QUARKUS

Supersonic. Subatomic. Java.

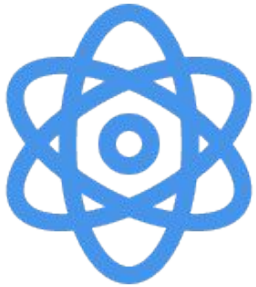
JBug Korea, Seoul, June 2019

Agenda - What you will get out of this session?

- You will learn What is Quarkus
  - Designed from ground-upCloud Native
  - Performance, footprint
- You will learn Why Quarkus
  - Use cases
  - How Quarkus can help you develop faster (Live reload, ..)
- You will learn How to use it.
  - Demo

A stack to write Java apps

Cloud Native,



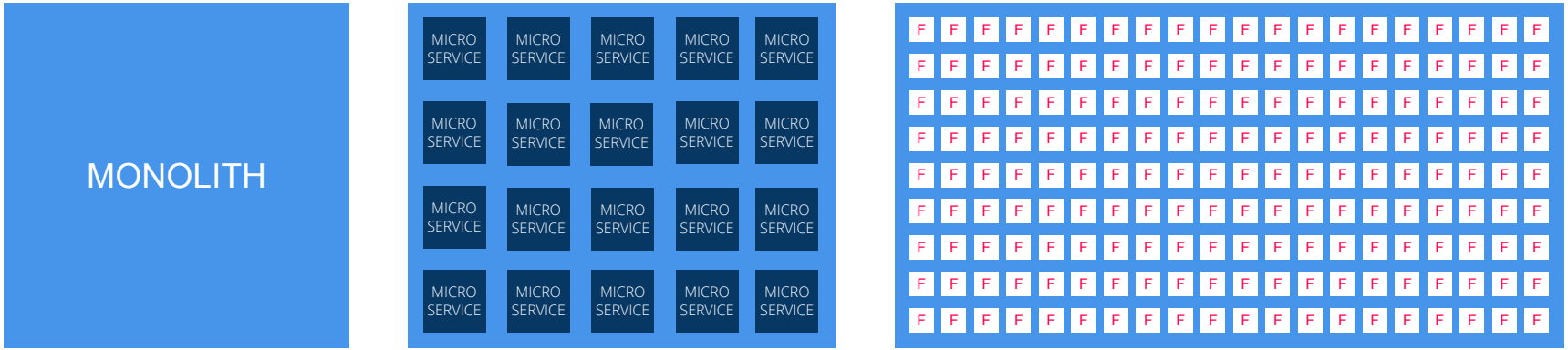
Microservices,

Serverless

DEMO

WHY QUARKUS?

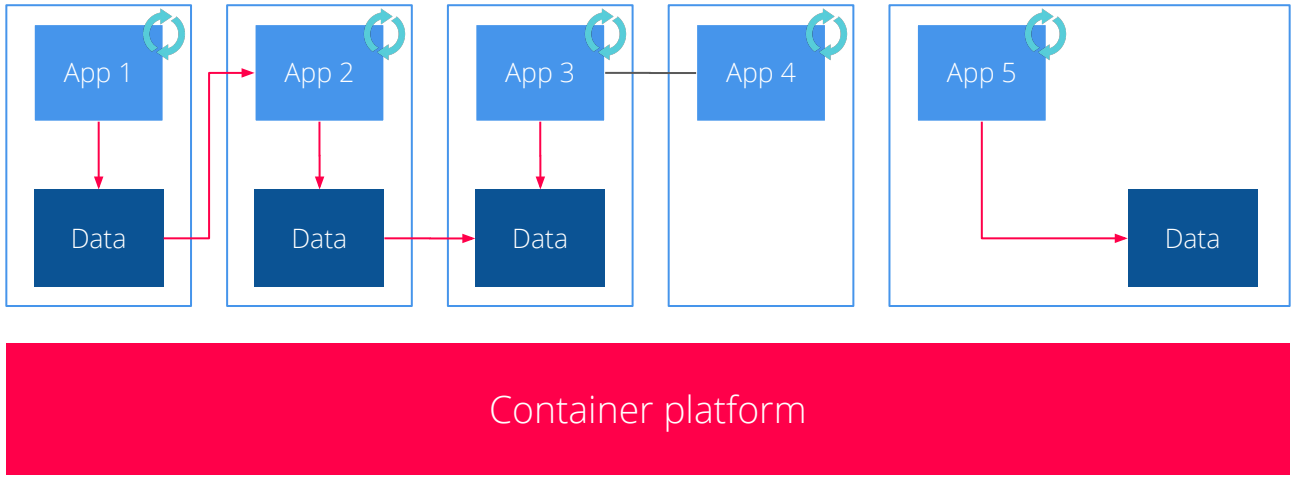
From monolith to...



- 1 monolith ≈ 20 microservices ≈ 200 functions
- Scale to 1 vs scale to 0
- Start up time

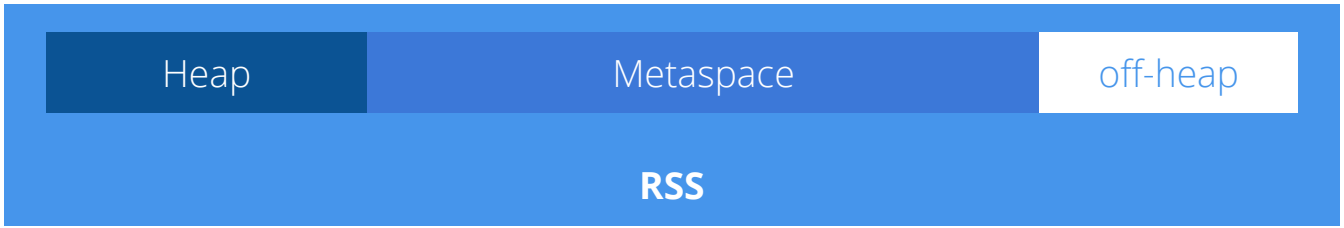
Why go through this pain?

Agility, Scalability, Faster Business Reactivity

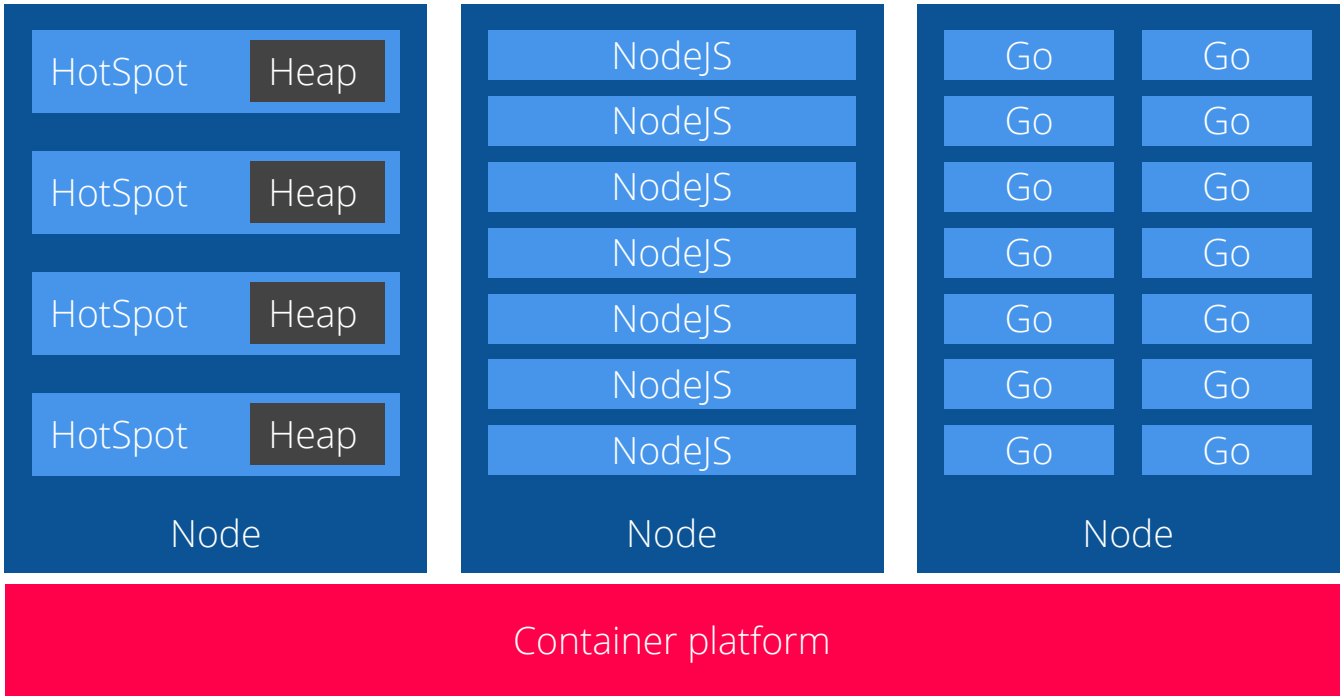


The hidden truth about Java + containers

- Startup overhead
  - # of classes, bytecode, JIT
- Memory overhead
  - # of classes, metadata, compilation



The hidden truth about Java + containers



Java on cloud native: a Red Hat journey

Standards

- Java EE streamlining
- Eclipse MicroProfile

Runtimes

- WildFly on OpenShift

Hardware architectures

- Raspberry Pi and Plug Computer
- Android

Ahead of time compilation

- Lead gcj
- Looked at Dalvik, Avian, Excelsior JET

OpenJDK

- Container ergonomics
- JVM metadata reduction

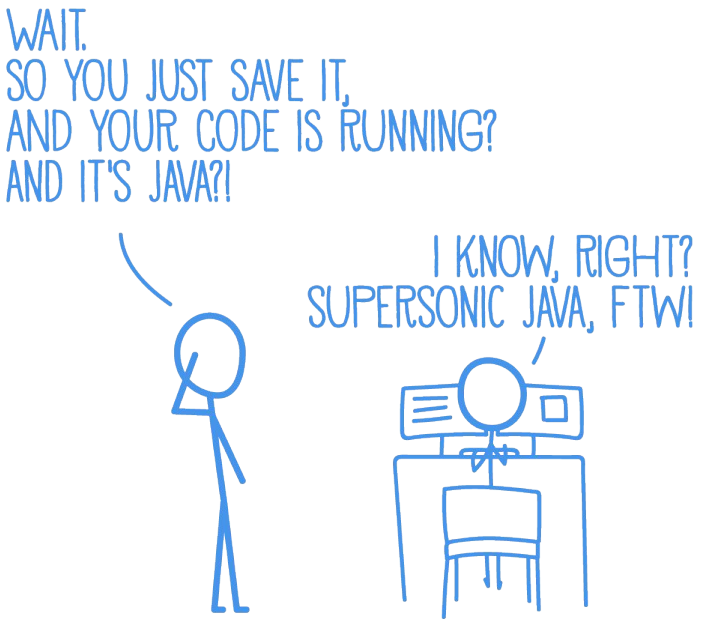
WHAT IS QUARKUS?

QUARK: elementary particle / **US**: hardest thing in computer science

Benefit No. 1: Developer Joy

A cohesive platform for optimized developer joy:

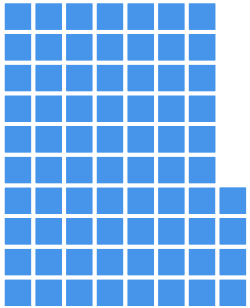
- Based on standards, but not limited
- Unified configuration
- Zero config, live reload in the blink of an eye
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation



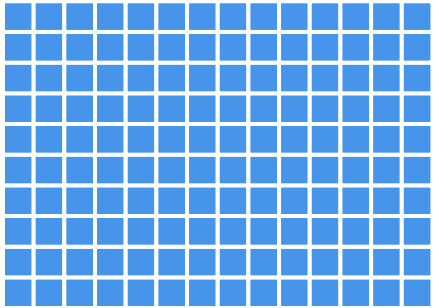
Benefit No. 2: Supersonic Subatomic Java

REST

Quarkus + GraalVM
13 MB



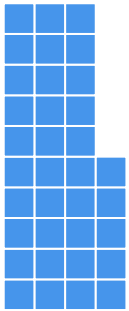
Quarkus + OpenJDK
74 MB



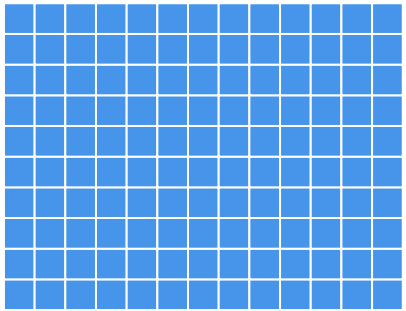
Traditional Cloud-Native Stack
140 MB

Benefit No. 2: Supersonic Subatomic Java

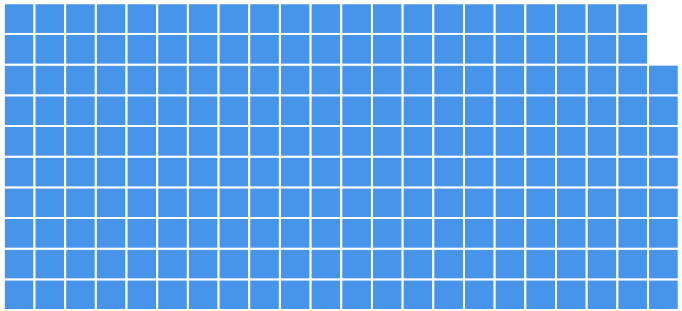
REST + CRUD



Quarkus + GraalVM
35 MB

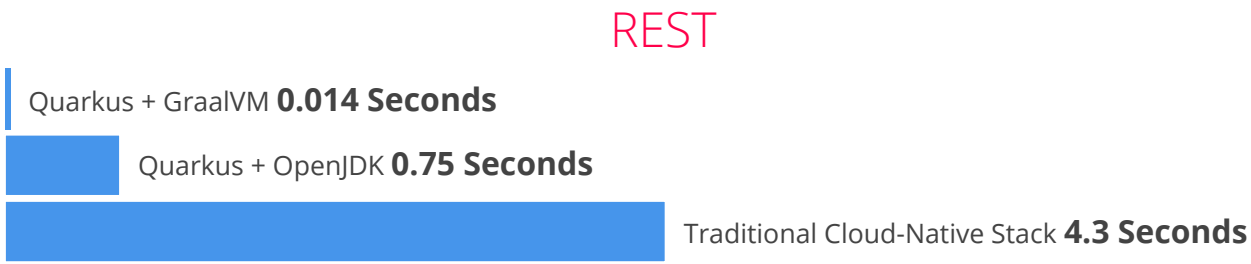


Quarkus + OpenJDK
130 MB



Traditional Cloud-Native Stack
218 MB

Benefit No. 2: Supersonic Subatomic Java



Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
      return say.hello();
}
```

```
@Inject @Stream("kafka")
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
      return reactiveSay;
}
```

- Combine both Reactive and imperative development in the same application
- Use the technology that fits your use-case
- Key for reactive systems based on event driven apps

Benefit No. 4: Best of Breed Frameworks & Standards

Eclipse Vert.x

Hibernate

RESTEasy

Apache Camel

Eclipse MicroProfile

Netty

Kubernetes

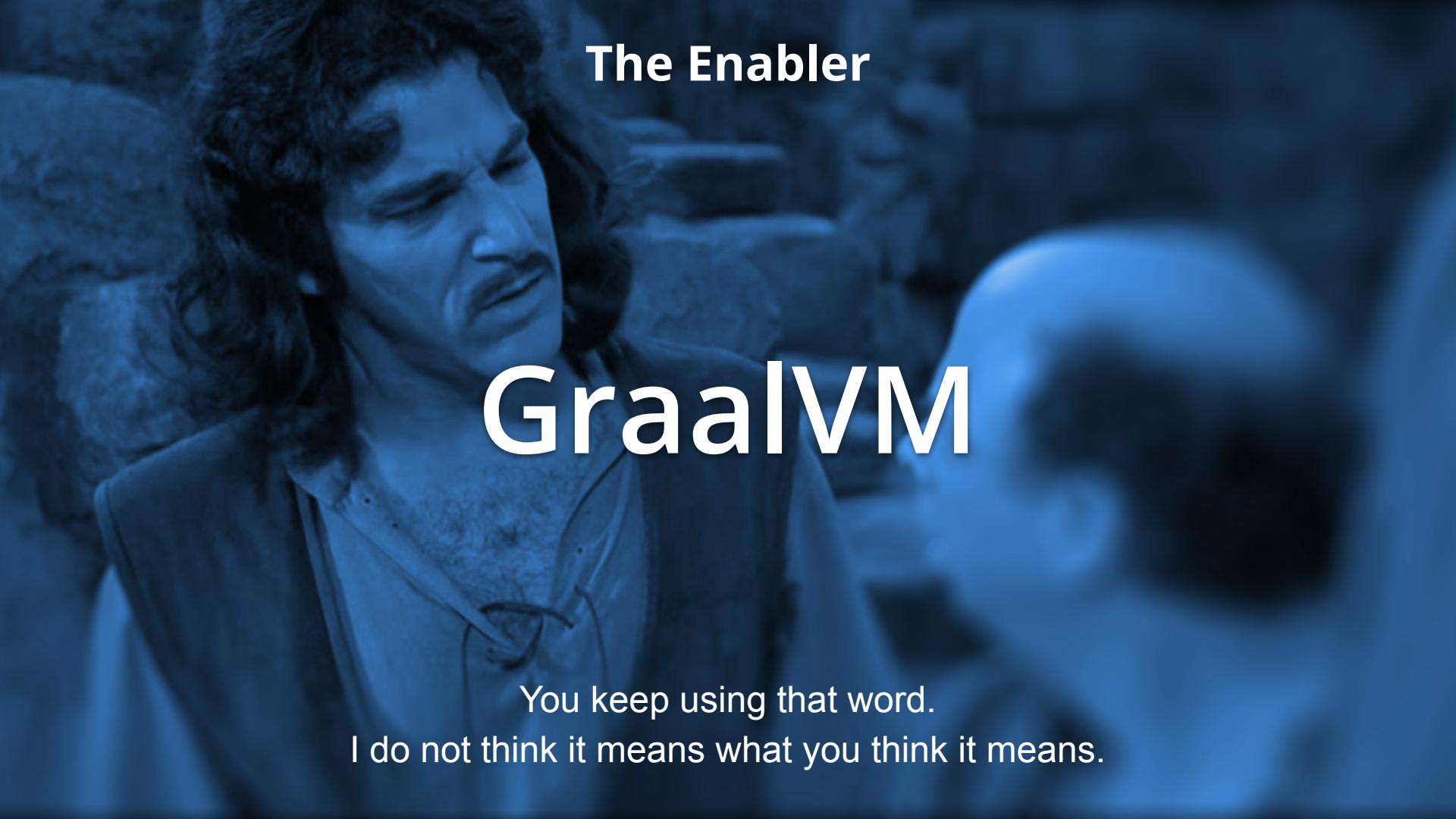
OpenShift

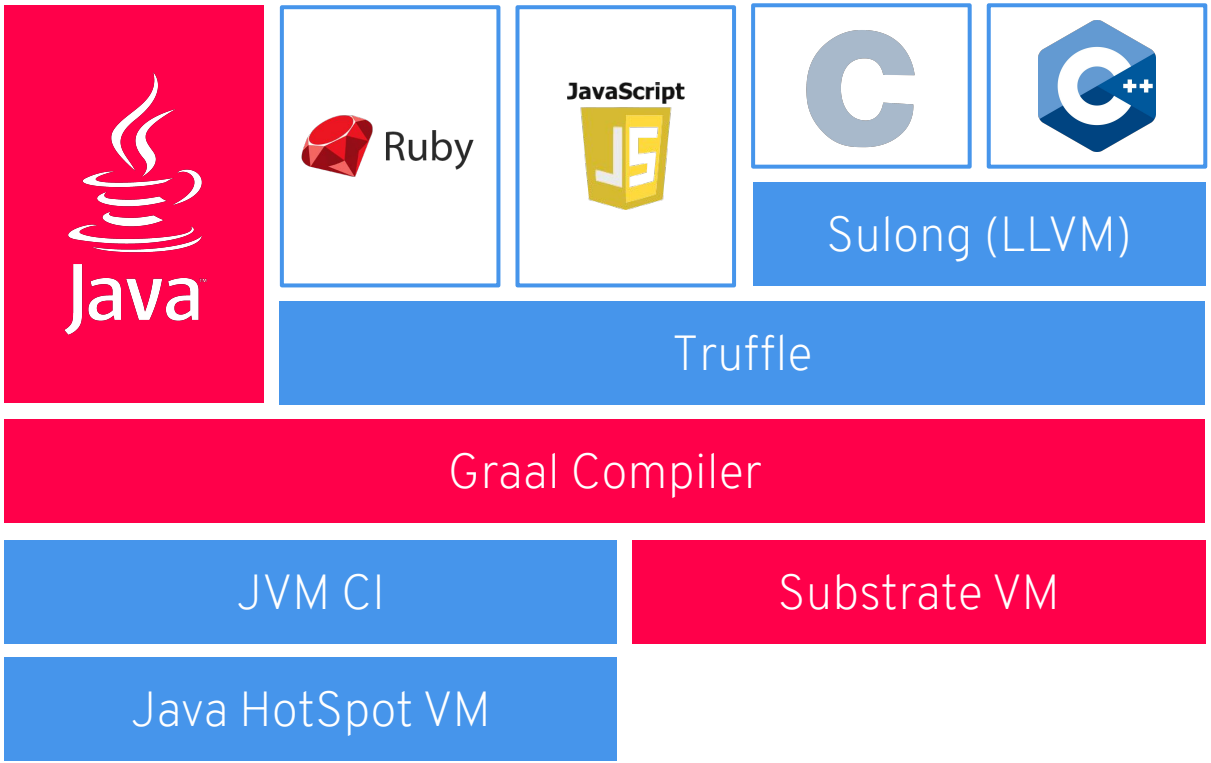
Jaeger

Prometheus

Apache Kafka

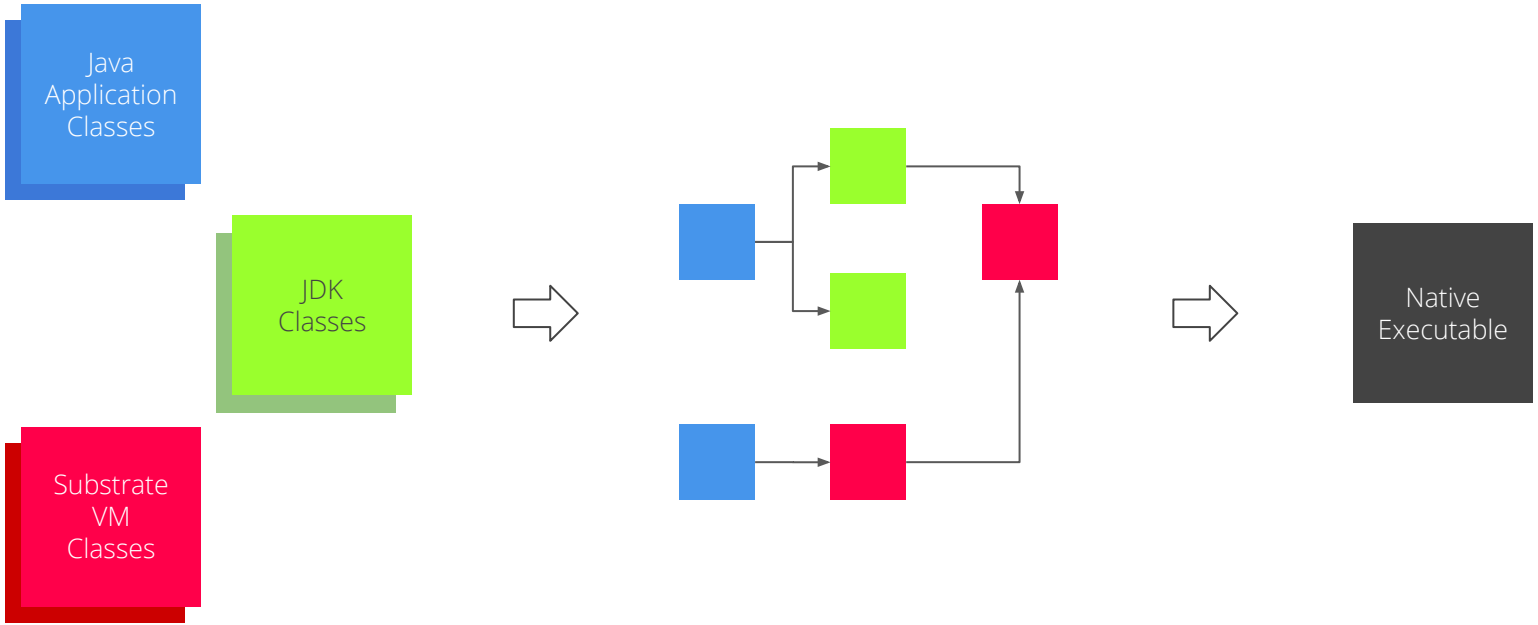
Infinispan


The Enabler
GraalVM
You keep using that word.
I do not think it means what you think it means.


Java
Ruby
JavaScript
Sulong (LLVM)
Truffle
Graal Compiler
JVM CI
Substrate VM
Java HotSpot VM

Dead code elimination

Closed-world assumption



The Dark Side

Not supported

- Dynamic classloading
- InvokeDynamic & Method handles
- Finalizer
- Security manager
- JVMTI, JMX, native VM Interfaces

OK with caveats in usage

- Reflection (manual list)
- Dynamic proxy (manual list)
- JNI (manual list)
- Static initializers
- Lambda, Threads (OK, pfff!)
- References (similar)

The Dark Side

The Good Parts

Not supported

- Dynamic classloading
- InvokeDynamic & Method handles
- Finalizer
- Security manager
- JVMTI, JMX, native VM Interfaces

OK with caveats in usage

- Reflection (manual list)
- Dynamic proxy (manual list)
- JNI (manual list)
- Static initializers
- Lambda, Threads (OK, pfff!)
- References (similar)

HOW QUARKUS WORKS

Move startup time to build time

What does a framework do at startup time

- Parse config files
- Classpath & classes scanning
  - for annotations, getters or other metadata
- Build framework metamodel objects
- Prepare reflection and build proxies
- *Start and open IO, threads etc*

Build time benefits

Do the work once, not at each start

All the bootstrap classes are no longer loaded

Less time to start, less memory used

Less or no reflection nor dynamic proxy

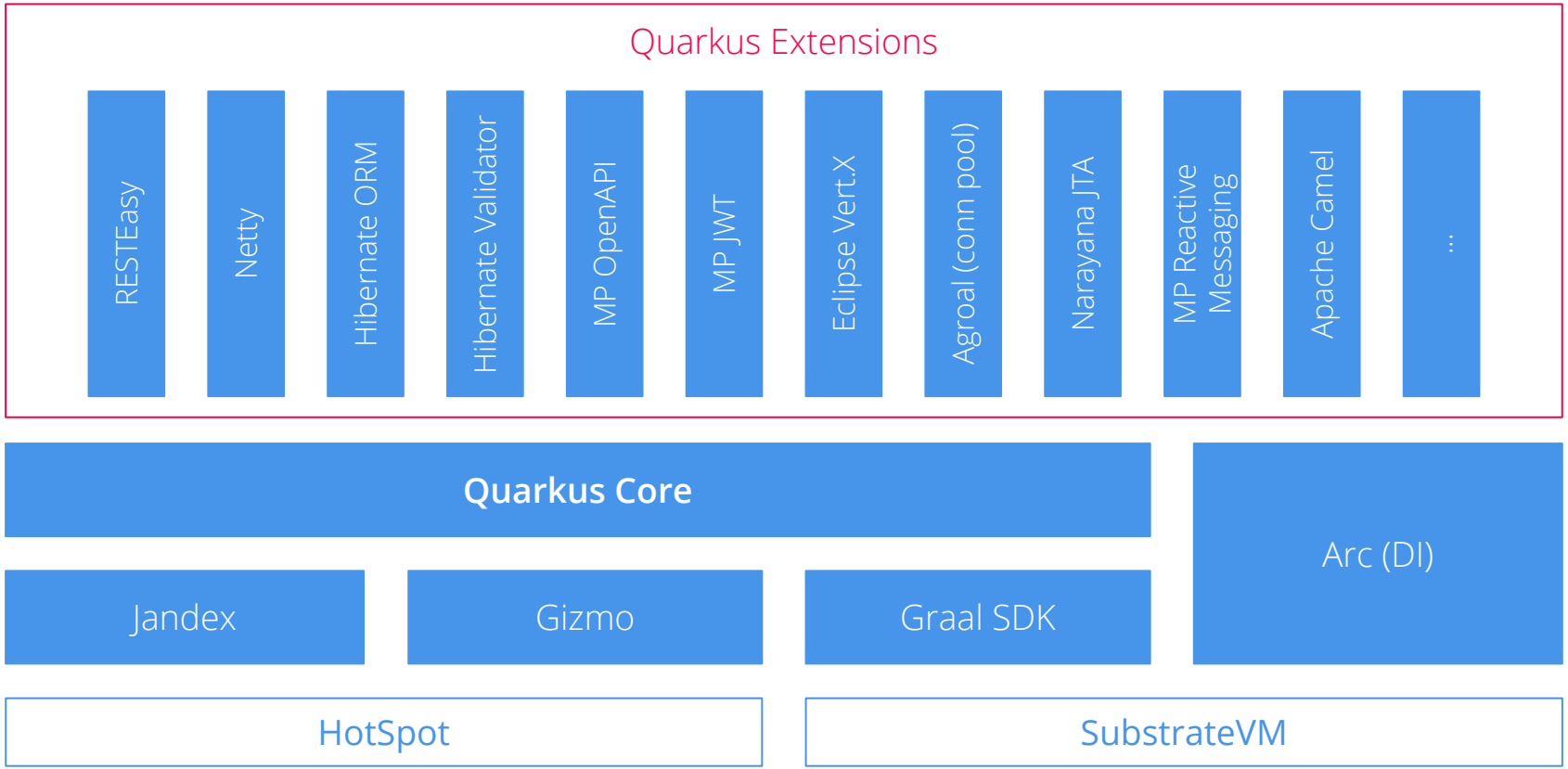
More Build time benefits

Drives the gathering of metadata needed by GraalVM

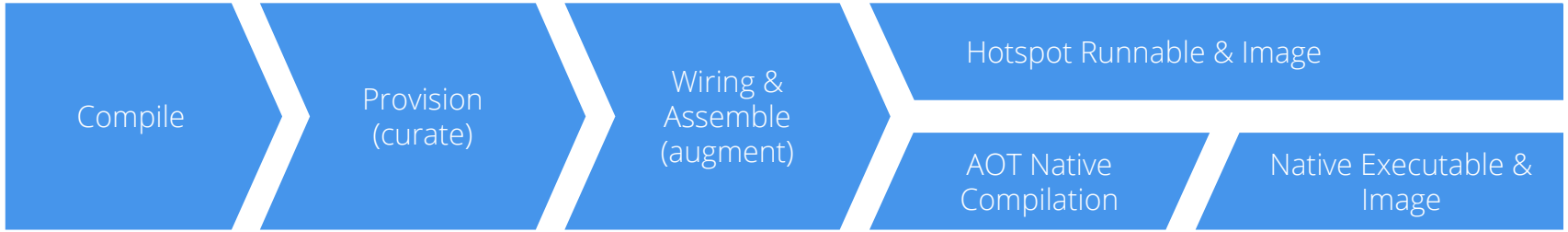
- based on framework knowledge
- Classes using reflection, resources, etc

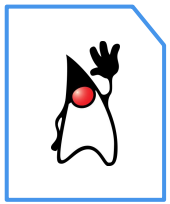
Minimize dependencies

Help dead code elimination

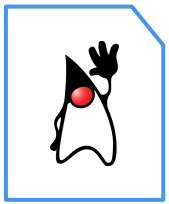

Quarkus Extensions
RESTEasy
Netty
Hibernate ORM
Hibernate Validator
MP OpenAPI
MP JWT
Eclipse Vert.X
Agroal (conn pool)
Narayana JTA
MP Reactive Messaging
Apache Camel
...
Quarkus Core
Arc (DI)
Jandex
Gizmo
Graal SDK
HotSpot
SubstrateVM

Build Process





app.jar



frameworks

Runnable java app

native-app

MORE QUARKUS

More is better

Testing is running

```
@QuarkusTest
public class HelloResourceTest {

    @Inject HelloService service;

    @Test
    public void testHelloEndpoint() {
        assertEquals(
           "Hello Quarkus",
          service.greeting("Quarkus")
          );
    }
}
```

Fast start

Full start

Injection

Define entity with Panache

```java
@Entity
public class Todo extends PanacheEntity {
   // id is inherited
   public String title;
   public boolean completed;
   public String url;

   @Column(name = "ordering")
   public int order;

   public static List<Todo> search(String query) {
       return list("completed = ?1 and title like ?2", false, query);
   }

   public static List<Todo> findNotCompleted() {
       return list("completed", false);
   }

   public static long deleteCompleted() {
       return delete("completed", true);
   }
}
```

Accessing entities with Panache

Convenient methods for lazy people

(aka good devs)

```
@GET
public List<Todo> getAll() {
   return Todo.listAll(Sort.by("order"));
}

@POST @Transactional
public void addTodo(@Valid Todo todo) {
   return todo.persist();
}

@GET @Path("/search/{query}")
public List<Todo> search(@PathParam("query") @NotBlank String query) {
   return Todo.search(query);
}
```

More Hibernate with Panache

Pagination

Sorting

ActiveRecord or Repository pattern

```java
@Entity
public class Todo extends PanacheEntity {
   // id is inherited
   public String title;
   public boolean completed;
   public String url;

   public static List<Todo> findNotCompleted() {
       return list("completed", false);
   }
}

@Path("/api")
public class TodoResource {
   @GET
   public List<Todo> getAll() {
      return Todo.listAll(Sort.by("order"));
   }
}
```

```java
@Entity
public class Todo {
   @Id @GeneratedValue public Long id;
   public String title;
   public boolean completed;
   public String url;
}

@ApplicationScoped
public class TodoRepo extends
      PanacheRepository<Person> {
   public List<Todo> findNotCompleted() {
       return list("completed", false);
   }
}

@Path("/api")
public class TodoResource {
   @Inject TodoRepo repo;
   @GET
   public List<Todo> getAll() {
      return repo.listAll(Sort.by("order"));
   }
}
```

Observability

Open API and Swagger UI

Metrics

Health

Flyway

Kafka

Security: JWT, Keycloak...

```
@Path(“api”) @Produces(MediaType.TEXT_PLAIN)
public class UsersResource {

   @PermitAll
   @NoCache
   public String publicStuff() {
       return “Hello world”;
   }

   @GET @Path("/admin")
   @RolesAllowed("admin")
   public String admin() {
       return "Secret handshake baby!";
   }
}
```

JWT

Keycloak

etc

<do your own extension>

FUTURE

The best way to predict it is to invent it

Developer Joy and Cloud Native

Developer Joy

- Start from nothing
  - Download CLI and go
  - Download Image
- Remote dev mode
- S2I chains
- Built in tooling to deploy and setup OpenShift

Cloud Native

- Easy to use client APIs for
  - RDG service
  - RH-SSO service
  - AMQ Streams
- MicroProfile
- Reactive

Foundation Layer and Serverless

Serverless

- Function framework
- Tool chain to help deploy functions

Foundation Layer

- Foundational layer
- “The Application Environment”
- Camel-K
- Automation
  - Project Submarine
- Infinispan
- Keycloak

Quarkus Benefits

Developer Joy

Supersonic Subatomic Java

Unifies

imperative and reactive

Best of breed

libraries and standards

Thank you.

QUARKUS

https://quarkus.io

https://quarkusio.zulipchat.com

@quarkusio