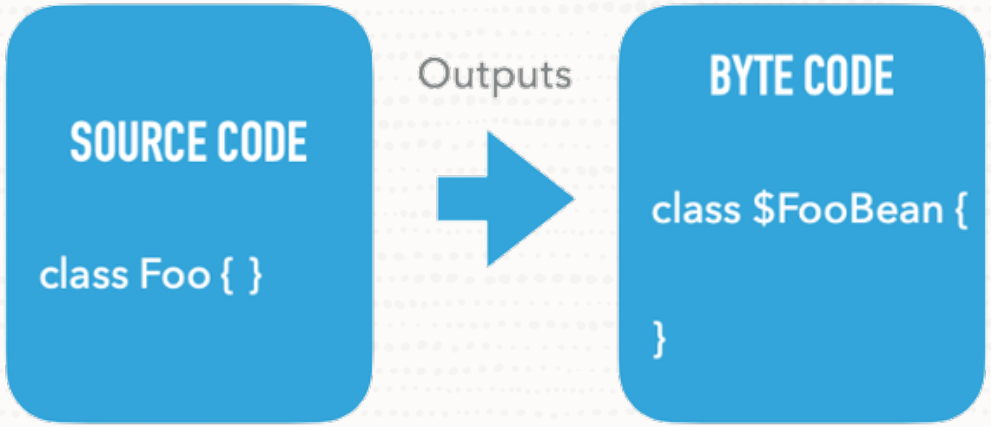
AOT: Ahead Of Time

Micronaut computes at **build time**:

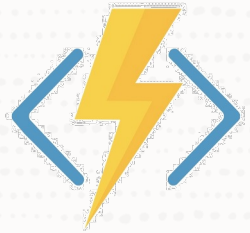
- All dependency and configuration injection.
- Annotation metadata (meta annotations)
- AOP proxies.
- Bean introspections.
- All other framework infrastructure.

At **runtime**:

- No reflection.
- No proxy generation.
- No dynamic classloading.
- No classpath scanning.



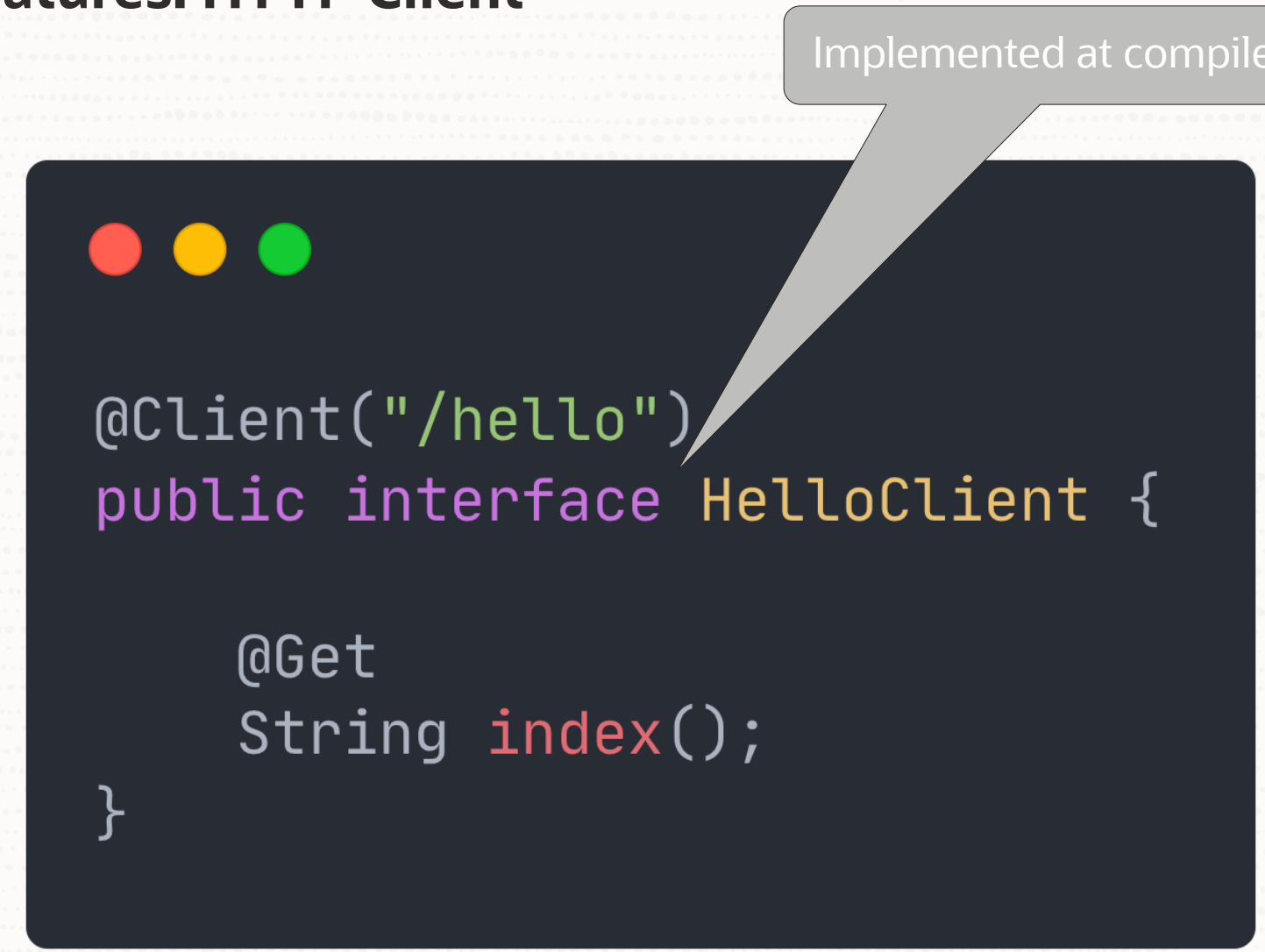
Any language, build tool, test framework, reactive library, cloud, ...



Micronaut Hello World

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

    @Get(produces = MediaType.TEXT_PLAIN)
    public String index() {
        return "Hello World";
    }
}
```

Micronaut features: HTTP Client



Micronaut features: message-driven applications

Kafka

- Producers.
- Consumers.
- Kafka Streams.

RabbitMQ

- Producers.
- Consumers.
- RPC.

MQTT

- Publishers.
- Subscribers.

NATS

- Producers.
- Consumers.

Micronaut features: data access

Micronaut features: data access

Interface implemented at compile time

```
@JdbcRepository(dialect = Dialect.ORACLE)
public interface UserRepository extends CrudRepository<User, Long> {

    List<User> findByNameOrderByName(String name);

    @Query("SELECT * FROM user WHERE enabled = false")
    List<User> findDisabled();
}
```

Query generated from the method name, compile-time validated

Can have custom queries, inserts, etc

Micronaut features: security

1. Authentication providers.
   - Bult-in support for LDAP and OAuth 2.0 password grant flow.
2. Security rules.
   - Configuration or annotation-based.
3. Authorization strategies.
   - Basic auth, session, JWT (JWKS, JWS), X.509.
4. OAuth 2.0.
   - Authorization code, client credentials and password grants.
   - OpenID Connect.
   - Ease of integration with providers such as Okta, Auth0, AWS Cognito, Keycloak and more.

Micronaut features: Test Resources

1. Automatic provisioning of external resources needed during development or testing.
   - Containers, binaries, etc.

2. Built-in support:
   - MariaDB
   - MySQL
   - Oracle
   - PostgreSQL
   - SQL Server
   - Elasticsearch
   - Redis
   - MongoDB
   - Neo4j
   - Hashicorp Vault
   - Kafka
   - RabbitMQ
   - MQTT
   - Localstack

3. Integrated with JUnit Platform (JUnit / Spock / Kotest) for zero-configuration Testcontainers.

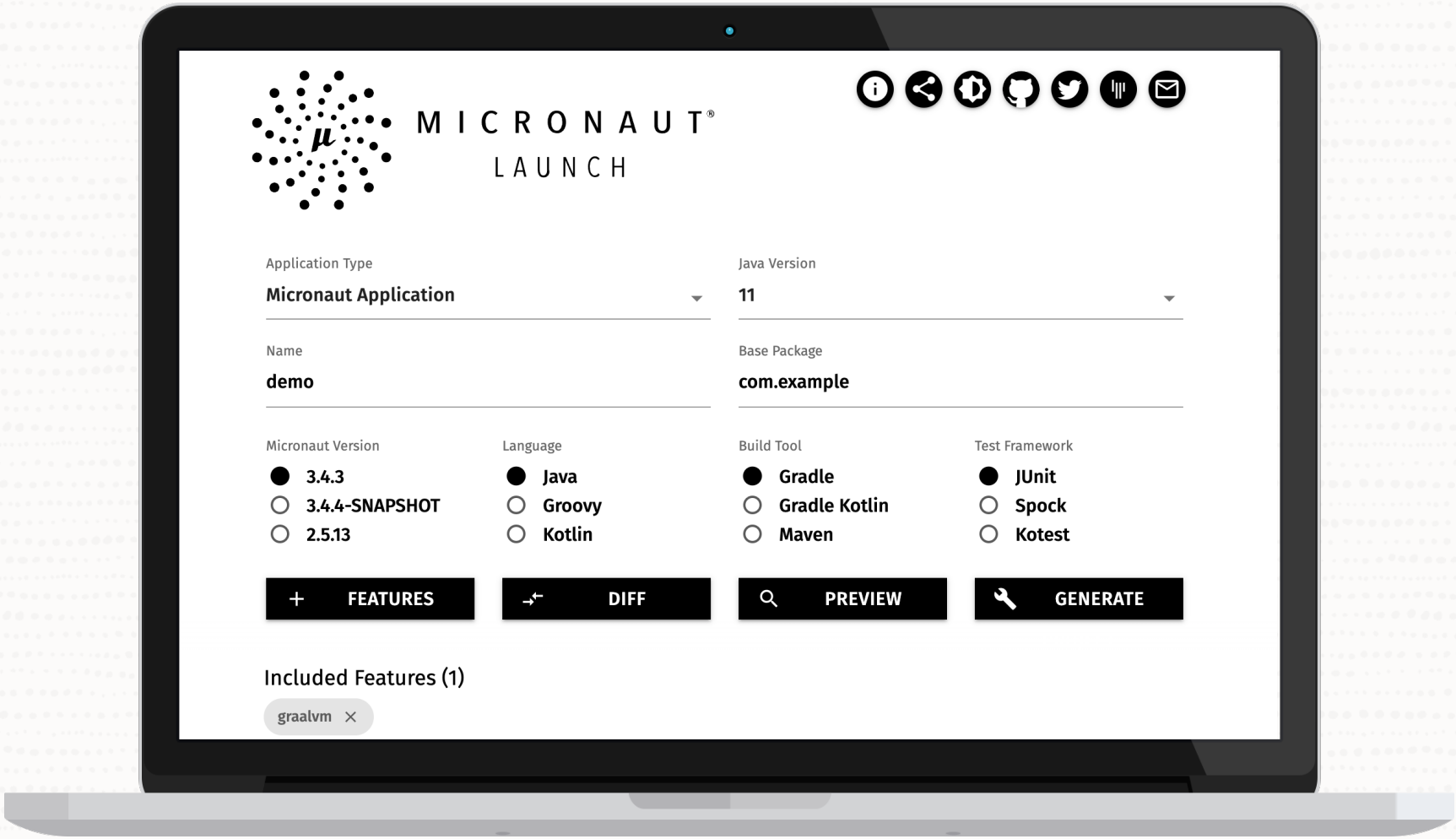
Micronaut features

1. Dependency injection.
   - Can use JSR-330 (`@Inject`) or Spring (`@Autowired`) annotations.
2. Configuration.
   - Properties, YAML, TOML, Groovy, Config4K.
3. Validation support.
   - Built-in: reflection free, faster startup, smaller JAR, reduced memory.
   - Hibernate Validation: full Bean Validation API compliance.
4. AOP: Aspect-Oriented Programming.
   - Compile-time, reflection free.

Micronaut features: misc

1. Distributed Tracing.
   - Zipkin, Jaegger.
2. Service Discovery and Distributed Configuration.
   - Consul, Eureka, Kubernetes.
3. Monitoring.
   - Micrometer, JMX, Elasticsearch.
4. API development.
   - GraphQL, GRPC, Open API.
5. Containers.
   - Docker, Kubernetes, Testcontainers.
6. Cache.
   - Jcache, Redis, Ehcache, Hazelcast, Infinispan, Oracle Coherence.
7. Email.
8. Error handling.

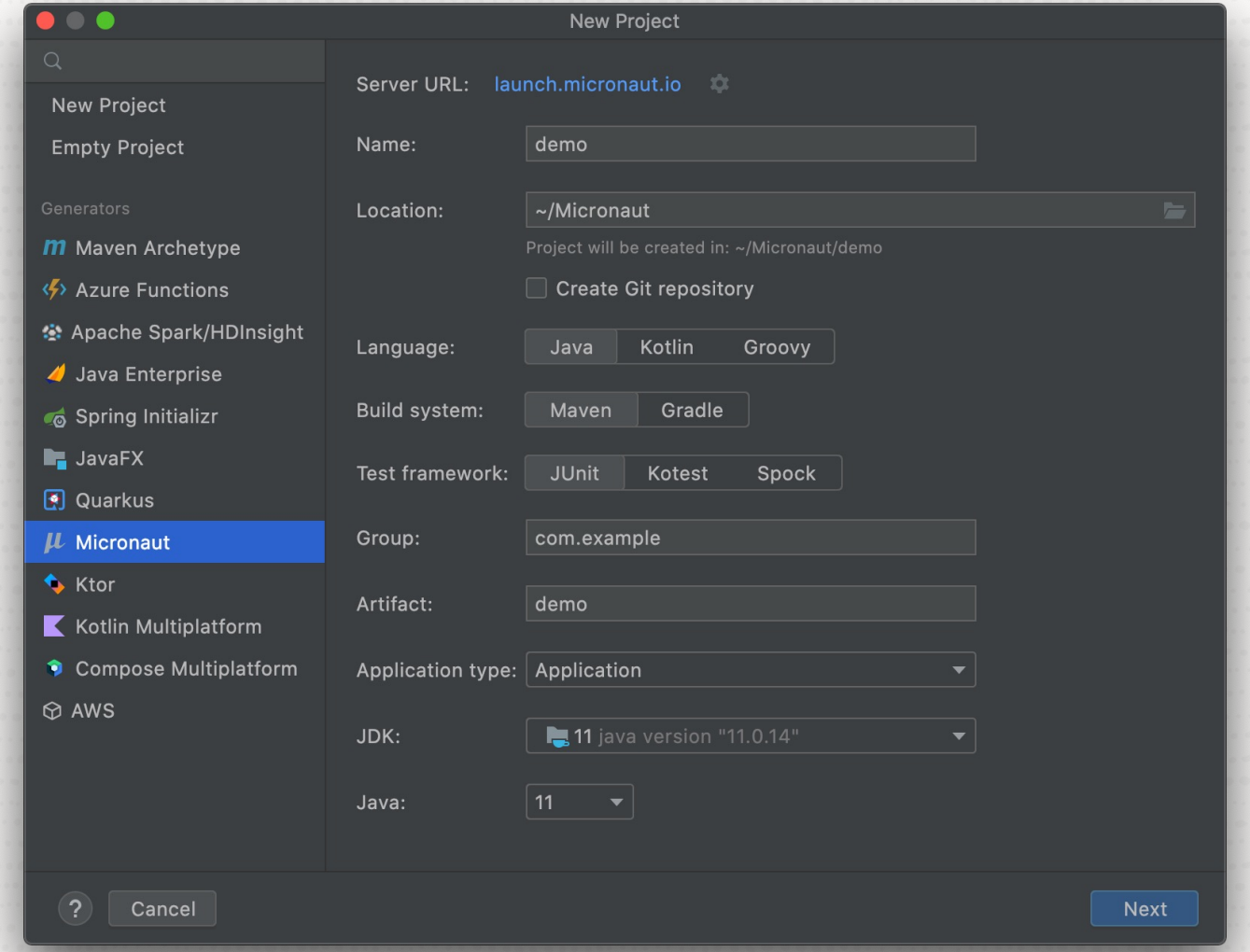
Getting Started: Micronaut Launch



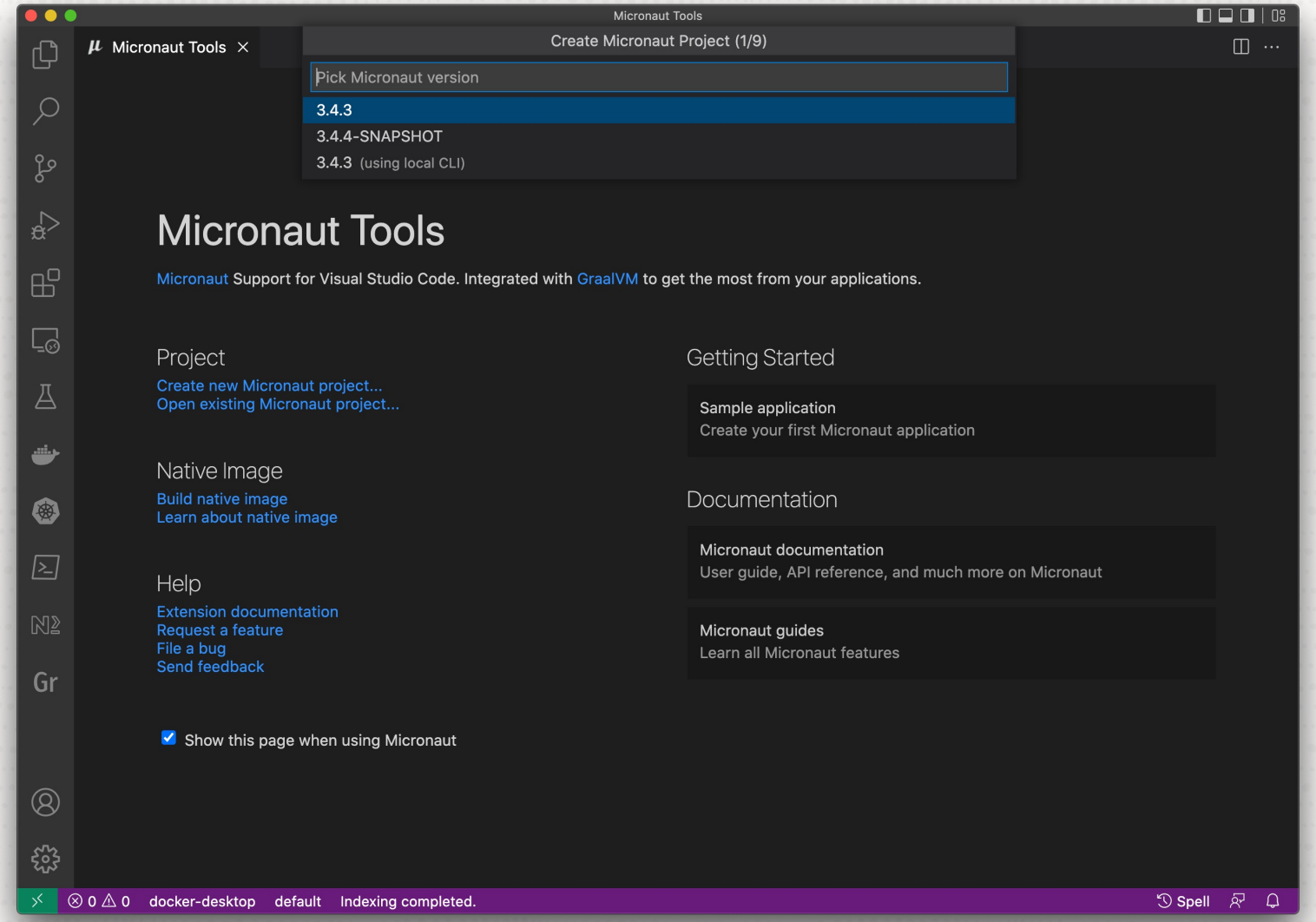
https://launch.micronaut.io

Getting Started: Micronaut CLI

Getting Started: Intellij IDEA



Getting Started: Visual Studio Code



Micronaut 4: what's new

1. Java 17 required.
   - Virtual threads support (Project Loom).
   - @Client implementation based on java.net.http.HttpClient (JEP 321)
   - For older versions of Java use Micronaut 3.x.
   - Other languages
     - Kotlin 1.8, KSP support.
     - Apache Groovy 4.
2. GraalVM improvements.
   - Reachability metadata enabled.
   - Improved build times and reduced memory consumption.
3. Micronaut Data v4
   - Hibernate 6.

Micronaut 4: what's new

3. Smaller, lighter runtime.
   - Performance closer to raw Netty (~200ms startup).
   - Some modules spun off core: Service Discovery, Retry, Session, Validation, Websocket.
   - Some dependencies now optional: Jackson, SnakeYAML, Caffeine.

4. Other features:
   - HTTP client/server filter methods: annotation-based and without reactive APIs.
   - Annotation-based CORS configuration.
   - HTTP/3 support.
   - Prefer Micronaut Serialization over Jackson.
   - Disabled cloud environment deduction by default.

Micronaut 4: what's new

5. Compile-time expression language

Checked at compile time

```
@Scheduled(fixedRate = "1s",condition = "#{!jc.paused}")
public void run(ExampleJobControl jc) {
    System.out.println("Job Running");
    this.jobRan = true;
}
```


Micronaut 4: what's new

5. Compile-time expression language

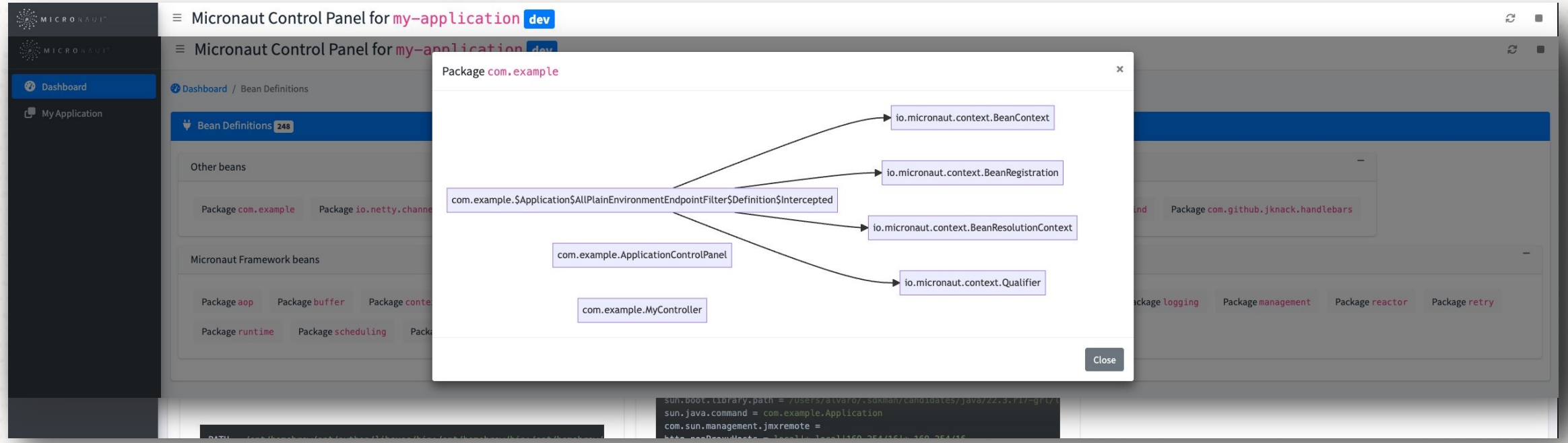
```
@Controller
public class MyController {

    @Get
    @RouteCondition("#{request.headers.getFirst('Authorization')?.contains('foo')}")
    String hello() {
        return "ok";
    }
}
```


Micronaut 4: what's new

6. Bean mappers: built-in, type-safe alternative to MapStruct for type mapping built on EL.

```java
@Singleton
public interface ProductMappers {
    // perform mathematical operations, and provde formatting
    @Mapping(to = "price", from = "#{product.price * 2}", format = "$#.00")
    // traverse the object graph
    @Mapping(to = "distributor", from = "#{product.distributor.name}")
    // combined with custom properties
    @Mapping(to = "storeName", from = "#{this.storeName}")
    ProductDTO toProductDTO(Product product);

    default String getStoreName() {
        return "My Awesome Storefront!";
    }
}
```

Micronaut 4 Control Panel



Micronaut 4 coming soon: source generation

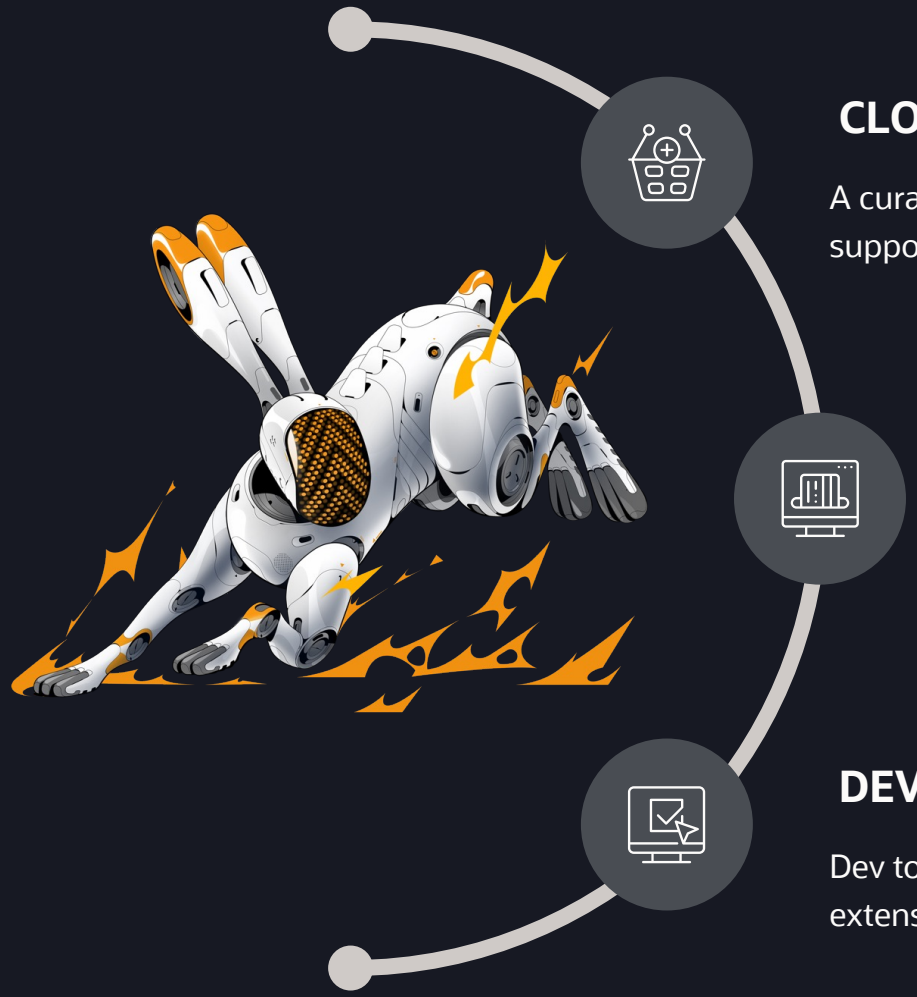
```java
import io.micronaut.sourcegen.annotations.Builder;

@Builder
public record Person(Long id, String name) {}
```

Generates `PersonBuilder` at compile-time

```java
@Test
public void buildsPerson() {
    var person = PersonBuilder.builder()
        .id(123L)
        .name("Cédric")
        .build();
    assertEquals("Cédric", person.name());
    assertEquals(123L, person.id());
}
```

Graal Cloud Native (GCN)

What is Graal Cloud Native?



CLOUD-AGNOSTIC MODULES

A curated set of Micronaut modules that provide platform-independent support for core cloud services – object store, secrets, streaming, and more.

BUILT FOR GRAALVM NATIVE IMAGE

Easily compile ahead-of-time with GraalVM Native Image into small executables that start instantly, and use less memory/CPU.

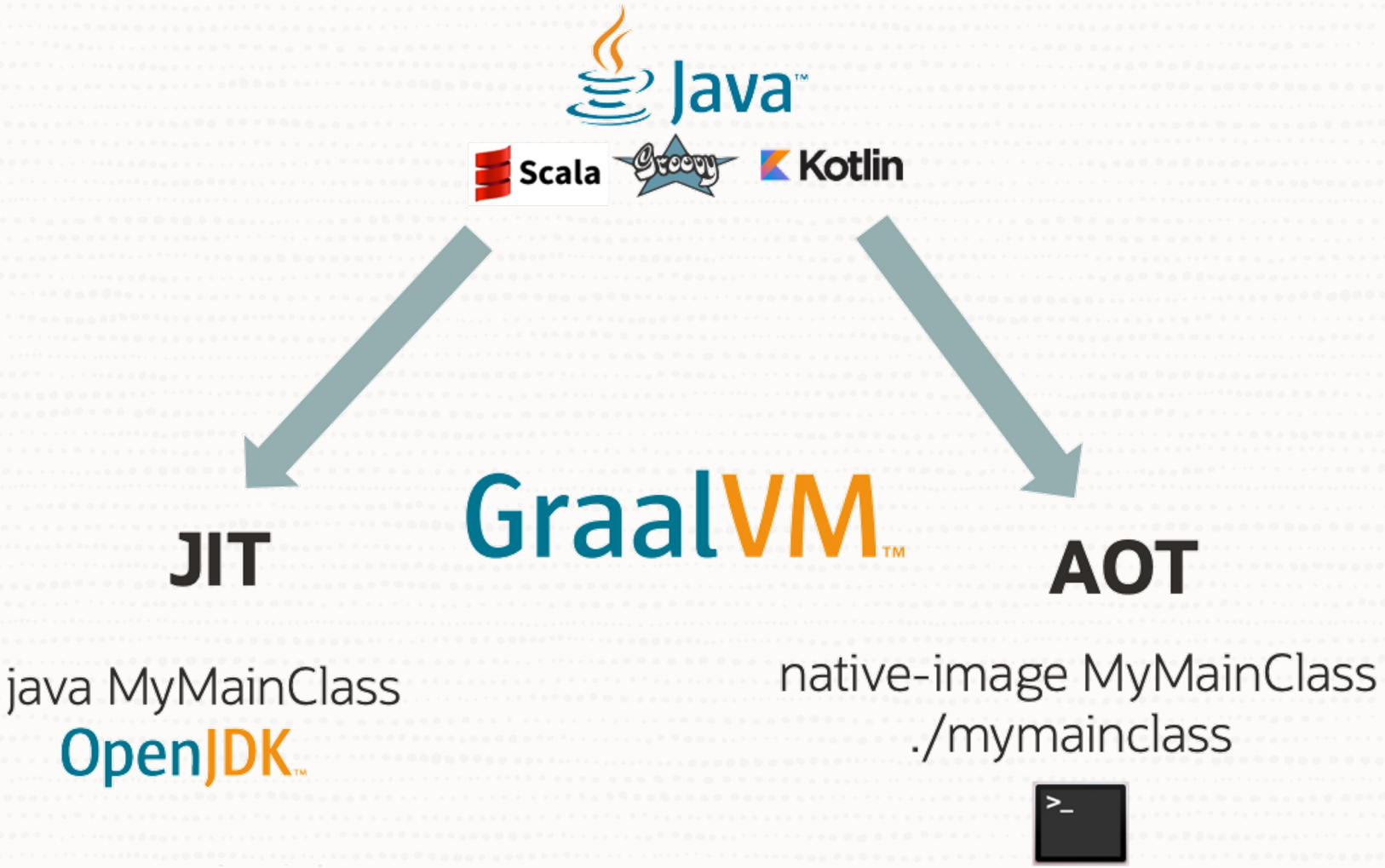
DEV TOOLS AND VS CODE EXTENSIONS

Dev tools to generate multicloud application starter templates. VS Code extensions for local development and cloud deployment.


Graal Cloud Native
Build portable cloud-native Java microservices that start instantly and use fewer resources to reduce compute costs.
Graal Cloud Native (GCN) is a curated set of open source Micronaut® framework modules designed from the ground up to be compiled ahead-of-time with GraalVM Native Image resulting in native executables ideal for microservices.
Get Started
Multicloud Portable Applications
Graal Cloud Native lets you take full advantage of powerful cloud services without coupling to proprietary platform APIs. The lack of standard APIs across clouds makes it nearly impossible to write portable applications, but with
https://graal.cloud

GraalVM™

What is GraalVM

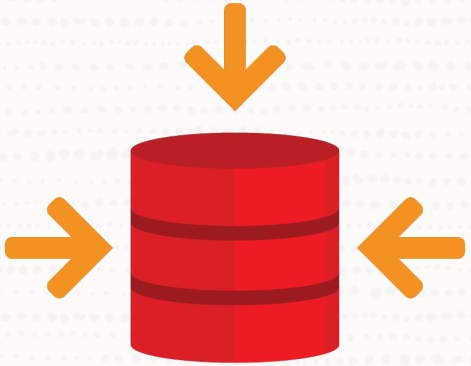


Native Image Goals

Start Fast

Low Resource Usage

Minimize Vulnerability



Compact Packaging

Micronaut and GraalVM

As low as

6ms

startup time

Consuming

18MB

memory footprint

Increased throughput up to

>30%

and reduced latency

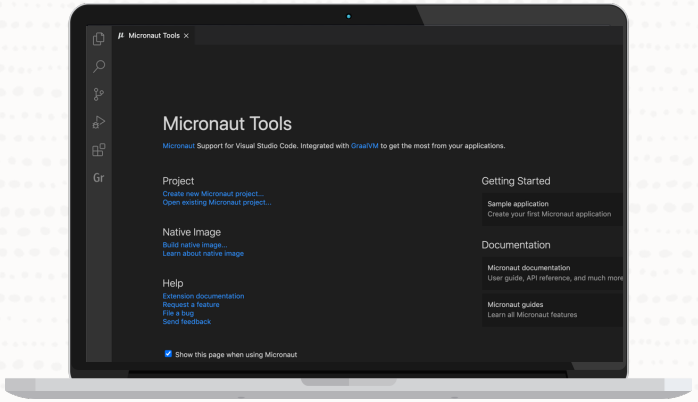
Micronaut and GraalVM

1. The best integration possibly, supported by teams working together at Oracle Labs.
   - GraalVM EE includes optimisations for Micronaut for increased performance and throughput.
   - GraalVM EE license included in Oracle Cloud.

2. Micronaut is ready for GraalVM Native Image since day 1.
   - No reflection, no runtime proxies, no bytecode generation, no dynamic classloading.

3. GraalVM Extenstion Pack for Visual Studio Code.

GraalVM™
Extension Pack



GraalVM Reachability Metadata Repository

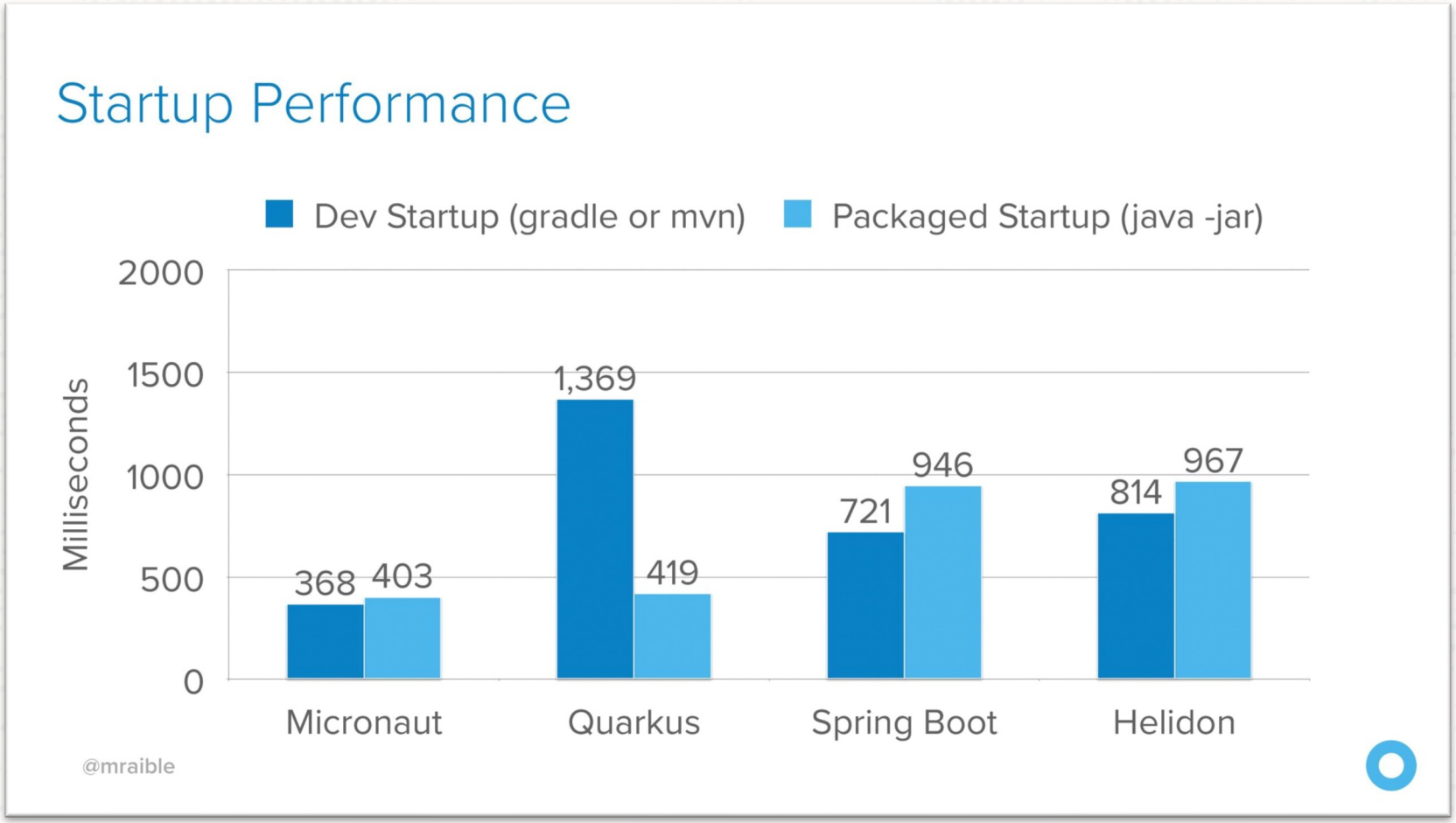
Community-driven collection of GraalVM reachability metadata for open-source libraries.

- Actively maintained by a collective effort of VMware, Oracle and others.
- Enabled in GraalVM Native Build Tools (Maven and Gradle plugins).
- Integrated by default in Spring Boot and Micronaut.


```
# Maven
$ mvn -Pnative native:compile

# Gradle
$ gradle nativeCompile
```

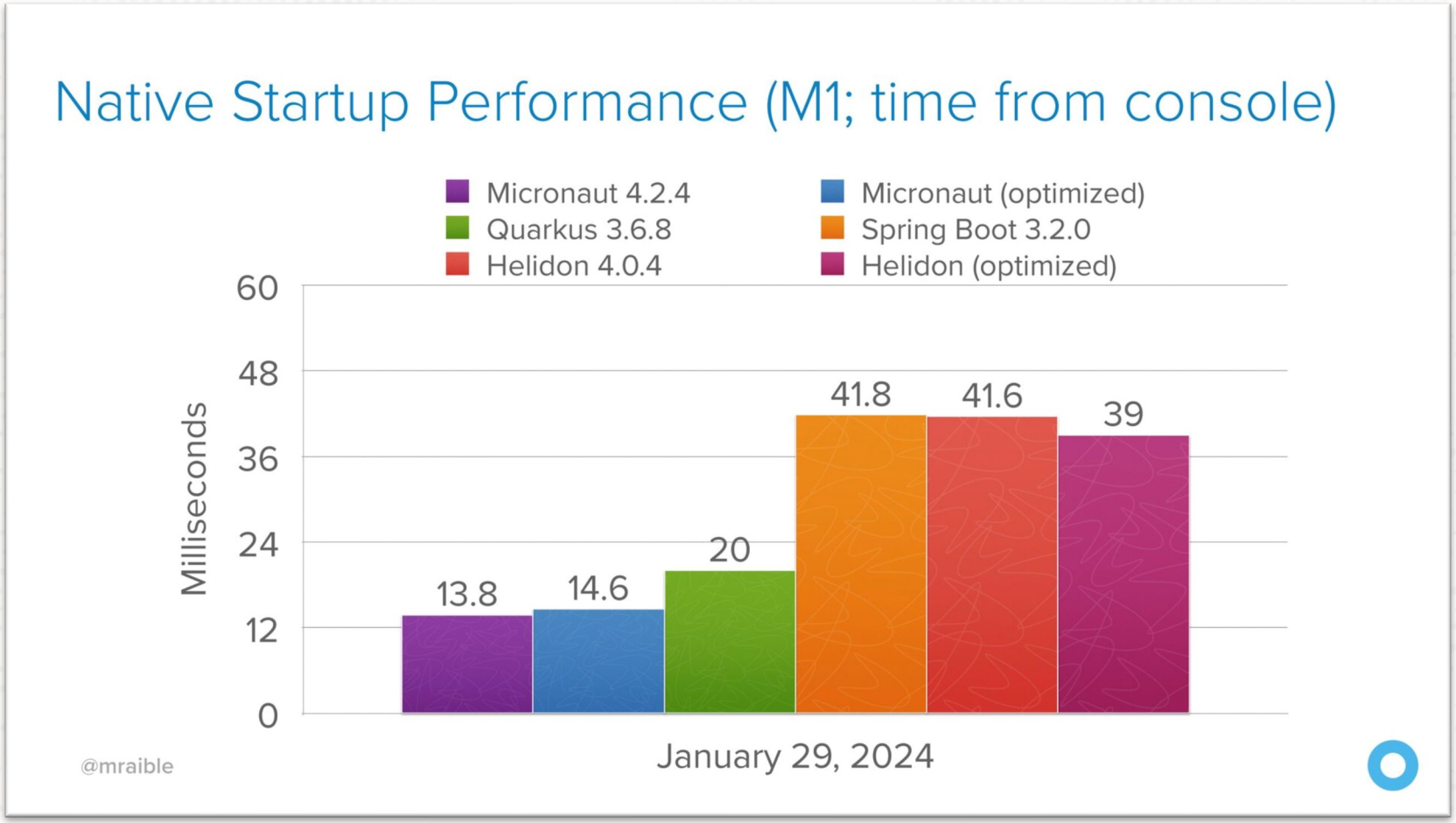
```
# Maven
$ mvn package -Dpackaging=native-image

# Gradle
$ gradle nativeCompile
```

Micronaut is the fastest to startup



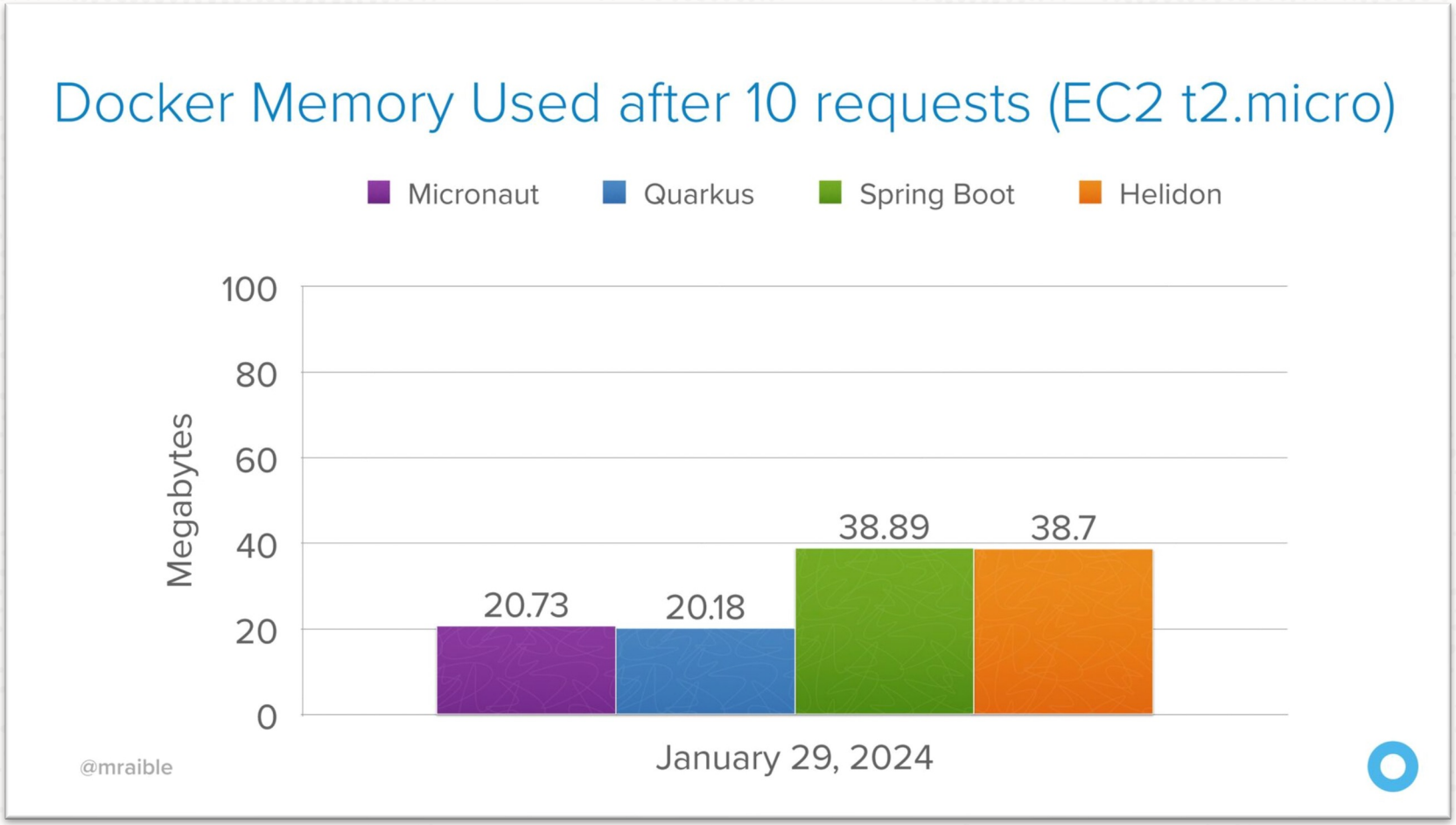
Source: https://speakerdeck.com/mraible/comparing-native-java-rest-api-frameworks-jchampions-conference-2024

Micronaut is the fastest to startup



Source: https://speakerdeck.com/mraible/comparing-native-java-rest-api-frameworks-jchampions-conference-2024

Micronaut consumes the less memory



Source: https://speakerdeck.com/mraible/comparing-native-java-rest-api-frameworks-jchampions-conference-2024

Micronaut and GraalVM powering Disney+

https://aws.amazon.com/blogs/opensource/improving-developer-productivity-at-disney-with-serverless-and-open-source/

Demo

Micronaut and GraalVM

Q &A


Álvaro Sánchez-Mariscal
@alvaro_sanchez
Thank you
ConFoo.CA
DEVELOPER CONFERENCE


ORACLE