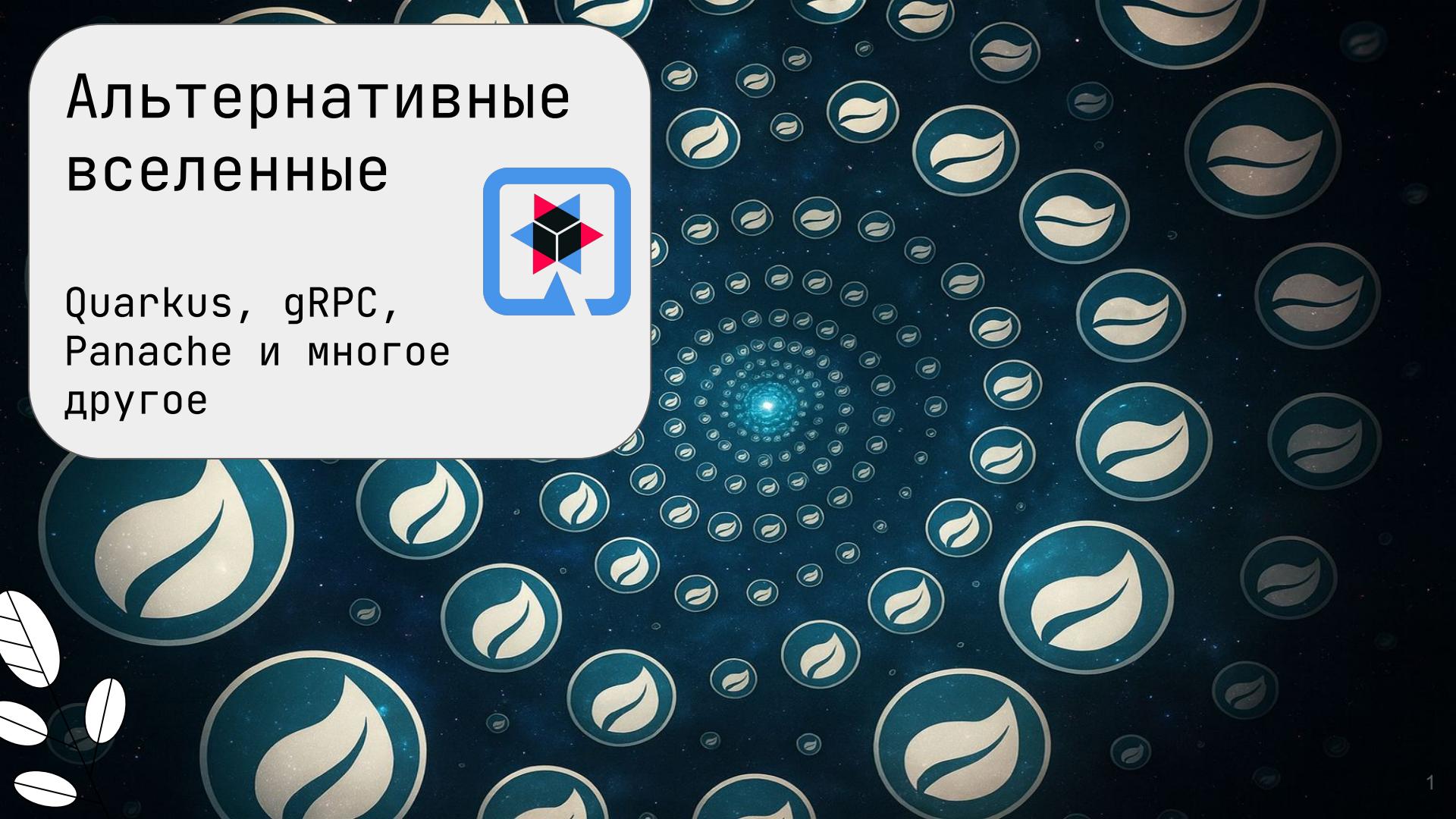
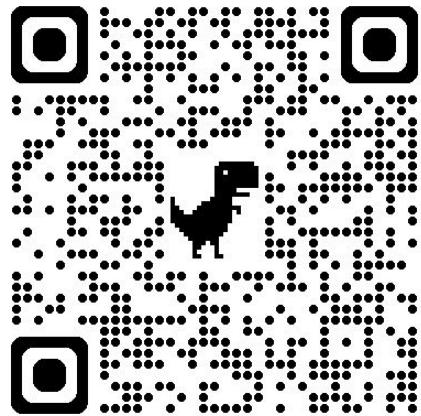


Альтернативные вселенные

Quarkus, gRPC,
Panache и многое
другое



\$ whoami?

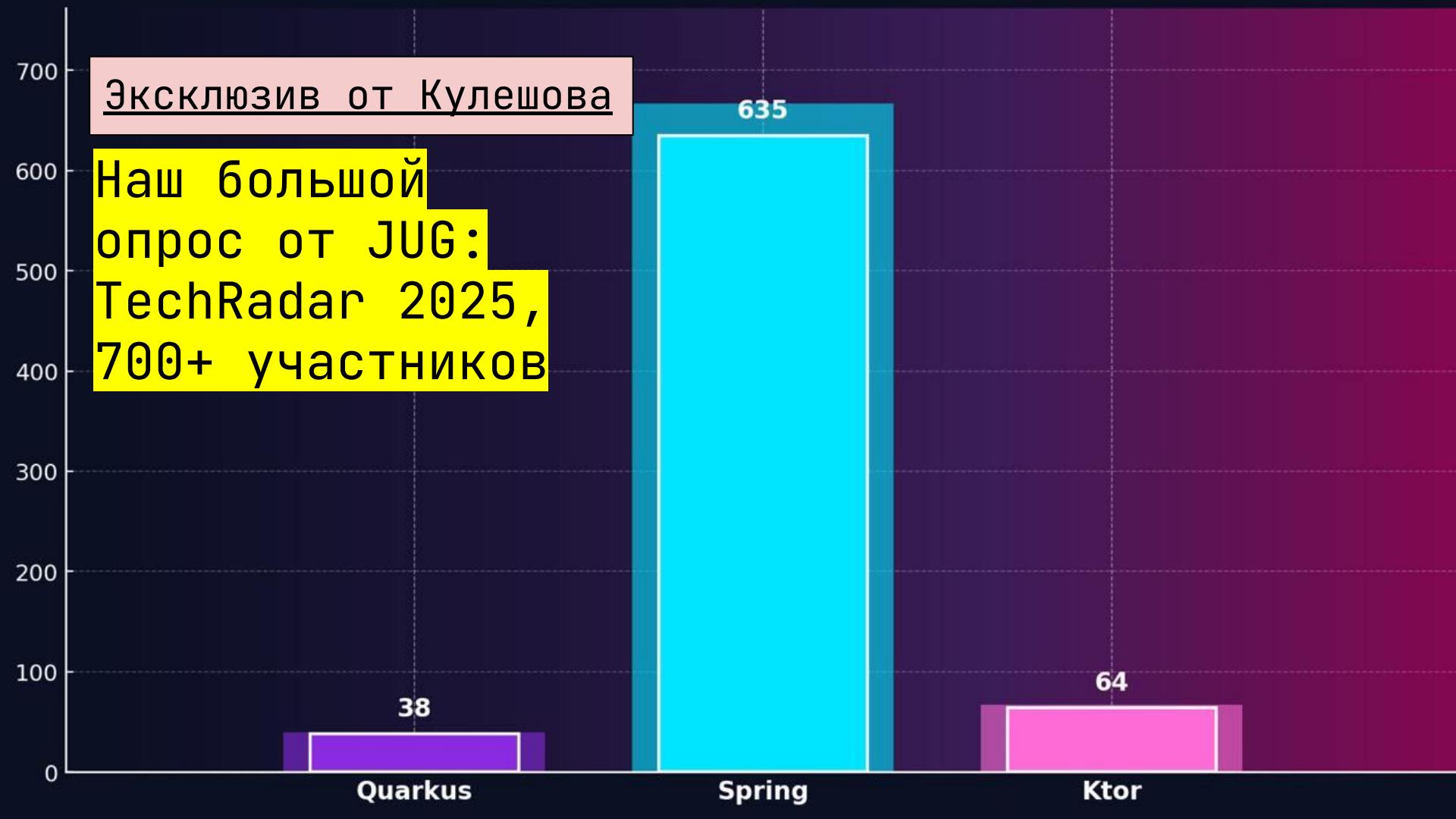


Андрей Кулешов



Эксклюзив от Кулешова

Наш большой
опрос от JUG:
TechRadar 2025,
700+ участников



«Стена плача» Java. На что хотят перейти

Не подходит для фронта, HFT, реалтайма

Показывает себя так себе

Отказаться от Kotlin в пользу Java

с Java на Kotlin

Новую Java

с Java на Golang

Перейти с JPA на JDBC с Kotlin+Spring на Golang

хочу на пенсию

с Quarkus на Spring

со Спринга на что угодно

На последний Spring Boot

с Hibernate на что угодно

Версия Java — SberJDK



Play Framework



PowerMock

Слезть «с иглы» Spring



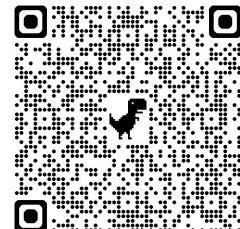
с Hibernate на что угодно

Слезь «с иглы» Spring



Кирилл Толкачев

X tolkv



«Стена плача» Java. На что хотят перейти

Не подходит для фронта, HFT, реалтайма

Показывает себя так себе

Отказаться от Kotlin в пользу Java

с Java на Kotlin

Новую Java

с Java на Golang

Перейти с JPA на JDBC с Kotlin+Spring на Golang

хочу на пенсию

с Quarkus на Spring

со Спринга на что угодно

На последний Spring Boot

с Hibernate на что угодно

Версия Java — SberJDK



Play Framework



PowerMock

Слезть «с иглы» Spring



с Hibernate на что угодно

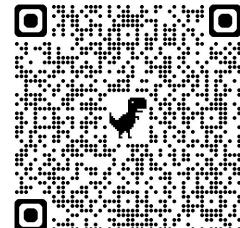
Слезь

«иглы»

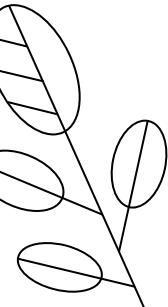


Кирилл Толкачев

X tolkv



6% опрошенных
хотели бы переехать
со **Spring** на **Quarkus**



«Стена плача» Java. На что хотят перейти

Не подходит для фронта, HFT, реалтайма

Показывает себя так себе

Отказаться от Kotlin в пользу Java

с Java на Kotlin

Новую Java

с Java на Golang

Перейти с JPA на JDBC с Kotlin+Spring на Golang

хочу на пенсию

с Quarkus на Spring

со Спринга на что угодно

На последний Spring Boot

с Hibernate на что угодно

Версия Java — SberJDK



Play Framework



PowerMock

Слезть «с иглы» Spring

Был и один герой,
который не молчал



с Hibernate на что угодно

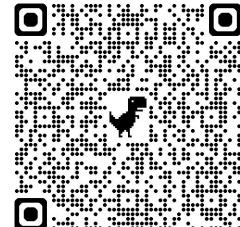
Слезь

«иглы»



Кирилл Толкачев

X tolkv

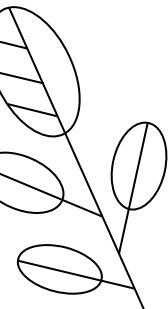
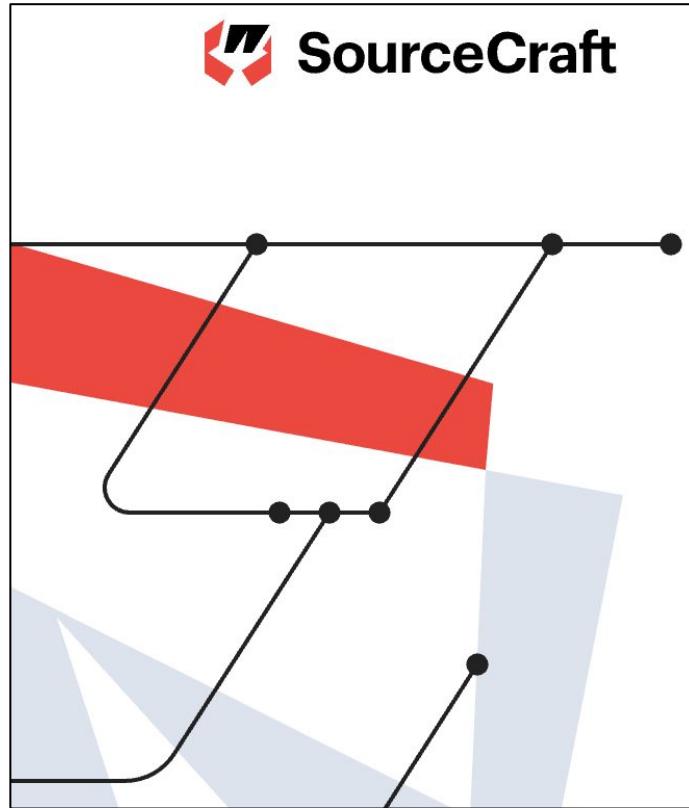
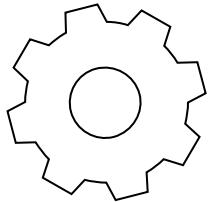


iPoint × JUG.RU
group

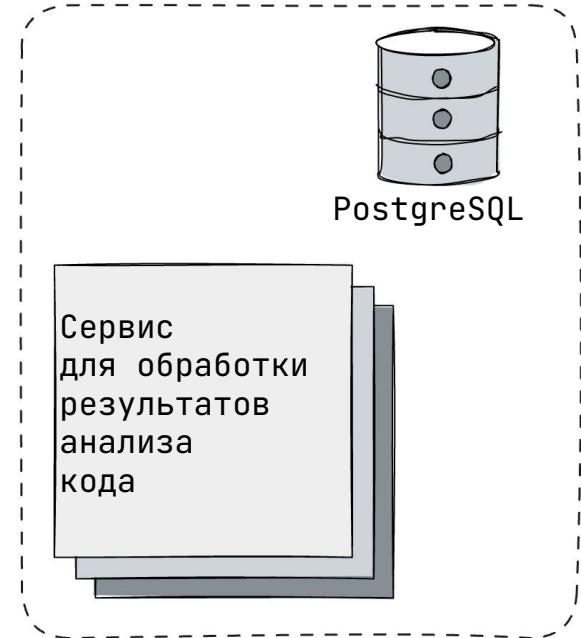
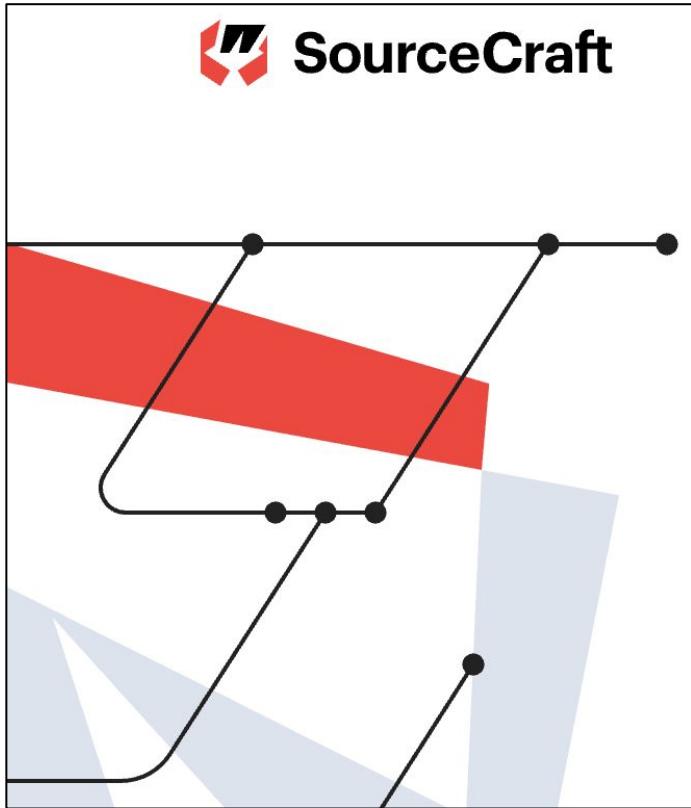
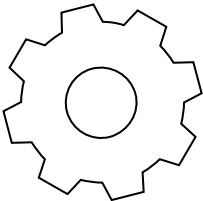
СБЕР

SM Lab

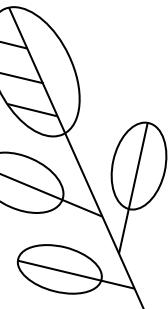
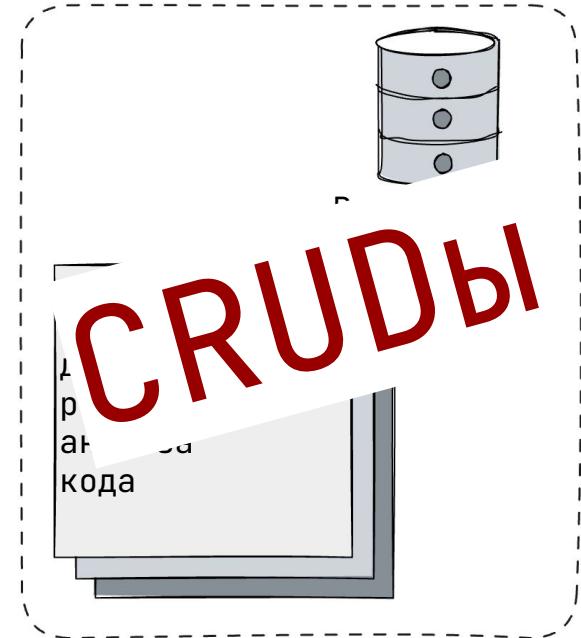
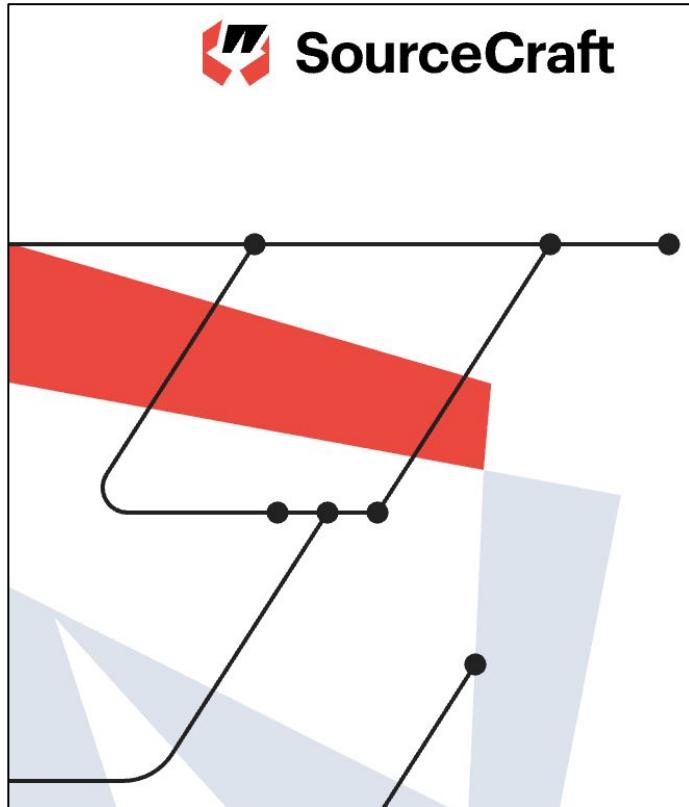
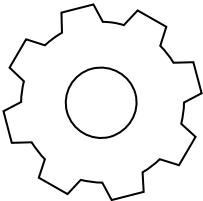
Моя история



Моя история



Моя история



РАБОТАЕТ – НЕ ТРОЖЬ



Слезть «с иглы» Spring

РАБОТАЕТ – ~~НЕ ТРОЖЬ~~

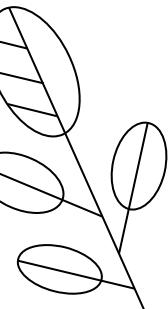
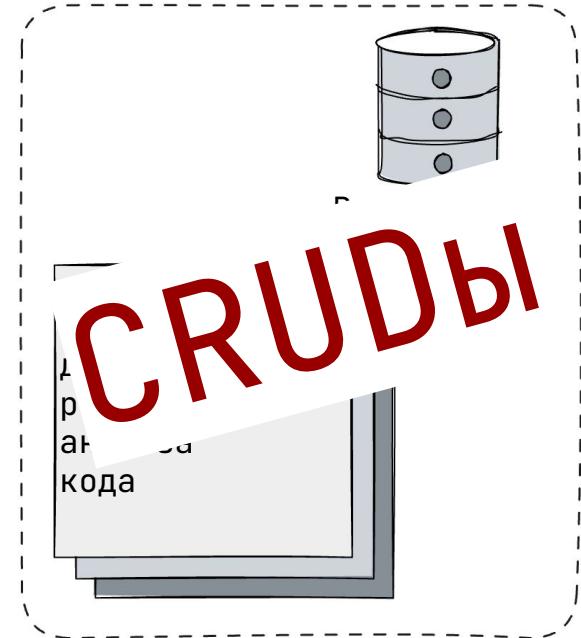
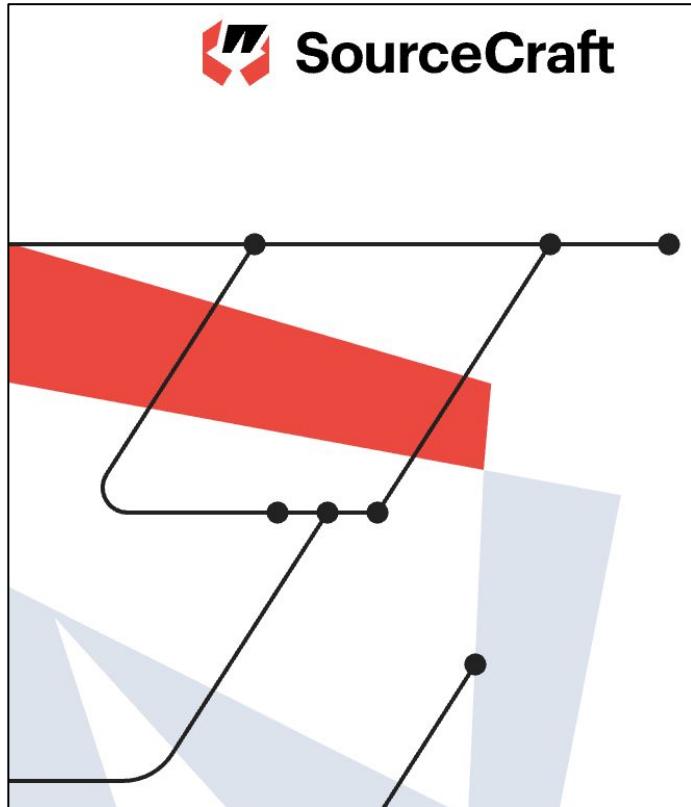
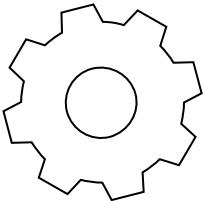


Слезть «с иглы» Spring

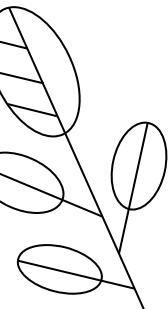
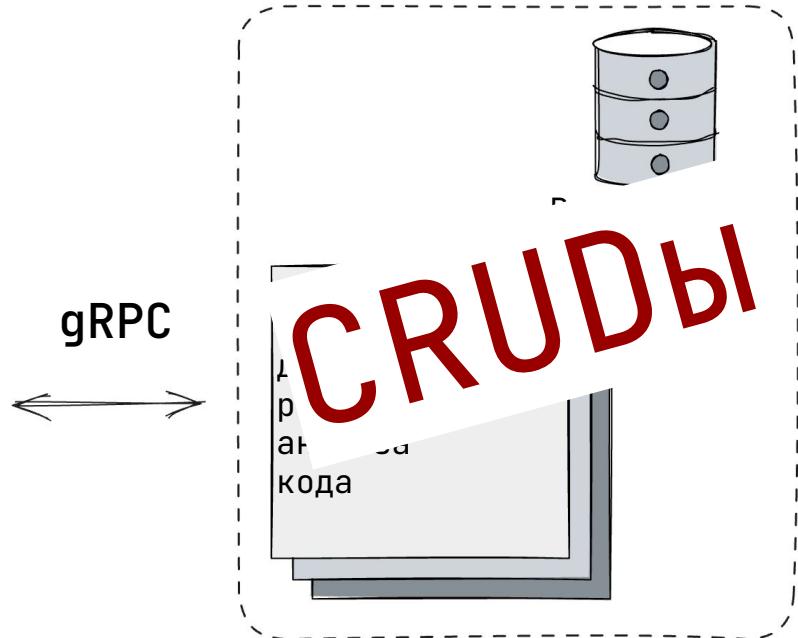
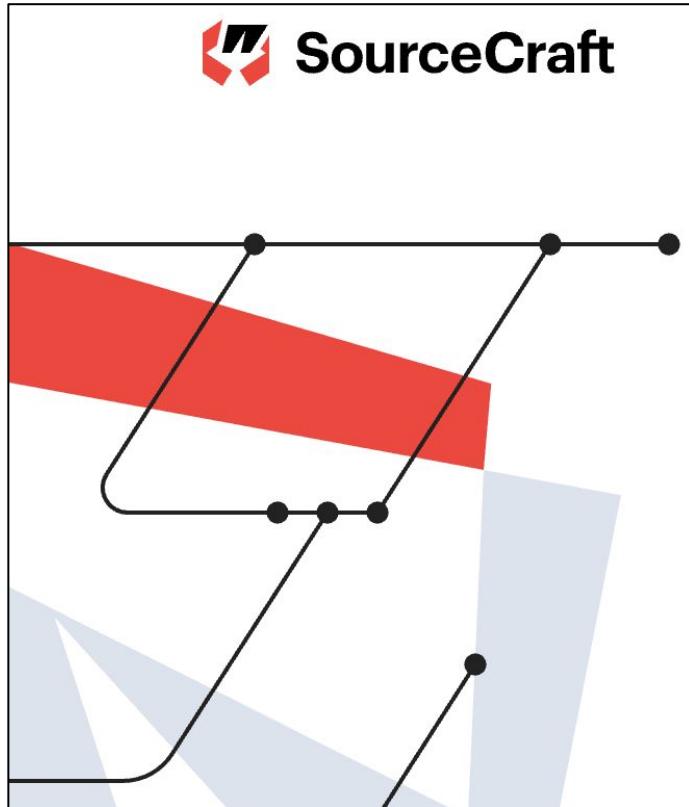
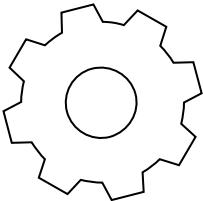
РАБОТАЕТ – ~~НЕ ТРОЖЬ~~



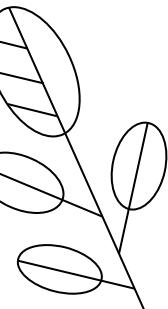
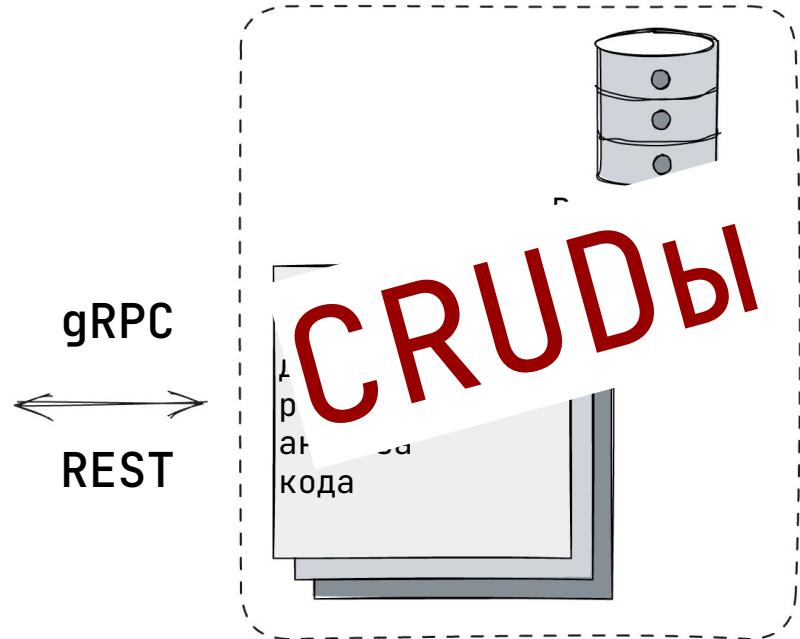
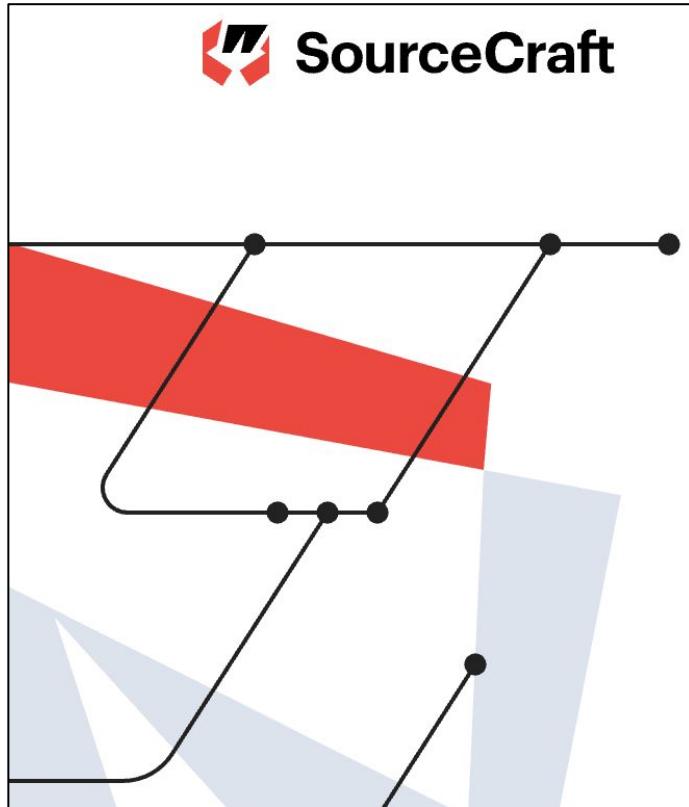
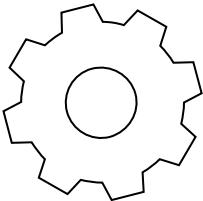
Моя история



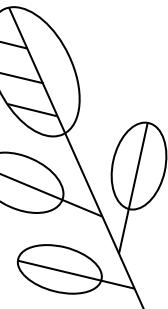
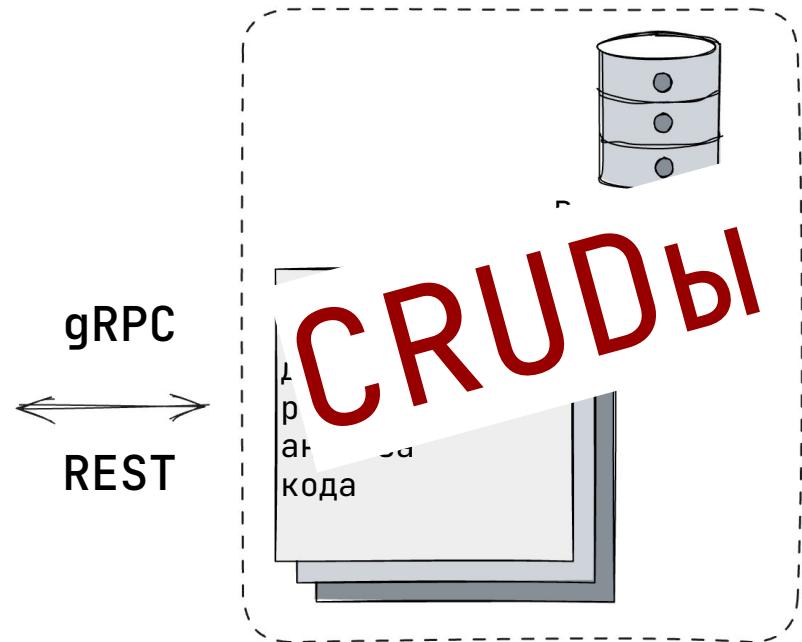
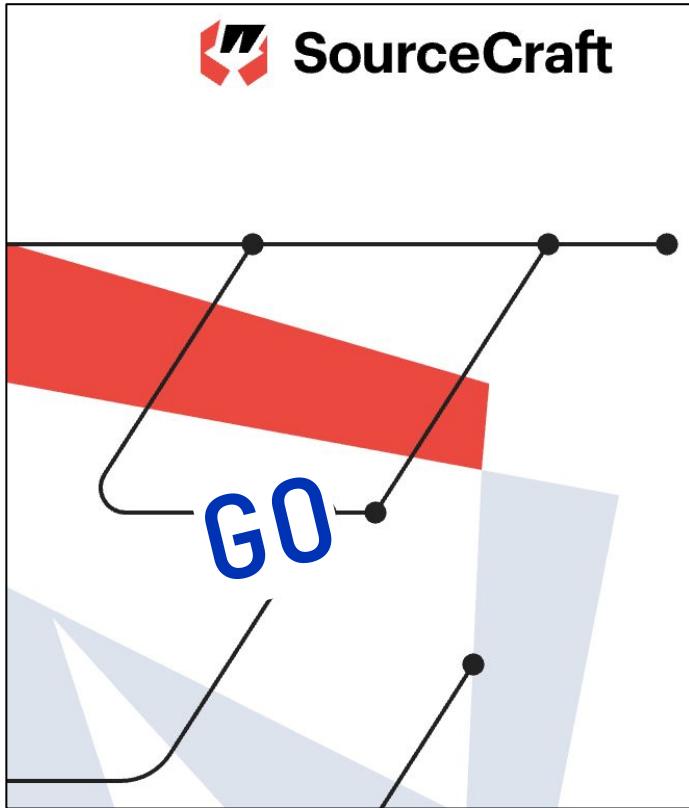
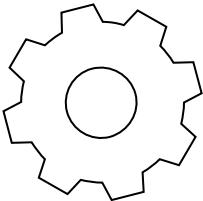
Моя история



Моя история



Моя история



отличная экосистема

Оч хороший фреймворк,
а спринг - ну его

Мы все сервисы переводим
на Quarkus сейчас, у нас
в компании стандарт де-
факто

Завязывают на всё своё,
редхатовское

Мне очень нравится -
отличная поддержка и
активное коммьюнити

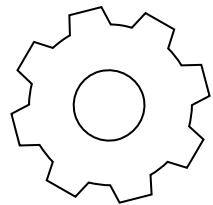
Всё из коробки же,
очень enterprise-ready

Точно будет
конфетно-букетный период, а потом...

Очень уж они заточены
как-будто под быстрые
промышленные сценарии

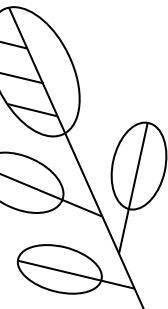
GraalVM же, натив!

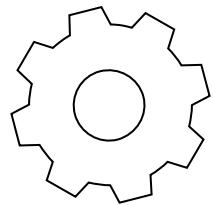
Шаг в сторону -
начинаются
проблемы



Про что поговорим?

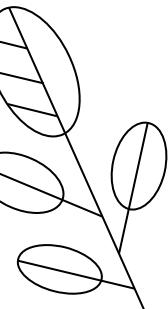
- Мой личный опыт использования

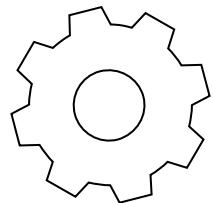




Про что поговорим?

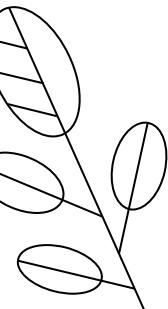
- Мой личный опыт использования
- Базовые вещи и моя **субъективная** оценка UX,
не стоит ждать революционности





Про что поговорим?

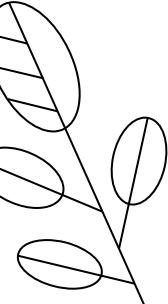
- Мой личный опыт использования
- Базовые вещи и моя **субъективная** оценка UX, не стоит ждать революционности
- Нет цели никого убеждать срочно переходить на Quarkus



**Дисклеймер: последний раз объясняю, что не
буду потрошить никого**



Знакомство



Для тех, кто хочет прямых сравнений

Round up

Getting started

Database access

Rest endpoints

Websocket client

Websocket

Virtual Threads

Scheduling & startup

Dev experience

Rest client & startup-hook

Native image



35:12 / 39:23 • Bonus Round ▶

ORA



SUPERSONIC/ SUBATOMIC/ JAVA

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

Now Available

QUARKUS 3.25.3

[Read the release notes](#)

SUPERSONIC/ SUBATOMIC/ JAVA

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

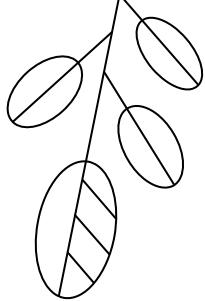
Now Available

QUARKUS 3.25.3

[Read the release notes](#)



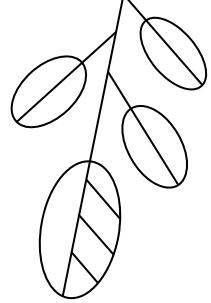
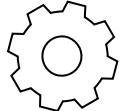
Азы и терминология



Netty (HTTP/Сетевой слой)



Азы и терминология



Vert.x (EventLoop/Worker Pool/итд.)

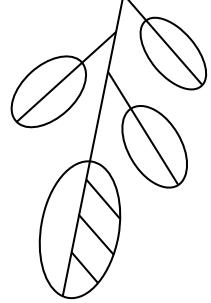
Netty (HTTP/Сетевой слой)





Азы и терминология

RESTEasy - реализация JAX-RS/Jakarta REST



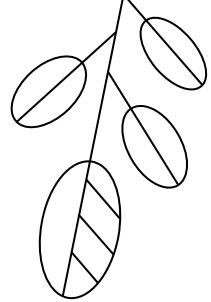
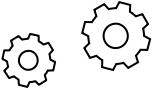
RESTEasy

Vert.x (EventLoop/Worker Pool/итд.)

Netty (HTTP/Сетевой слой)



Азы и терминология



Default Quarkus 2.8-

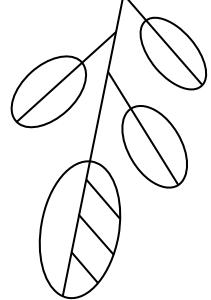
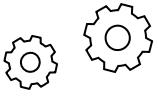
quarkus-resteasy

Vert.x (EventLoop/Worker Pool/итд.)

Netty (HTTP/Сетевой слой)



Азы и терминология



Default Quarkus 2.8-

quarkus-resteasy
RESTEasy Classic

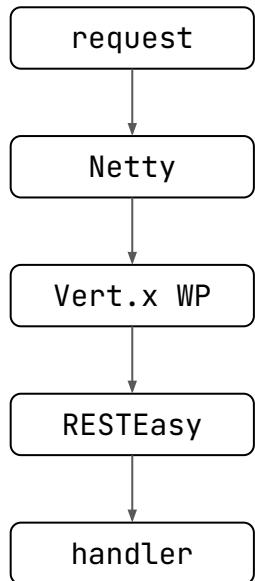
Vert.x (EventLoop/Worker Pool/итд.)

Netty (HTTP/Сетевой слой)



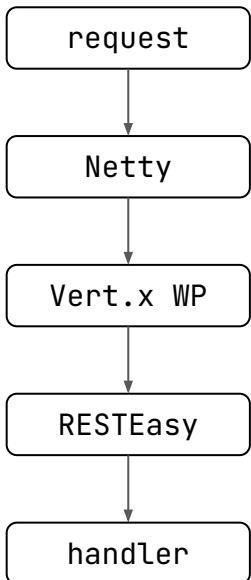
Азы и терминология

quarkus-resteasy



Азы и терминология

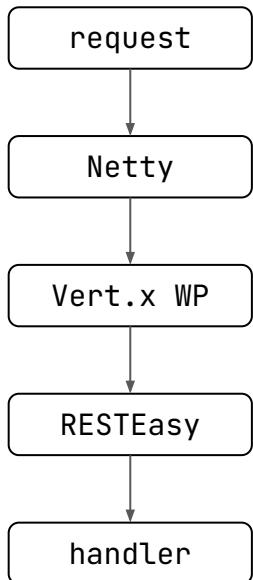
quarkus-resteasy



```
// Берется поток из WP под запрос
@Path("/users")
public class UserResource {
    @GET
    public List<User> getUsers() {
    }
}
```

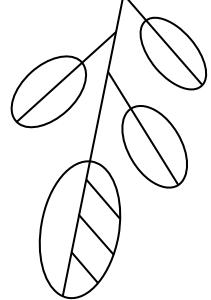
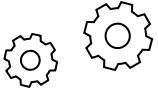
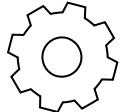
Азы и терминология

quarkus-resteasy



```
// Берется поток из WP под запрос
@Path("/users")
public class UserResource {
    @GET
    public List<User> getUsers() {
        // Императивно выполняется в worker thread
        // Поток заблокирован до завершения операции
        // Блокирующий вызов БД
        return userService.findAll();
    }
}
```

Азы и терминология



Default Quarkus 2.8+

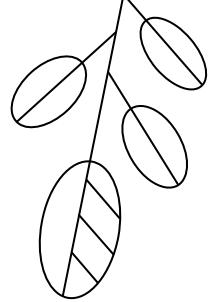
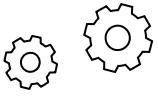
quarkus-resteasy-reactive
RESTEasy Reactive

Vert.x (EventLoop/Worker Pool/итд.)

Netty (HTTP/Сетевой слой)



Азы и терминология



Default Quarkus 2.8+

quarkus-rest

~~quarkus-resteasy reactive~~

RESTEasy Reactive

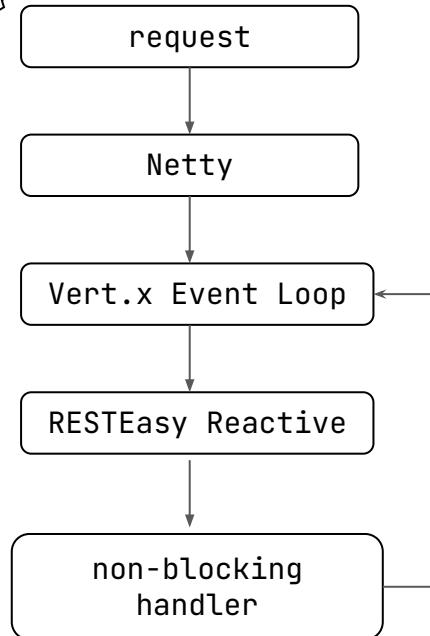
Vert.x (EventLoop/Worker Pool/итд.)

Netty (HTTP/Сетевой слой)



Азы и терминология

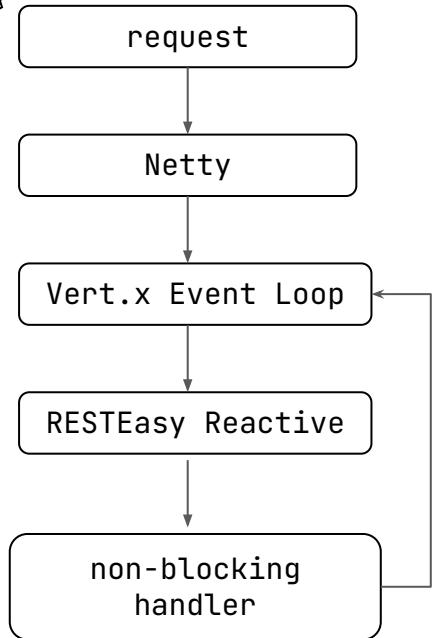
quarkus-rest(easy-reactive)



```
@Path("/users")
public class UserResource {
    @GET
    public Uni<List<String>> getUsers() {
        }
}
```

Азы и терминология

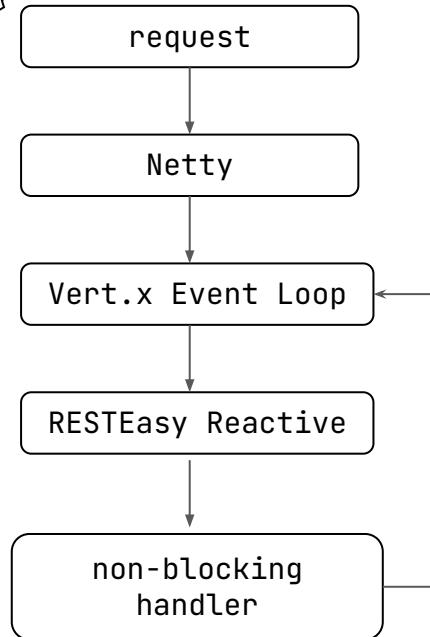
quarkus-rest(easy-reactive)



```
@Path("/users")
public class UserResource {
    @GET
    public Uni<List<String>> getUsers() {
        // Выполняется в event loop thread
        // Поток НЕ блокируется, сразу отдает Uni<
        return userService.findAll();
    }
}
```

Азы и терминология

quarkus-rest(easy-reactive)



```
@Path("/users")
public class UserResource {
    @GET
    public Uni<List<String>> getUsers() {
        // Выполняется в event loop thread
        // Поток НЕ блокируется, сразу отдает Uni<
        return userService.findAll();
    }
}
```

Реактивный фреймворк

```
package io.smallrye.mutiny;
```

[smallrye-mutiny](#)

Public

An Intuitive Event-Driven Reactive Programming Library for Java



Java



871



137



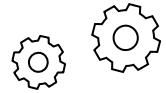
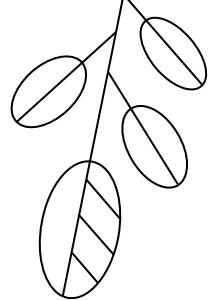
Реактивный фреймворк

package io.smallrye.mutiny;



Как и `project.reactor`:
Event Driven + Reactive Streams

Реактивный фреймворк



package io.smallrye.mutiny;

smallrye-mutiny Public

An Intuitive Event-Driven Reactive Programming Library for Java

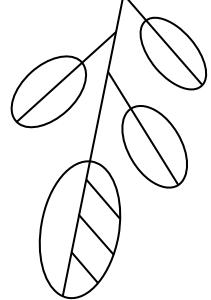
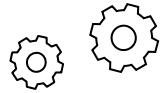
Java ⭐ 871 ⚡ 137

Как и `project.reactor`:
`Event Driven + Reactive Streams`

- + По заверениям авторов проще для интеграции с Vert.x



Реактивный фреймворк



package io.smallrye.mutiny;

smallrye-mutiny Public

An Intuitive Event-Driven Reactive Programming Library for Java

Java ⭐ 871 ⚡ 137

Как и `project.reactor`:
Event Driven + Reactive Streams

- + По заверениям авторов проще для интеграции с Vert.x



Азы и терминология

smallrye/mutiny

Uni

```
@GET  
@Path("/user/{id}")  
public Uni<User> getUser(Long id) {  
    return userService.findById(id);  
    {"id": 1, "name": "John"}  
}
```

Азы и терминология

smallrye/mutiny

Multi

```
// Multi - streaming response (SSE или chunked)
@GET
@Path("/users/stream")
@Produces(MediaType.SERVER_SENT_EVENTS)
public Multi<User> streamUsers() {
    return userService.streamAll();

    // data: {"id": 1, "name": "John"}
    //
    // data: {"id": 2, "name": "Jane"}
    //
    // data: {"id": 3, "name": "Bob"}
    //
    ...
}
```



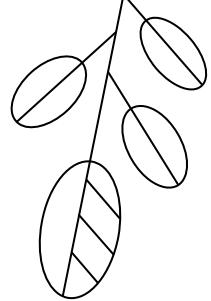
Азы и терминология

smallrye/mutiny

Quarkus'у на этапе build-time важна
сигнатура метода

Азы и терминология

smallrye/mutiny



Quarkus'у на этапе build-time важна
сигнатура метода

если **Uni/Multi** → помечается как
non-blocking route → вешается на
event-loop (**I/O thread**)



Азы и терминология

smallrye/common

@Blocking/@NonBlocking

@GET

Ничего не указываем

```
public String getTest() {  
  
    System.out.println(Thread.currentThread().getName());  
    return "Test";  
}
```

Азы и терминология

smallrye/common

@Blocking/@NonBlocking

@GET

```
Или явно @Blocking  
public String getTest() {  
  
    System.out.println(Thread.currentThread().getName());  
    return "Test";  
}
```

Азы и терминология

smallrye/common

@Blocking/@NonBlocking

@GET

```
Или явно @Blocking  
public String getTest() {  
  
    System.out.println(Thread.currentThread().getName());  
    return "Test";  
}
```

executor-thread-1



Азы и терминология

smallrye/common

@Blocking/@NonBlocking

```
@GET  
@NonBlocking  
public String getTest() {  
  
    System.out.println(Thread.currentThread().getName());  
    return "Test";  
}
```

Азы и терминология

smallrye/common

@Blocking / @NonBlocking

```
@GET  
@NonBlocking  
public String getTest() {  
  
    System.out.println(Thread.currentThread().getName());  
    return "Test";  
}
```

Или Uni/Multi возвращаемое значение без NonBlocking

Азы и терминология

smallrye/common

@Blocking / @NonBlocking

```
@GET  
@NonBlocking  
public String getTest() {  
  
    System.out.println(Thread.currentThread().getName());  
    return "Test";  
}
```

Или Uni/Multi возвращаемое значение без NonBlocking

vert.x-eventloop-thread-1

Азы и терминология

smallrye/common

`@Blocking/@NonBlocking`

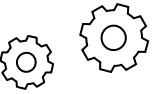
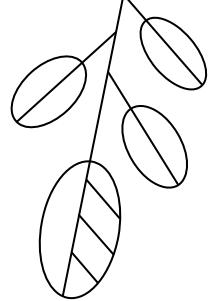
*java.lang.IllegalStateException: You have attempted
to perform a blocking operation on a IO thread.
This is not allowed, as blocking the IO thread will
cause major performance issues with your
application. If you want to perform blocking
EntityManager operations make sure you are doing it
from a worker thread.*

Азы и терминология

smallrye/common

@Blocking/@NonBlocking

Нет жесткого разделения моделей, как WebFlux и MVC



Азы и терминология

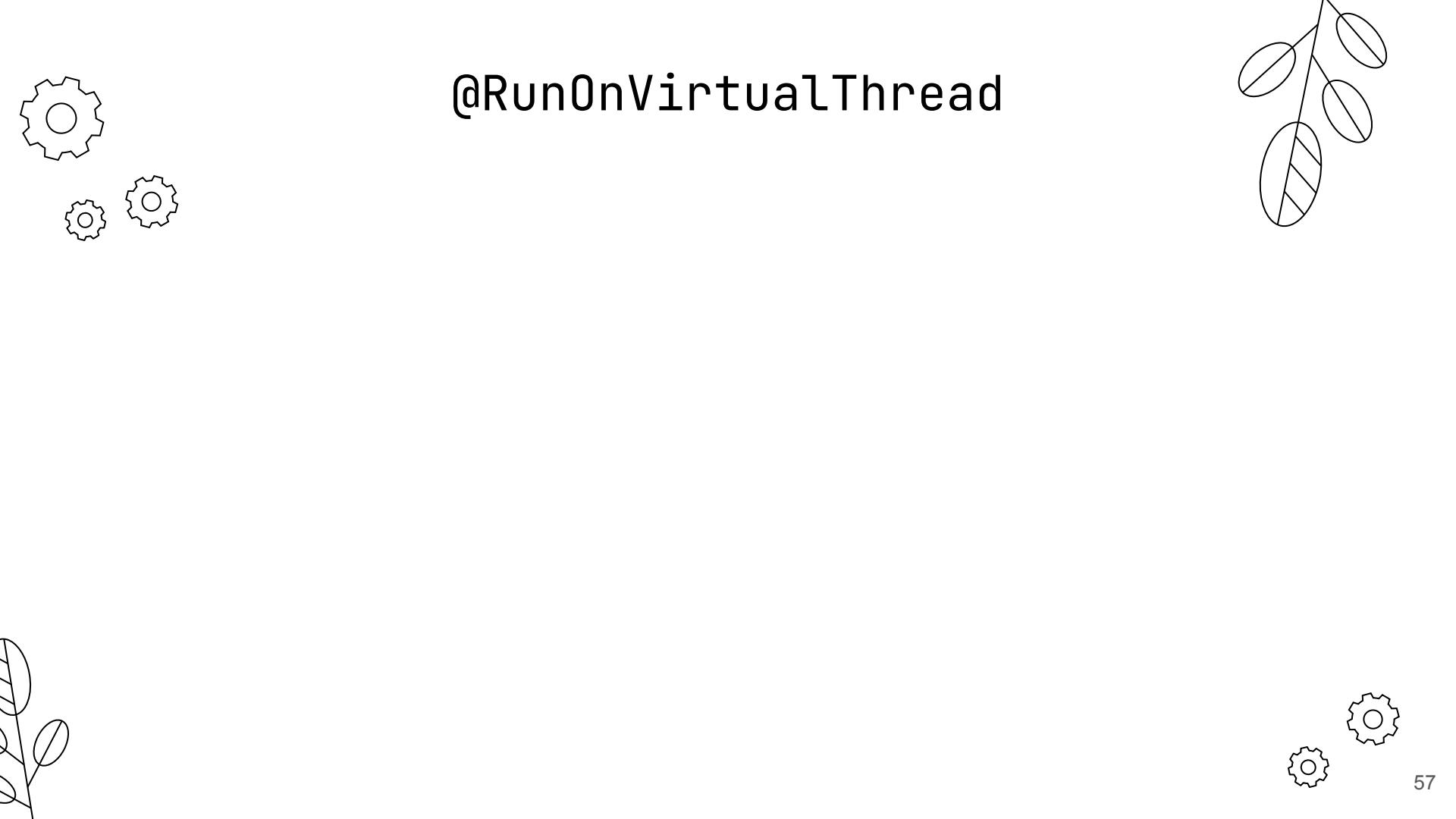
smallrye/common

@Blocking/@NonBlocking

Нет жесткого разделения моделей, как WebFlux и MVC

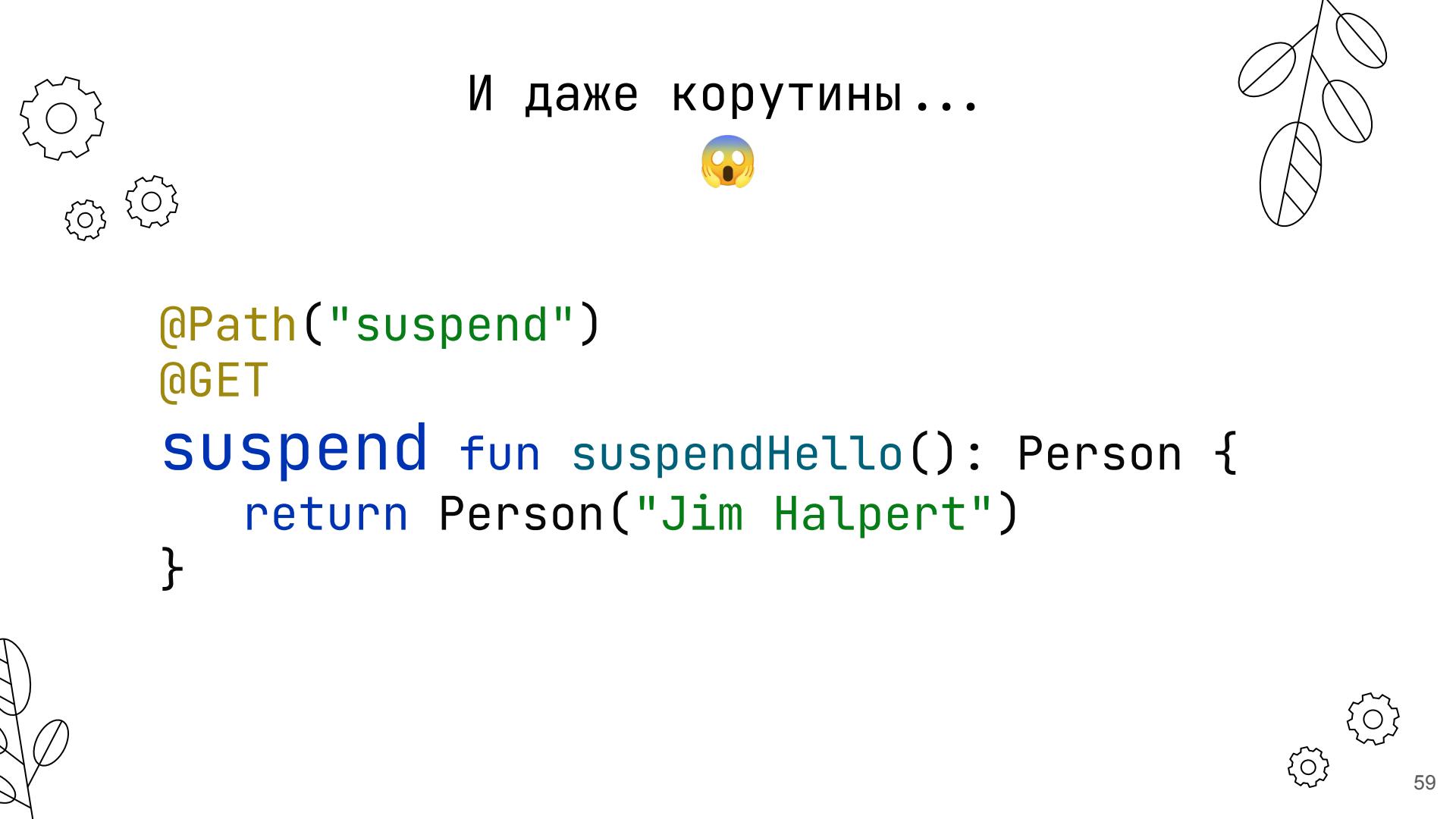
Method signature	Dispatching strategy
T method(...)	Worker thread
Uni<T> method(...)	I/O thread
CompletionStage<T> method(...)	I/O thread
Multi<T> method(...)	I/O thread
Publisher<T> method(...)	I/O thread
@Transactional CompletionStage<T> method(...)	Worker thread

@RunOnVirtualThread



@RunOnVirtualThread

As mentioned above, not everything can run safely on virtual threads. The risk of **monopolization** can lead to **high-memory usage**. Also, there are situations where the virtual thread cannot be unmounted from the carrier thread. This is called **pinning**. Finally, some libraries use **ThreadLocal** to store and reuse objects. Using virtual threads with these libraries will lead to massive allocation, as the intentionally pooled objects will be instantiated for every (disposable and generally short-lived) virtual thread.

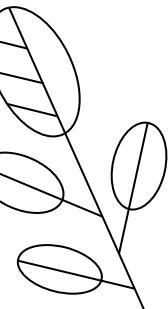


И даже корутины...

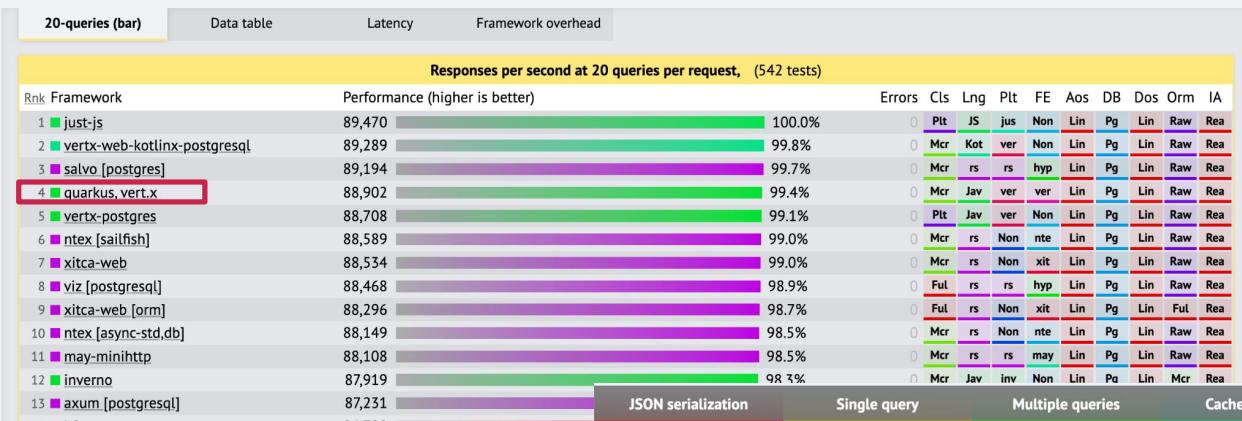


```
@Path("suspend")
@GET
suspend fun suspendHello(): Person {
    return Person("Jim Halpert")
}
```

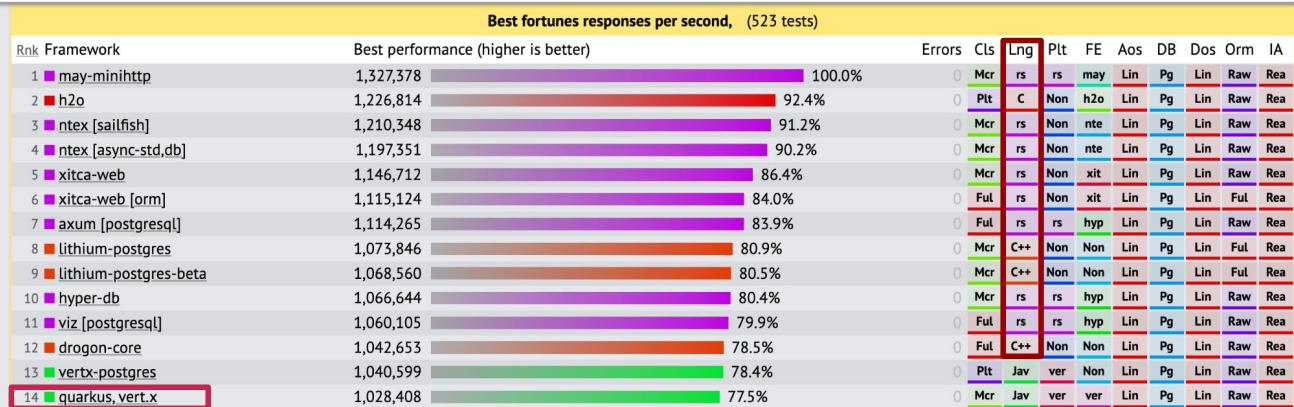
Быстро?



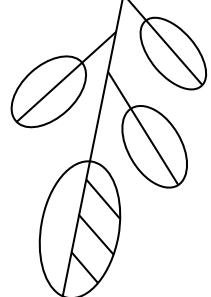
Multiple queries



Fortunes

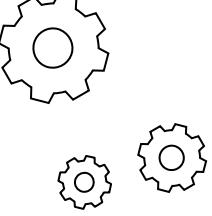


Техемпойер -
плохой бенчмарк,
но ...



Про скорость

- Quarkus жертвует “универсальностью” и “совместимостью”



Про скорость

- Quarkus жертвует “универсальностью” и “совместимостью”
- Берите, что дают: ядро фреймворка ограничено тем, что удобно разработчикам, не распухает



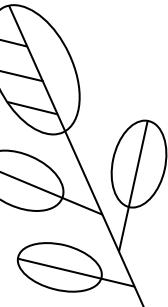
Про скорость

- Quarkus жертвует “универсальностью” и “совместимостью”
- Берите, что дают: ядро фреймворка ограничено тем, что удобно разработчикам, не распухает
- Попытка переносить большую часть логики на build time (например, **DI**)

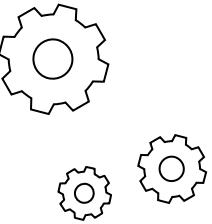
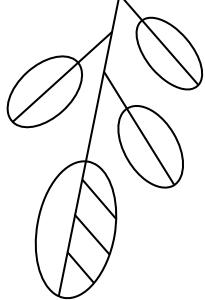
CDI

jakarta.enterprise.cdi

Arc



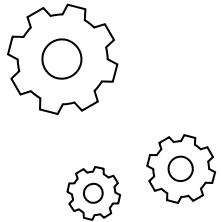
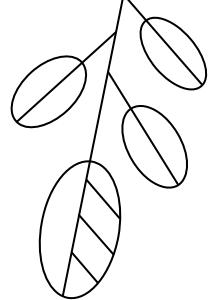
Азы и терминология



Arc - приблизительная
имплементация спецификации
Jakarta Contexts and
Dependency Injection 4.1



Азы и терминология

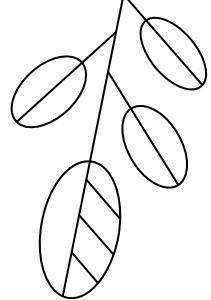
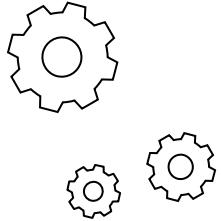


Arc - приблизительная
имплементация спецификации
Jakarta Contexts and
Dependency Injection 4.1

`quarkus.arc.strict-compatibility=true`

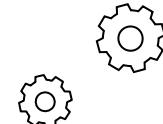


Азы и терминология

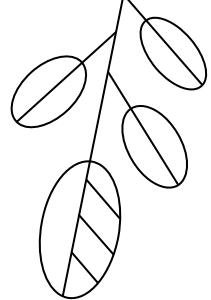
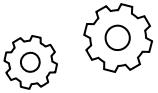


Arc - приблизительная
имплементация спецификации
Jakarta Contexts and
Dependency Injection 4.1

Работает за счет **ASM**, библиотеки для
манипуляций с байткодом



Азы и терминология



Arc - приблизительная
имплементация спецификации
[Jakarta Contexts and](#)
[Dependency Injection 4.1](#)

Работает за счет **ASM**, библиотеки для
манипуляций с байткодом



Андрей Кулешов: "Компиляторные плагины в Котлин:
почувствуй себя системным программистом"



Contexts and Dependency Injection

```
@ApplicationScoped  
public class UserService {  
    public User findById(Long id) {  
        return new User(id, "John");  
    }  
}
```

Singleton, с дополнительными сгенерированными обертками и прокси

Contexts and Dependency Injection

```
@ApplicationScoped  
public class OrderService {  
  
    // ===== Field atau setter  
    @Inject  
    UserService userService;  
  
}  
}
```

Contexts and Dependency Injection

```
@ApplicationScoped
public class OrderService {

    // ===== Field или setter
    @Inject
    UserService userService;

    // ===== Конструктор
    private final PaymentService paymentService;

    @Inject
    public OrderService(PaymentService paymentService) {
        this.paymentService = paymentService;
    }

}
```

Contexts and Dependency Injection

```
@ApplicationScoped
public class OrderService {

    // ===== Field или setter
    @Inject
    UserService userService;

    // ===== Конструктор
    private final PaymentService paymentService;

    @Inject
    public OrderService(PaymentService paymentService) {
        this.paymentService = paymentService;
    }

    // ===== Метод
    @Inject
    void configure(ConfigService config, LogService log) {
        // Вызывается после создания объекта
    }
}
```

Contexts and Dependency Injection

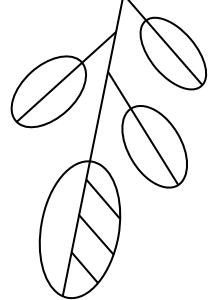
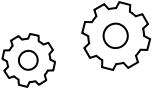
@RequestScoped

```
public class RequestContext {  
    private String requestId =  
        UUID.randomUUID().toString();  
  
    public String getRequestId() {  
        return requestId;  
    }  
}
```

Новый экземпляр, например для каждого HTTP запроса



Contexts and Dependency Injection

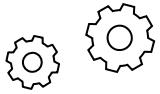


@Dependent - создается новый экземпляр
каждый раз при инъекции

@Singleton - похож на **@ApplicationScoped**,
но не создается proxy



ARC / Jakarta Interceptors



@Transactional

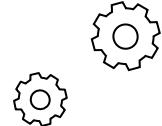
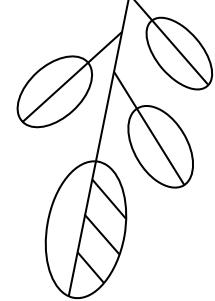
@CircuitBreaker

@RateLimit

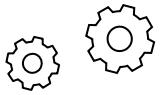
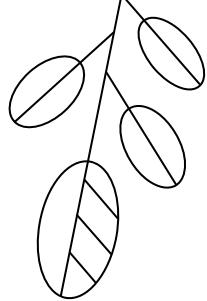
@Timeout

@Retry

@Fallback



ARC / Jakarta Interceptors



@Transactional

Arc во время **build-time**:

@CircuitBreaker

@RateLimit

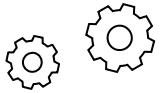
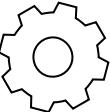
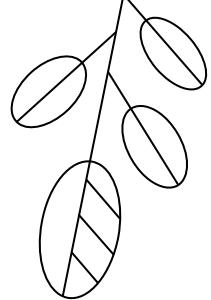
@Timeout

@Retry

@Fallback



ARC / Jakarta Interceptors



@Transactional

Arc во время build-time:

- Находит всё, что имеет аннотации-интерсепторы и тд

@CircuitBreaker

@RateLimit

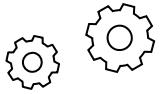
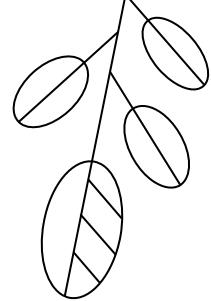
@Timeout

@Retry

@Fallback



ARC / Jakarta Interceptors



@Transactional

@CircuitBreaker

@RateLimit

@Timeout

@Retry

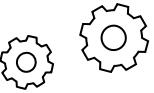
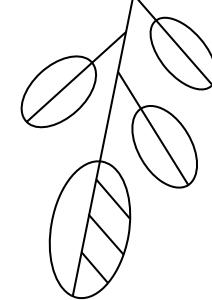
@Fallback

Arc во время build-time:

- Находит всё, что имеет аннотации-интэрсепторы и тд
- Генерирует байткод, который не вызывает метод напрямую, а вставляет вызов цепочки интэрсепторов



ARC / Jakarta Interceptors



@Transactional

@CircuitBreaker

@RateLimit

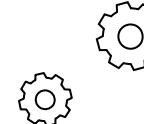
@Timeout

@Retry

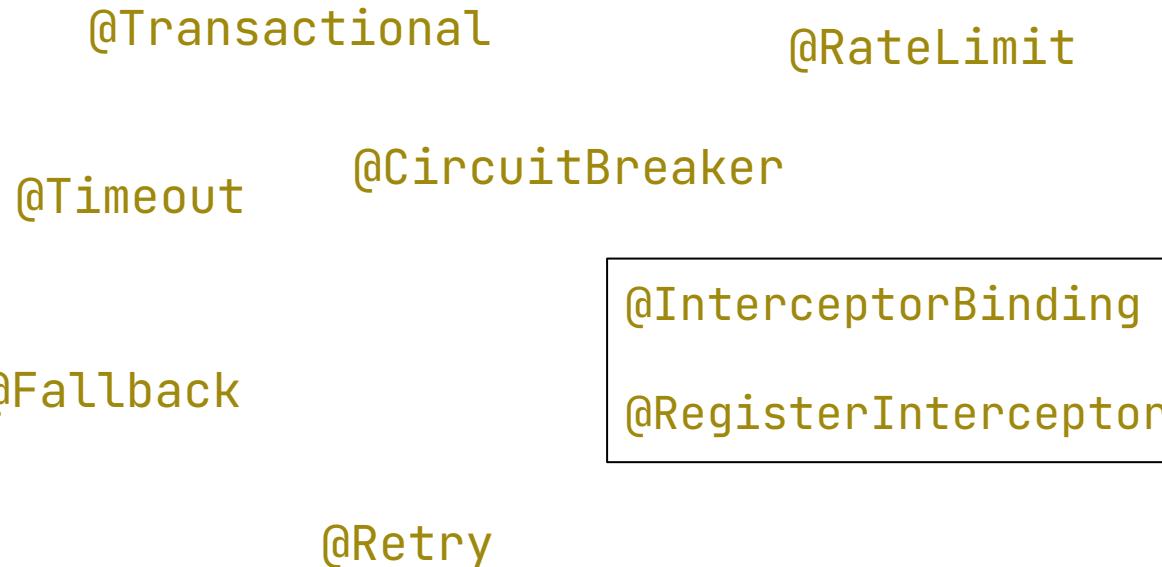
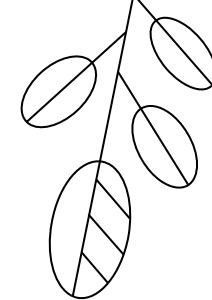
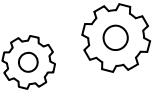
@Fallback

Arc во время build-time:

- Находит всё, что имеет аннотации-интерсепторы и тд
- Генерирует байткод, который не вызывает метод напрямую, а вставляет вызов цепочки интерсепторов
- В runtime, при вызове метода, сначала выполняются интерсепторы, потом и сам метод



ARC / Jakarta Interceptors



Bytecode-gen

```
@ApplicationScoped  
class ProcessingService {  
    fun createCommitAndRepoEntities() {  
        ...  
    }  
}
```

Subclass - реальный объект

The screenshot shows a Java decompiled class named `ProcessingService_Subclass.class`. The code is as follows:

```
Decompiled .class file, bytecode version: 55.0 (Java 11)

package dev.sourcecraft.api.utils;

import ...

// $FF: synthetic class
public class ProcessingService_Subclass extends ProcessingService implements Subclass { no usages
    private volatile boolean arc$constructed;
    private InterceptedMethodMetadata arc$1;
    private InterceptedMethodMetadata arc$2;
    private InterceptedMethodMetadata arc$3;
    private InterceptedMethodMetadata arc$4;
    private InterceptedMethodMetadata arc$5;
    private InterceptedMethodMetadata arc$6;
    private InterceptedMethodMetadata arc$7;
    private InterceptedMethodMetadata arc$8;
    private InterceptedMethodMetadata arc$9;
    private InterceptedMethodMetadata arc$10;
    private InterceptedMethodMetadata arc$11;
    private InterceptedMethodMetadata arc$12;
    private InterceptedMethodMetadata arc$13;
    private InterceptedMethodMetadata arc$14;
    private InterceptedMethodMetadata arc$15;
```

The left sidebar shows a tree view of the project structure, with `ProcessingService_Subclass` selected. The bottom right corner of the window has a circular icon.

Subclass - интерсепторы в логике

ProcessingService_Subclass.class

Decompiled .class file, bytecode version: 55.0 (Java 11)

```
public class ProcessingService_Subclass extends ProcessingService implements Subclass { no usages
    private void arc$initMetadata0(Map var1, Map var2) {
        Method var62 = Reflections.findMethod(ProcessingService.class, methodName: "setS3Service", va
        ProcessingService_Subclass$$function$14 var63 = new ProcessingService_Subclass$$function$14
        InterceptedMethodMetadata var64 = new InterceptedMethodMetadata((List)var5, var62, (Set)var7,
        this.arc$14 = var64;
        Class[] var65 = new Class[]{ScanResultsPersistenceService.class};
        Method var66 = Reflections.findMethod(ProcessingService.class, methodName: "setScanResultsPer
        ProcessingService_Subclass$$function$15 var67 = new ProcessingService_Subclass$$function$15
        InterceptedMethodMetadata var68 = new InterceptedMethodMetadata((List)var5, var66, (Set)var7,
        this.arc$15 = var68;
    }

    public CommitRepo createCommitAndRepoEntities(String var1, String var2, long var3) {
        Object[] var5 = new Object[]{var1, var2, var3};
        if (this.arc$constructed) {
            try {
                InterceptedMethodMetadata var6 = this.arc$1;
                return (CommitRepo)InvocationContexts.performAroundInvoke(target: this, var5, var6);
            } catch (RuntimeException var8) {
                throw (Throwable)var8;
            }
        } else {
            return this.createCommitAndRepoEntities$$superforward(var1, var2, var3);
        }
    }
}
```

Bean-конструктор

The screenshot shows a Java decompiled class named `ProcessingService_Bean.class`. The code implements `InjectableBean` and `Supplier`. It contains several nested try blocks, each attempting to inject a different provider. The providers are obtained from `InterceptorProviderSupplier` and `injectProviderSupplier`. The code uses reflection to get methods from providers and set them on the `ProcessingService` object. The class also contains a private method `doCreate` which creates a `ProcessingService_Subclass` instance.

```
public class ProcessingService_Bean implements InjectableBean, Supplier { no usages

    public Object get() { return this; }

    private ProcessingService doCreate(CreationalContext var1) {
        Object var2 = this.interceptorProviderSupplier6.get();
        ProcessingService var5 = (ProcessingService)(new ProcessingService_Subclass(var1, (InjectableReferenceProvider)var2));
        try {
            Object var3 = this.injectProviderSupplier1.get();
            CreationalContextImpl var4 = CreationalContextImpl.child((InjectableReferenceProvider)var3);
            Object var6 = ((InjectableReferenceProvider)var3).get((CreationalContext)var4);
            var5.reportParsingService = (ReportParsingService)var6;
        } catch (RuntimeException var28) {
            throw (Throwable)(new RuntimeException("Error injecting dev.sourcecraft.utils.ReportParsingService"));
        }

        try {
            Object var8 = this.injectProviderSupplier2.get();
            CreationalContextImpl var9 = CreationalContextImpl.child((InjectableReferenceProvider)var8);
            Object var10 = ((InjectableReferenceProvider)var8).get((CreationalContext)var9);
            var5.repository4Commit = (Repository4Commit)var10;
        } catch (RuntimeException var27) {
            throw (Throwable)(new RuntimeException("Error injecting dev.sourcecraft.repositories.Repository4Commit"));
        }

        try {
            Object var12 = this.injectProviderSupplier3.get();
        }
    }
}
```

Client-proxy

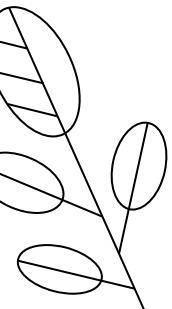
The screenshot shows a Java decompiled class file named `ProcessingService_ClientProxy.class`. The code is annotated with line numbers and syntax highlighting. The class implements the `ClientProxy` interface and extends `ProcessingService`. It uses `ArcContainer` to manage beans and contexts.

```
1 > /...
5
6 package dev.sourcecraft.api.utils;
7
8 > import ...
24
25 // $FF: synthetic class
26 public class ProcessingService_ClientProxy extends ProcessingService implements ClientProxy { no usage
27     private final InjectableBean bean;
28     private final InjectableContext context;
29
30     public ProcessingService_ClientProxy(String var1) {
31         ArcContainer var3 = Arc.container();
32         InjectableBean var2 = var3.bean(var1);
33         this.bean = var2;
34         Class var4 = var2.getScope();
35         Object var5 = var3.getContexts(var4).get(0);
36         this.context = (InjectableContext)var5;
37     }
38
39     private ProcessingService arc$delegate() {
40         InjectableBean var1 = this.bean;
41         return (ProcessingService)ClientProxies.getApplicationScopedDelegate(this.context, var1);
42     }
43
44 ⓘ >
45
46     public Object arc_contextualInstance() { return this.arc$delegate(); }
47 }
```

META-INF/jandex.idx

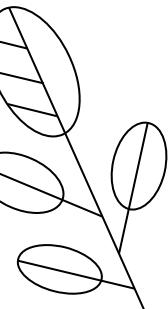
@Unremovable

Что отдаленно напоминает?





подсказка



JEP 483: AOT Class Loading & Linking

JEP 514: AOT Command-Line Ergonomics



JEP 483: AOT Class Loading & Linking

JEP 514: AOT Command-Line Ergonomics

- AOT cache
- *Не требуется обнаруживать, линковать, отдельно загружать классы на старте*



JEP 483: AOT Class Loading & Linking

JEP 514: AOT Command-Line Ergonomics

- AOT cache
- Не требуется обнаруживать, линковать, отдельно загружать классы на старте
- *Spring PetClinic*  **42%** (21k classes)
- Поможет ли Quarkus'у?!



Вкатываемся в UX!

CONFIGURE YOUR APPLICATION

Group org.acme

Artifact code-with-quarkus

Build Tool Maven

 MORE OPTIONS 0

Generate your application (Esc + ←)



Search ▾



Search across all available extensions: rest, hibernate, web, data...

 View the full list of available extensions ▾

Start from an extensions preset



REST service

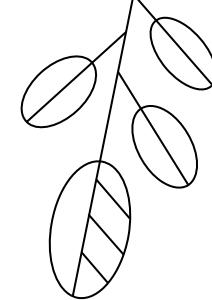
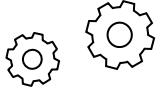


REST service with database

Event driven service with
Kafka

Command-line tool

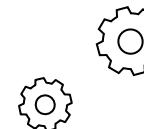
Что мне сразу бросилось в глаза?



```
% quarkus create && cd code-with-quarkus  
Creating an app (default project type, see --help).  
Looking for the newly published extensions in registry.quarkus.io
```

```
-----  
applying codestarts...  
java  
maven  
quarkus  
config-properties  
tooling-dockerfiles  
tooling-maven-wrapper  
rest-codestart
```

```
-----  
[SUCCESS] ✅ quarkus project has been successfully generated in:  
→ /Users/akuleshov7/projects/code-with-quarkus
```



The screenshot shows the IntelliJ IDEA interface with a Java project named "code-with-quarkus". The project structure is displayed in the left-hand navigation pane, and the README.md file is open in the main editor area.

Project Structure:

- code-with-quarkus (~/projects/code-with-quarkus)
 - .idea
 - .mvn
 - src
 - main
 - docker
 - Dockerfile.jvm
 - Dockerfile.legacy-jar
 - Dockerfile.native
 - Dockerfile.native-micro
 - java
 - org.acme
 - GreetingResource
 - resources
 - test
 - java
 - org.acme
 - GreetingResourceIT
 - GreetingResourceTest- .dockignore
- .gitignore
- mvnw
- mvnw.cmd
- pom.xml
- README.md
- External Libraries
- Scratches and Consoles

README.md Content:

```
M README.md x G GreetingResource.java

# code-with-quarkus

This project uses Quarkus, the Supersonic Subatomic Java Framework.

If you want to learn more about Quarkus, please visit its website: <https://quarkus.io/>

## Running the application in dev mode

You can run your application in dev mode that enables live coding using:

shell script
./mvnw quarkus:dev
```

> **_NOTE:_** Quarkus now ships with a Dev UI, which is available in dev mode only at <http://localhost:8080/q/dev>.

Packaging and running the application

The application can be packaged using:

shell script
./mvnw package
```

It produces the `quarkus-run.jar` file in the `target/quarkus-app/` directory. Be aware that it's not an _über-jar_ as the dependencies are copied into the `target/quarkus-app/` directory.

The application is now runnable using `java -jar target/quarkus-app/quarkus-run.jar`.

If you want to build an _über-jar_, execute the following command:

shell script
./mvnw package -Dquarkus.package.jar.type=uber-jar
```

```

# ./mvnw quarkus:dev

```
--/---\ / / / - | / - \ / / / / / / --/
-/ / / / / / / _/ , _/ , < / / / / \ \ \ /
--____/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
2025-08-19 23:59:46,859 INFO [io.quarkus] (Quarkus Main Thread) code-with-quarkus 1.0.0-SNAPSHOT on JVM (powered by Quarkus 3.25.3)
2025-08-19 23:59:46,860 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2025-08-19 23:59:46,860 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, compose, kubernetes, rest, smallrye-contex
t-propagation, smallrye-health, vertx]

--
Tests paused
Press [e] to edit command line args (currently ''), [r] to resume testing, [o] Toggle test output, [:] for the terminal, [h] for mor
e options>
```

quarkus.test.continuous-testing=enabled

# ./mvnw quarkus:dev



```
--
2025-08-19 23:59:46,859 INFO [io.quarkus] (Quarkus Main Thread) code-with-quarkus 1.0.0-SNAPSHOT on JVM (powered by Quarkus 3.25.3)
2025-08-19 23:59:46,860 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2025-08-19 23:59:46,860 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, compose, kubernetes, rest, smallrye-conte
xt-propagation, smallrye-health, vertx]
```

--  
**Tests paused**

```
Press [e] to edit command line args (currently ''), [r] to resume testing, [o] Toggle test output, [:] for the terminal, [h] for mor
e options>
```

quarkus.test.continuous-testing=enabled

# ./mvnw quarkus:dev



```
--
2025-08-19 23:59:46,859 INFO [io.quarkus] (Quarkus Main Thread) code-with-quarkus 1.0.0-SNAPSHOT on JVM (powered by Quarkus 3.25.3)
2025-08-19 23:59:46,860 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2025-08-19 23:59:46,860 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, compose, kubernetes, rest, smallrye-conte
xt-propagation, smallrye-health, vertx]
```

--  
**Tests paused**

```
Press [e] to edit command line args (currently ''), [r] to resume testing, [o] Toggle test output, [:] for the terminal, [h] for mor
e options>
```

quarkus.test.continuous-testing=enabled

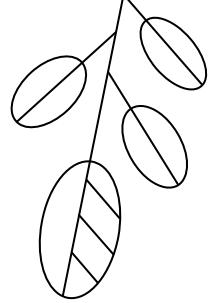
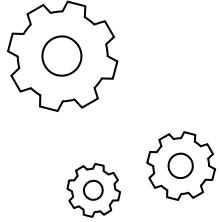
- + Hot reload, но этим пользователей Spring не удивить

## RuntimeUpdatesProcessor

<https://github.com/quarkusio/quarkus/blob/803ec3a617795cc036ecfd0160bb011e9e60c248/core/deployment/src/main/java/io/quarkus/deployment/dev/RuntimeUpdatesProcessor.java>



Сразу куча  
девсервисов из  
коробки





## Configuration

Extensions

Configuration

Workspace

Endpoints

Continuous Testing

Dev Services

Build Metrics

Readme

Dependencies

| Phase | Name                                            | Value                                  |
|-------|-------------------------------------------------|----------------------------------------|
| 🔒     | quarkus.analytics.disabled                      | <input type="checkbox"/>               |
| 🔒     | quarkus.analytics.timeout                       | 3000                                   |
| 🔒     | quarkus.analytics.uri.base                      |                                        |
| 🔒     | quarkus.application.name                        | code-with-quarkus                      |
| 🔒     | quarkus.application.ui-header                   | {applicationName} (powered by Quarkus) |
| 🔒     | quarkus.application.version                     | 1.0.0-SNAPSHOT                         |
| 🔒     | quarkus.arc.auto-inject-fields                  | <input checked="" type="checkbox"/>    |
| 🔒     | quarkus.arc.auto-producer-methods               | <input checked="" type="checkbox"/>    |
| 🔒     | quarkus.arc.context-propagation.enabled         | <input checked="" type="checkbox"/>    |
| 🔒     | quarkus.arc.detect-unused-false-positives       | <input checked="" type="checkbox"/>    |
| 🔒     | quarkus.arc.detect-wrong-annotations            | <input checked="" type="checkbox"/>    |
| 🔒     | quarkus.arc.dev-mode.generate-dependency-graphs |                                        |
| 🔒     | quarkus.arc.dev-mode.monitoring-enabled         | <input type="checkbox"/>               |
| 🔒     | quarkus.arc.exclude-dependency.*.artifact-id    |                                        |
| 🔒     | quarkus.arc.exclude-dependency.*.classifier     |                                        |
| 🔒     | quarkus.arc.exclude-dependency.*.group-id       |                                        |





Extensions

Configuration

Workspace

Endpoints

Continuous Testing

Dev Services

Build Metrics

Readme

Dependencies

## Endpoints

## Resource Endpoints

| URL    | Description                        |
|--------|------------------------------------|
| /hello | GET (produces:text/plain;charset=U |

## Additional endpoints

| URL                                       | Description            |
|-------------------------------------------|------------------------|
| http://localhost:9003/q/arc               | CDI Overview           |
| http://localhost:9003/q/arc/beans         | Active CDI Beans       |
| http://localhost:9003/q/arc/observers     | Active CDI Observers   |
| http://localhost:9003/q/arc/removed-beans | Removed CDI Beans      |
| http://localhost:9003/q/dev-ui            | Dev UI                 |
| http://localhost:9003/q/health            | Health Check           |
| http://localhost:9003/q/health-ui         | Health UI              |
| http://localhost:9003/q/health/group      | Health Group Check     |
| http://localhost:9003/q/health/group/*    | Health Group Check     |
| http://localhost:9003/q/health/live       | Health Liveness Check  |
| http://localhost:9003/q/health/ready      | Health Readiness Check |
| http://localhost:9003/q/health/started    | Health Started Check   |





## Continuous Testing

[Run all](#) [Run failed](#)  Only run failing tests  Display tags (if available)

Extensions

Configuration

Workspace

Endpoints

Continuous Testing

Dev Services

Build Metrics

Readme

Dependencies

Tags ▾ Test Class ▾ Name ▾  
 Q </> org.acme.GreetingResourceTest testHelloEndpoint()





Dev UI

## Workspace



Extensions



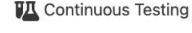
Configuration



Workspace



Endpoints



Continuous Testing



Dev Services



Build Metrics



Readme



Dependencies



GreetingResource.java



```
1 package org.acme;
2
3 import jakarta.ws.rs.GET;
4 import jakarta.ws.rs.Path;
5 import jakarta.ws.rs.Produces;
6 import jakarta.ws.rs.core.MediaType;
7
8 @Path("/hello")
9 public class GreetingResource {
10
11 @GET
12 @Produces(MediaType.TEXT_PLAIN)
13 public String hello() {
14 return "Hello from Quarkus REST";
15 }
16 }
17
```



src/main/java/org/acme/GreetingResource.java saved successfully



## Dependencies

-  Extensions
  -  Configuration
  -  Workspace
  -  Endpoints
  -  Continuous
  -  Dev Services
  -  Build Metrics
  -  Readme
  -  Dependencies



Не “паришься” по поводу  
локального/тестового запуска

# Не “паришься” по поводу локального/тестового запуска

```
quarkus.datasource.devservices.enabled=true
quarkus.datasource.devservices.image-name=postgres:15
```

# Не “паришься” по поводу локального запуска или тестов

```
quarkus.datasource.devservices.enabled=true
quarkus.datasource.devservices.image-name=postgres:15
```

```
quarkus.s3.devservices.enabled=true
quarkus.aws.devservices.localstack.image-name=localstack/localstack:3.3.0
```

# Не “паришься” по поводу локального запуска или тестов

```
quarkus.datasource.devservices.enabled=true
quarkus.datasource.devservices.image-name=postgres:15
```

```
quarkus.s3.devservices.enabled=true
quarkus.aws.devservices.localstack.image-name=localstack/localstack:3.3.0
```

```
quarkus.kafka.devservices.enabled=true
quarkus.kafka.devservices.image-name=confluentinc/cp-kafka:7.4.0
```

# gRPC из коробки

Сразу **три** варианта:

1. старая Vert.x gRPC имплементация с **отдельным** gRPC сервером
2. новая Vert.x gRPC имплементаци **поверх текущего** HTTP сервера (тот же порт, без оверхеда)
3. xDS gRPC обертка над grpc-java с **отдельным** gRPC Netty сервером

# gRPC из коробки

Сразу **три** варианта:

1. старая Vert.x gRPC имплементация с отдельным gRPC сервером
2. новая Vert.x gRPC имплементаци поверх текущего HTTP сервера (**тот же порт**, без оверхеда)
3. xDS gRPC обертка над grpc-java с отдельным gRPC Netty сервером

# gRPC из коробки

```
option java_package = "hello";

service GreeterService {
 rpc SayHello (HelloRequest) returns (HelloReply);
}
```

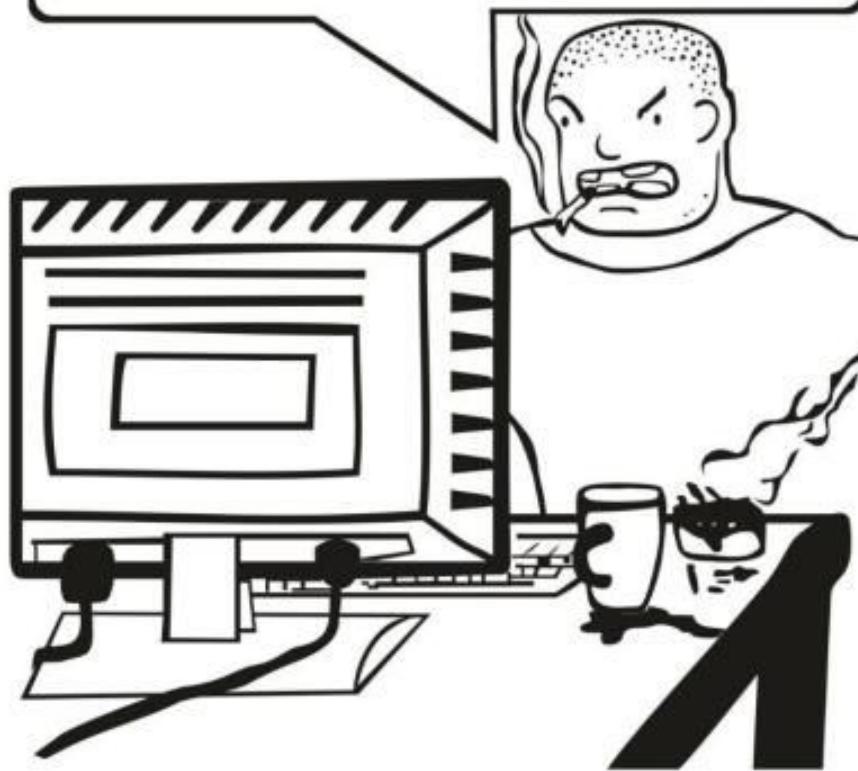
# gRPC из коробки

```
import io.quarkus.grpc.GrpcService;
import hello.Greeter;

@GrpcService
public class HelloService implements GreeterService {

 @Override
 public Uni<HelloReply> sayHello(HelloRequest request) {
 return Uni.createFrom().item(() →
 HelloReply.newBuilder().setMessage(
 "Hello " + request.getName()).build()
);
 }
}
```

Так, ну ладно, разобрались.  
А че они там писали про  
**Kubernetes Native?**



# SUPERSONIC/ SUBATOMIC/ JAVA

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

Now Available

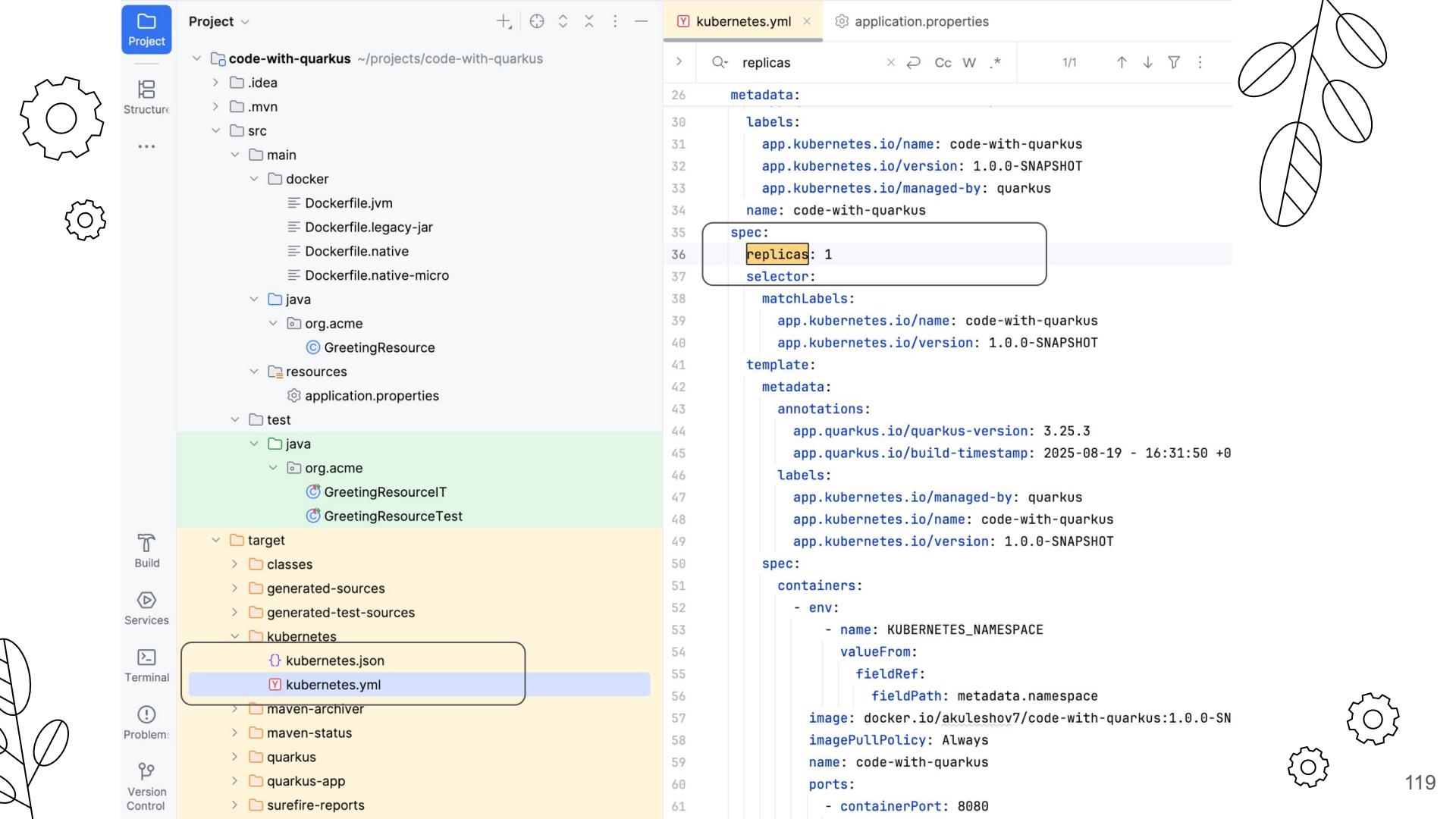
**QUARKUS 3.25.3**

[Read the release notes](#)

# Kubernetes Native

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

./mvnw package

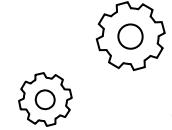
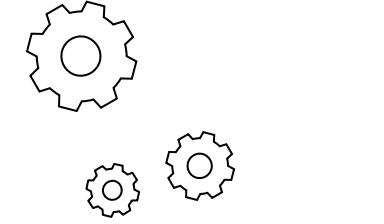
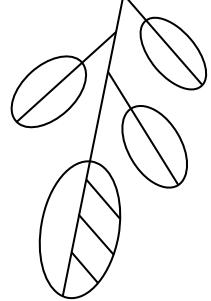


Kubernetes.yml application.properties

```
> Q- replicas x ↻ Cc W .*
26 metadata:
27 labels:
28 app.kubernetes.io/name: code-with-quarkus
29 app.kubernetes.io/version: 1.0.0-SNAPSHOT
30 app.kubernetes.io/managed-by: quarkus
31 name: code-with-quarkus
32
33 spec:
34 replicas: 1
35 selector:
36 matchLabels:
37 app.kubernetes.io/name: code-with-quarkus
38 app.kubernetes.io/version: 1.0.0-SNAPSHOT
39
40 template:
41 metadata:
42 annotations:
43 app.quarkus.io/quarkus-version: 3.25.3
44 app.quarkus.io/build-timestamp: 2025-08-19 - 16:31:50 +0
45
46 labels:
47 app.kubernetes.io/managed-by: quarkus
48 app.kubernetes.io/name: code-with-quarkus
49 app.kubernetes.io/version: 1.0.0-SNAPSHOT
50
51 spec:
52 containers:
53 - env:
54 - name: KUBERNETES_NAMESPACE
55 valueFrom:
56 fieldRef:
57 fieldPath: metadata.namespace
58
59 image: docker.io/akuleshov7/code-with-quarkus:1.0.0-SN
60 imagePullPolicy: Always
61 name: code-with-quarkus
62 ports:
63 - containerPort: 8080
```

# application.properties

- + quarkus.kubernetes.replicas=7



# application.properties

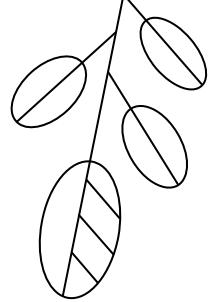
- + quarkus.kubernetes.replicas=7

```
spec:
 replicas: 7
 ...
```

# application.properties

- + quarkus.kubernetes.replicas=7
- + dependency quarkus-smallrye-health

```
spec:
 replicas: 7
 ...
```

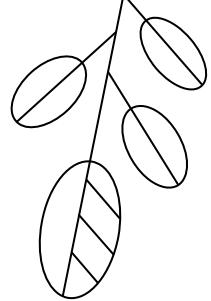


# application.properties

- + quarkus.kubernetes.replicas=7
- + dependency quarkus-smallrye-health

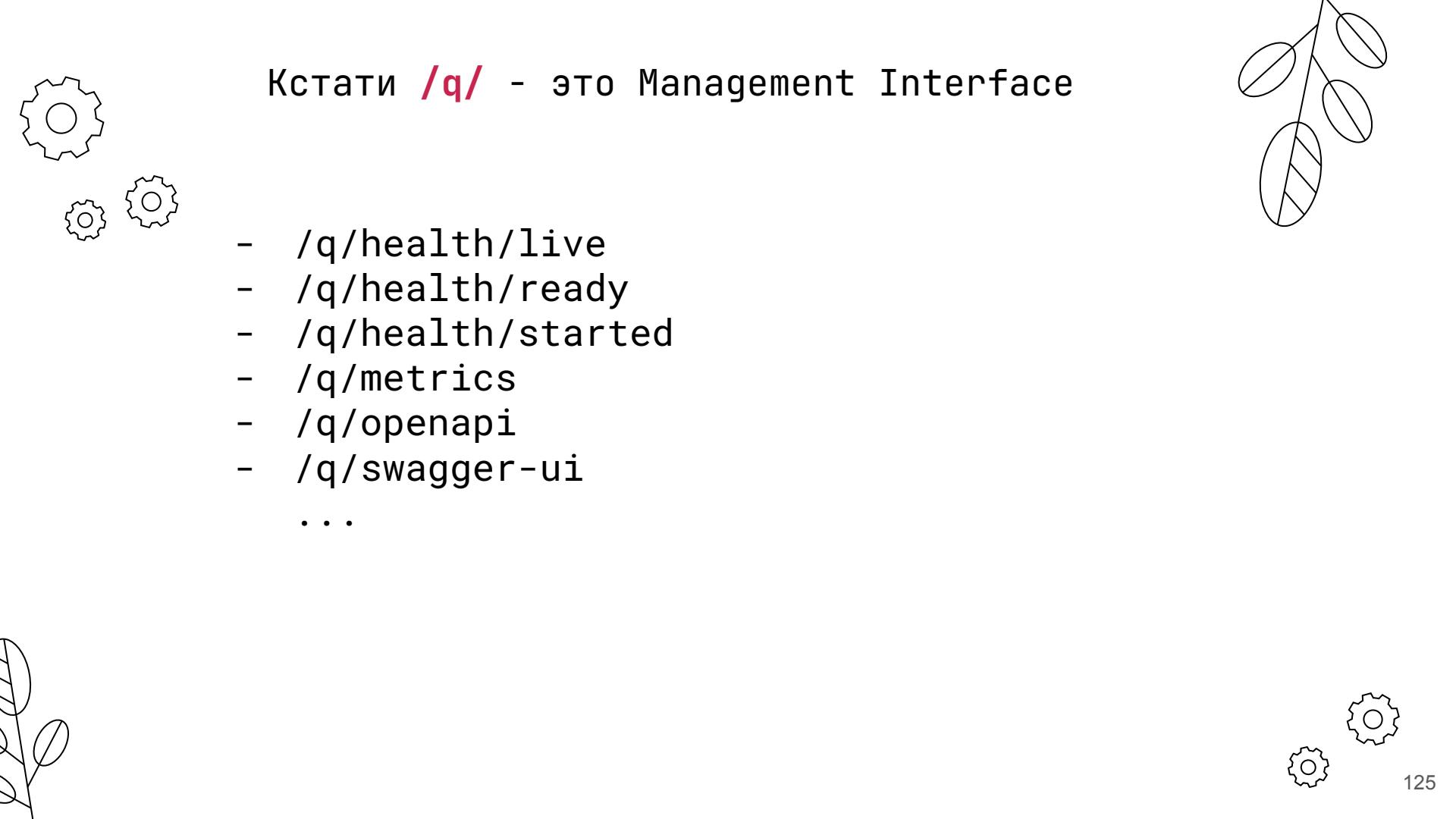


```
spec:
 replicas: 7
...
livenessProbe:
...
 httpGet:
 path: /q/health/live
 port: 8080
 scheme: HTTP
...
readinessProbe:
 failureThreshold: 3
 httpGet:
 path: /q/health/ready
 port: 8080
 scheme: HTTP
```



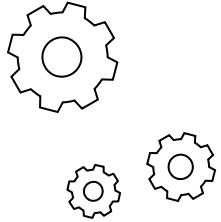
```
@Liveness
@ApplicationScoped
public class AppLivenessCheck implements HealthCheck {
 @Override
 public HealthCheckResponse call() {
 return HealthCheckResponse.named("app-liveness")
 .status(isApplicationRunning())
 .build();
 }
}

@Readiness
@ApplicationScoped
public class DatabaseReadinessCheck implements HealthCheck {
 @Override
 public HealthCheckResponse call() {
 if(isDatabaseReady()) {
 return HealthCheckResponse.up("database-readiness");
 } else {
 return HealthCheckResponse.down("database-readiness");
 }
 }
}
```

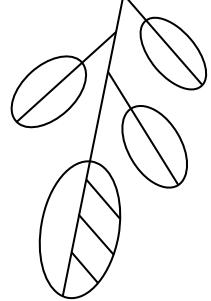


Кстати */q/* - это Management Interface

- /q/health/live
- /q/health/ready
- /q/health.started
- /q/metrics
- /q/openapi
- /q/swagger-ui
- ...



И **/q/** легко вынести на отдельный HTTP сервер/порт, чтобы разделить от основной логики



```
quarkus.management.enabled=true
```

```
на 0.0.0.0 по дефолту
quarkus.management.port=9004
```





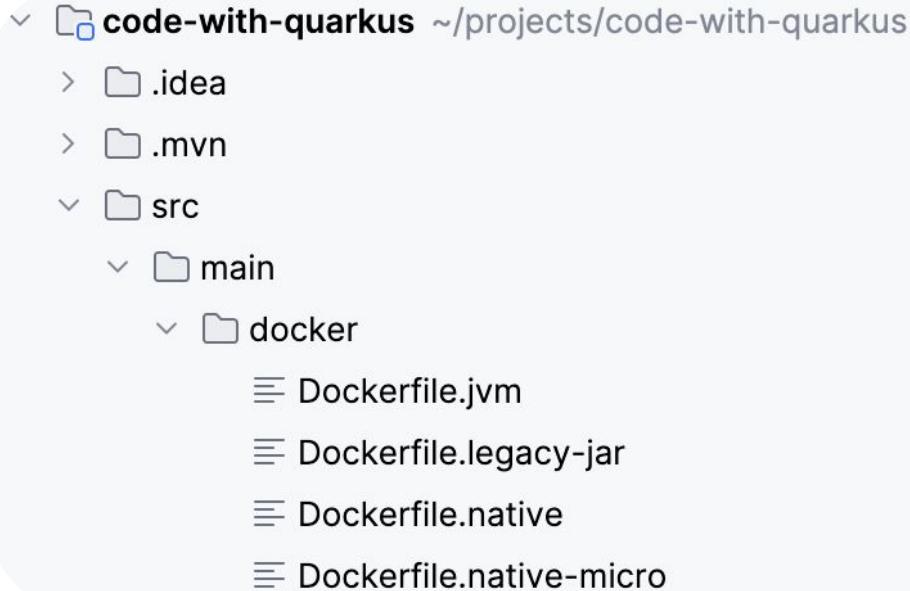
# SUPERSONIC/ SUBATOMIC/ JAVA

A Kubernetes Native Java stack tailored for OpenJDK HotSpot and  
GraalVM, crafted from the best of breed Java libraries and standards.

Now Available

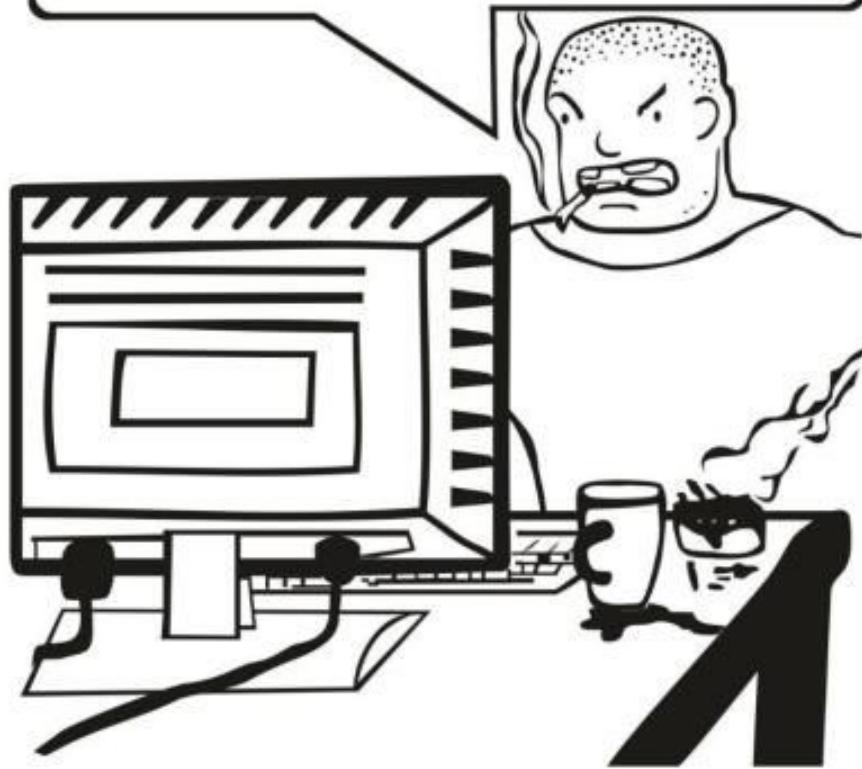
**QUARKUS 3.25.3**

[Read the release notes](#)



```
./mvnw package -Dnative -Dquarkus.native.container-build=true
```

Ладно, ясно, а в сортах  
чего они разбираются?





# SUPERSONIC/ SUBATOMIC/ JAVA

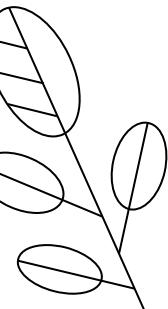
A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

Now Available

**QUARKUS 3.25.3**

[Read the release notes](#)

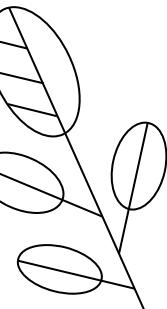
# Ваши любимые ORM





# Ваши любимые ORM

Урезанный Hibernate ORM с Panache





pro.jvm

6 136 members, 1903 online



...

Dima 💙

admin

Хибер был создан, чтобы Михалча 30летнюю ипотеку за 3 месяца закрывал

думайте



12



1



16 13:31

GIF



13:31

133

# Active Record Pattern

```
@Entity
public class Person extends PanacheEntity {
 public String name;
}
```

# Active Record Pattern

```
@Entity
public class Person extends PanacheEntity {
 public String name;
}
```

```
Person p = new Person();
p.name = "Dima";
p.persist();
```

```
List<Person> persons = Person.listAll();
Person byName = Person.find("name", "Dima").firstResult();
Person byName = Person.find("name =?1", "Dima").firstResult();
```

# Active Record Pattern

```
@Entity
public class Person extends PanacheEntity {
 public String name;
}
```

```
Person p = new Person();
p.name = "Dima";
p.persist();
```

```
List<Person> persons = Person.listAll();
Person byName = Person.find("name", "Dima").firstResult();
Person byName = Person.find("name =?1", "Dima").firstResult();
```

# Repository Pattern

```
@ApplicationScoped
public class PersonRepository
 implements PanacheRepository<Person> {
 ...
}

repo.persist(new Person("Dima", 30));
List<Person> people = repo.listAll();
```

# Datasource

```
Default
quarkus.datasource.db-kind=postgresql
quarkus.datasource.username=user1
quarkus.datasource.password=pass1
quarkus.datasource.jdbc.url=jdbc:postgresql://

Named datasource: orders
quarkus.datasource.orders.db-kind=mysql
quarkus.datasource.orders.username=user2
quarkus.datasource.orders.password=pass2
quarkus.datasource.orders.jdbc.url=jdbc:mysql://
```

# Datasource



```
Default
quarkus.datasource.db-kind=postgresql
quarkus.datasource.username=user1
quarkus.datasource.password=pass1
quarkus.datasource.jdbc.url=jdbc:postgresql://

Named datasource: orders
quarkus.datasource.orders.db-kind=mysql
quarkus.datasource.orders.username=user2
quarkus.datasource.orders.password=pass2
quarkus.datasource.orders.jdbc.url=jdbc:mysql://
```

Пул JDBC коннекшенов через Agroal (свой Hikari)

```
@ApplicationScoped
public class MyService {

 @Inject
 @PersistenceUnit("orders")
 EntityManager em;

 public Person findById(Long id) {
 return em.find(Person.class, id);
 }
}
```

jakarta.persistence

# Native Query

```
@Transactional
public List<UserDTO> findUsersByIds(List<Long> userIds) {
 String queryStr = """
 SELECT id, name
 FROM users
 WHERE id IN (:userIds)
 """;

 Query query = em.createNativeQuery(queryStr, "UserMapping");
 query.setParameter("userIds", userIds);

 return query.getResultList();
}
```

## Есть и “нестандартная” Native Query

```
@Entity
@NamedNativeQuery(
 name = "Person.getByAge",
 query = "SELECT * FROM person WHERE age > :age",
 resultClass = Person.class
)
public class Person extends PanacheEntity {
 public String name;
 public int age;
}
```

# Что еще?

# Что еще?



@Transactional

JBoss / Narayana Transaction Manager

# Что еще?



@Transactional

JBoss / Narayana Transaction Manager

hibernate-reactive-panache



- [ChecklistForNewProjects](#)
- [DocumentingYourExtension](#)
- [Home](#)
- [Infrastructure for Quarkus 3.x: Manual upgrade](#)
- [Migrating to the Sonatype Portal API using JReleaser](#)
- [Release](#)
- [NamingConventions](#)
- [Infrastructure for Quarkus 3.x](#)

<https://hub.quarkiverse.io/>  
<https://github.com/quarkiverse>

## Welcome to the Quarkiverse Hub!

Quarkus is a Kubernetes Native Java stack tailored for the OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards, as well as an ever growing ecosystem of its extensions. At the beginning we were accepting all the contributed extensions in the core Quarkus repository. Eventually though it grew too large and lead to a maintenance overhead. In addition to that, in some cases it's simply not always sensible to include an extension into the main Quarkus repository. For example, an extension might want to have a release cadence which is independent of the core repository's lifecycle.

Quarkiverse is our solution to create a "home" for such extensions.

## What is Quarkiverse

The [Quarkiverse GitHub organization](#) provides repository hosting (including build, CI and release publishing setup) for Quarkus extension projects contributed by the community.

Quarkus extensions hosted in the Quarkiverse organization can easily be included into the Quarkus extensions catalog available on [code.quarkus.io](https://code.quarkus.io), [extensions.quarkus.io](https://extensions.quarkus.io), and the Quarkus command line tools (such as `mvn quarkus:list-extensions`, `gradle listExtensions`). To stay listed, the only requirement is that the extension keeps functioning, stays up-to-date and cause no harm.

## Why Quarkiverse

In the early days and actually up until recently, Quarkus extensions contributed by the community members (including the core Quarkus t IntelliJ IDEA CE) welcomed in the main

### quarkus-jooq Public

Quarkus Jooq Extension

[quarkus-extension](#)

Java · Apache License 2.0 · 19 · 77 · 20 · 2 · Updated last week

### quarkus-mcp-servers

Public

Model Context Protocol Servers in Quarkus

[mcp](#) [quarkus-app](#)

Java · Apache License 2.0 · 41 · 163 · 16 · 6 · Updated 3 weeks ago

### quarkus-scala3 Public

Quarkus Extension to support Scala 3

[quarkus-extension](#)

Java · Apache License 2.0 · 6 · 59 · 2 · 9 · Updated on Jan 9

### quarkus-amazon-services

Public

Quarkus Amazon Services extensions

[quarkus-extension](#)

Java · Apache License 2.0 · 62 · 48 · 4 · 4 · Updated 1 hour ago

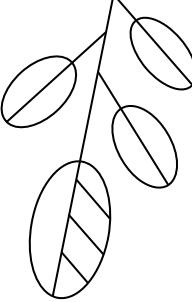
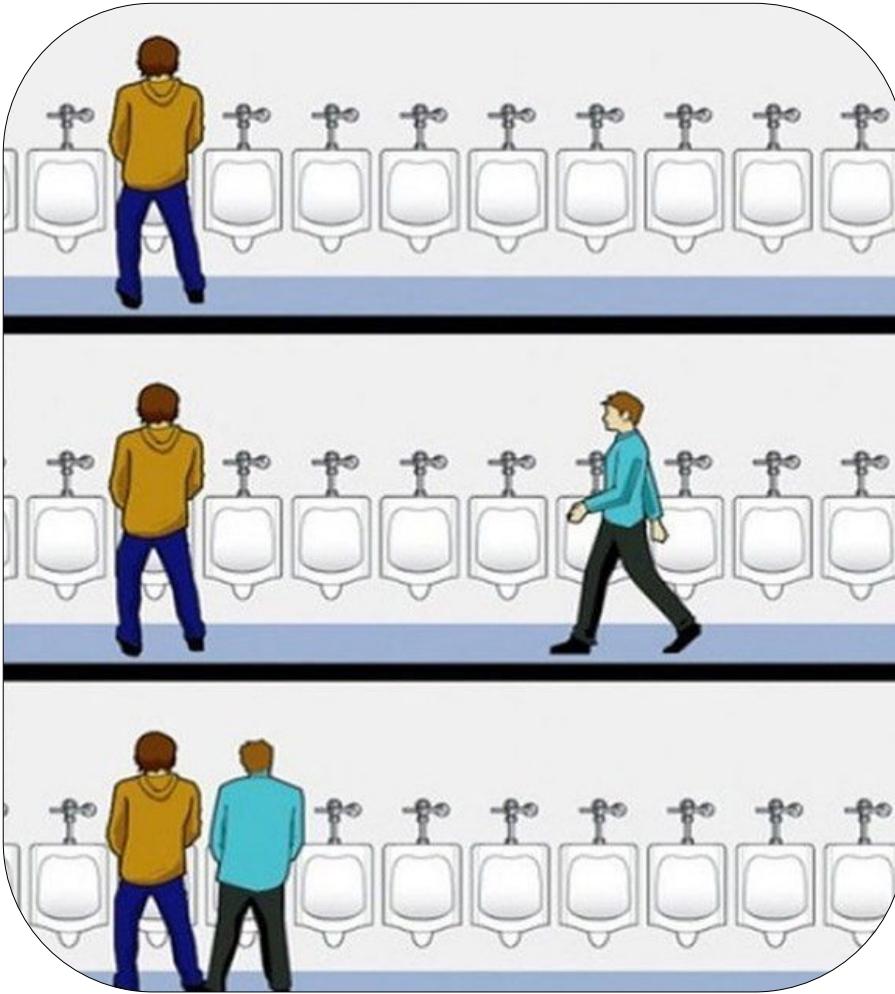
### quarkus-neo4j Public

Quarkus Neo4j extension

[neo4j](#) [graph](#) [graph-database](#) [quarkus-extension](#)

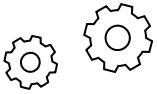
Java · Apache License 2.0 · 7 · 23 · 1 · 2 · Updated last week

# Выводы

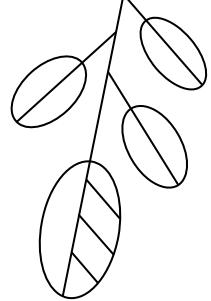




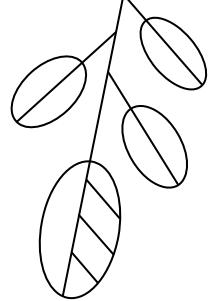
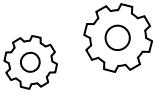
# Выводы



Quarkus - это:

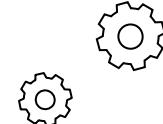


# Выводы



Quarkus - это:

- скорость



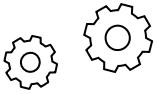
# Выводы

Quarkus - это:

- скорость
- отдельная экосистема

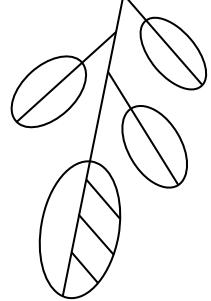


# Выводы

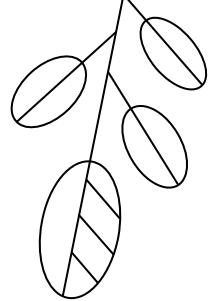
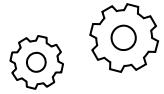


Quarkus - это:

- скорость
- отдельная экосистема
- делалось корпорацией для корпораций

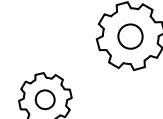


# Выводы

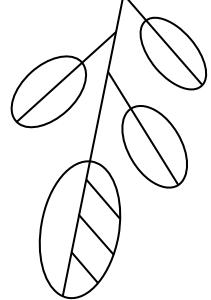
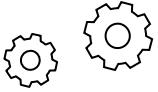


Quarkus - это:

- скорость
- отдельная экосистема
- делалось корпорацией для корпораций
- шероховатости, до сих пор полируется



# Выводы



Quarkus - это:

- скорость
- отдельная экосистема
- делалось корпорацией для корпораций
- шероховатости, до сих пор полируется
- быстро и безболезненно для PoC/MVP

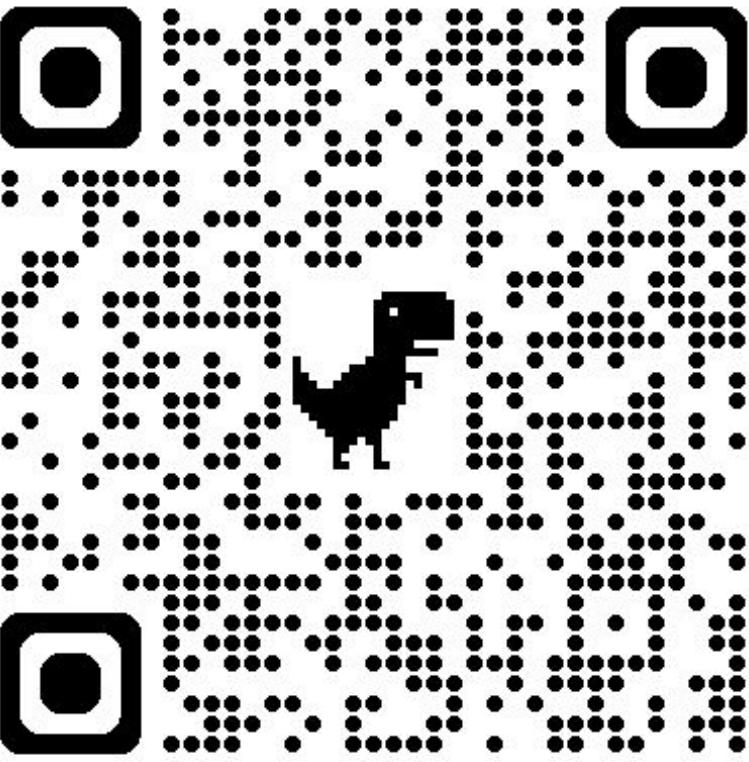




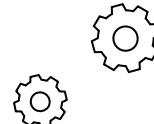
Спасибо!  
Q&A



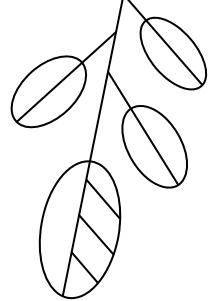
Обратная связь



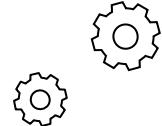
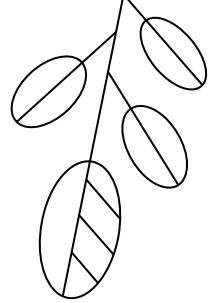
Мой канал



Андрей, с чем  
ты неожиданным  
сталкивался?



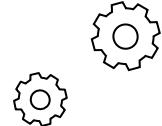
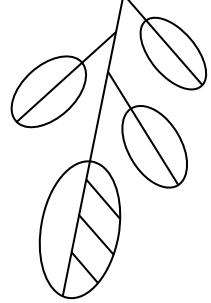
# 1



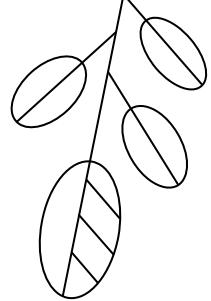
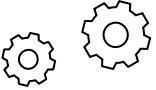


# JBoss Logging

2



# Столкновение с метриками



*By default, the metrics are exported using the Prometheus format **application/openmetrics-text**, you can revert to the former format by specifying the Accept request header to text/plain  
(curl -H "Accept: text/plain" localhost:8080/q/metrics/).*





# Configurable metrics format #49152

[New issue](#)[Open](#)

#49581



axidex opened 3 weeks ago

...

## Description

Some custom metrics exporters sidecars don't have ability to add custom headers to requests. It would be great if this could be added to the quarkus.micrometer config.

By default, the metrics are exported using the Prometheus format application/openmetrics-text, you can revert to the former format by specifying the Accept request header to text/plain (curl -H "Accept: text/plain" localhost:8080/q/metrics/).

## Implementation ideas

I would suggest something like this:

```
quarkus.micrometer.export.prometheus.format=plain
quarkus.micrometer.export.prometheus.format=openmetrics
```



4



axidex

added kind/enhancement 3 weeks ago



quarkus-bot

added area/config area/metrics area/smallrye 3 weeks ago

### Assignees

No one assigned

### Labels

[area/metrics](#) [kind/enhancement](#)

### Type

No type

### Projects

No projects

### Milestone

No milestone

### Relationships

None yet

### Development

[Code with agent mode](#)

Format property added in micrometer  
prometheus config  
quarkusio/quarkus

163

Open

Format property added in micrometer prometheus config #49581

axidex wants to merge 4 commits into [quarkusio:main](#) from [axidex:feature/micrometer-prom...](#)



## Configuration

```
Use legacy Prometheus text format by default
quarkus.micrometer.export.prometheus.format=plain

Use OpenMetrics format by default (default behavior)
quarkus.micrometer.export.prometheus.format=openmetrics
```



## Behavior

- When no Accept header is provided, the configured format is used
- When Accept header is provided (e.g., text/plain or application/openmetrics-text), it takes precedence over configuration
- This allows custom metrics exporters/sidecars that don't support custom Accept headers to work with the plain text format

## Issues

- Fixes [Configurable metrics format](#) #49152



quarkus-bot bot commented 5 days ago



df3c38f



...

/cc [@brunobat](#) (micrometer), [@ebullient](#) (micrometer)



axidex marked this pull request as ready for review 5 days ago

gastaldi reviewed 3 days ago

### Labels

[area/metrics](#)

### Projects

None yet

### Milestone

No milestone

### Development

Successfully merging this pull request may close these issues.

### [Configurable metrics format](#)

### Notifications

Customize

[Subscribe](#)

You're not receiving notifications from this thread.

### 2 participants



[View reviewed changes](#)

**Open**

Format property added in micrometer prometheus config #49581

axidex wants to merge 4 commits into [quarkusio:main](#) from [axidex:feature/micrometer-prom...](#)



## Configuration

```
Use legacy Prometheus text format by default
quarkus.micrometer.export.prometheus.format=plain

Use OpenMetrics format by default (default behavior)
quarkus.micrometer.ex
```



## Behavior

- When no Accept header
- When Accept header is application/prometheus
- This allows custom metrics format

## Issues

- Fixes [Configurable](#)



↳ [Format\\_property](#)

↳ [quarkus-bot bot](#)



[quarkus-bot bot comm](#)

/cc [@brunobat](#) (micrometer)



👁️ [axidex](#) marked this pull request as ready for review 5 days ago



👁️ [gastaldi](#) reviewed 3 days ago



over configuration  
the plain text

✓ df3c38f

...

## Labels

[area/metrics](#)

## Projects

None yet

## Milestone

No milestone

## Development

Successfully merging this pull request may close these issues.

[Configurable metrics format](#)

## Notifications

Customize

[Subscribe](#)

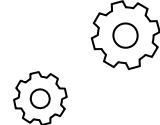
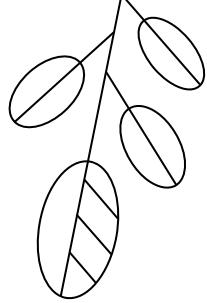
You're not receiving notifications from this thread.

## 2 participants



[View reviewed changes](#)

# 3



# Неожиданные проблемы в неожиданных местах

```
@GET
@Path("/{organization}/{repo}/{commit}")
fun getResults(): Response {
 val list = listOf(B("1"), B("2"))
 return Response.ok(list).build()
}

@Serializable
data class B(
 val b: String
)
```

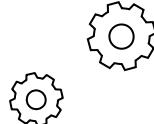
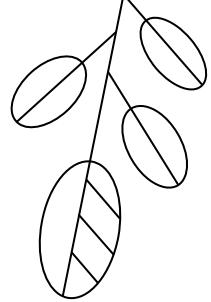
# Неожиданные проблемы в неожиданных местах

```
@GET
@Path("/{organization}/{repo}/{commit}")
fun getResults(): Response {
 val list = listOf(B("1"), B("2"))
 return Response.ok(list).build()
}
```

```
@Serializable
data class B(
 val b: String
)
```

SerializationException: Serializer for class 'ArrayList' is not found.

4



# Is automatic validation of Protobuf messages in place? #42802

Unanswered preslavrachev asked this question in Q&A

 preslavrachev on Aug 27, 2024

Take the following Protobufs spec:

```
message SomeMessageRequest {
 string name = 1;
 optional string label = 2;
}
```

I also have a Grpc service that is supposed to accept requests of that type. A quick demo allowed me to send an empty message to that service, and it still hit the endpoint. I was expecting an error, because the `name` field is not marked as `optional`, but that wasn't the case. The response from the server was a clear 200 OK.

What is the way to validate incoming Protobuf messages in Quarkus, other than field-by-field checking?

 2 

1 comment

Oldest Newest Top

 cescoffier on Sep 2, 2024 Maintainer

No, there is no automatic validation. We follow the same default behavior of gRPC Java. If that's not the case anymore, please open an issue.

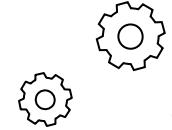
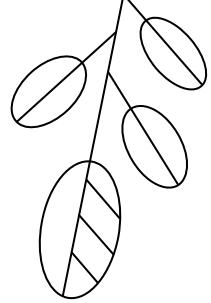
 1 

0 replies

Write a reply

# Генерация из proto, валидации и плагины. Голая и куцая имплементация

# 5



# Quarkus 3.26.1, 3.20.2.2 and 3.15.6.2 - Emergency releases



By [Guillaume Smet](#)

Today, we released Quarkus 3.26.1, 3.20.2.2 and 3.15.6.2 to fix an important regression introduced in Vert.x 4.5.18.

We recommend upgrading to these releases as soon as possible if you are using a Quarkus version using Vert.x 4.5.18 (i.e. 3.26.0, 3.25.4, 3.20.2.1, and 3.15.6.1).

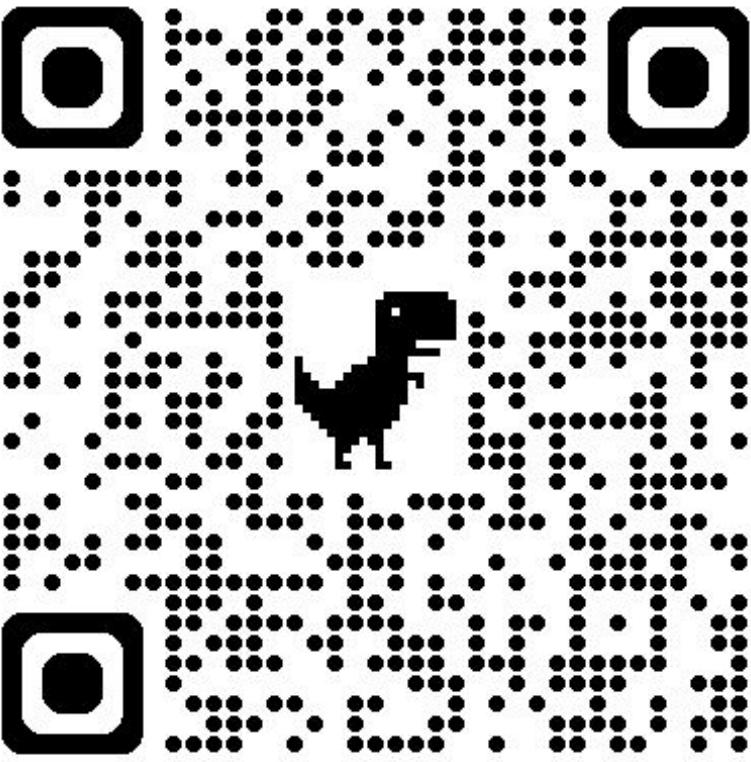
Quarkus 3.26.1 also contains some other unrelated fixes.



Спасибо!  
Q&A



Обратная связь



Мой канал

