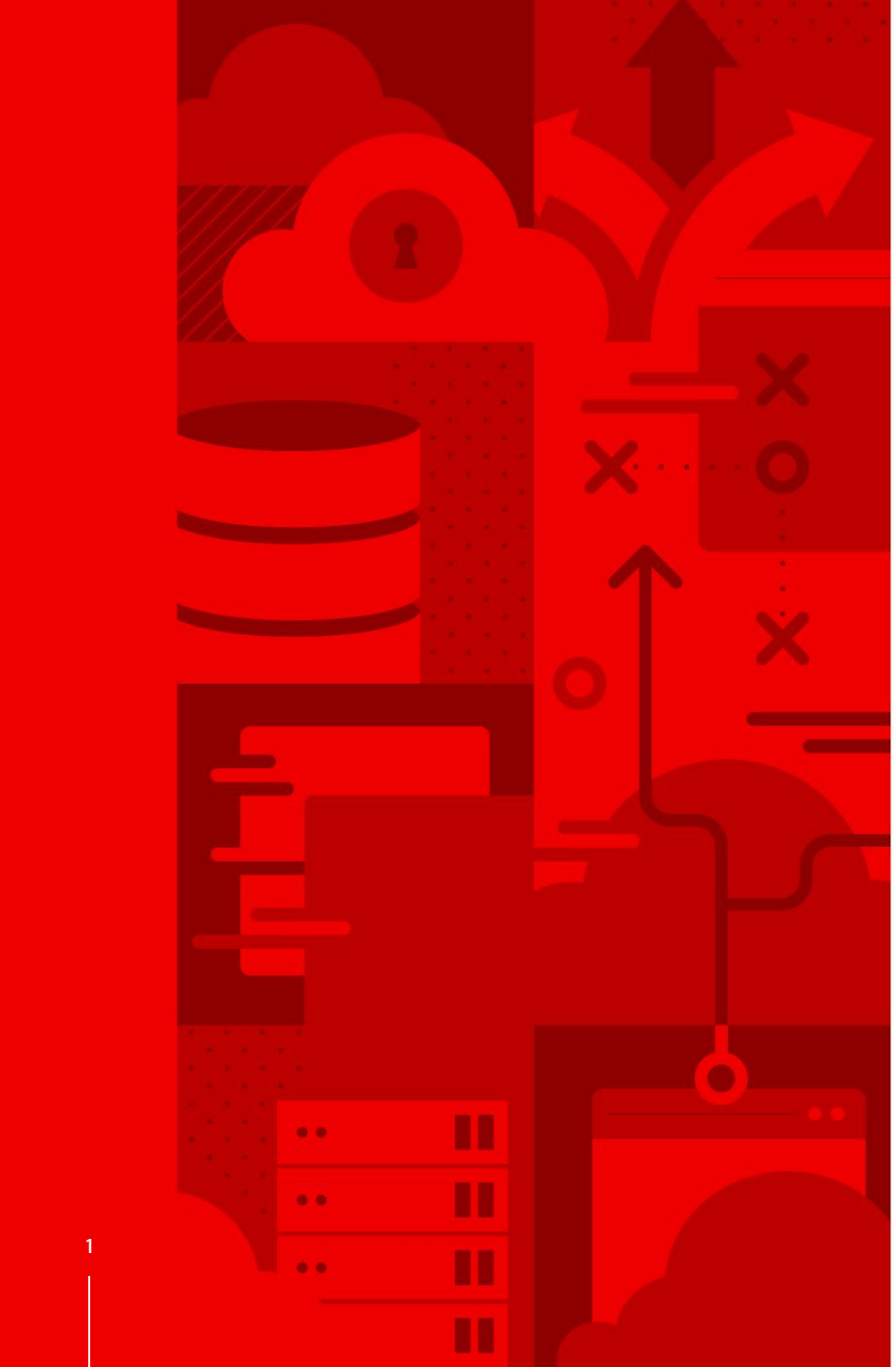


What every Java developer needs to know about serverless

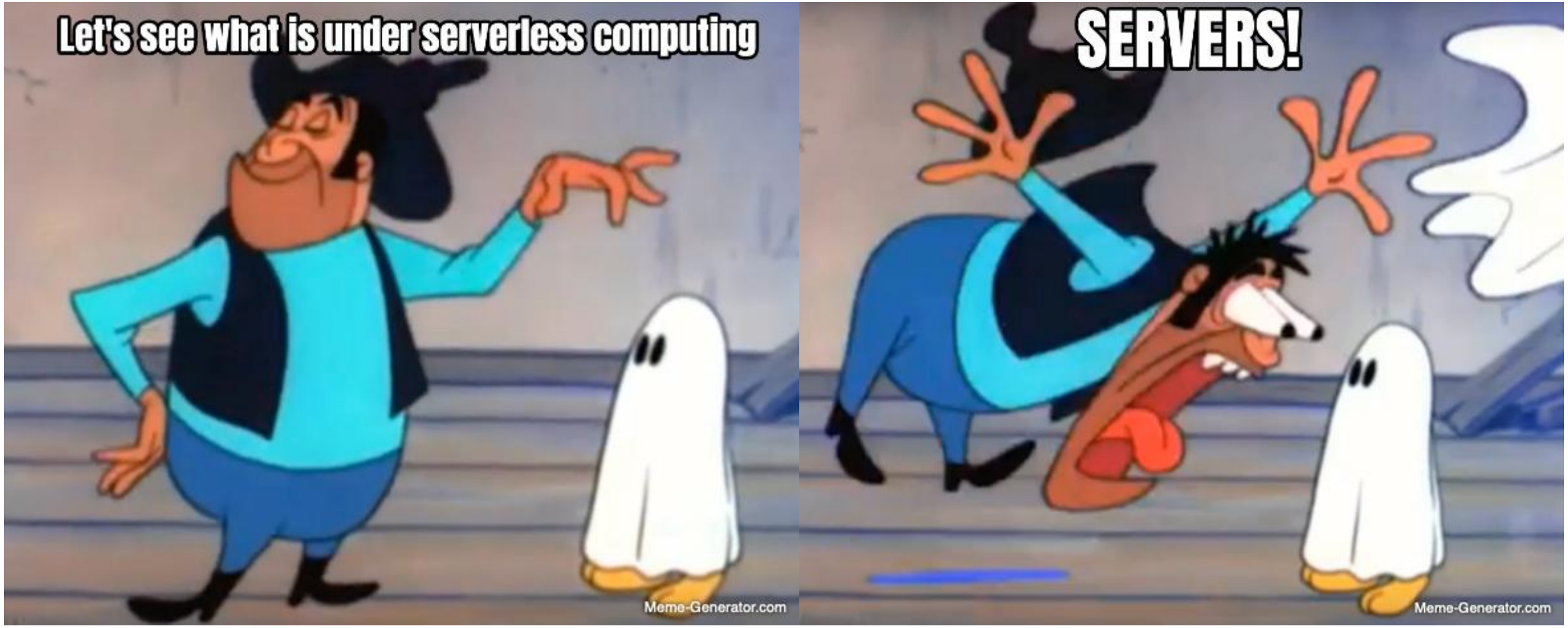
Helber Belmiro
Open Source Software Engineer
Working on Kogito and CNCF Serverless Workflow

Red Hat

What we'll discuss today

- What is serverless computing?
- Why should you care?
- What about Java?
- Evolution
- Tools
- Demo

Red Hat


Let's see what is under serverless computing
Meme-Generator.com
SERVERS!
Meme-Generator.com

CLOUD NATIVE
COMPUTING FOUNDATION

> Building and running applications that do not require server management. A finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

Cloud Native Computing Foundation
Serverless Working Group

Source:
https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview

Two personas

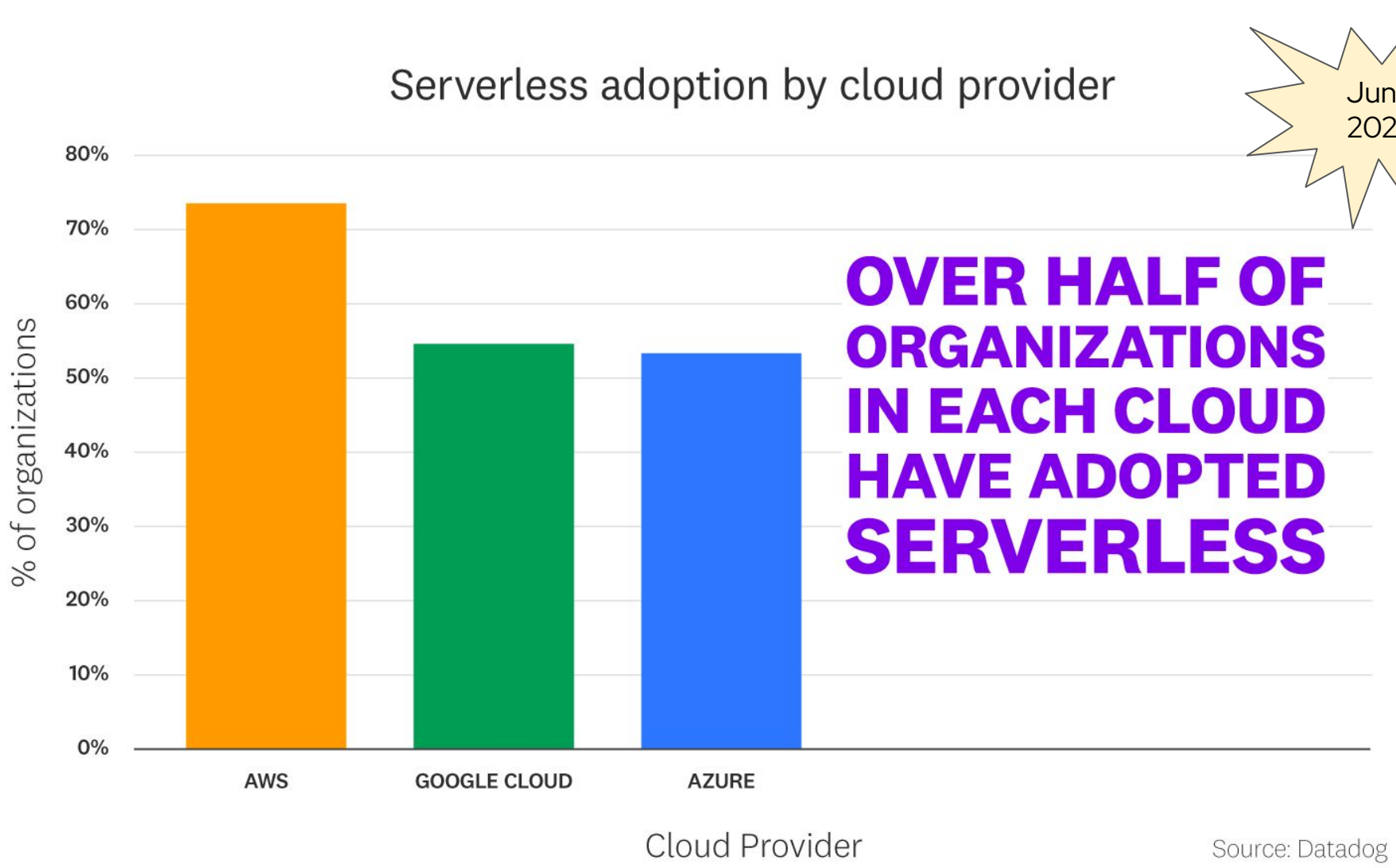
1. **Developer** - writes code for, and benefits from the serverless platform which provides them the point of view that there are no servers nor that their code is always running

2. **Provider** - deploys the serverless platform for an external or internal customer

A serverless platform may provide one or both of the following

1. **Functions-as-a-Service (FaaS)**
   - Typically event-driven computing
   - Functions that are triggered by events or HTTP requests
   - Executed as needed, scaling without the need to manage servers or any other underlying infrastructure

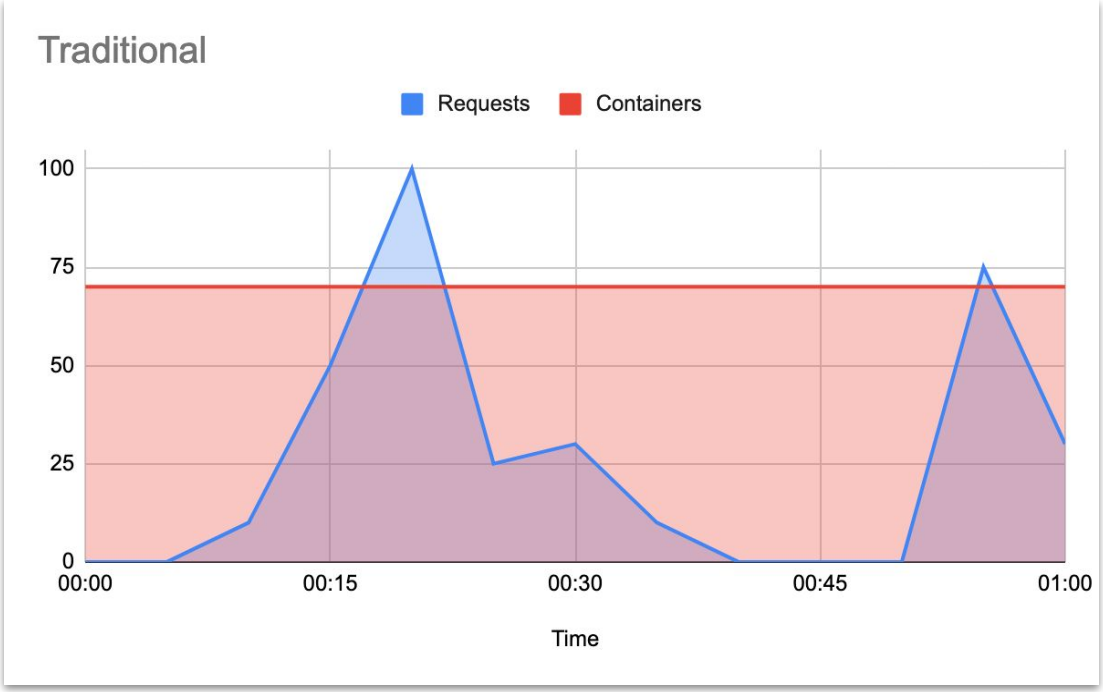
2. **Backend-as-a-Service (BaaS)**
   - Third-party API-based services
   - Replace core subsets of functionality in an application
   - Auto-scales and operates transparently

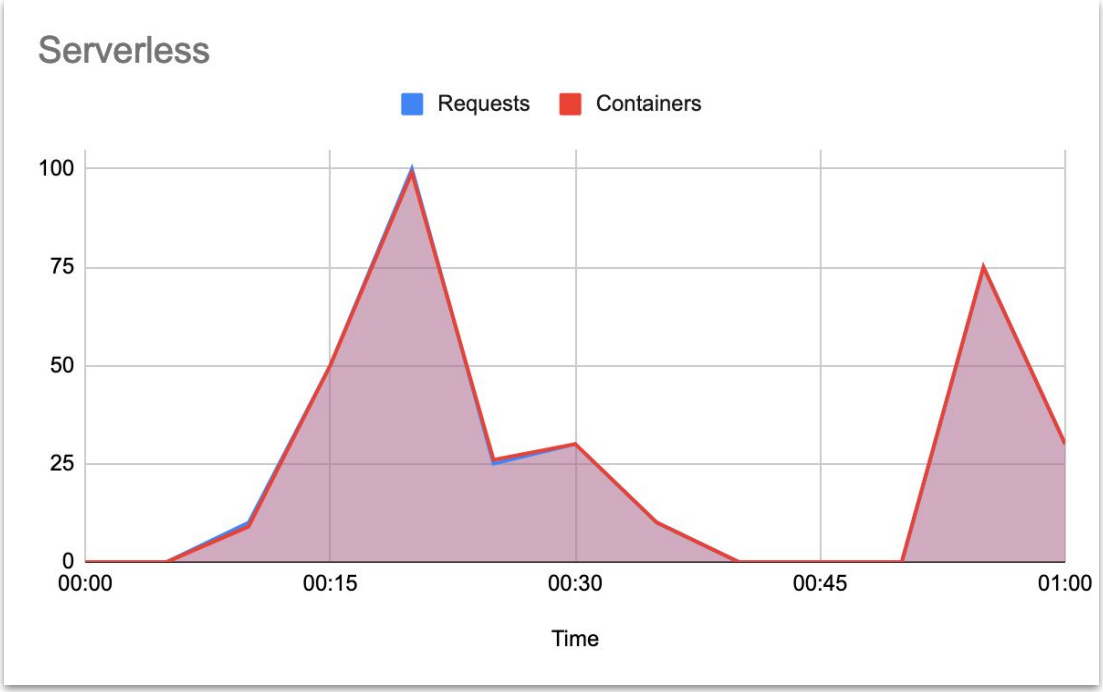


Source:
https://www.datadoghq.com/state-of-serverless/

Benefits

Serverless reduces operational costs





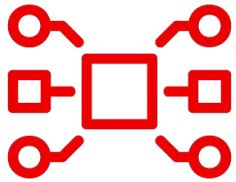
Benefits

Reduces costs

- You pay for compute time as it's needed
- Increases developer productivity

Helps enable DevOps adoption

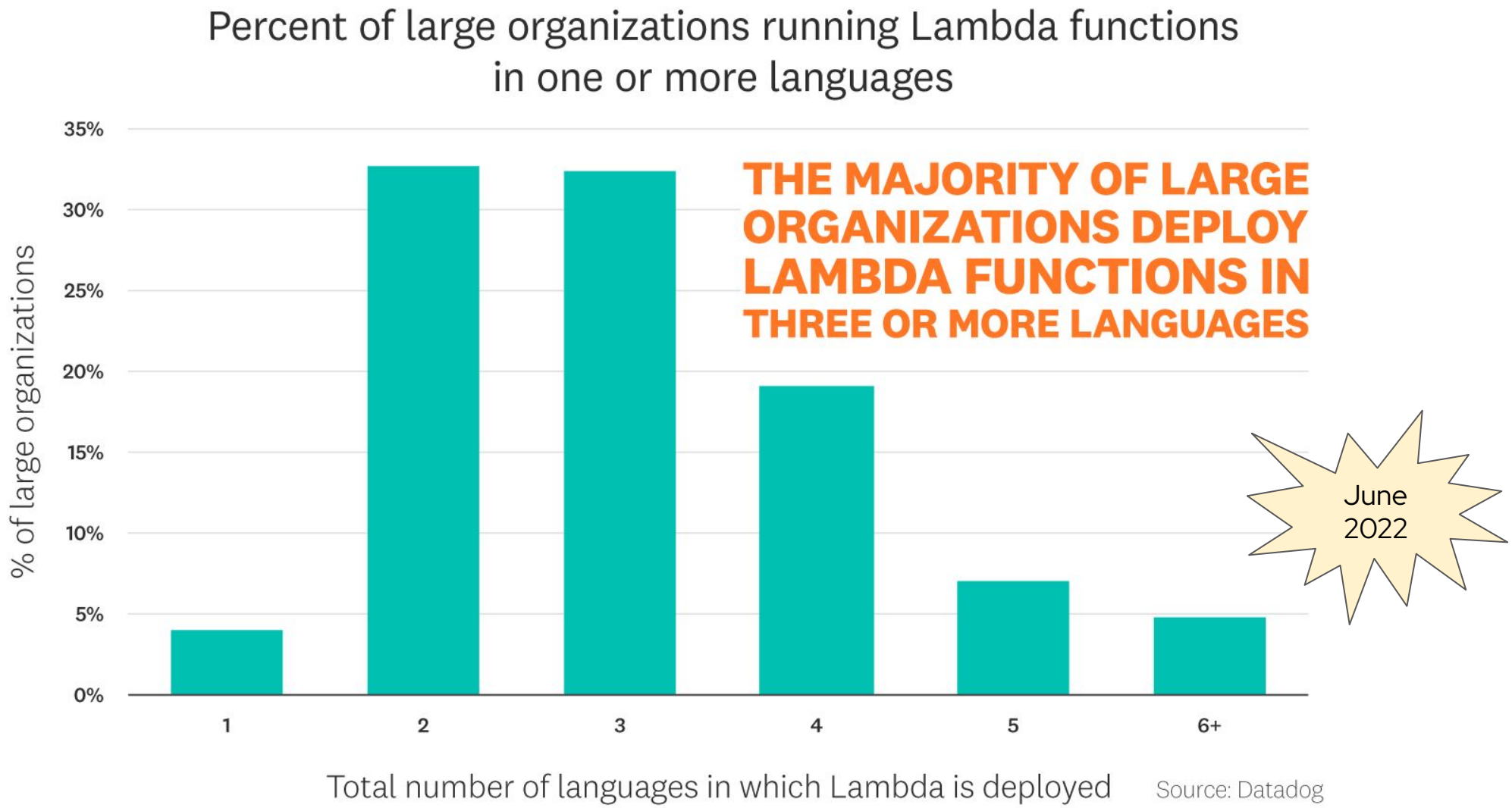
Developers don't need to explicitly describe the infrastructure they need operations to provision for them



Enables agility

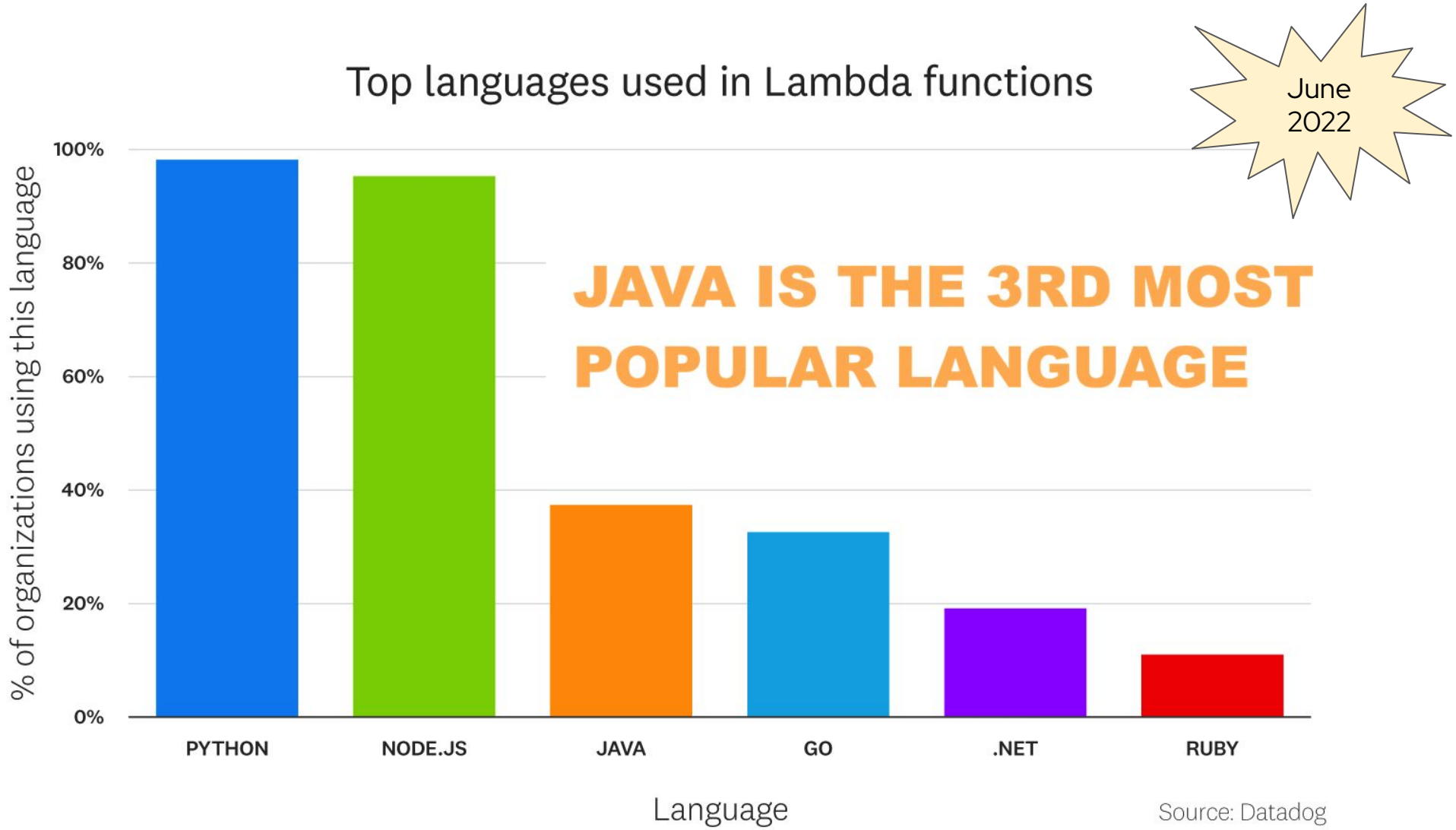
Allows to streamline app development even further by incorporating entire components from third-party BaaS offerings

Sooner or later you'll have to face it



Source:
https://www.datadoghq.com/state-of-serverless/

What about Java?



Source:
https://www.datadoghq.com/state-of-serverless/

Serverless and Java in 2018

Serverless with Java

Serverless with Python and Node.js

FaaS

2015

Serverless 1.0 - FaaS

- AWS Lambda - 2015
- Azure Functions - 2016
- IBM Cloud Functions - 2016
- Google Cloud Functions - 2018

FaaS

2015

Serverless 1.0 - FaaS

```java
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.LambdaLogger;

public class HandlerInteger implements RequestHandler<Integer, Integer> {

  @Override
  public Integer handleRequest(Integer event, Context context) {
    LambdaLogger logger = context.getLogger();

    // process event
    logger.log("EVENT: " + event);
    logger.log("EVENT TYPE: " + event.getClass());

    // return amount of time remaining before timeout
    return context.getRemainingTimeInMillis();
  }
}
```

FaaS

2015

Serverless 1.0 - FaaS

- HTTP and few other sources
- Functions only
- Limited execution time (5-10 minutes)
- Limited local development experience
- Vendor lock-in

FaaS

2015

Serverless 1.0 - FaaS - Java

- Java 8 - 2014
- Designed for throughput
- Designed to be long-running
- **High startup time**
- **Can't scale fast**

FaaS

2015

Serverless 1.0 - FaaS - Java

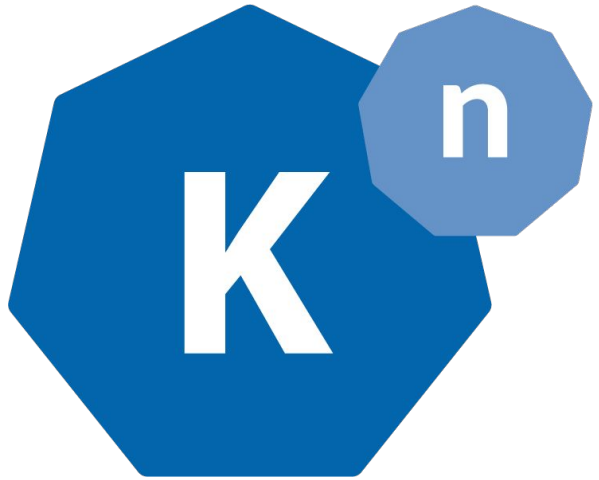
- Java EE 7 - Abandoned by Oracle
- Spring Framework 4 - De facto standard
- Reflection-based
- **High startup time**
- **Can't scale fast**

Serverless 1.5 - Kubernetes

- Frameworks that auto-scale containers
- Managed services that abstract K8s APIs
- Knative 0.1 - Late 2018
- Easy to debug and test locally
- Poliglot
- Portable - No vendor lock-in

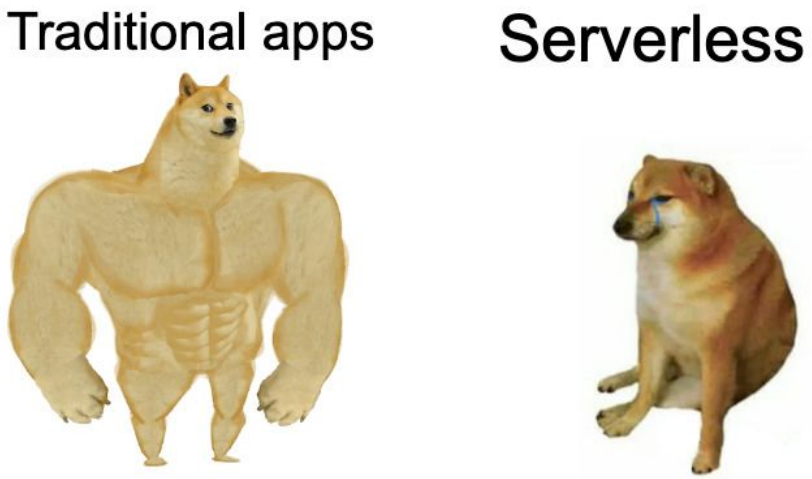
Serverless 1.5 - Knative

- Serverless on top of Kubernetes
- Auto-scale
- Scale to zero
- No vendor lock-in
- Hybrid cloud
- Any language

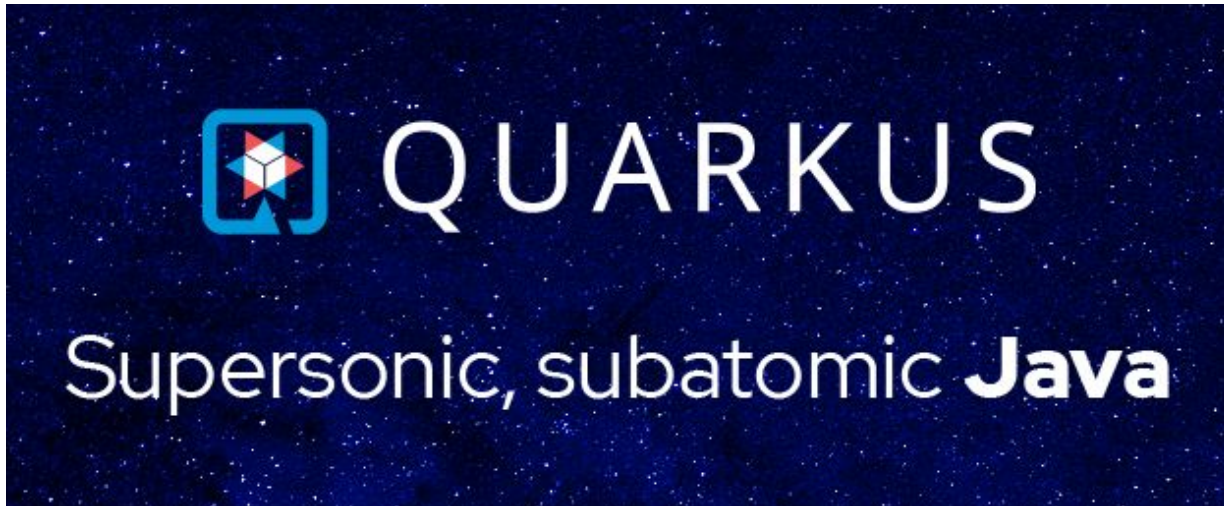


Serverless 1.5 - Java

- Java 11 - 2018 👍
- Better support to containers 👍
- **High startup time** 👎
- **Can't scale fast** 👎

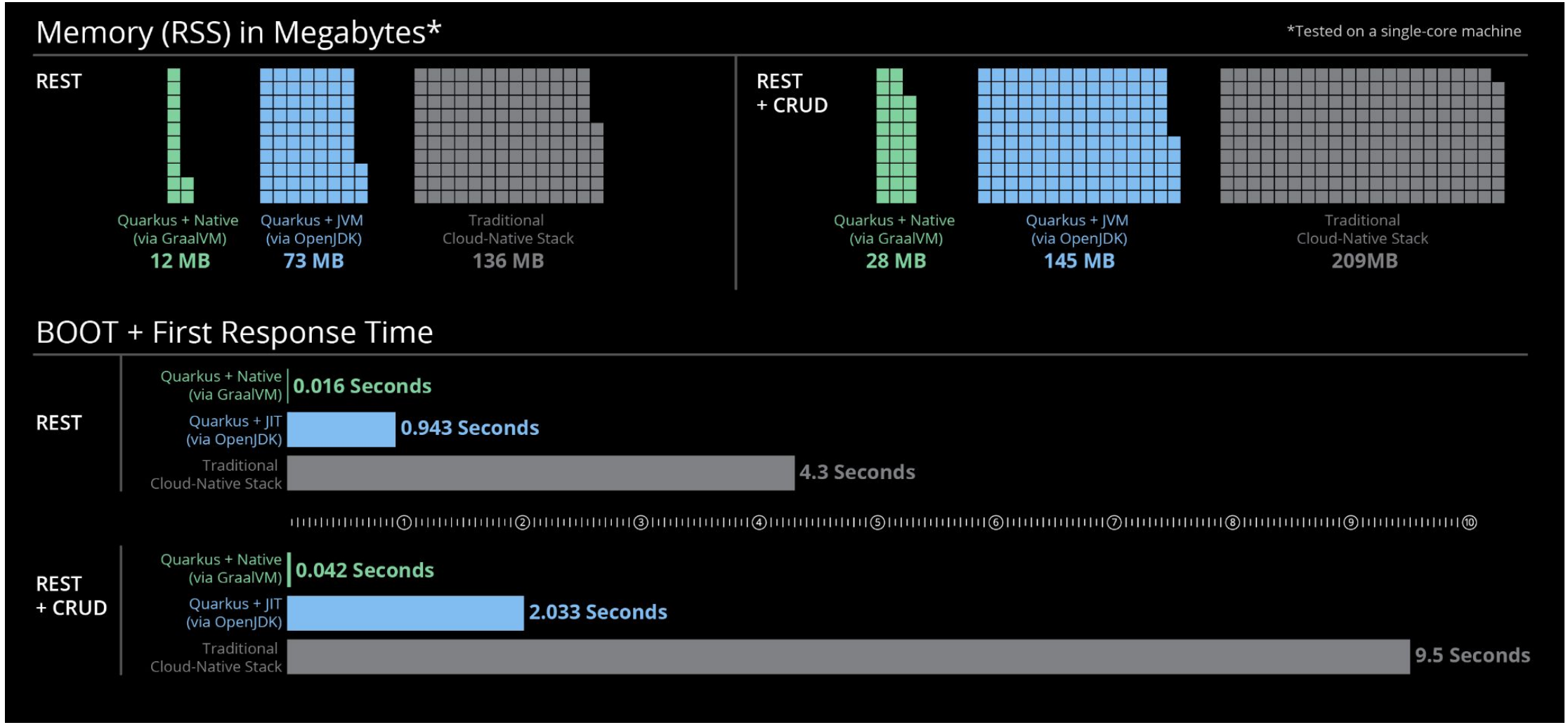


Serverless 1.5 - Quarkus



Serverless 1.5 - Quarkus

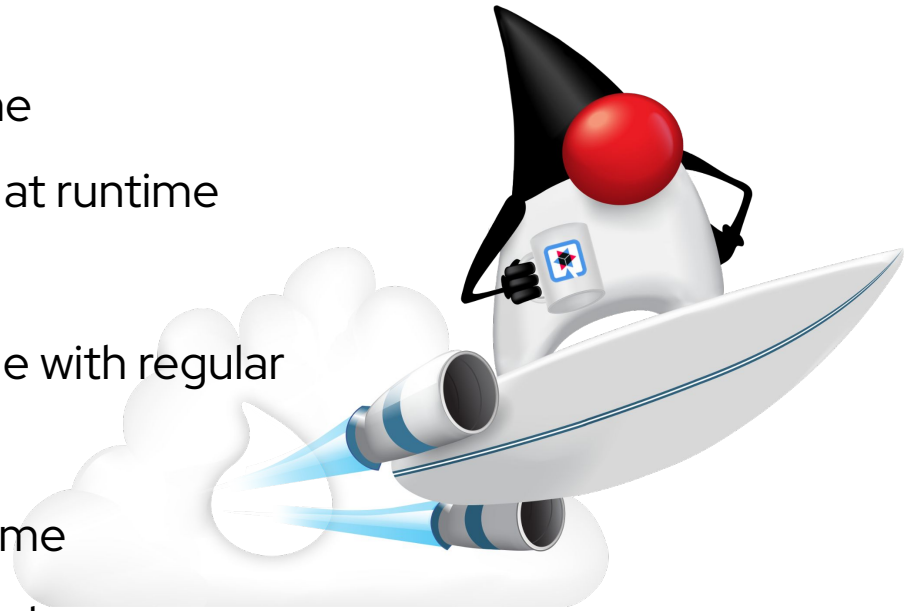
- Open source with a vibrant community
- Container first
- Kubernetes native
- **Supersonic: Superfast startup**
- **Subatomic: Low memory usage**
- **Can scale fast**



Source:
https://quarkus.io/

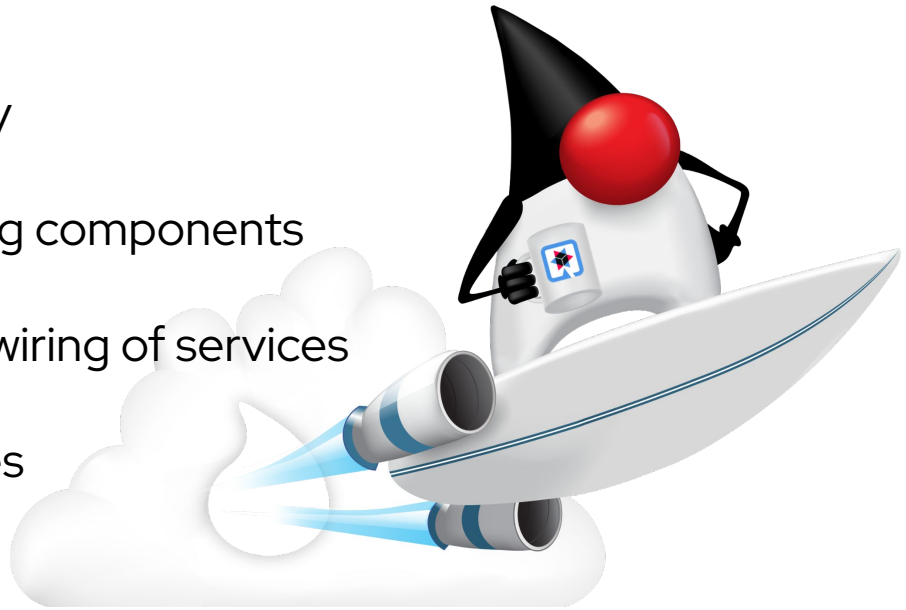
Quarkus - How to achieve that performance?

- Build-time processing
  - As much as possible is done at build time
  - The app only contains the classes used at runtime
- Reduced use of reflection
  - Reflection calls are replaced in build time with regular invocations
  - Dependency injection is done in build time
  - No expensive lookups when the app starts



Quarkus - Developer joy

- Live coding - code changes are reflected automatically
- Dev UI - visualize/configure extensions, logs and testing components
- Dev services - automatic provisioning and application wiring of services
- Continuous testing - instant feedback on code changes



Quarkus - Best libraries and standards

- CDI, JAX-RS, JPA, JTA, Vert.x, Camel...
- Implements MicroProfile
- Supports Spring APIs
- Hundreds of extensions

@helber_belmiro hbelmiro Red Hat

Deploying a Quarkus app to Knative

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-kubernetes</artifactId>
</dependency>
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-container-image-jib</artifactId>
</dependency>
```

```
$ mvn package \
    -Dquarkus.kubernetes.deploy=true \
    -Dquarkus.kubernetes.deployment-target=knative \
    -Dquarkus.container-image.group=dev.local/hbelmiro
```

FaaS

2015

2018

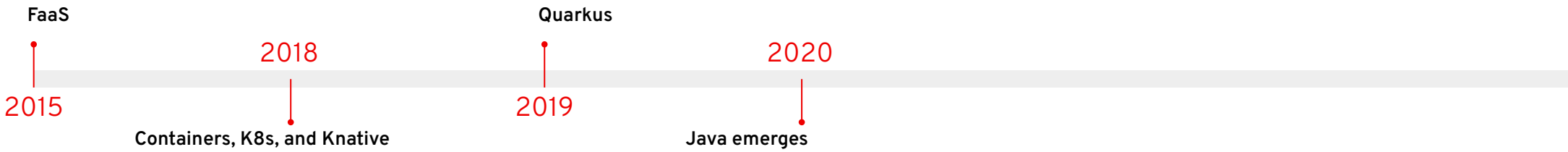
Containers, K8s, and Knative

Quarkus

2019

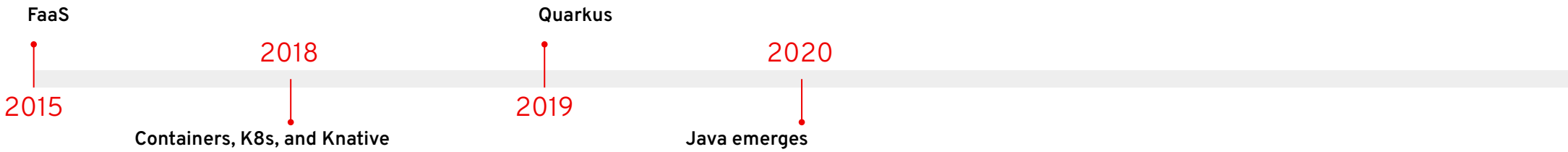
2020

Java emerges



Serverless 2.0 – State, integration, and orchestration

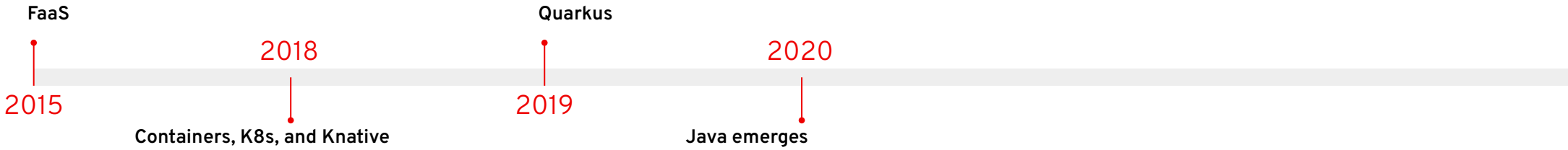
- State handling
- Enterprise integration patterns
- Advanced messaging capabilities
- Orchestration

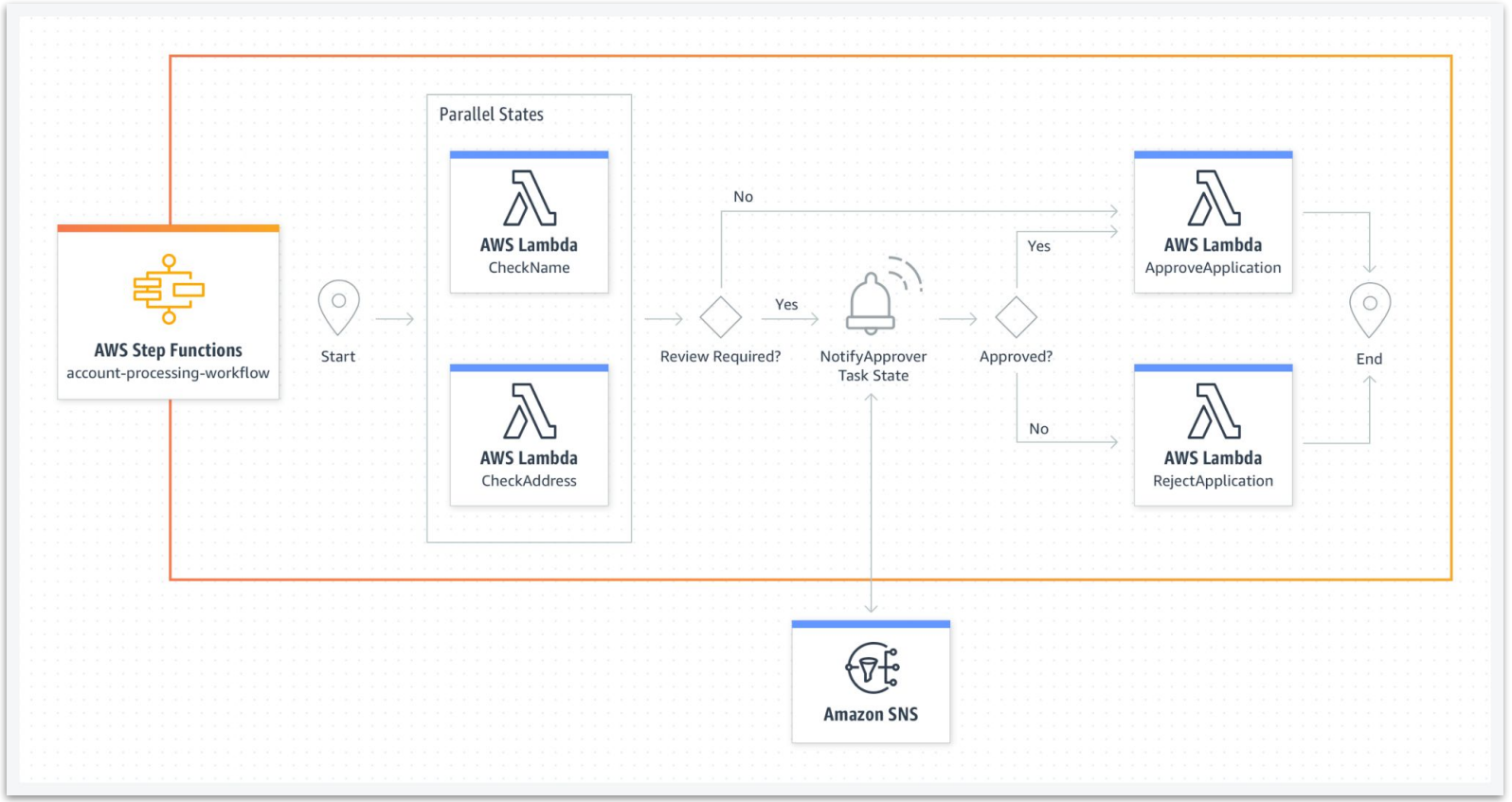


Serverless 2.0 - Orchestration

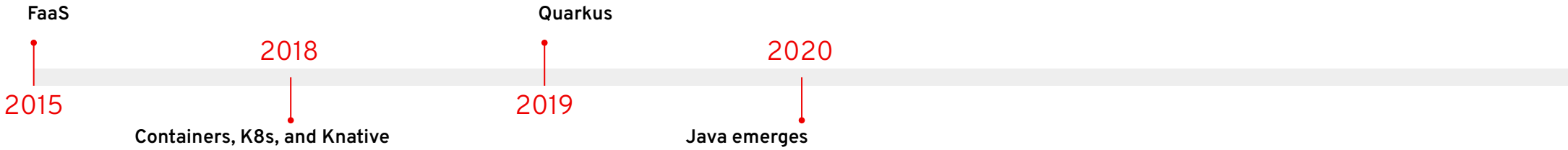
- AWS Step Functions
- Google Workflows
- Azure Durable Functions

Evolution



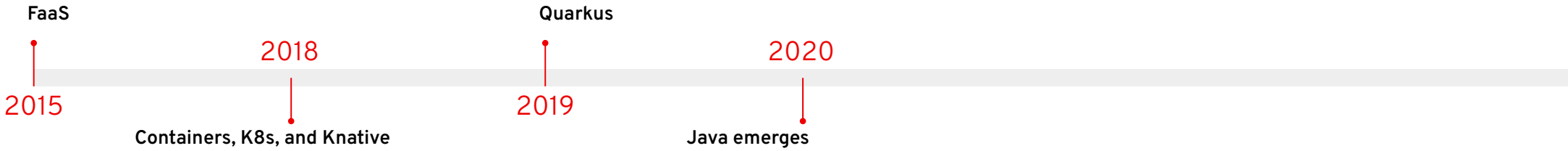


Source:
https://aws.amazon.com/step-functions/use-cases/



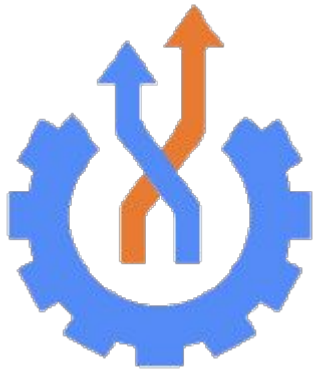
Serverless 2.0 - Orchestration

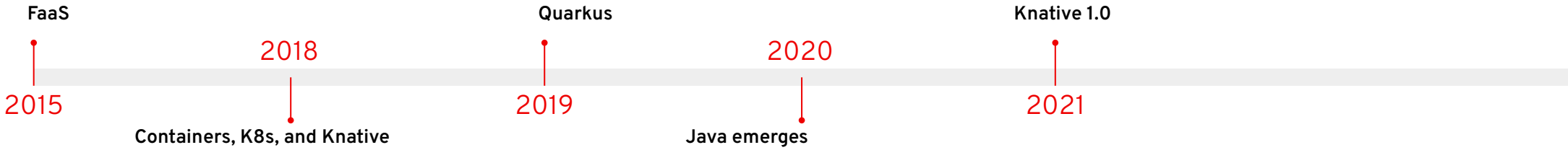
- Vendor lock-in (no portability and low productivity across platforms)
- Limits the potential for common libs, tooling, and infrastructure
- What about Knative?



Serverless 2.0 - CNCF Serverless Workflow

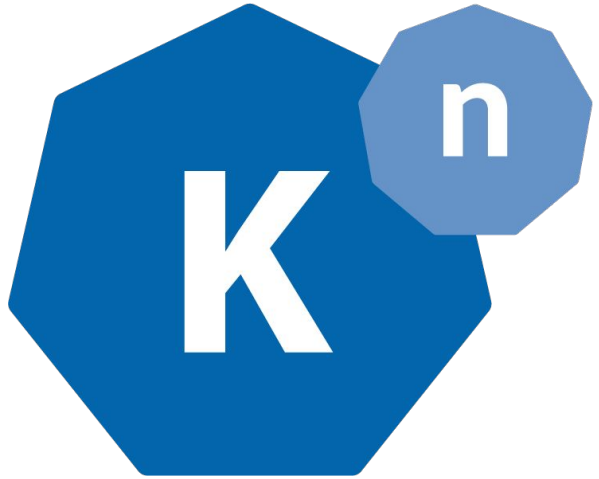
- Community driven
- Vendor neutral
- Open source
- Focus on standards (OpenAPI, CloudEvents, gRPC, GraphQL)
- Multi-language support (Java, Python, Typescript, Go, .NET)

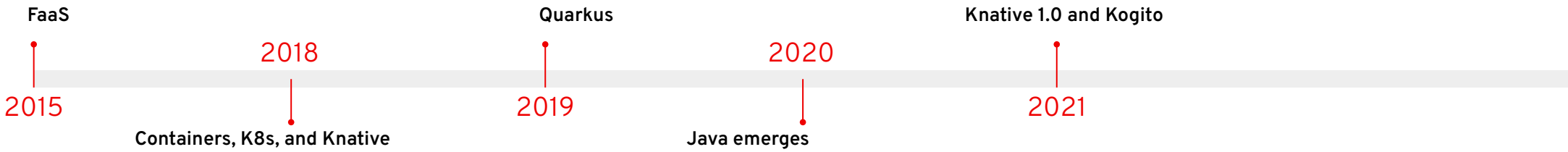




Knative 1.0

- Serving
- Eventing
- Apache Kafka Broker
- RabbitMQ Broker
- Knative Operator

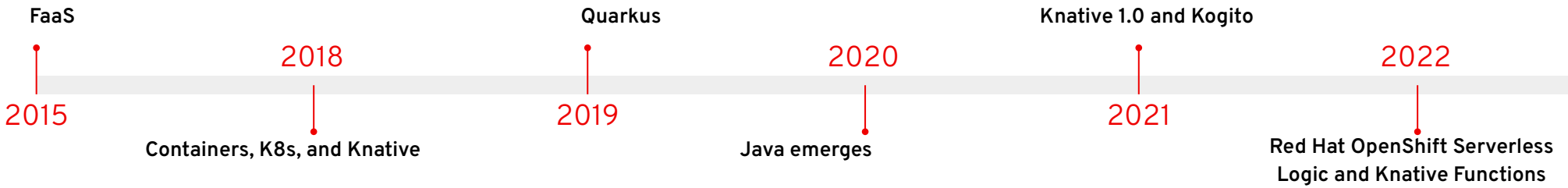




Kogito Serverless Workflow

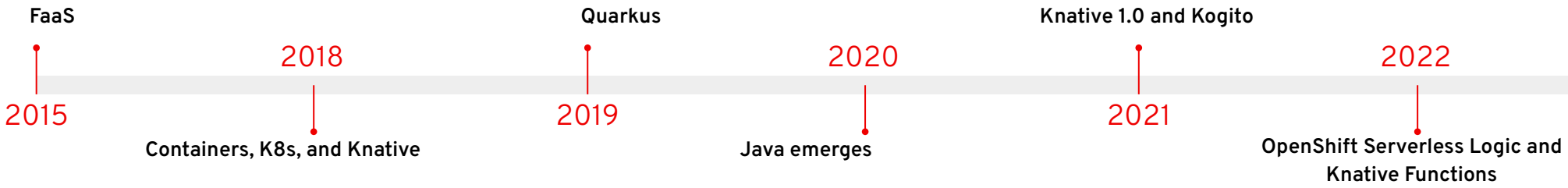
- Implements the CNCF Serverless Workflow specification
- Open source
- Built on top of Quarkus

https://github.com/kiegroup/kogito-docs/



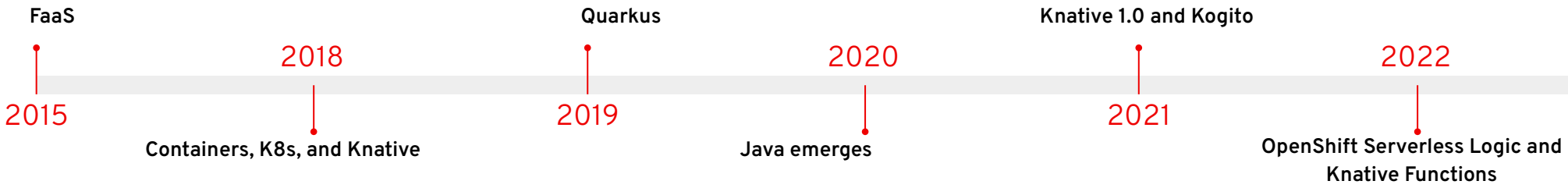
Red Hat OpenShift Serverless Logic

- Knative and Kogito under the hood
- Available as a Developer Preview in OpenShift Serverless 1.24.0



Knative Functions

- Available in Knative 1.8
- FaaS in Knative
- Uses Funqy

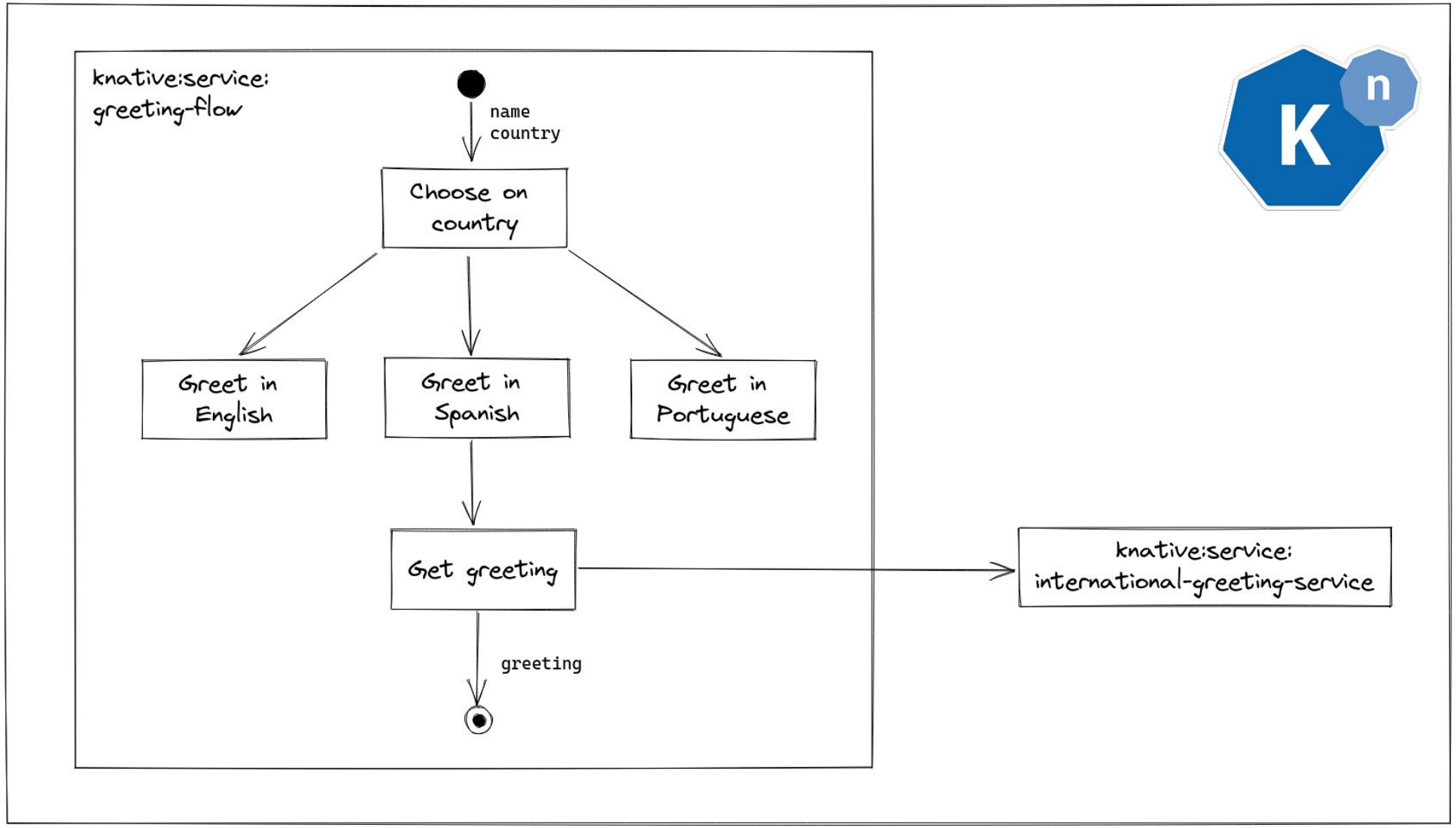


Knative Functions

```java
import io.quarkus.funqy.Funq;

public class Function {
    /**
     * This function simply echoes its input
     * @param input a Java bean
     * @return a Java bean
     */
    @Funq
    public Output function(Input input) {
        // Add business logic here
        return new Output(input.getMessage());
    }
}
```

Demo


knative:service:
greeting-flow
name
country
Choose on country
Greet in English
Greet in Spanish
Greet in Portuguese
Get greeting
greeting
knative:service:
international-greeting-service

Takeaways

- FaaS and BaaS
- You can use the cloud providers' APIs
- You can take a portable approach with Knative and CNCF Serverless Workflow
- Scale to zero
- Apps need to start and scale fast
- Usually short living apps
- Java is great for serverless

@helber_belmiro hbelmiro Red Hat

Questions?

Let's stay in touch:

thegreatapi.com

github.com/hbelmiro

linkedin.com/in/hbelmiro

twitter.com/helber_belmiro

Red Hat