

OpenTelemetry Instrumentation using Kotlin Multiplatform Compiler Plugins



Fabian Schoenberger
Markus Weninger, Institute for System Software

Viewing Note

- This presentation is designed with animations and is best experienced in PowerPoint. This PDF version is provided for your convenience for printing and sharing.

OpenTelemetry Instrumentation using Kotlin Multiplatform Compiler Plugins



Fabian Schoenberger
Markus Weninger, Institute for System Software

Motivation: OpenTelemetry

Motivation: OpenTelemetry

OpenTelemetry

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Span

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Span Time-measured unit of work within a trace. (Name, start, end, parent, ...)

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Span Time-measured unit of work within a trace. (Name, start, end, parent, ...)

Metrics

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Span Time-measured unit of work within a trace. (Name, start, end, parent, ...)

Metrics A measurement captured at runtime (Name, kind, unit, description)

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Span Time-measured unit of work within a trace. (Name, start, end, parent, ...)

Metrics A measurement captured at runtime (Name, kind, unit, description)

Logs

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of *measurements or other data* at remote points and their automatic transmission to receiving equipment *for monitoring*

Traces The path of a request through your application.

Span Time-measured unit of work within a trace. (Name, start, end, parent, ...)

Metrics A measurement captured at runtime (Name, kind, unit, description)

Logs A recording of an event (Message, context, ...)

Motivation: OpenTelemetry

OpenTelemetry Standard to collect, transport and store telemetry data

Telemetry Collection of measurements or other data at remote points and their automatic transmission to receiving equipment for monitoring

Traces The path of a request through your application.

Span Time-measured unit of work within a trace. (Name, start, end, parent, ...)

Metrics A measurement captured at runtime (Name, kind, unit, description)

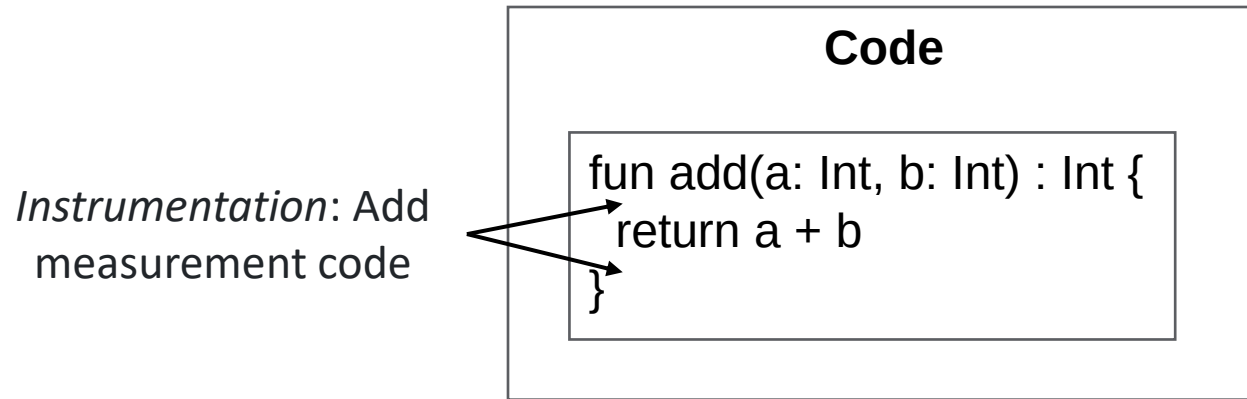
Logs A recording of an event (Message, context, ...)

Motivation: OpenTelemetry

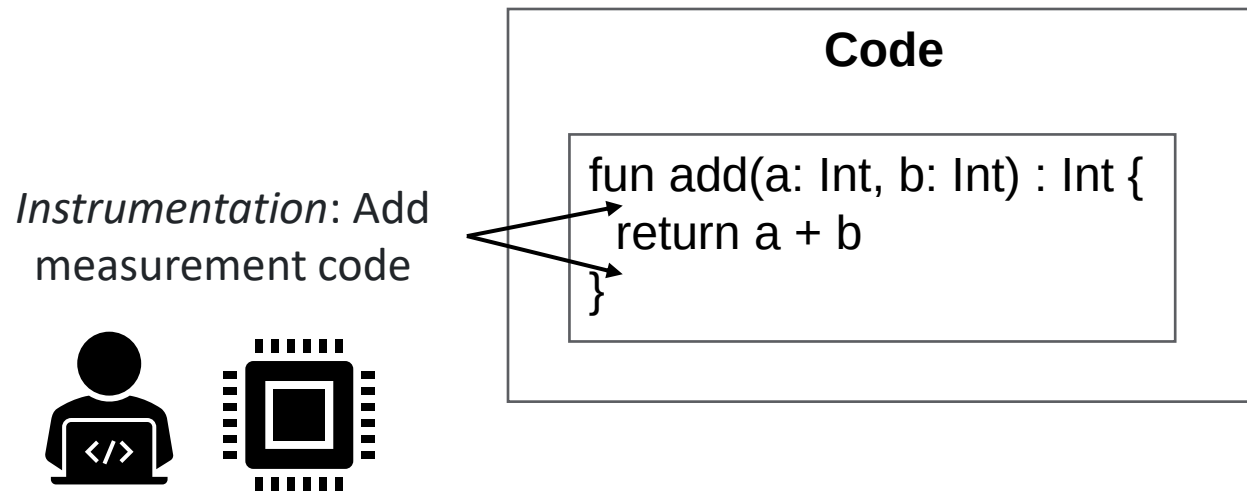
Code

```
fun add(a: Int, b: Int) : Int {  
    return a + b  
}
```

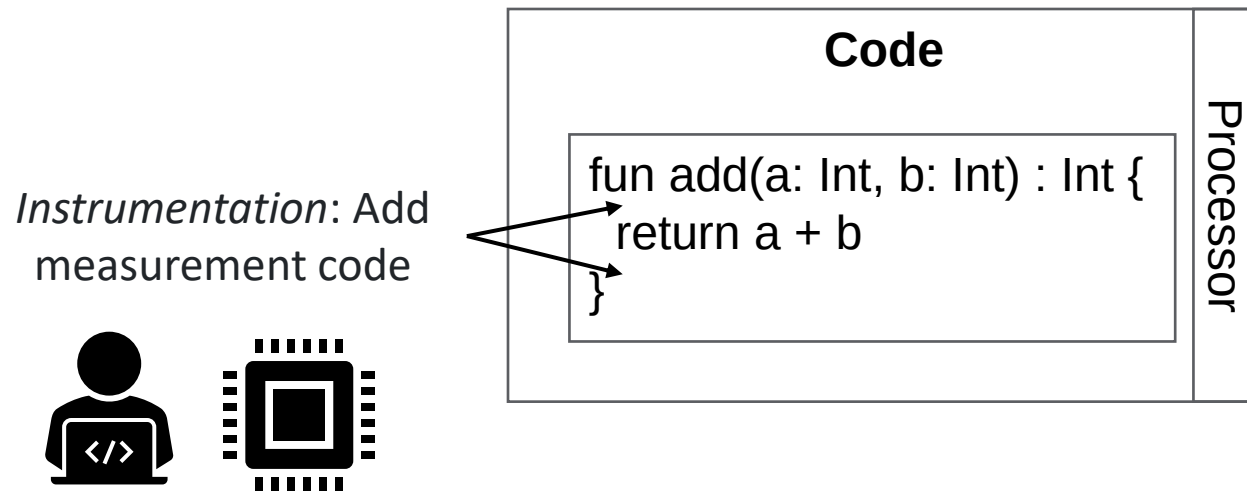
Motivation: OpenTelemetry

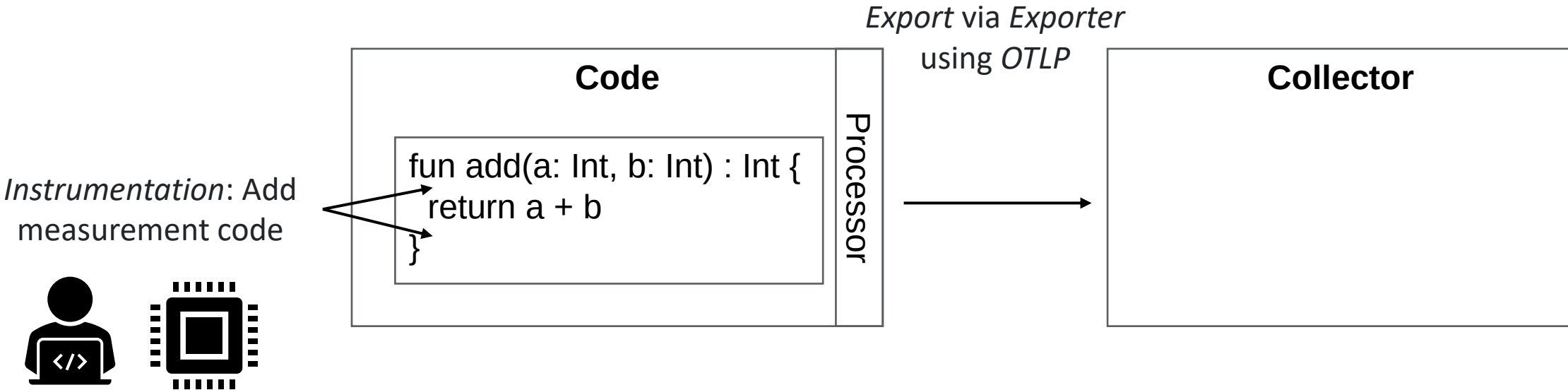


Motivation: OpenTelemetry

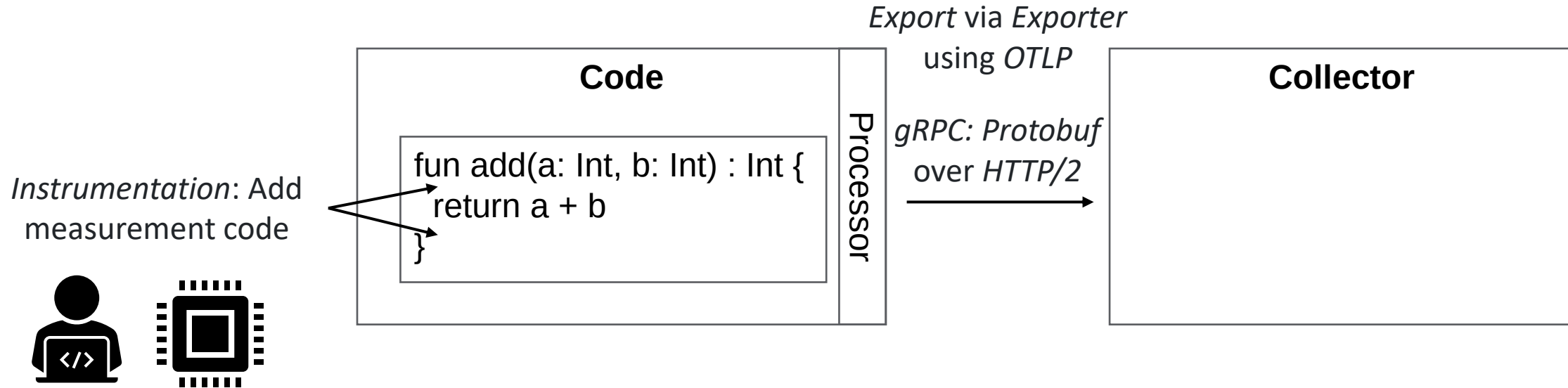


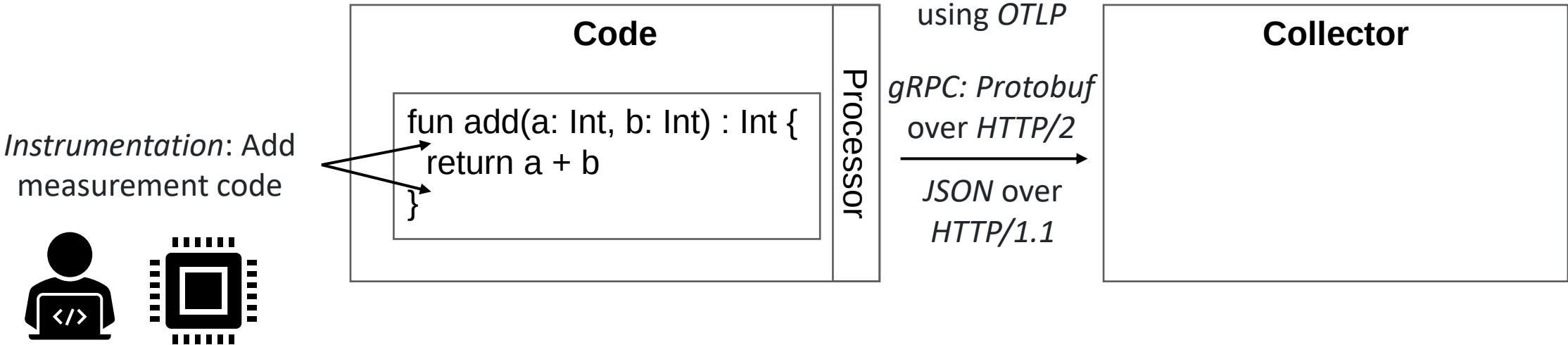
Motivation: OpenTelemetry



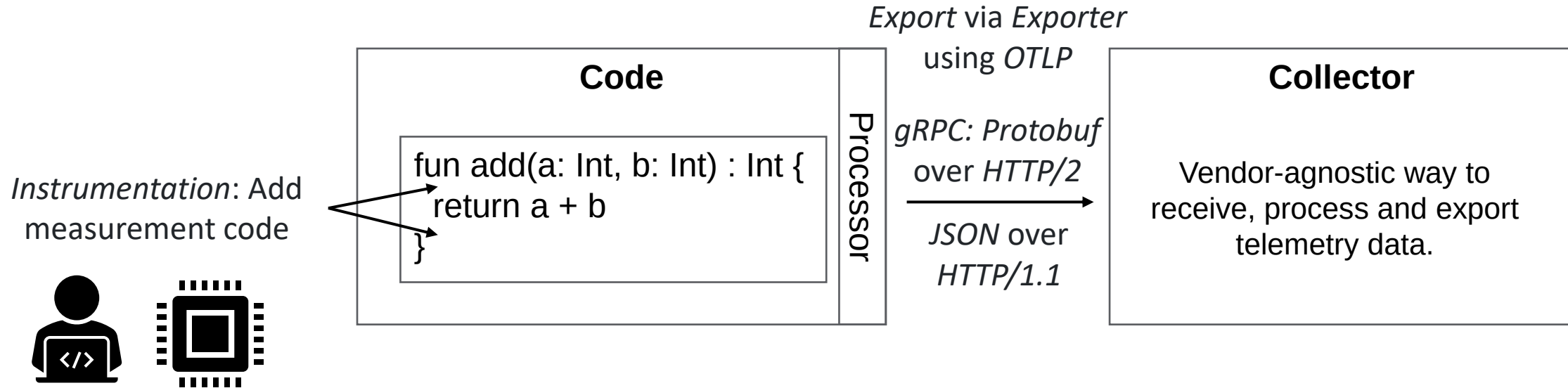


Motivation: OpenTelemetry

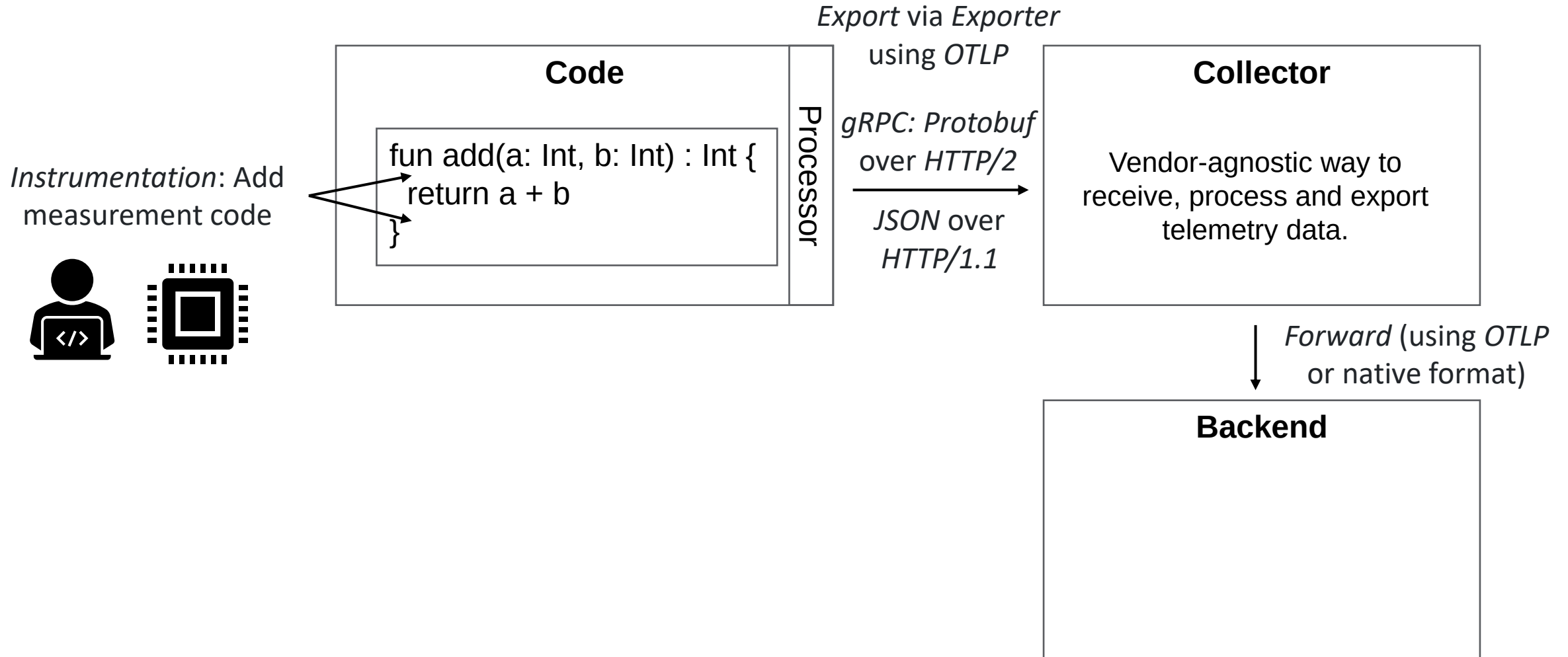




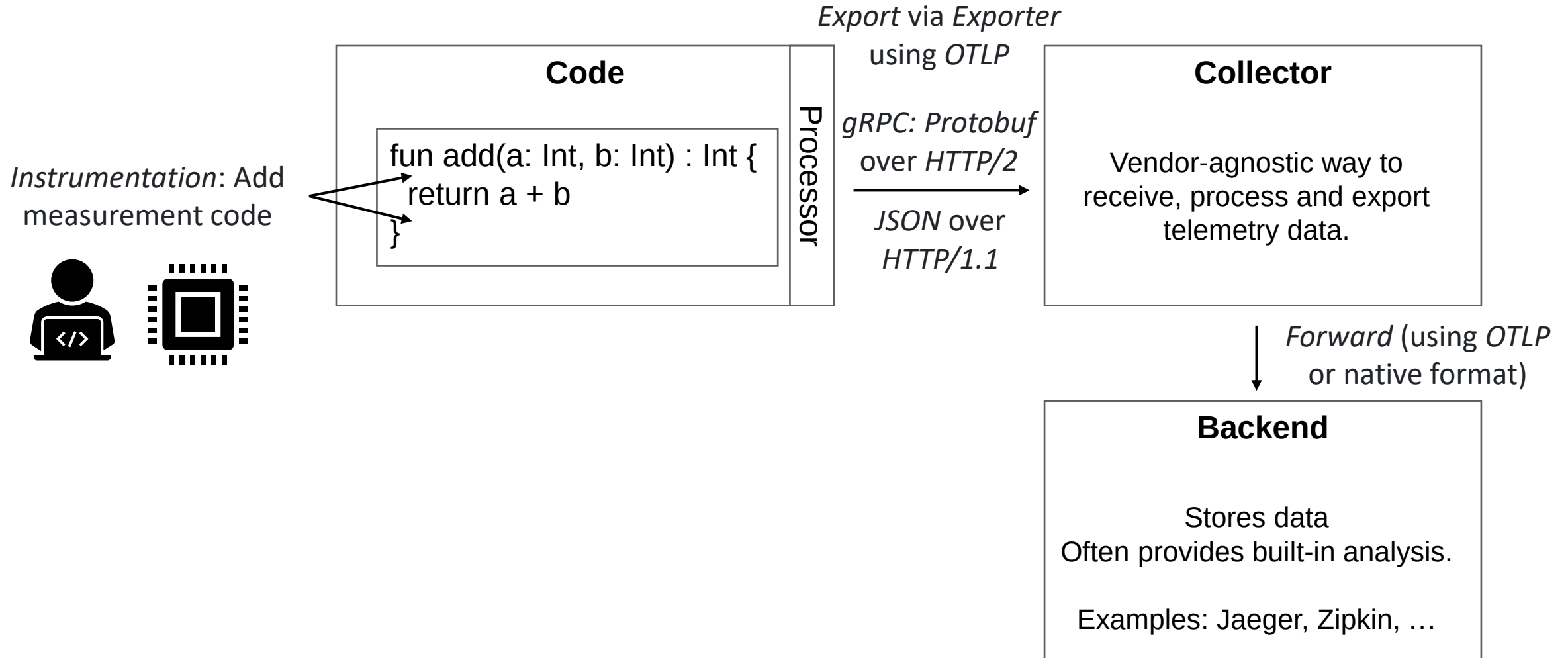
Motivation: OpenTelemetry



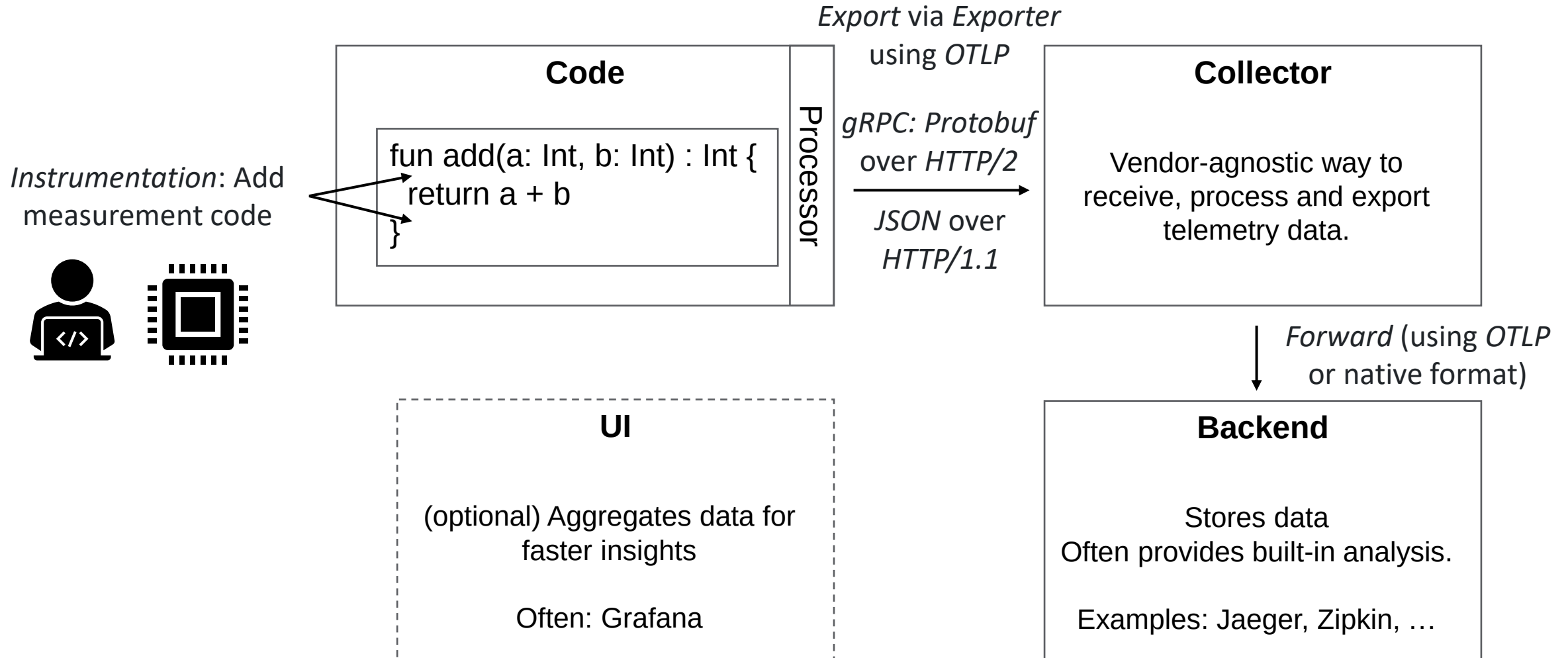
Motivation: OpenTelemetry



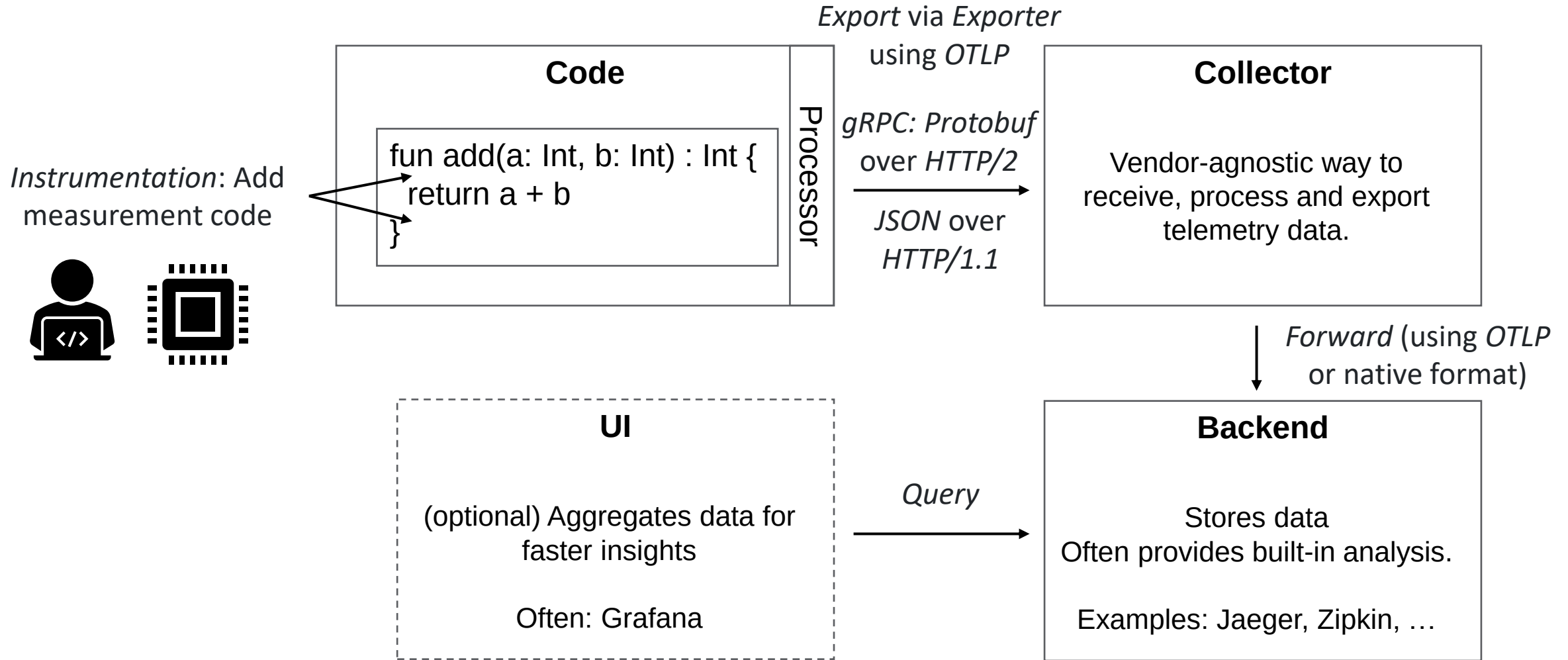
Motivation: OpenTelemetry



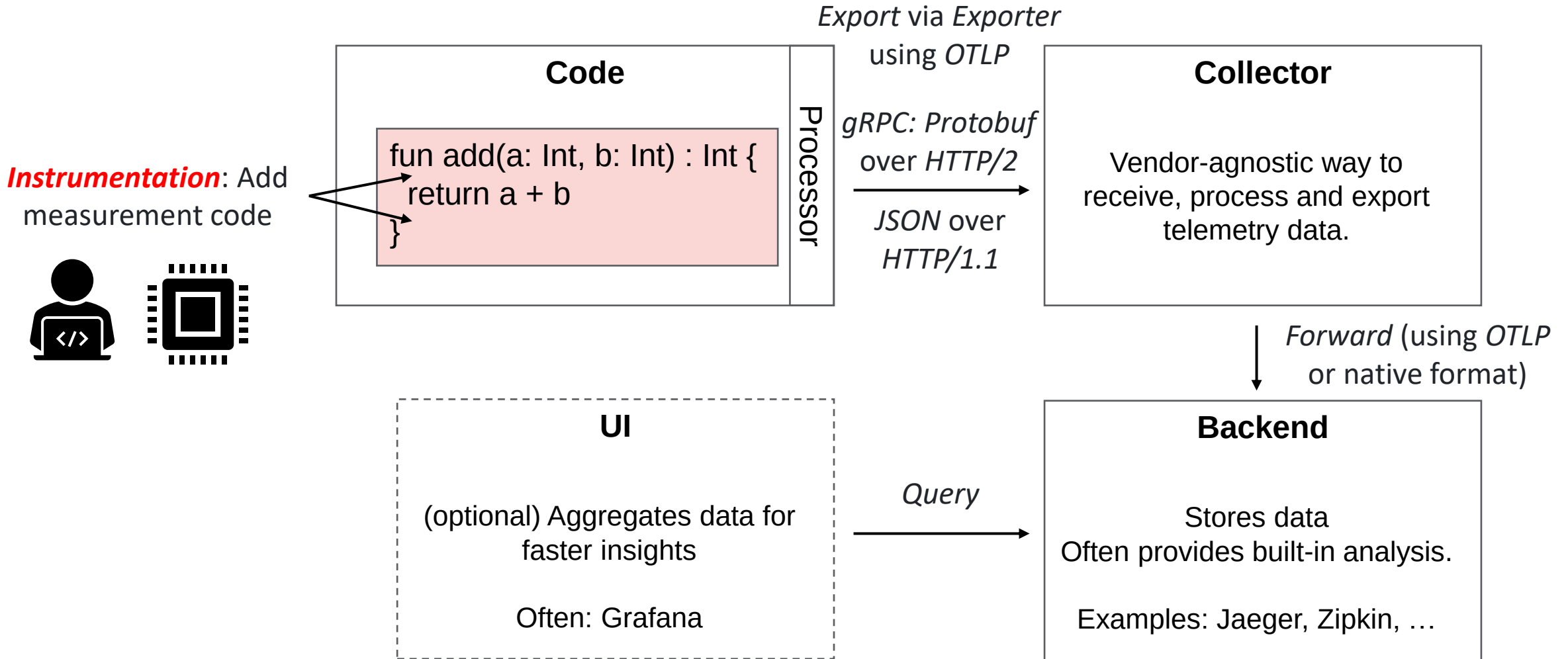
Motivation: OpenTelemetry



Motivation: OpenTelemetry



Motivation: OpenTelemetry



OpenTelemetry **Instrumentation** using Kotlin Multiplatform Compiler Plugins



Fabian Schoenberger
Markus Weninger, Institute for System Software

Motivation: Instrumentation

```
fun myFunction () {  
  val x = 1  
  val y = 1  
  val z = x + y  
}
```


Motivation: Instrumentation

```
fun myFunction () {  
  val x = 1  
  val y = 1  
  val z = x + y  
}
```

instrument



Motivation: Instrumentation

```
fun myFunction () {  
  val x = 1  
  val y = 1  
  val z = x + y  
}
```

instrument

```
fun myFunction () {  
  val span = startSpan()  
  try {  
    val x = 1  
    val y = 1  
    val z = x + y  
  } finally {  
    span.end()  
  }  
}
```

Motivation: Instrumentation

```
fun myFunction () {  
  val x = 1  
  val y = 1  
  val z = x + y  
}
```

instrument



```
fun myFunction () {  
  val span = startSpan()  
  try {  
    val x = 1  
    val y = 1  
    val z = x + y  
  } finally {  
    span.end()  
  }  
}
```

```
fun myFunction (a: Int, b: Int) = a * b - 3
```

Motivation: Instrumentation

```
fun myFunction () {  
  val x = 1  
  val y = 1  
  val z = x + y  
}
```

instrument

```
fun myFunction () {  
  val span = startSpan()  
  try {  
    val x = 1  
    val y = 1  
    val z = x + y  
  } finally {  
    span.end()  
  }  
}
```

```
fun myFunction (a: Int, b: Int) = a * b - 3
```

instrument

Motivation: Instrumentation

```
fun myFunction () {  
  val x = 1  
  val y = 1  
  val z = x + y  
}
```

instrument

```
fun myFunction () {  
  val span = startSpan()  
  try {  
    val x = 1  
    val y = 1  
    val z = x + y  
  } finally {  
    span.end()  
  }  
}
```

```
fun myFunction (a: Int, b: Int) = a * b - 3
```

instrument

```
fun myFunction (a: Int, b: Int) {  
  val span = startSpan()  
  try {  
    return a * b - 3  
  } finally {  
    span.end()  
  }  
}
```

Motivation: Instrumentation

```
fun myFunction () {  
    val x = 1  
    val y = 1  
    val z = x + y  
}
```

instrument

```
fun myFunction () {  
    val span = startSpan()  
    try {  
        val x = 1  
        val y = 1  
        val z = x + y  
    } finally {  
        span.end()  
    }  
}
```

```
fun myFunction (a: Int, b: Int) = a * b - 3
```

instrument

```
fun myFunction (a: Int, b: Int) {  
    val span = startSpan()  
    try {  
        return a * b - 3  
    } finally {  
        span.end()  
    }  
}
```



How to best instrument **Kotlin**?

OpenTelemetry Instrumentation using **Kotlin** Multiplatform Compiler Plugins



Fabian Schoenberger
Markus Weninger, Institute for System Software

OpenTelemetry Instrumentation using **Kotlin Multiplatform** Compiler Plugins



Fabian Schoenberger
Markus Weninger, Institute for System Software

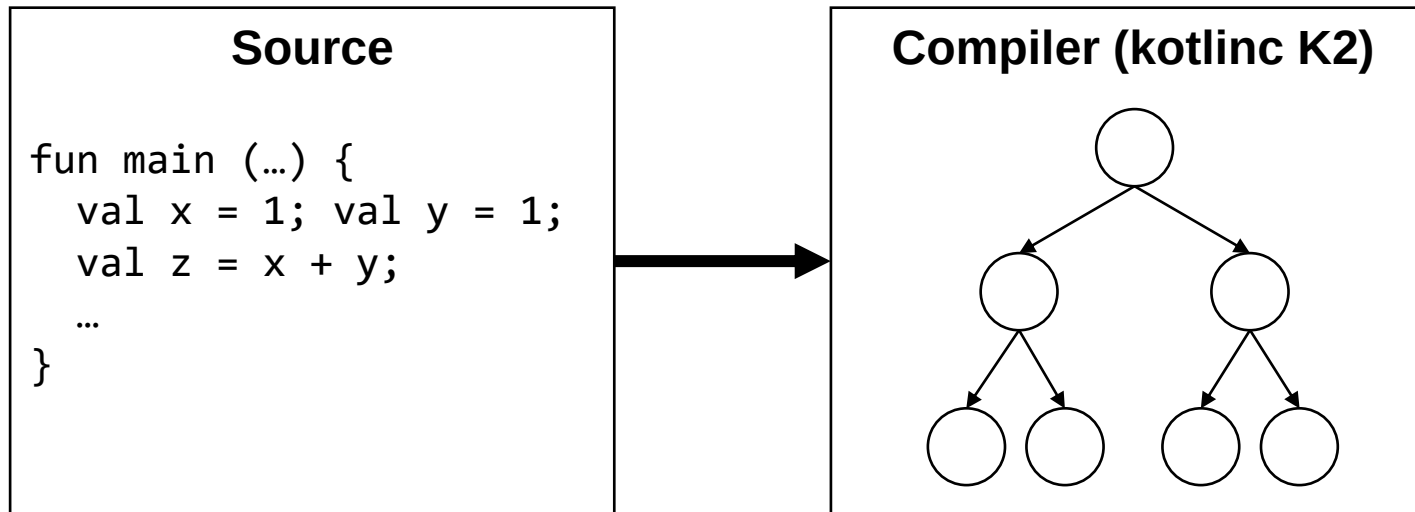
Motivation: Kotlin Multiplatform

Motivation: Kotlin Multiplatform

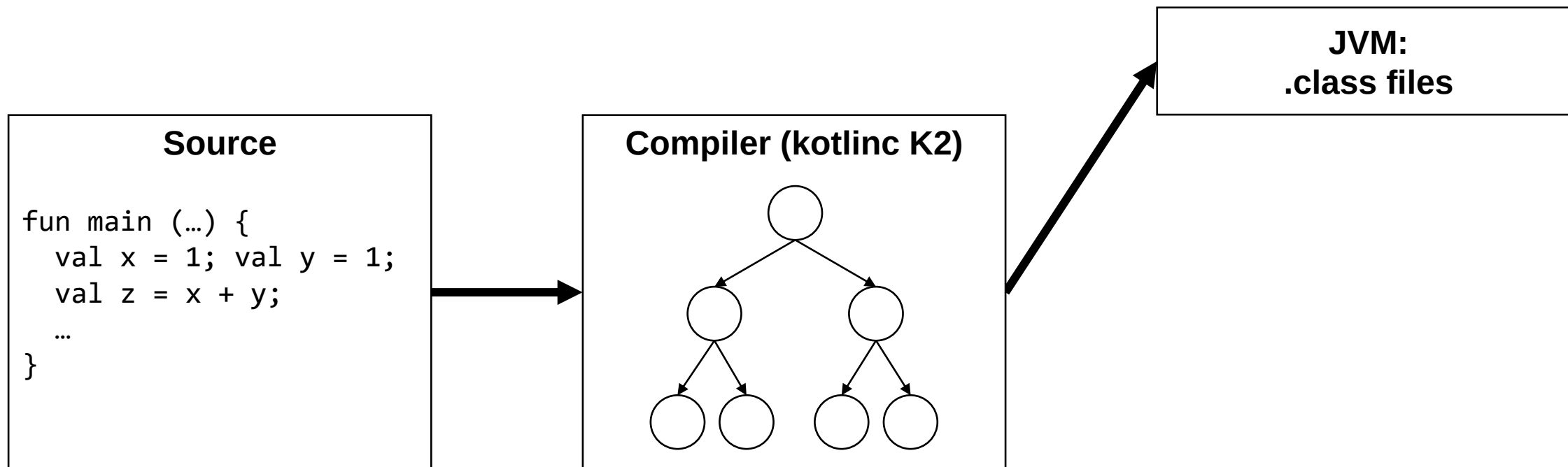
Source

```
fun main (...) {  
    val x = 1; val y = 1;  
    val z = x + y;  
    ...  
}
```

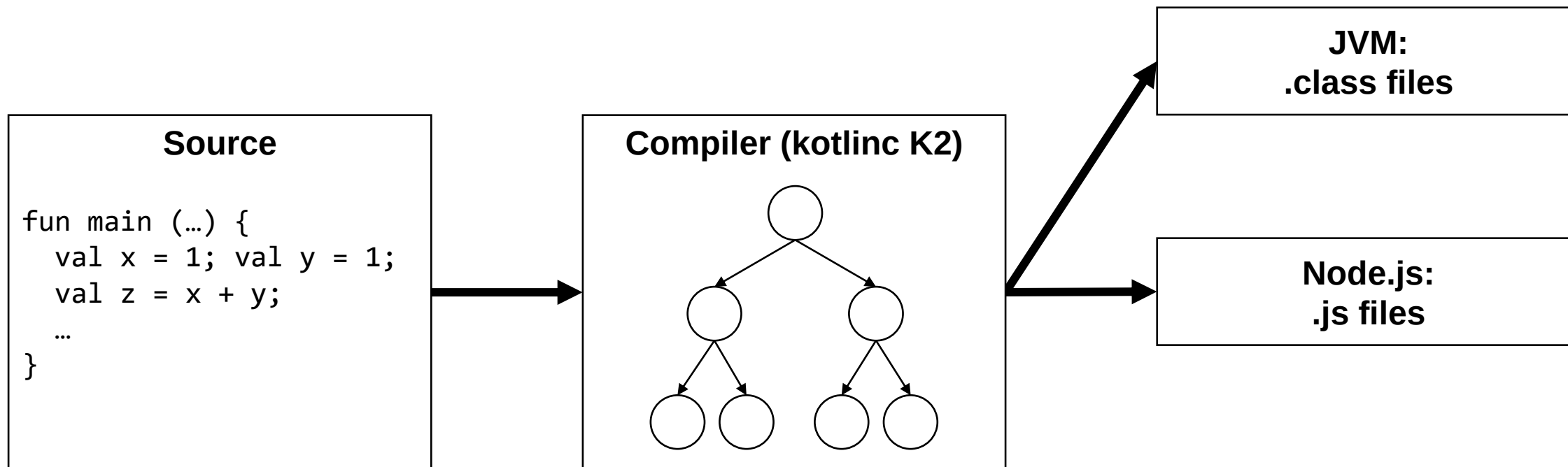
Motivation: Kotlin Multiplatform



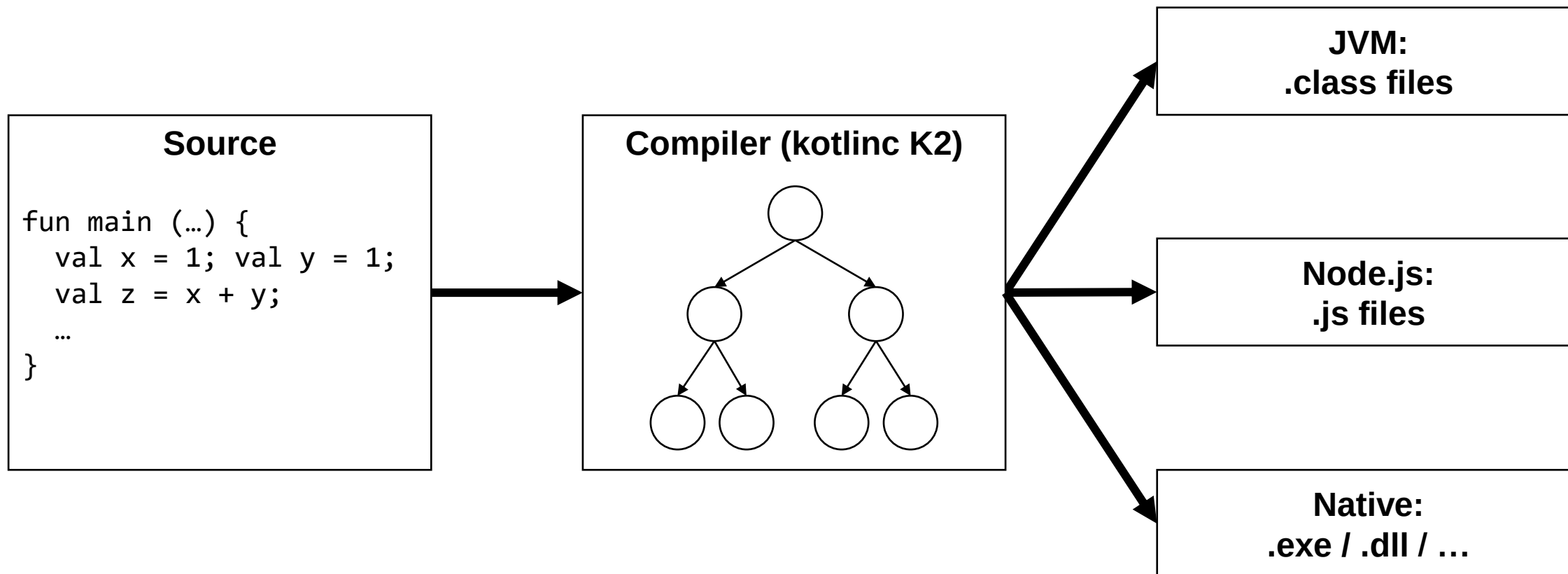
Motivation: Kotlin Multiplatform



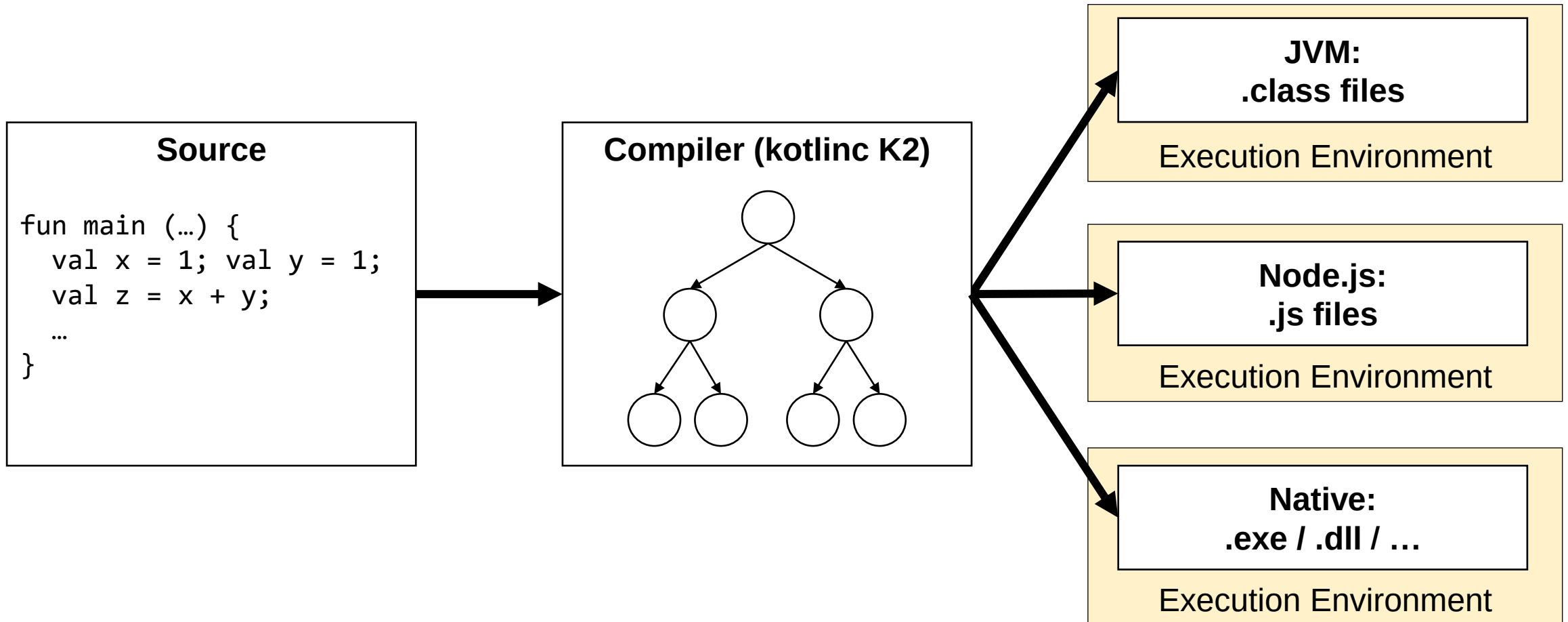
Motivation: Kotlin Multiplatform



Motivation: Kotlin Multiplatform



Motivation: Kotlin Multiplatform



Requirements for Kotlin Multiplatform Projects

Dependencies

Requirements for Kotlin Multiplatform Projects

Dependencies

(1) Rely on libraries that are available as multiplatform libraries

Requirements for Kotlin Multiplatform Projects

Dependencies

(1) Rely on libraries that are available as multiplatform libraries

```
sourceSets {  
    commonMain.dependencies {  
        implementation 'io.opentelemetry.kotlin.api'  
    }  
}
```

Requirements for Kotlin Multiplatform Projects

Dependencies

(1) Rely on libraries that are available as multiplatform libraries

```
sourceSets {  
    commonMain.dependencies {  
        implementation 'io.opentelemetry.kotlin.api'  
    }  
}
```

OpenTelemetry as
KMP library?

Requirements for Kotlin Multiplatform Projects

Dependencies

(1) Rely on libraries that are available as multiplatform libraries

```
sourceSets {  
    commonMain.dependencies {  
        implementation 'io.opentelemetry.kotlin.api'  
    }  
}
```

OpenTelemetry as KMP library?

(2) Have a separate dependency per target

Requirements for Kotlin Multiplatform Projects

Dependencies

(1) Rely on libraries that are available as multiplatform libraries

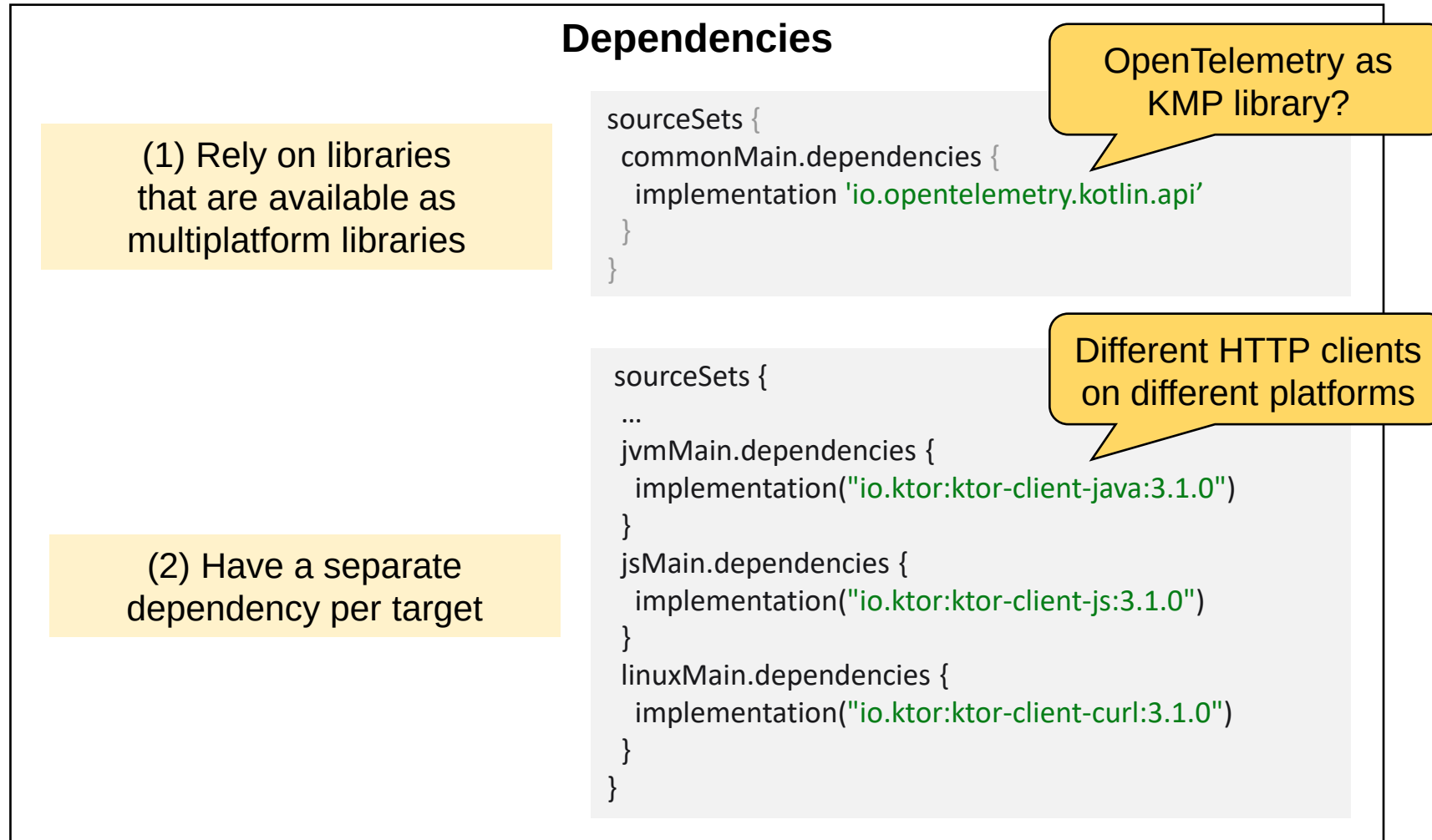
```
sourceSets {  
    commonMain.dependencies {  
        implementation 'io.opentelemetry.kotlin.api'  
    }  
}
```

OpenTelemetry as KMP library?

(2) Have a separate dependency per target

```
sourceSets {  
    ...  
    jvmMain.dependencies {  
        implementation("io.ktor:ktor-client-java:3.1.0")  
    }  
    jsMain.dependencies {  
        implementation("io.ktor:ktor-client-js:3.1.0")  
    }  
    linuxMain.dependencies {  
        implementation("io.ktor:ktor-client-curl:3.1.0")  
    }  
}
```

Requirements for Kotlin Multiplatform Projects



OpenTelemetry as Kotlin Multiplatform Projects?

OpenTelemetry as Kotlin Multiplatform Projects?

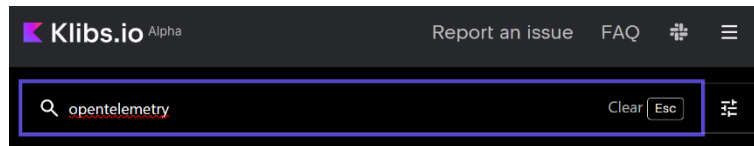
Official Repo

<https://klibs.io/>

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>



opentelemetry-kotlin

★49

by embrace-io

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, JS, Common

koog

★3.4k

by JetBrains

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k

★52

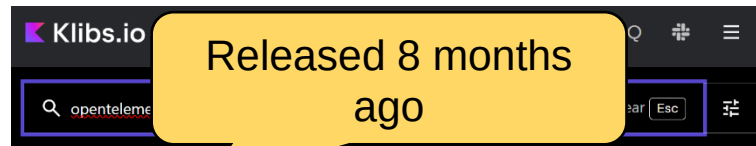
by smyrgeorge

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>



opentelemetry-kotlin ★49
by embrace-io

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0
Android JVM, JVM, Kotlin/Native, JS, Common

koog ★3.4k
by JetBrains

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0
Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

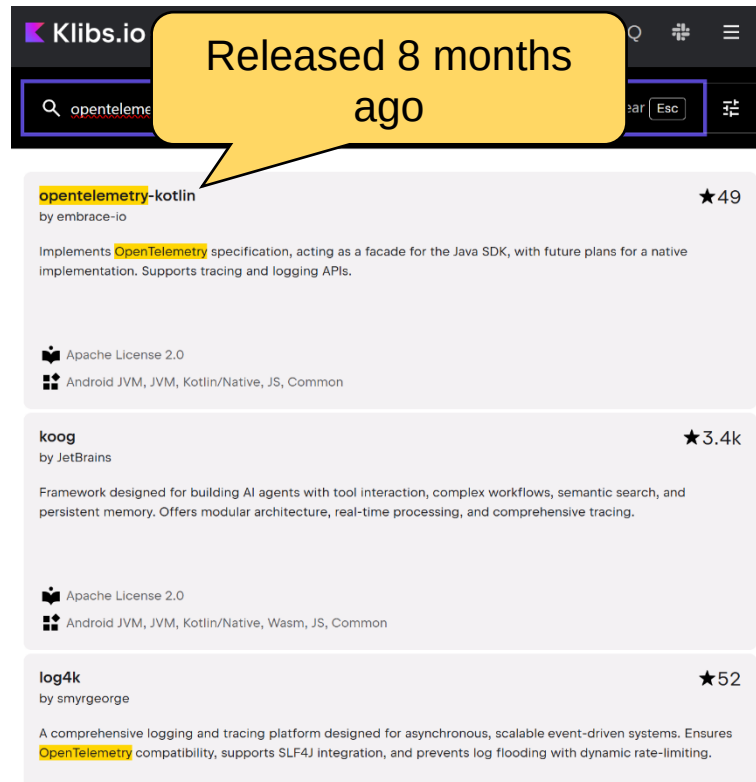
log4k ★52
by smyrgeorge

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>



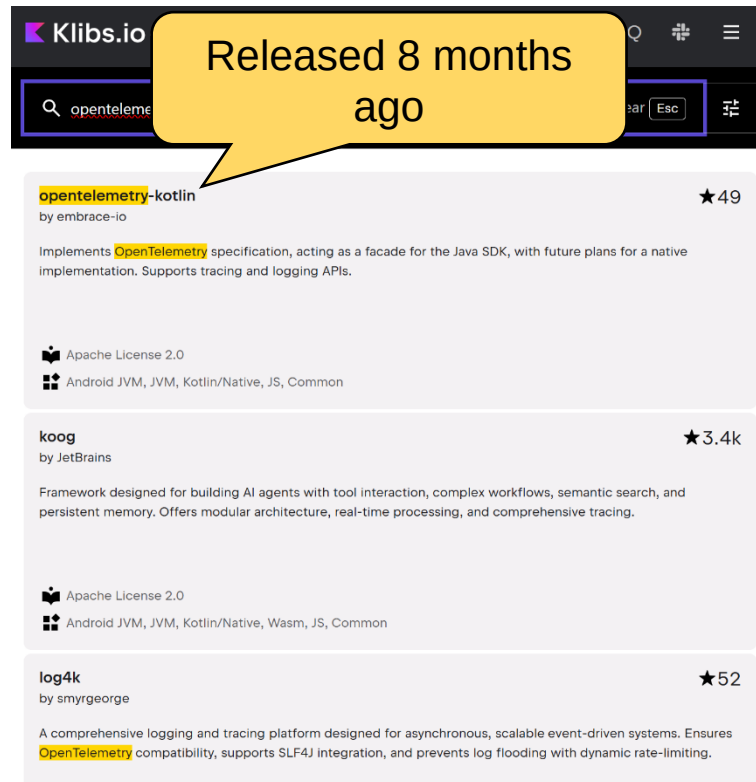
Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>



The screenshot shows the Klibs.io website with a search bar containing 'opentelemetry'. A yellow speech bubble points to the search results, stating 'Released 8 months ago'. The results list three projects: 'opentelemetry-kotlin' by embrace-io (49 stars), 'koog' by JetBrains (3.4k stars), and 'log4k' by smyrgeorge (52 stars). Each project entry includes a brief description and supported platforms.

Released 8 months ago

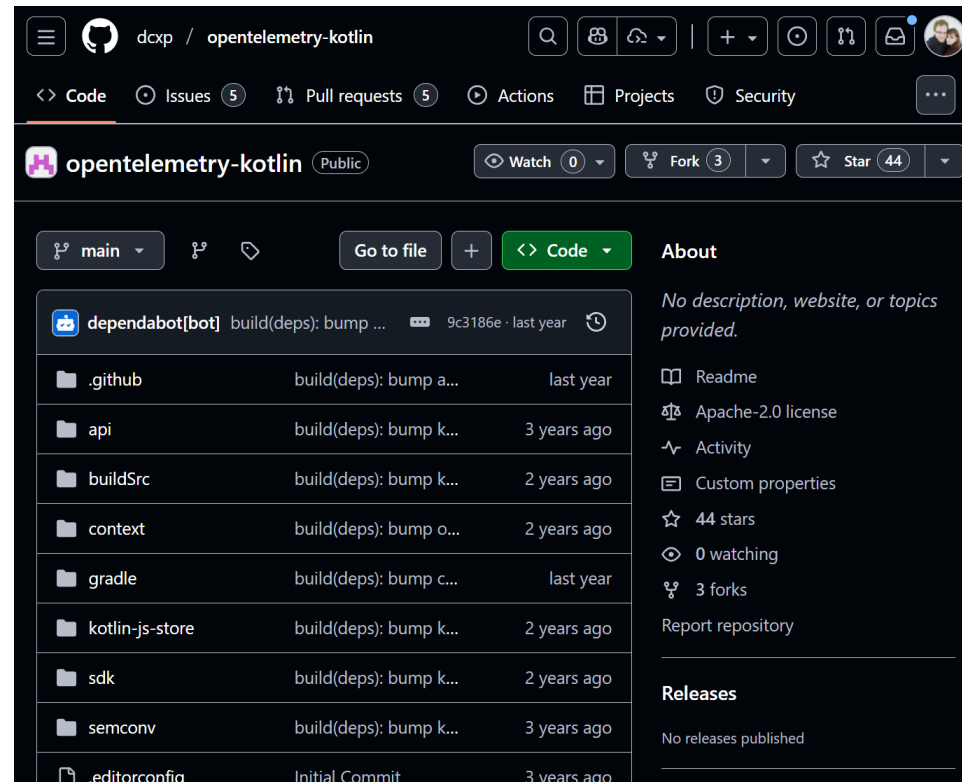
opentelemetry-kotlin by embrace-io ★49
Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.
Apache License 2.0
Android JVM, JVM, Kotlin/Native, JS, Common

koog by JetBrains ★3,4k
Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.
Apache License 2.0
Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k by smyrgeorge ★52
A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)



The screenshot shows the GitHub repository page for 'dcxp/opentelemetry-kotlin'. The repository is public and has 44 stars, 3 forks, and 0 watches. The file list shows a directory structure with files like '.github', 'api', 'buildSrc', 'context', 'gradle', 'kotlin-js-store', 'sdk', 'semconv', and 'editorconfig'. The right sidebar shows repository details: no description, Apache-2.0 license, 44 stars, 0 watching, 3 forks, and no releases published.

dcxp / opentelemetry-kotlin Public
Watch 0 Fork 3 Star 44

opentelemetry-kotlin Public
main
Go to file + <> Code

dependabot[bot] build(deps): bump ... 9c3186e · last year

File	Commit	Time
.github	build(deps): bump a...	last year
api	build(deps): bump k...	3 years ago
buildSrc	build(deps): bump k...	2 years ago
context	build(deps): bump o...	2 years ago
gradle	build(deps): bump c...	last year
kotlin-js-store	build(deps): bump k...	2 years ago
sdk	build(deps): bump k...	2 years ago
semconv	build(deps): bump k...	3 years ago
editorconfig	Initial Commit	3 years ago

About
No description, website, or topics provided.
Readme
Apache-2.0 license
Activity
Custom properties
44 stars
0 watching
3 forks
Report repository

Releases
No releases published

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>

Klibs.io

Released 8 months ago

opentelemetry-kotlin ★49
by embrace-io

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0
Android JVM, JVM, Kotlin/Native, JS, Common

koog ★3.4k
by JetBrains

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0
Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k ★52
by smyrgeorge

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)

dcxp / opentelemetry-kotlin

<> Code 5 Issues 5 Pull requests 5 Actions Projects Security

opentelemetry-kotlin Public Watch 0 Fork 3 Star 44

main Go to file + <> Code About

dependabot[bot] build(deps): bump ... 9c3186e

File	Commit	Time
.github	build(deps): bump a...	last year
api	build(deps): bump k...	3 years ago
buildSrc	build(deps): bump k...	2 years ago
context	build(deps): bump o...	2 years ago
gradle	build(deps): bump c...	last year
kotlin-js-store	build(deps): bump k...	2 years ago
sdk	build(deps): bump k...	2 years ago
semconv	build(deps): bump k...	3 years ago
editorconfig	Initial Commit	3 years ago

Released 3 years ago

Readme
Apache-2.0 license
Activity
Custom properties
44 stars
0 watching
3 forks
Report repository

Releases
No releases published

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>

Released 8 months ago

opentelemetry-kotlin ★49
by embrace-io

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0
Android JVM, JVM, Kotlin/Native, JS, Common

koog ★3.4k
by JetBrains

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0
Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k ★52
by smyrgeorge

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)

Released 3 years ago

dcxp / opentelemetry-kotlin

Code Issues (5) Pull requests (5) Actions Projects Security

opentelemetry-kotlin Public Watch (0) Fork (3) Star (44)

main Go to file Code About

File	Commit	Author	Time
.github	build(deps): bump...	dependabot[bot]	last year
api	build(deps): bump k...	dependabot[bot]	3 years ago
buildSrc	build(deps): bump k...	dependabot[bot]	2 years ago
context	build(deps): bump o...	dependabot[bot]	2 years ago
gradle	build(deps): bump c...	dependabot[bot]	last year
kotlin-js-store	build(deps): bump k...	dependabot[bot]	2 years ago
sdk	build(deps): bump k...	dependabot[bot]	2 years ago
semconv	build(deps): bump k...	dependabot[bot]	3 years ago
editorconfig	Initial Commit	dcxp	3 years ago

dependabot[bot] build(deps): bump... 9c3186e

Readme

Apache-2.0 license

Activity

Custom properties

44 stars

0 watching

3 forks

Report repository

Releases

No releases published

Collections

[terrakok/kmp-awesome: An awesome list that curates the best Kotlin Multiplatform libraries, tools and more.](#)

[AAkira/Kotlin-Multiplatform-Libraries: Kotlin Multiplatform Libraries.](#)

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>

Released 8 months ago

opentelemetry-kotlin by embrace-io ★49

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, JS, Common

koog by JetBrains ★3,4k

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k by smyrgeorge ★52

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)

Released 3 years ago

dcxp / opentelemetry-kotlin

Code Issues (5) Pull requests (5) Actions Projects Security

opentelemetry-kotlin Public Watch (0) Fork (3) Star (44)

main Go to file Code About

File	Commit	Author	Time
.github	build(deps): bump...	dependabot[bot]	last year
api	build(deps): bump k...	dependabot[bot]	3 years ago
buildSrc	build(deps): bump k...	dependabot[bot]	2 years ago
context	build(deps): bump o...	dependabot[bot]	2 years ago
gradle	build(deps): bump c...	dependabot[bot]	last year
kotlin-js-store	build(deps): bump k...	dependabot[bot]	2 years ago
sdk	build(deps): bump k...	dependabot[bot]	2 years ago
semconv	build(deps): bump k...	dependabot[bot]	3 years ago
editorconfig	Initial Commit	dcxp	3 years ago

dependabot[bot] build(deps): bump ... 9c3186e

Readme

Apache-2.0 license

Activity

Custom properties

44 stars

0 watching

3 forks

Report repository

Releases

No releases published

Collections

No OTEL 😞

[some. An awesome list that curates the best Kotlin Multiplatform libraries, tools and more.](#)

[AAkira/Kotlin-Multiplatform-Libraries: Kotlin Multiplatform Libraries.](#)

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>

Released 8 months ago

opentelemetry-kotlin by embrace-io ★49

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, JS, Common

koog by JetBrains ★3,4k

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k by smyrgeorge ★52

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)

Released 3 years ago

opentelemetry-kotlin Public

Watch 0 Fork 3 Star 44

main

Go to file

Code

About

File	Commit	Time
.github	build(deps): bump...	last
api	build(deps): bump k...	3 years ago
buildSrc	build(deps): bump k...	2 years ago
context	build(deps): bump o...	2 years ago
gradle	build(deps): bump c...	last year
kotlin-js-store	build(deps): bump k...	2 years ago
sdk	build(deps): bump k...	2 years ago
semconv	build(deps): bump k...	3 years ago
editorconfig	Initial Commit	3 years ago

dependabot[bot] build(deps): bump ... 9c3186e

Readme

Apache-2.0 license

Activity

Custom properties

44 stars

0 watching

3 forks

Report repository

Releases

No releases published

Collections

No OTEL ☹️

[some. An awesome list that curates the best Kotlin Multiplatform libraries, tools and more.](#)

No OTEL ☹️

[Kotlin-Multiplatform-Libraries: Kotlin Multiplatform Libraries.](#)

OpenTelemetry as Kotlin Multiplatform Projects?

Official Repo

<https://klibs.io/>

Released 8 months ago

opentelemetry-kotlin by embrace-io ★49

Implements **OpenTelemetry** specification, acting as a facade for the Java SDK, with future plans for a native implementation. Supports tracing and logging APIs.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, JS, Common

koog by JetBrains ★3,4k

Framework designed for building AI agents with tool interaction, complex workflows, semantic search, and persistent memory. Offers modular architecture, real-time processing, and comprehensive tracing.

Apache License 2.0

Android JVM, JVM, Kotlin/Native, Wasm, JS, Common

log4k by smyrgeorge ★52

A comprehensive logging and tracing platform designed for asynchronous, scalable event-driven systems. Ensures **OpenTelemetry** compatibility, supports SLF4J integration, and prevents log flooding with dynamic rate-limiting.

Github

[dcxp/opentelemetry-kotlin](https://github.com/dcxp/opentelemetry-kotlin)

Released 3 years ago

opentelemetry-kotlin Public

Watch 0 Fork 3 Star 44

main

Go to file

Code

About

File	Commit	Time
.github	build(deps): bump a...	last
api	build(deps): bump k...	3 years ago
buildSrc	build(deps): bump k...	2 years ago
context	build(deps): bump o...	2 years ago
gradle	build(deps): bump c...	last year
kotlin-js-store	build(deps): bump k...	2 years ago
sdk	build(deps): bump k...	2 years ago
semconv	build(deps): bump k...	3 years ago
editorconfig	Initial Commit	3 years ago

dependabot[bot] build(deps): bump ... 9c3186e

Readme

Apache-2.0 license

Activity

Custom properties

44 stars

0 watching

3 forks

Report repository

Releases

No releases published

Collections

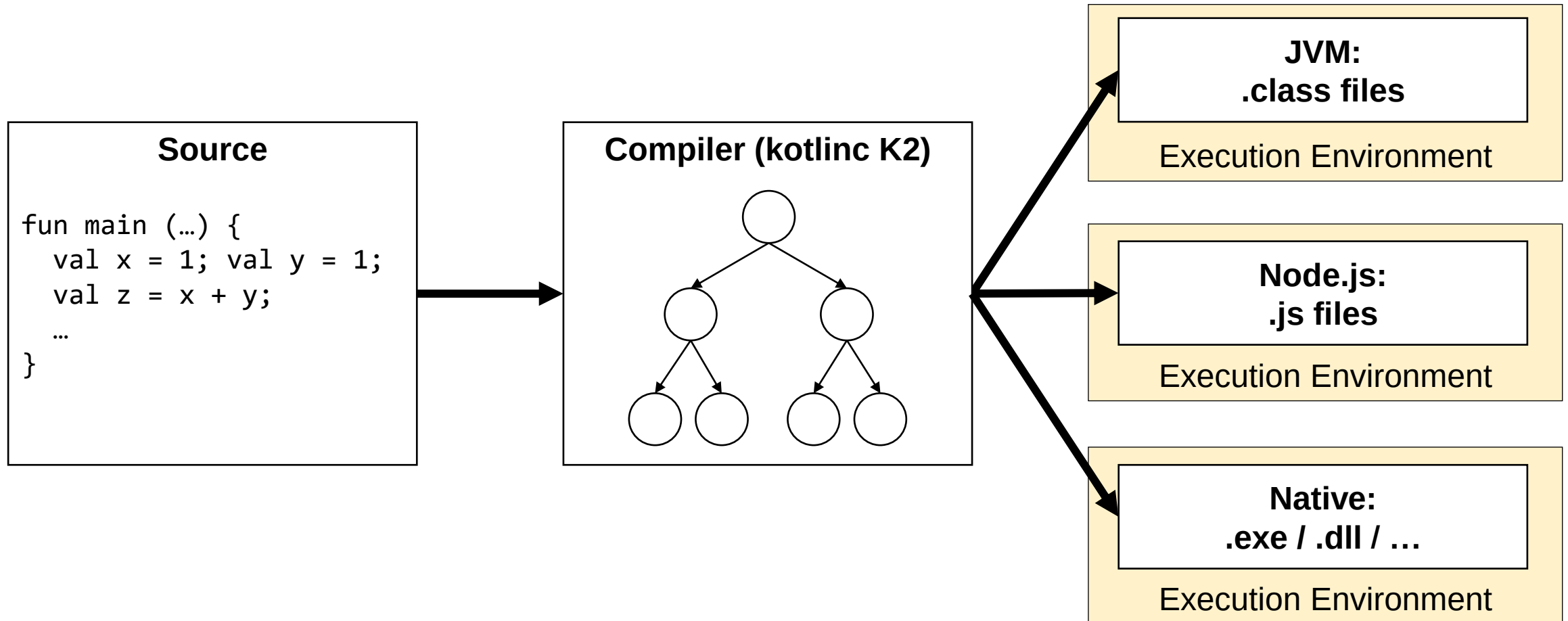
No OTEL 😞

[some. An awesome list that curates the best Kotlin Multiplatform libraries, tools and more.](#)

No OTEL 😞

[Kotlin-Multiplatform-Libraries: Kotlin Multiplatform Libraries.](#)

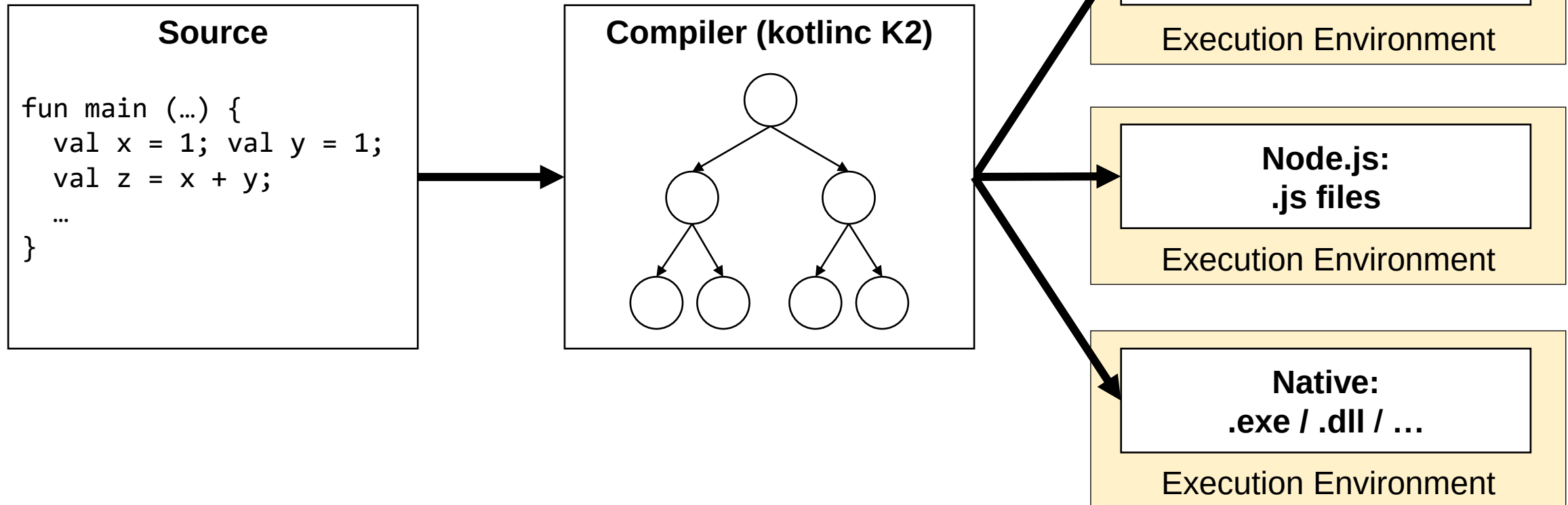
OpenTelemetry Instrumentation



OpenTelemetry Instrumentation



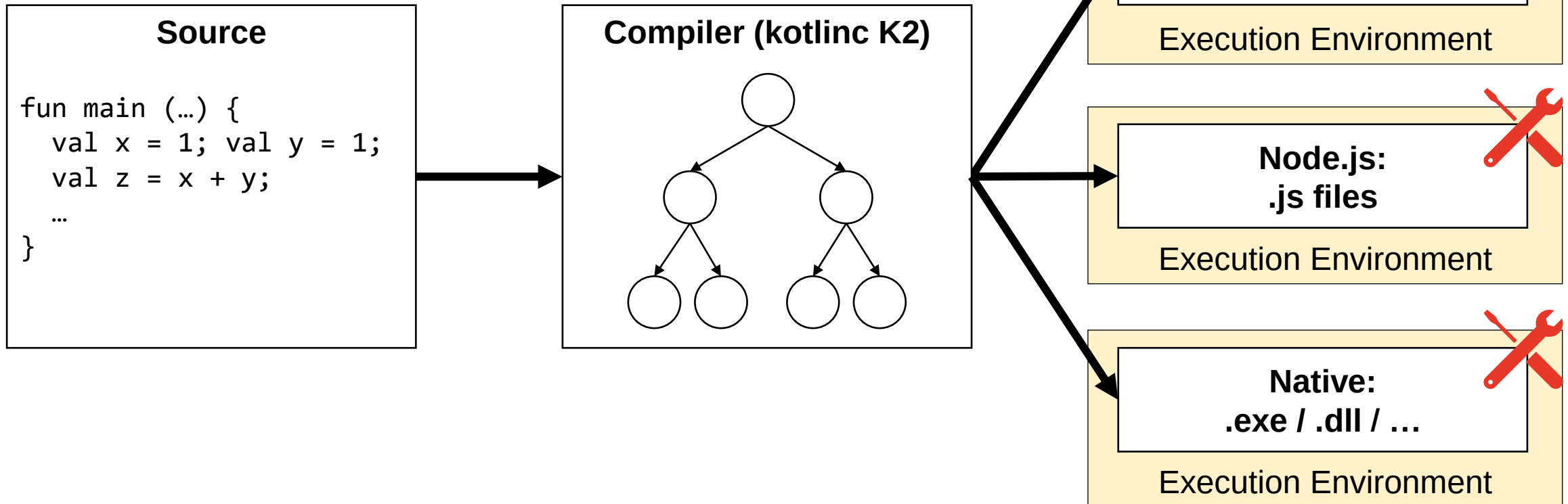
Where to add OTel



OpenTelemetry Instrumentation



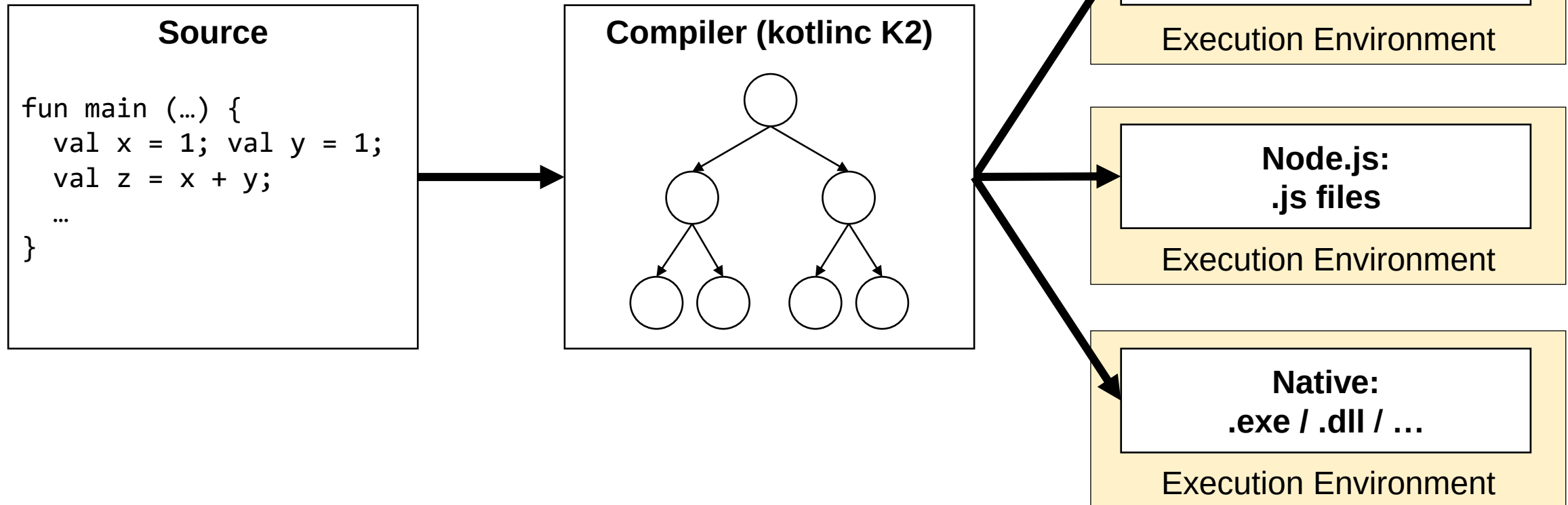
Where to add OTel



OpenTelemetry Instrumentation



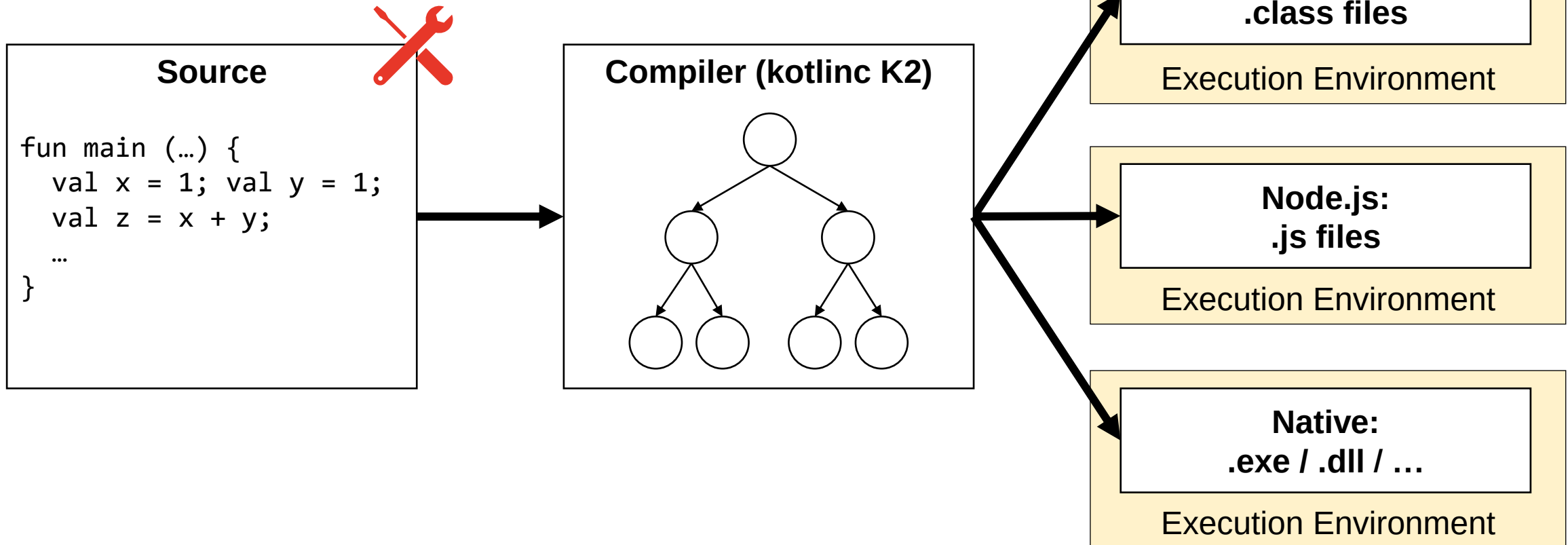
Where to add OTel



OpenTelemetry Instrumentation



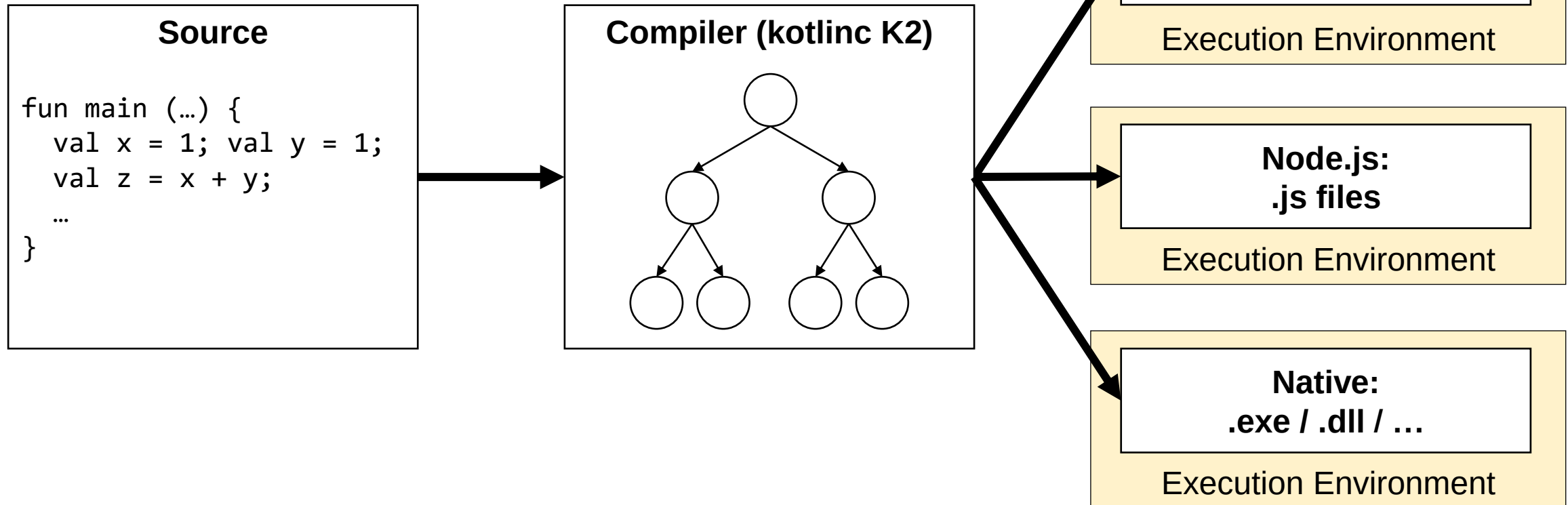
Where to add OTel



OpenTelemetry Instrumentation



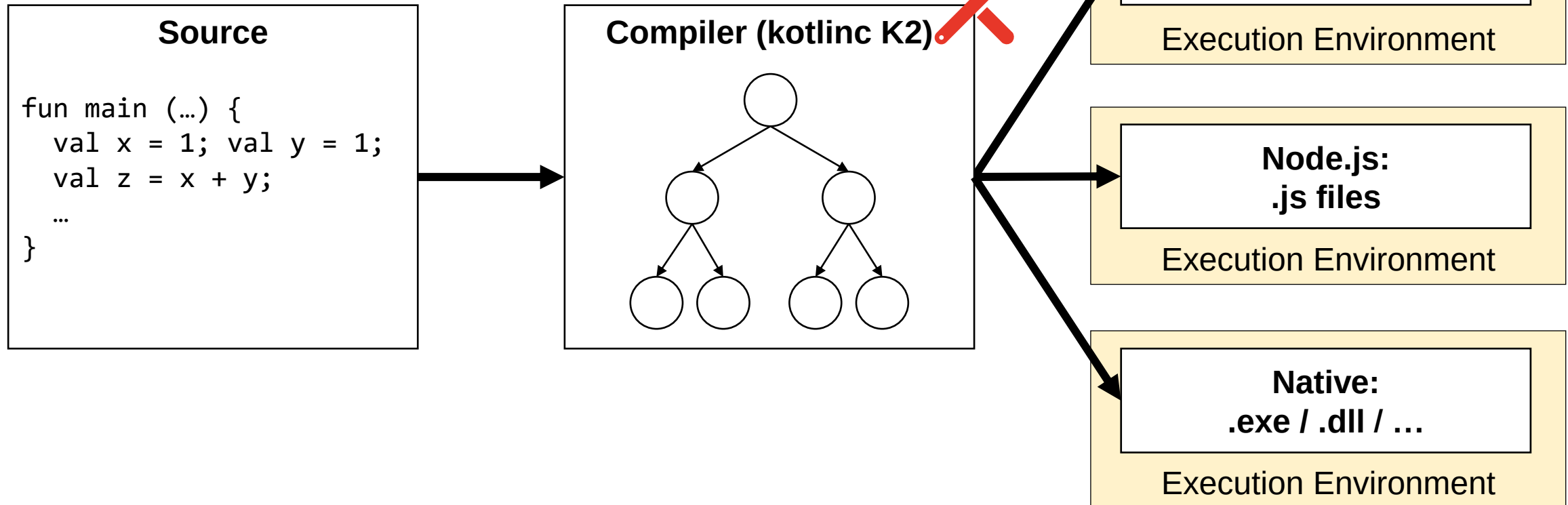
Where to add OTel



OpenTelemetry Instrumentation



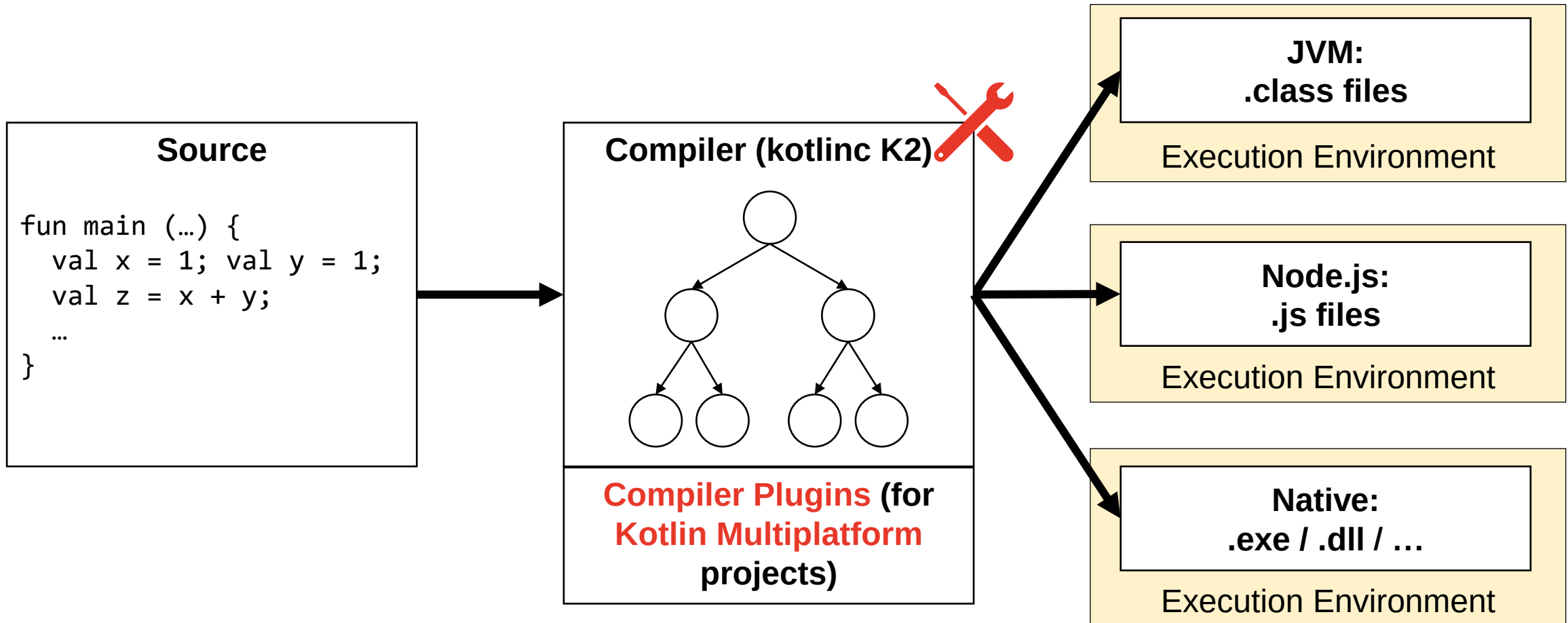
Where to add OTel



OpenTelemetry Instrumentation



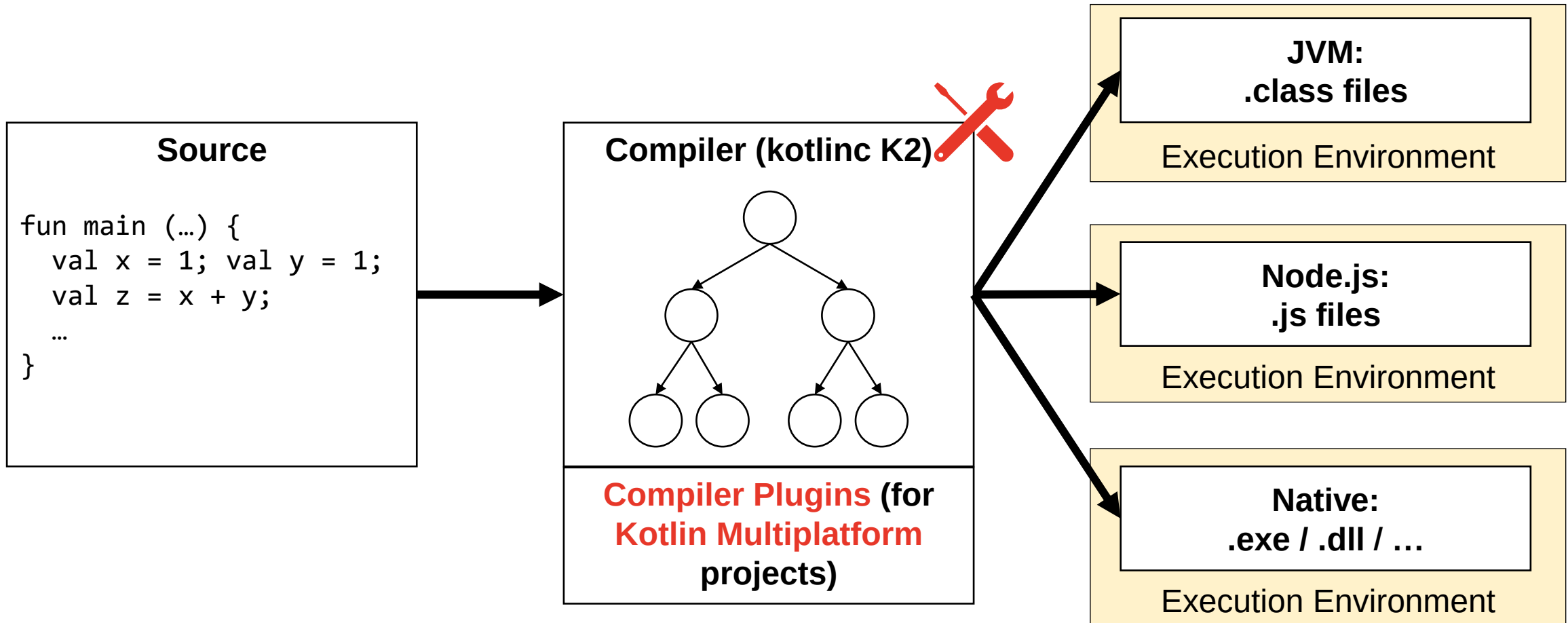
Where to add OTel



OpenTelemetry Instrumentation



Where to add OTel

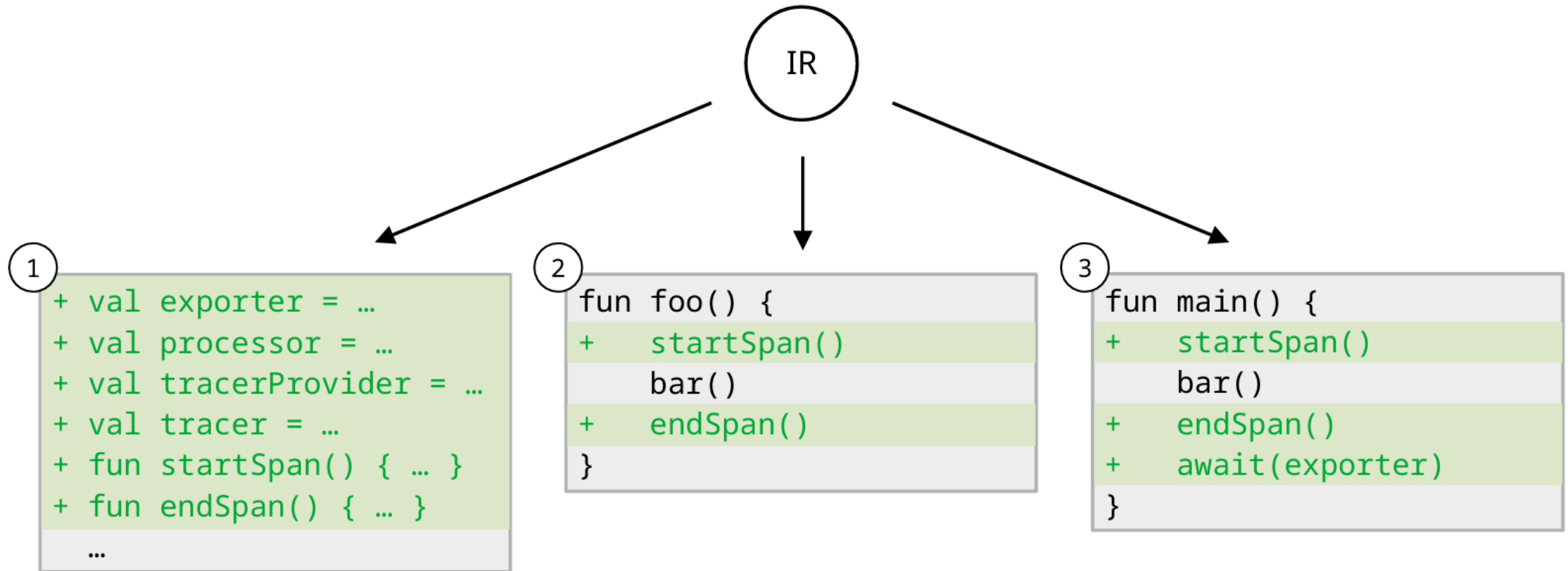


OpenTelemetry Instrumentation using Kotlin Multiplatform Compiler Plugins

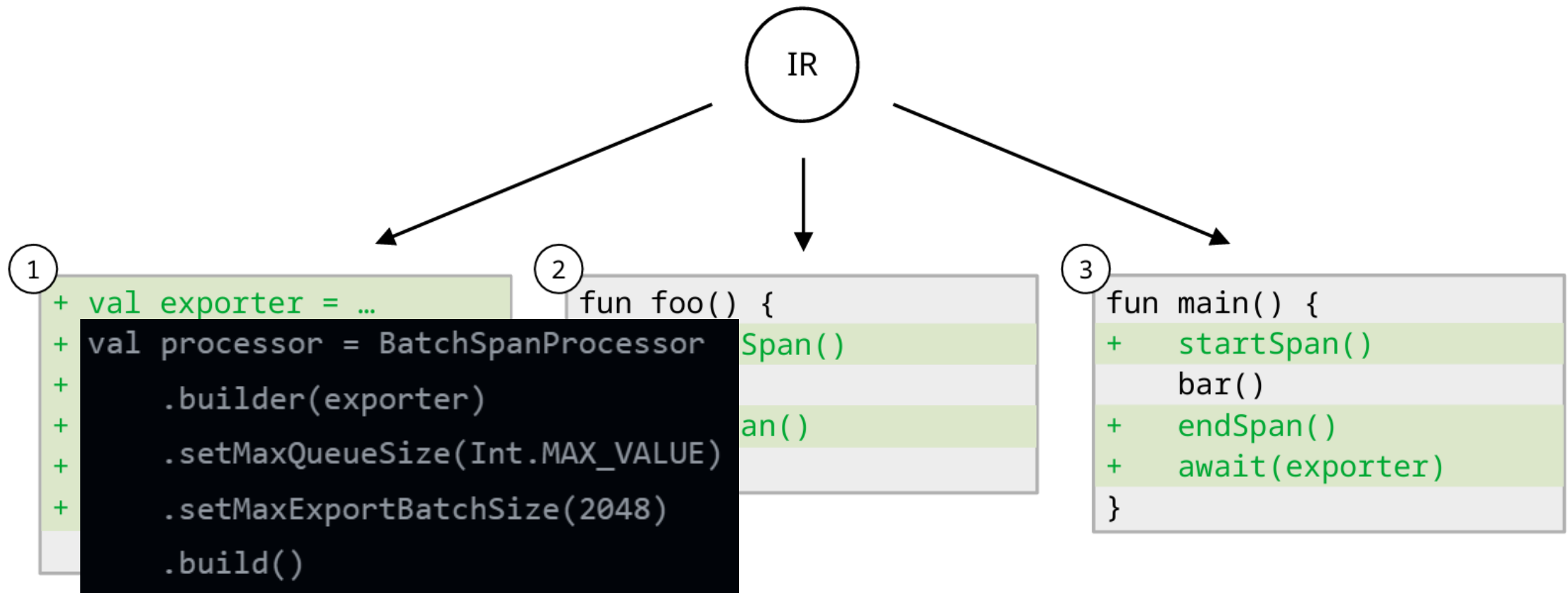


Fabian Schoenberger
Markus Weninger, Institute for System Software

Our IR Instrumentation Plugin



Our IR Instrumentation Plugin



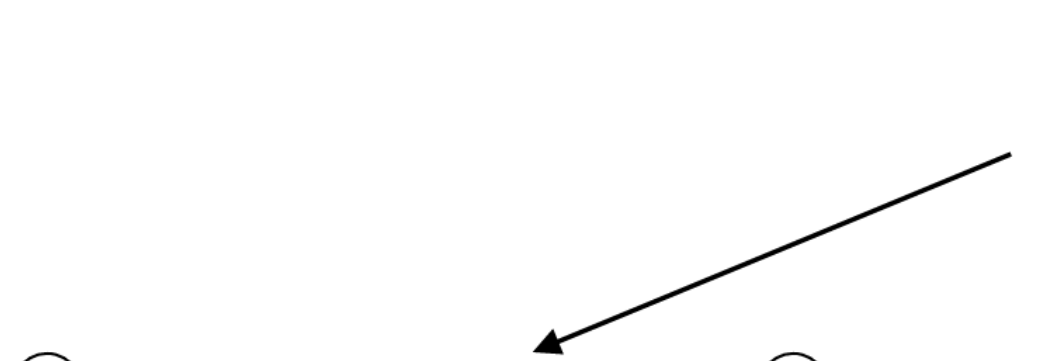
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+   .builder(exporter)  
+   .setMaxQueueSize(Int.MAX_VALUE)  
+   .setMaxExportBatchSize(2048)  
+   .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

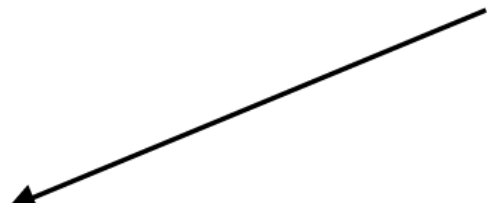
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+   .builder(exporter)  
+   .setMaxQueueSize(Int.MAX_VALUE)  
+   .setMaxExportBatchSize(2048)  
+   .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
                dispatchReceiver = irGetObject(ProcessorCompanion)  
            }  
        }  
    }  
}
```

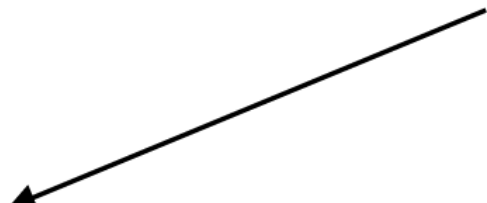
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+   .builder(exporter)  
+   .setMaxQueueSize(Int.MAX_VALUE)  
+   .setMaxExportBatchSize(2048)  
+   .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

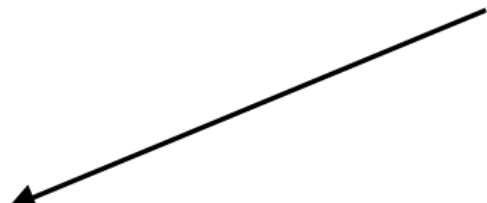

Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

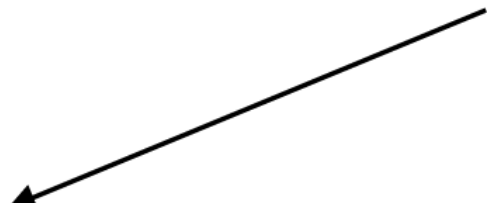
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
                dispatchReceiver = irGetObject(ProcessorCompanion)  
            }  
        }  
    }  
}
```

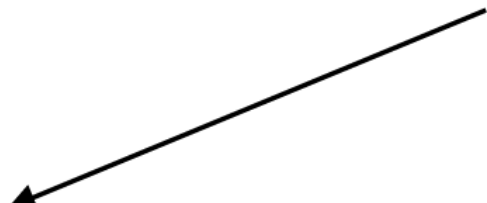
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
                dispatchReceiver = irGetObject(ProcessorCompanion)  
            }  
        }  
    }  
}
```

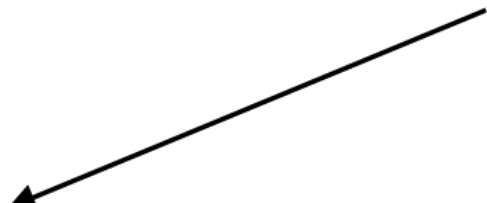
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

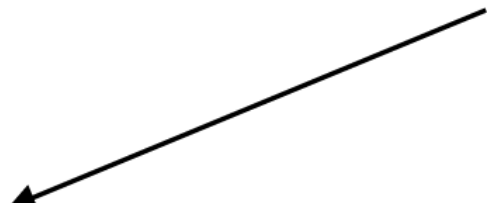
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

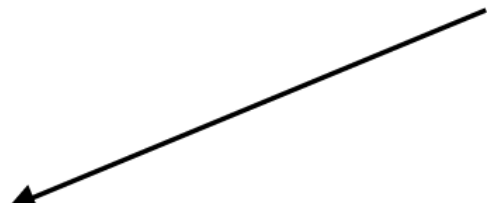
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

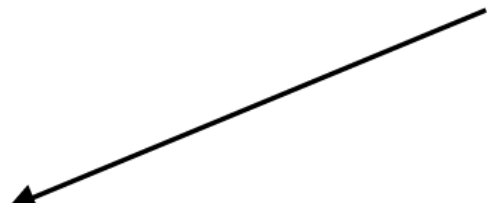
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+ .builder(exporter)  
+ .setMaxQueueSize(Int.MAX_VALUE)  
+ .setMaxExportBatchSize(2048)  
+ .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```

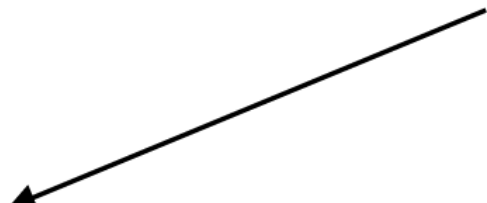
Our IR Instrumentation Plugin

1

```
+ val exporter = ...  
+ val processor = BatchSpanProcessor  
+   .builder(exporter)  
+   .setMaxQueueSize(Int.MAX_VALUE)  
+   .setMaxExportBatchSize(2048)  
+   .build()
```

2

```
fun foo()
```



```
val processor = buildField(  
    name = "_processor",  
    type = Processor.type(),  
    static = true,  
) {  
    initializer = expression {  
        call(ProcessorBuilder_build) {  
            dispatchReceiver = call(ProcessorBuilder_setMaxExportBatchSize) {  
                argument(0, irInt(Int.MAX_VALUE))  
            }  
            dispatchReceiver = call(ProcessorBuilder_setMaxQueueSize) {  
                argument(0, irInt(2048))  
            }  
            dispatchReceiver = call(ProcessorCompanion_builder) {  
                argument(0, irGetField(null, exporter))  
            }  
            dispatchReceiver = irGetObject(ProcessorCompanion)  
        }  
    }  
}
```


One Major Oversight

 README  Apache-2.0 license  

Opentelemetry - Kotlin

This repository contains a port of [Opentelemetry-Java](#) in to Kotlin Multiplatform

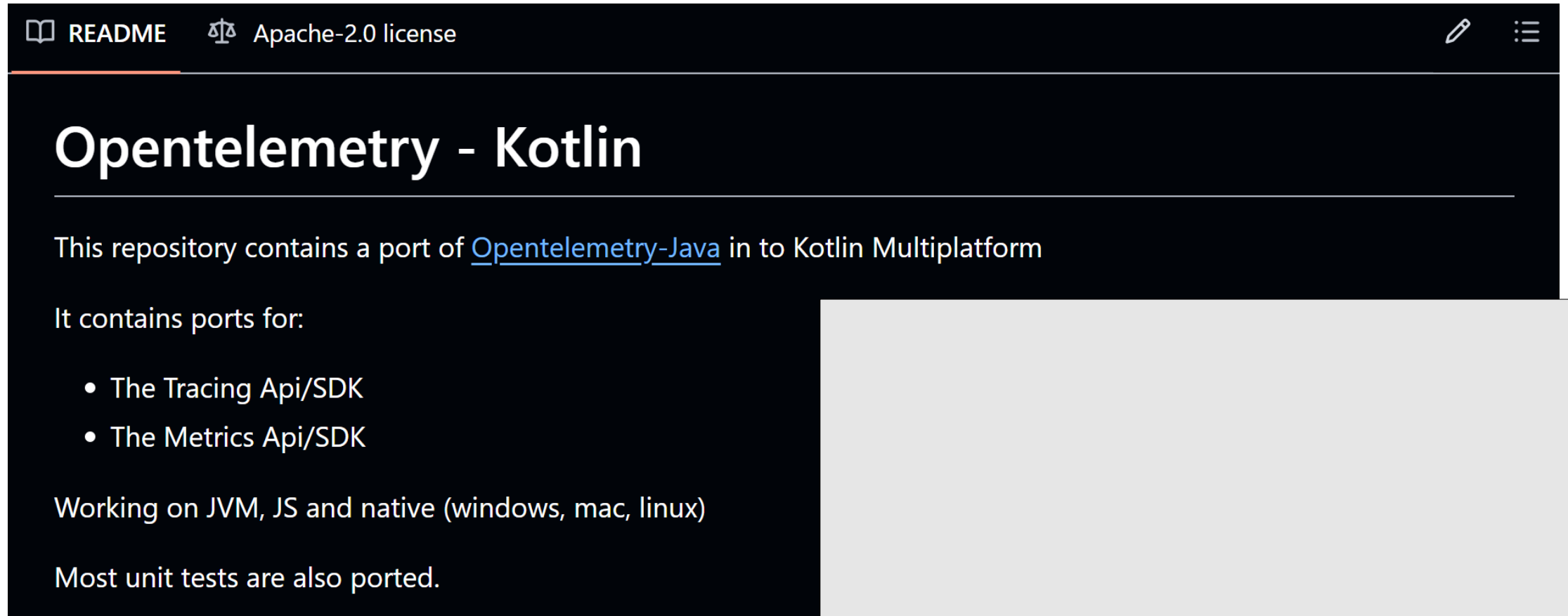
It contains ports for:

- The Tracing Api/SDK
- The Metrics Api/SDK

Working on JVM, JS and native (windows, mac, linux)

Most unit tests are also ported.

One Major Oversight



The screenshot shows the README for the 'Opentelemetry - Kotlin' repository. At the top, there are links for 'README' and 'Apache-2.0 license'. The title 'Opentelemetry - Kotlin' is prominently displayed. Below the title, a paragraph states that the repository is a port of 'Opentelemetry-Java' to Kotlin Multiplatform. A section titled 'It contains ports for:' lists two items: 'The Tracing Api/SDK' and 'The Metrics Api/SDK'. Further down, it mentions 'Working on JVM, JS and native (windows, mac, linux)' and 'Most unit tests are also ported.'.

README Apache-2.0 license

Opentelemetry - Kotlin

This repository contains a port of [Opentelemetry-Java](#) in to Kotlin Multiplatform

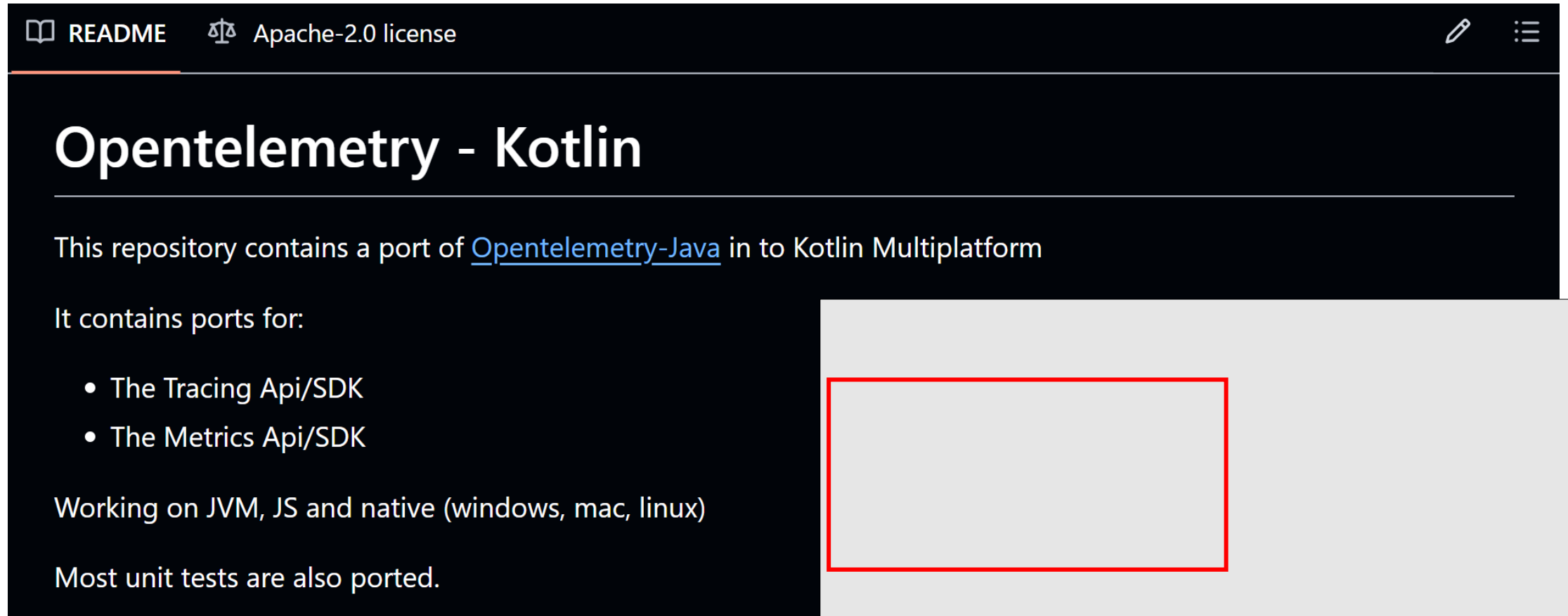
It contains ports for:

- The Tracing Api/SDK
- The Metrics Api/SDK

Working on JVM, JS and native (windows, mac, linux)

Most unit tests are also ported.

One Major Oversight



The screenshot shows the README for the 'Opentelemetry - Kotlin' repository. At the top, there are links for 'README' and 'Apache-2.0 license'. The title 'Opentelemetry - Kotlin' is prominently displayed. Below the title, a paragraph states that the repository is a port of 'Opentelemetry-Java' to Kotlin Multiplatform. A bulleted list indicates that it contains ports for 'The Tracing Api/SDK' and 'The Metrics Api/SDK'. Further down, it mentions that the project is 'Working on JVM, JS and native (windows, mac, linux)' and that 'Most unit tests are also ported'. A large, light gray rectangular area on the right side of the README is highlighted with a red border, representing a missing or placeholder image.

README Apache-2.0 license

Opentelemetry - Kotlin

This repository contains a port of [Opentelemetry-Java](#) in to Kotlin Multiplatform

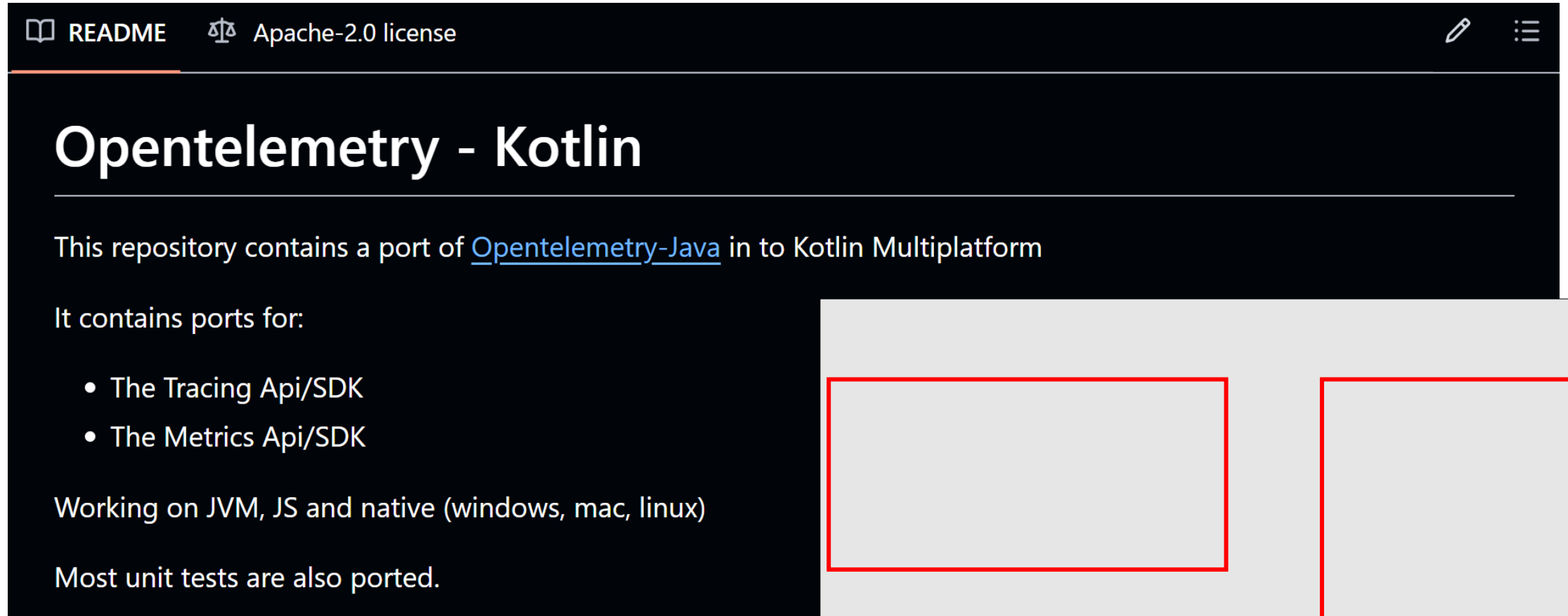
It contains ports for:

- The Tracing Api/SDK
- The Metrics Api/SDK

Working on JVM, JS and native (windows, mac, linux)

Most unit tests are also ported.

One Major Oversight



The screenshot shows the README for the 'Opentelemetry - Kotlin' repository. At the top, there are links for 'README' and 'Apache-2.0 license'. The title 'Opentelemetry - Kotlin' is prominently displayed. Below the title, a paragraph states that the repository is a port of 'Opentelemetry-Java' to Kotlin Multiplatform. A section titled 'It contains ports for:' lists two items: 'The Tracing Api/SDK' and 'The Metrics Api/SDK'. Further down, it mentions 'Working on JVM, JS and native (windows, mac, linux)' and 'Most unit tests are also ported.' To the right of the text, there is a large light gray rectangular area containing two empty red rectangular boxes, one horizontal and one vertical, which appear to be placeholders for images or diagrams.

README Apache-2.0 license

Opentelemetry - Kotlin

This repository contains a port of [Opentelemetry-Java](#) in to Kotlin Multiplatform

It contains ports for:

- The Tracing Api/SDK
- The Metrics Api/SDK

Working on JVM, JS and native (windows, mac, linux)

Most unit tests are also ported.

One Major Oversight

📖 README ⚖️ Apache-2.0 license ✎ ☰

Opentelemetry - Kotlin

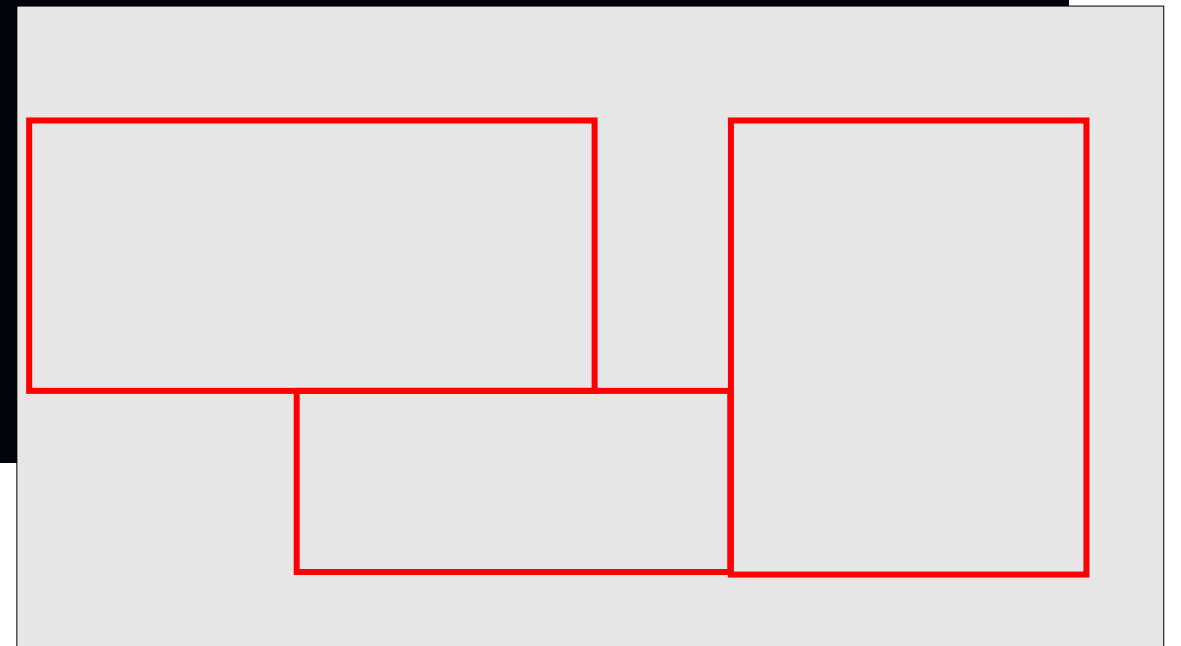
This repository contains a port of [Opentelemetry-Java](#) in to Kotlin Multiplatform

It contains ports for:

- The Tracing Api/SDK
- The Metrics Api/SDK

Working on JVM, JS and native (windows, mac, linux)

Most unit tests are also ported.



One Major Oversight

📖 README ⚖️ Apache-2.0 license ✎ ☰

Opentelemetry - Kotlin

This repository contains a port of [Opentelemetry-Java](#) in to Kotlin Multiplatform

It contains ports for:

- The Tracing Api/SDK
- The Metrics Api/SDK

Working on JVM, JS and native (windows, mac, linux)

Most unit tests are also ported.

Exporter NOT part of SDK! ☹️

Exporter

```
class OtlpExporter(  
    val host: String,  
    val service: String  
) : SpanExporter {  
    private val client = HttpClient()  
    private val scope = CoroutineScope(Dispatchers.Default)  
    private val jobs: MutableList<Job> = mutableListOf()  
  
    suspend fun await() {  
        jobs.joinAll()  
    }  
  
    override fun export(  
        spans: Collection<SpanData>  
    ): CompletableResultCode {  
        val job = scope.launch {  
            val payload = spans.serialize(service)  
            client.post("http://$host/v1/traces") {  
                contentType(ContentType.Application.Json)  
                setBody(payload)  
            }  
        }  
        jobs.add(job)  
  
        return CompletableResultCode.ofSuccess()  
    }  
  
    override fun flush(): CompletableResultCode {  
        return CompletableResultCode.ofSuccess()  
    }  
  
    override fun shutdown() = flush()  
}
```

Exporter

```
class OtlpExporter(  
    val host: String,  
    val service: String  
) : SpanExporter {  
    private val client = HttpClient()  
    private val scope = CoroutineScope(Dispatchers.Default)  
    private val jobs: MutableList<Job> = mutableListOf()
```

```
    override fun export(  
        spans: Collection<SpanData>  
    ): CompletableResultCode {  
        val job = scope.launch {  
            val payload = spans.serialize(service)  
            client.post("http://$host/v1/traces") {  
                contentType(ContentType.Application.Json)  
                setBody(payload)  
            }  
        }  
        jobs.add(job)  
  
        return CompletableResultCode.ofSuccess()  
    }  
}
```


Exporter

```
class OtlpExporter(  
    val host: String,
```

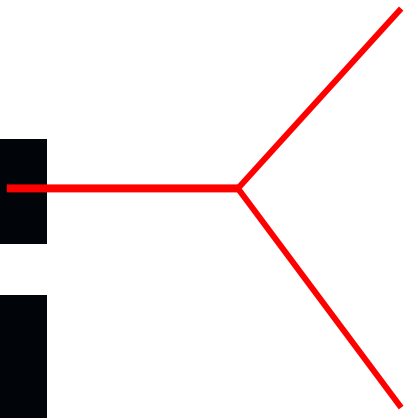
```
private val client = HttpClient()
```

```
private val scope = CoroutineScope(Dispatchers.Default)  
private val jobs: MutableList<Job> = mutableListOf()
```

```
override fun export(  
    spans: Collection<SpanData>  
): CompletableResultCode {  
    val job = scope.launch {  
        val payload = spans.serialize(service)  
        client.post("http://$host/v1/traces") {  
            contentType(ContentType.Application.Json)  
            setBody(payload)  
        }  
    }  
    jobs.add(job)  
  
    return CompletableResultCode.ofSuccess()  
}
```

Exporter

```
class OtlpExporter(  
    val host: String,  
  
    private val client = HttpClient()  
  
    private val scope = CoroutineScope(Dispatchers.Default)  
    private val jobs: MutableList<Job> = mutableListOf()  
  
    override fun export(  
        spans: Collection<SpanData>  
    ): CompletableResultCode {  
        val job = scope.launch {  
            val payload = spans.serialize(service)  
            client.post("http://$host/v1/traces") {  
                contentType(ContentType.Application.Json)  
                setBody(payload)  
            }  
        }  
        jobs.add(job)  
  
        return CompletableResultCode.ofSuccess()  
    }  
}
```



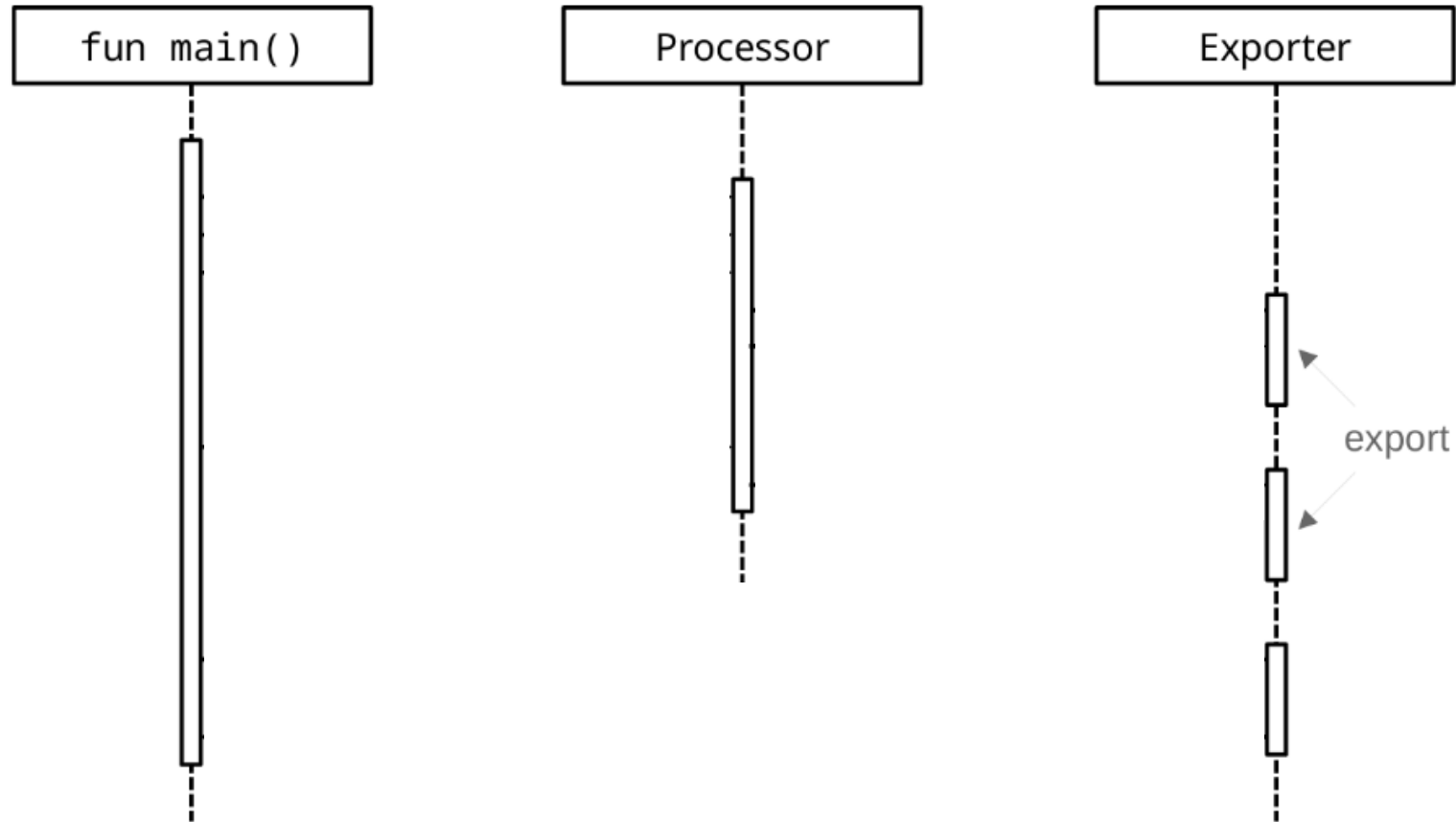
Exporter

```
class OtlpExporter(  
    val host: String,  
  
    private val client = HttpClient()
```

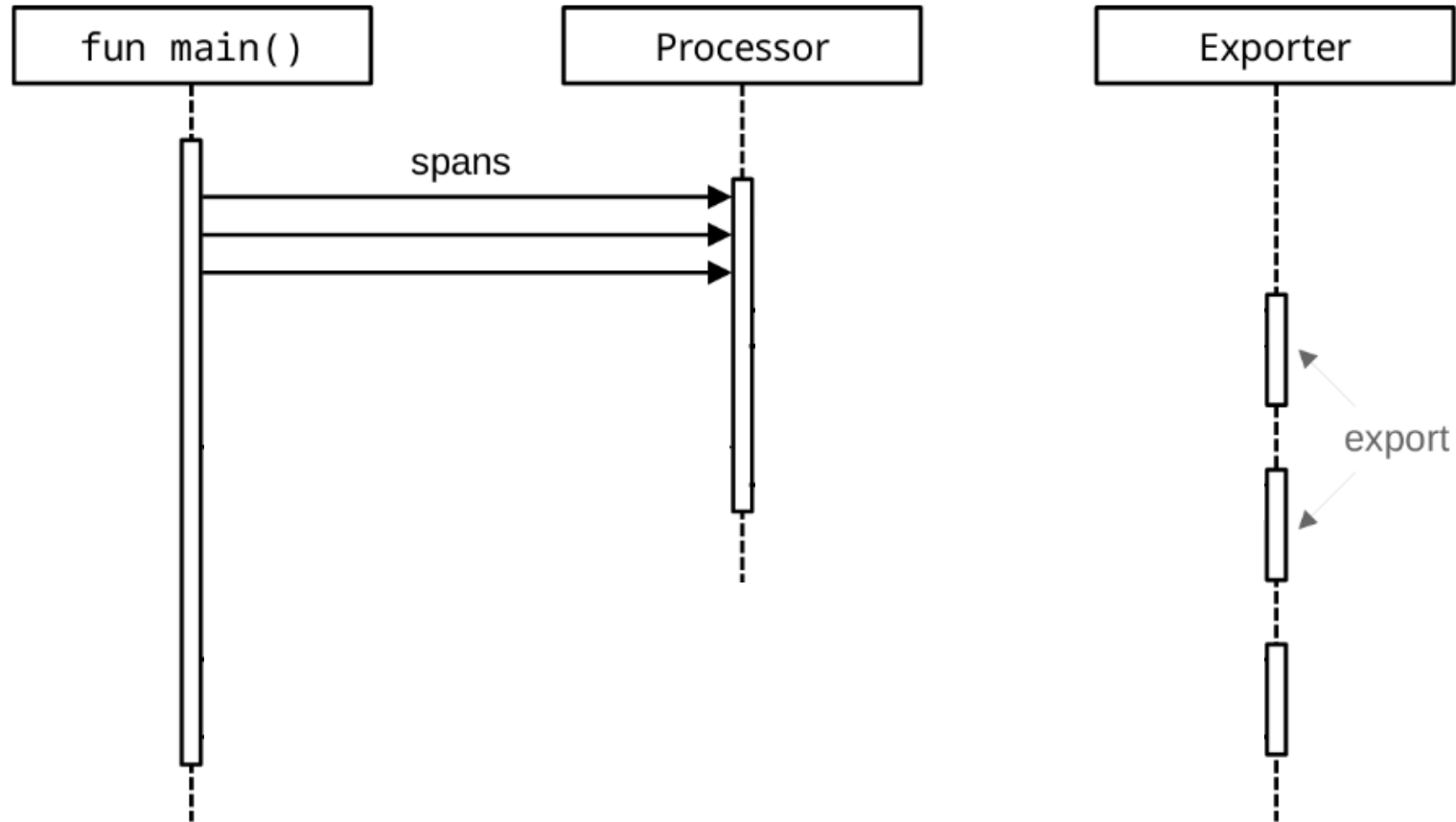
```
    private val scope = CoroutineScope(Dispatchers.Default)  
    private val jobs: MutableList<Job> = mutableListOf()  
  
    override fun export(  
        spans: Collection<SpanData>  
    ): CompletableResultCode {  
        val job = scope.launch {  
            val payload = spans.serialize(service)  
            client.post("http://$host/v1/traces") {  
                contentType(ContentType.Application.Json)  
                setBody(payload)  
            }  
        }  
        jobs.add(job)  
  
        return CompletableResultCode.ofSuccess()  
    }  
}
```

```
sourceSets {  
    ...  
    jvmMain.dependencies {  
        implementation("io.ktor:ktor-client-java:3.1.0")  
    }  
    jsMain.dependencies {  
        implementation("io.ktor:ktor-client-js:3.1.0")  
    }  
    linuxMain.dependencies {  
        implementation("io.ktor:ktor-client-curl:3.1.0")  
    }  
}
```

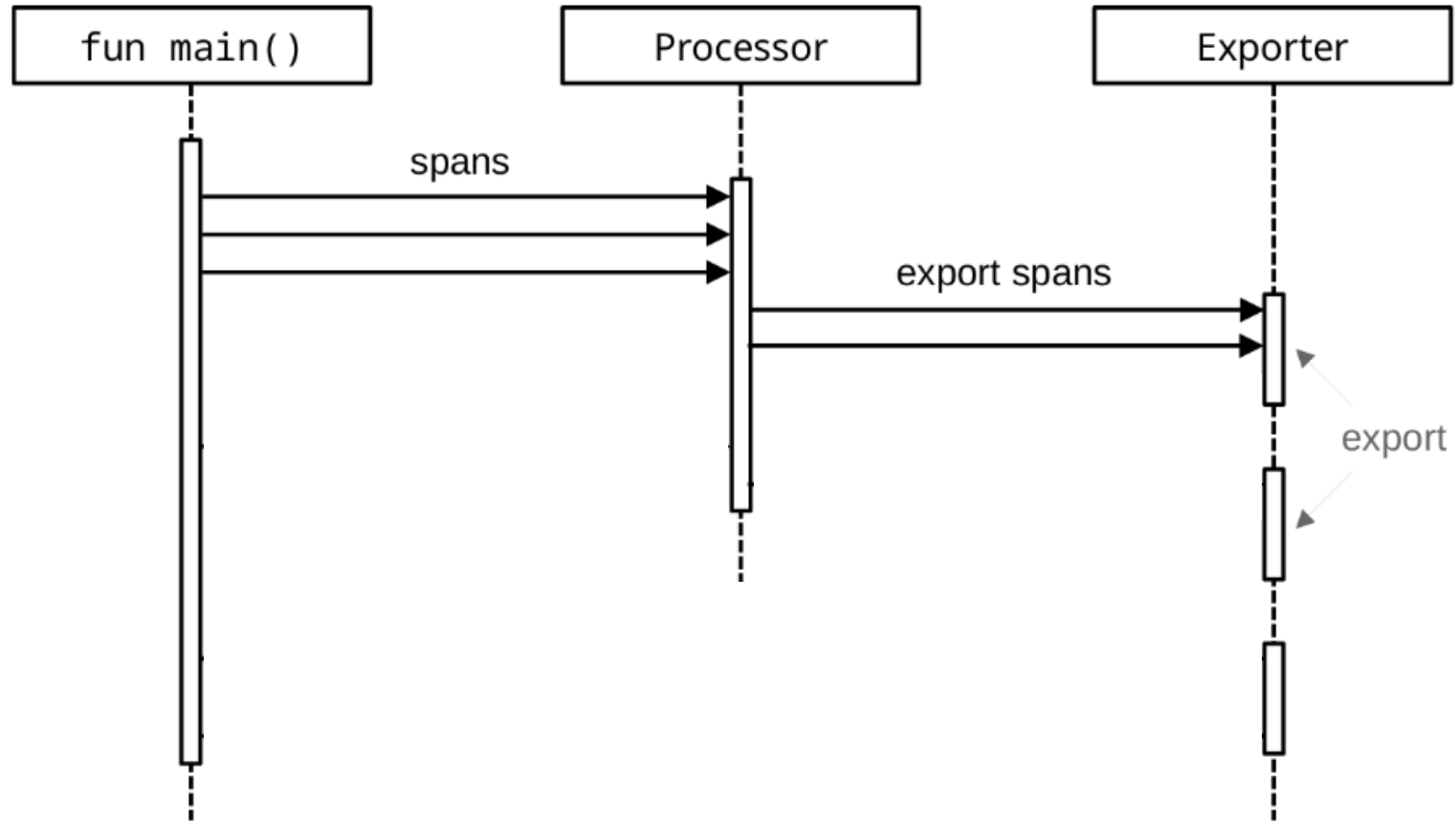
Waiting for the Exporter



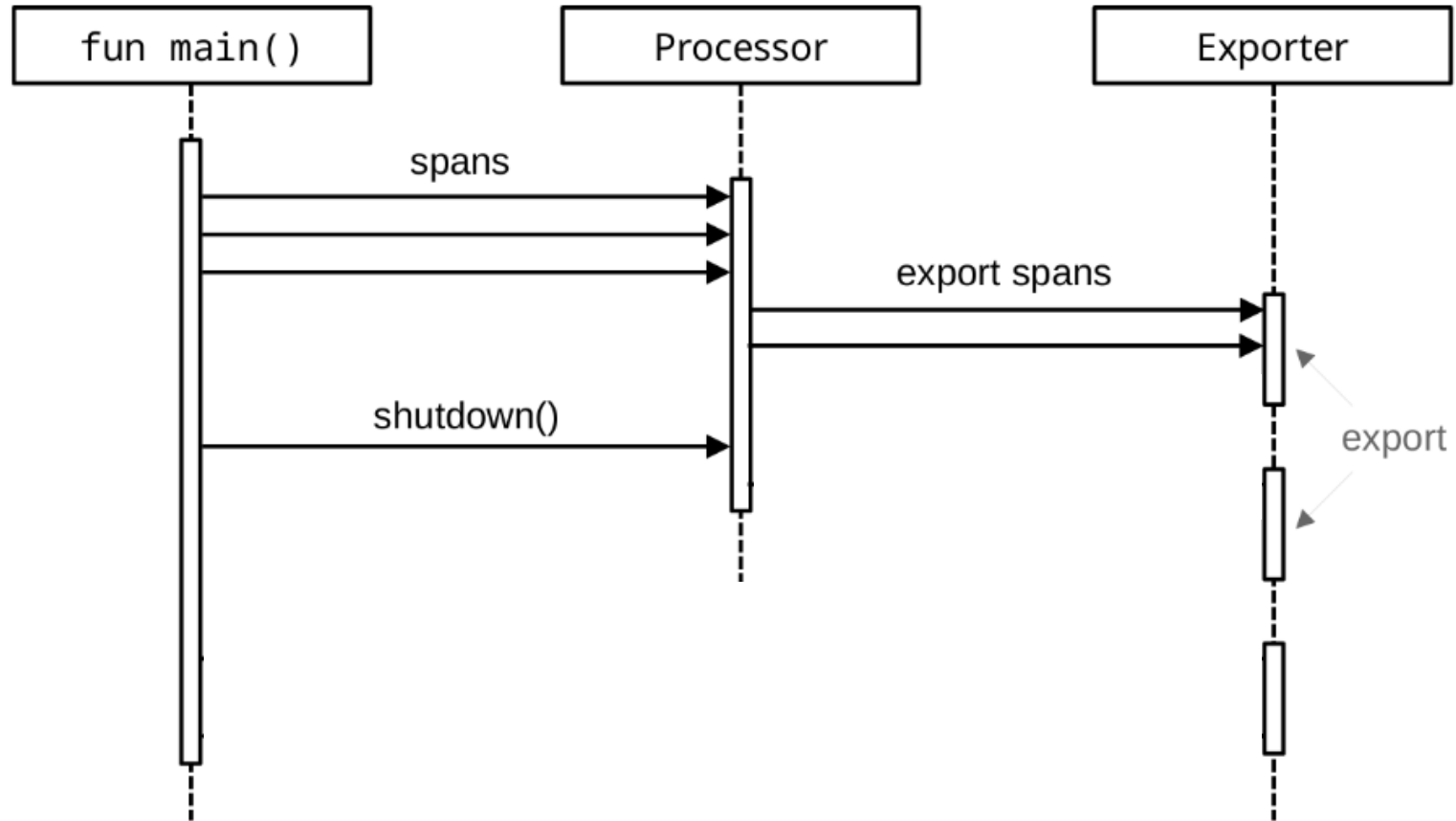
Waiting for the Exporter



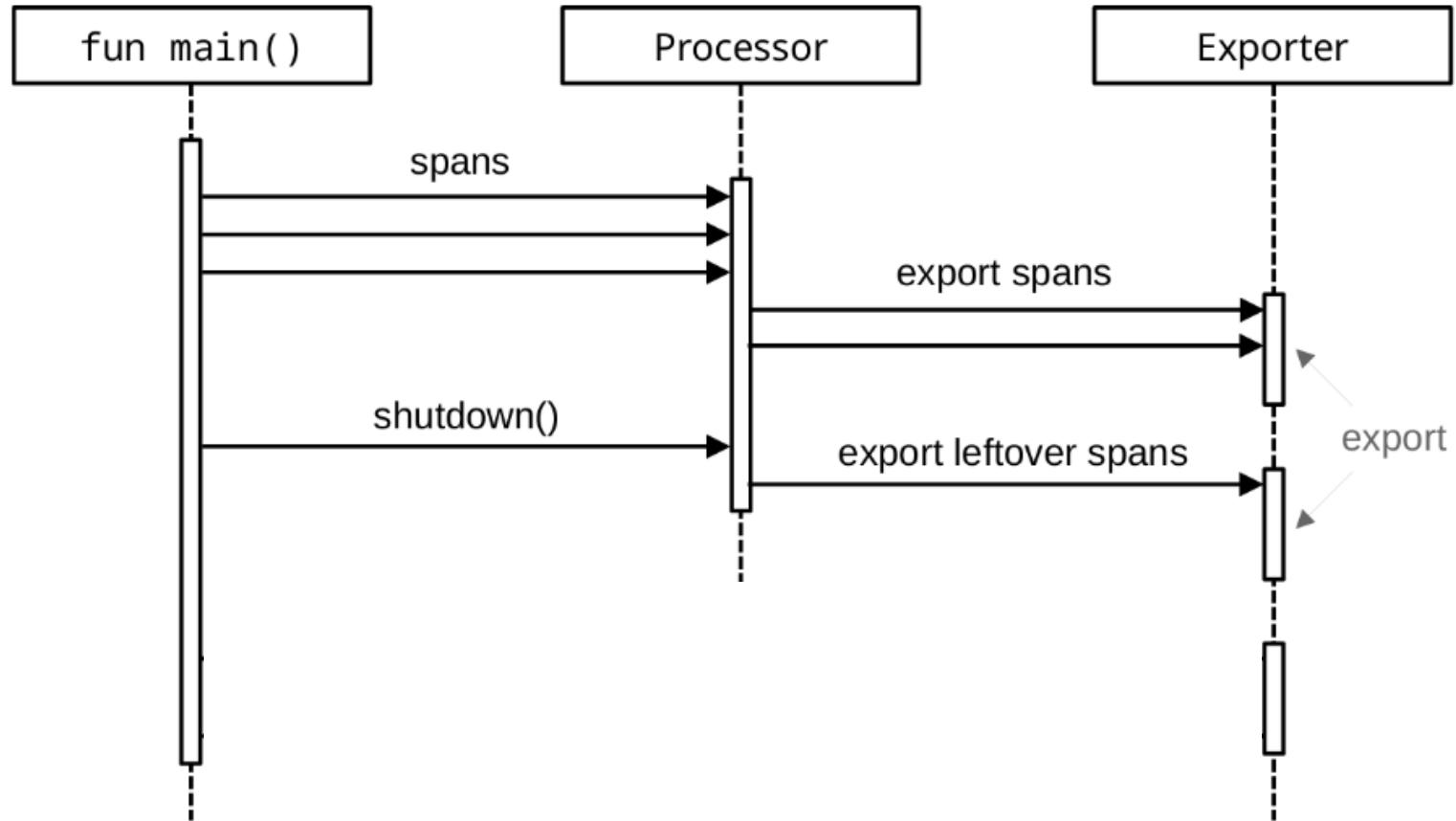
Waiting for the Exporter



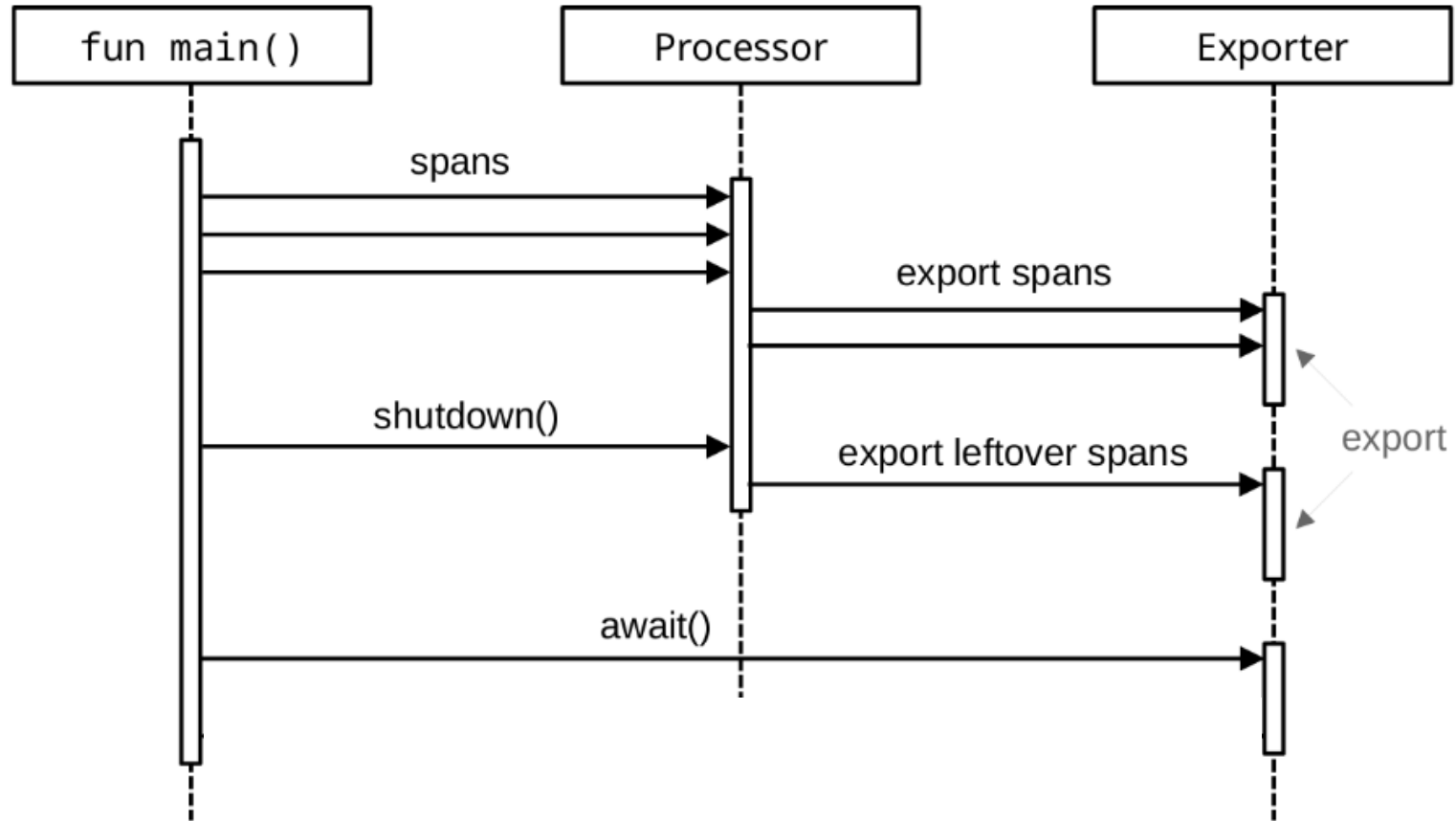
Waiting for the Exporter



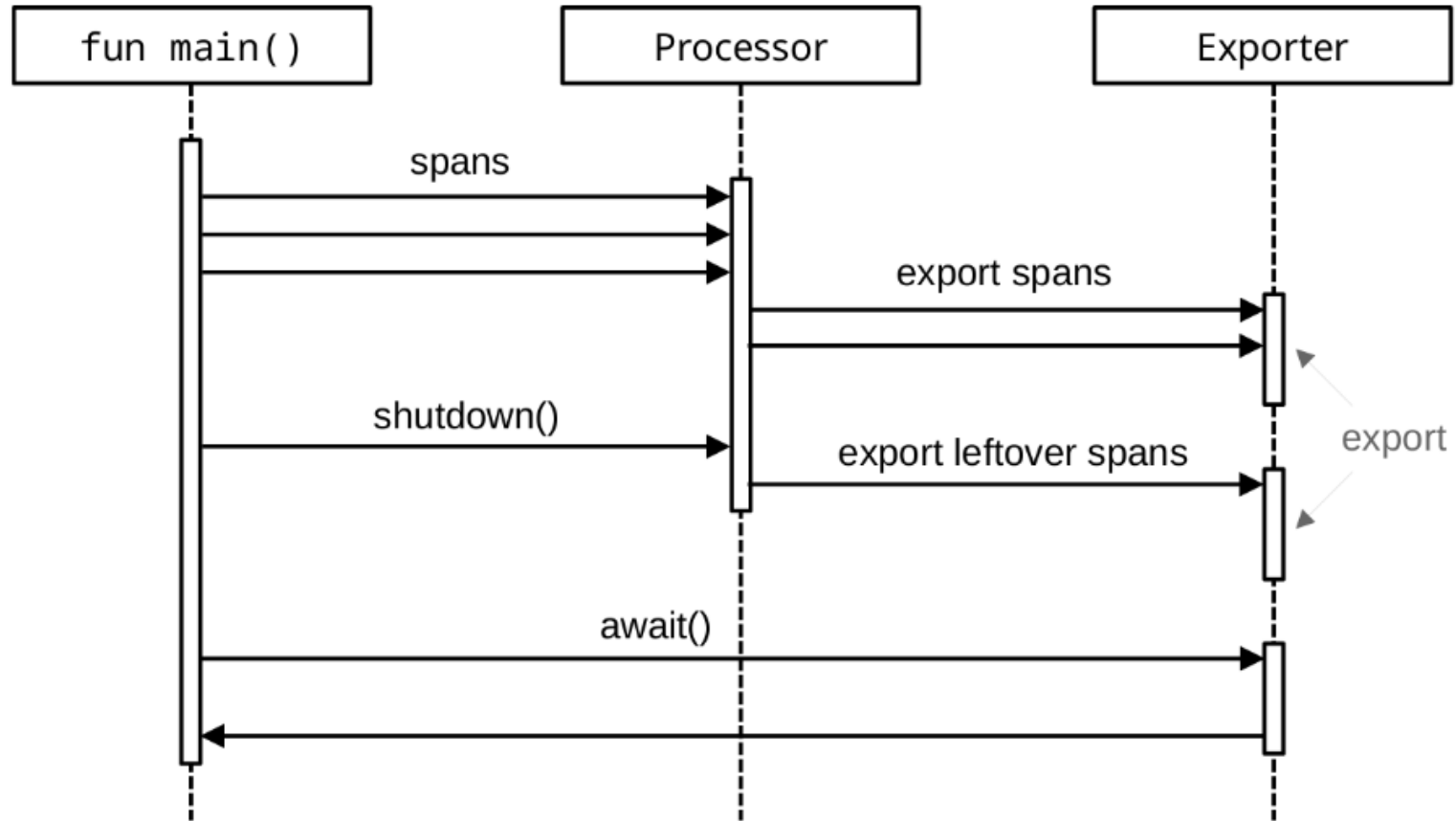
Waiting for the Exporter



Waiting for the Exporter



Waiting for the Exporter



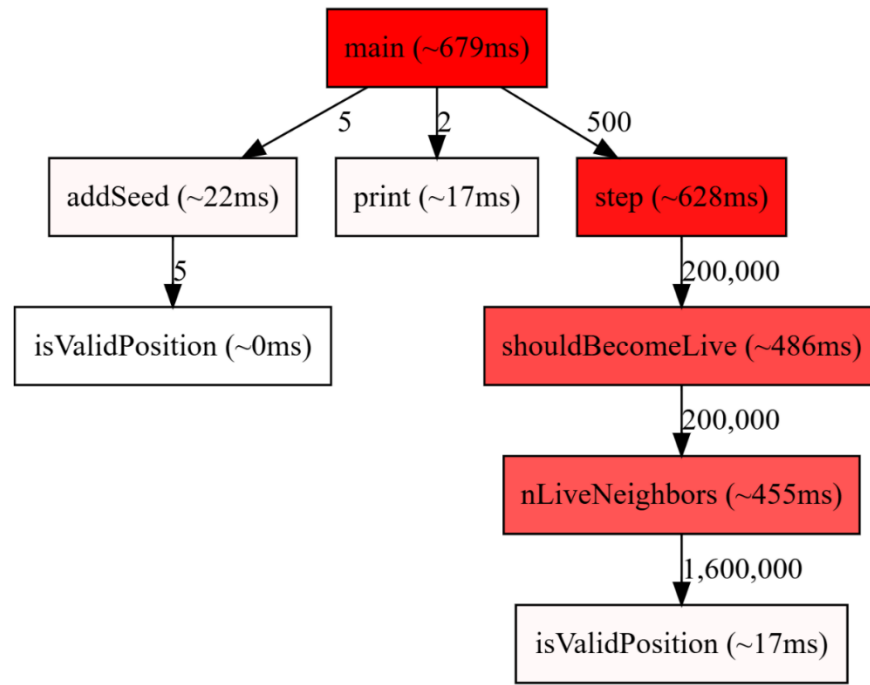
Results from last year (kperf with custom format)

Results from last year (kperf with custom format)

```
>;1      // main entered  
>;2      // sayHello entered  
<;230    // sayHello left after 0.23ms  
>;2      // sayHello entered  
<;250    // sayHello left after 0.25ms  
<;7500   // main left after 7.5ms
```

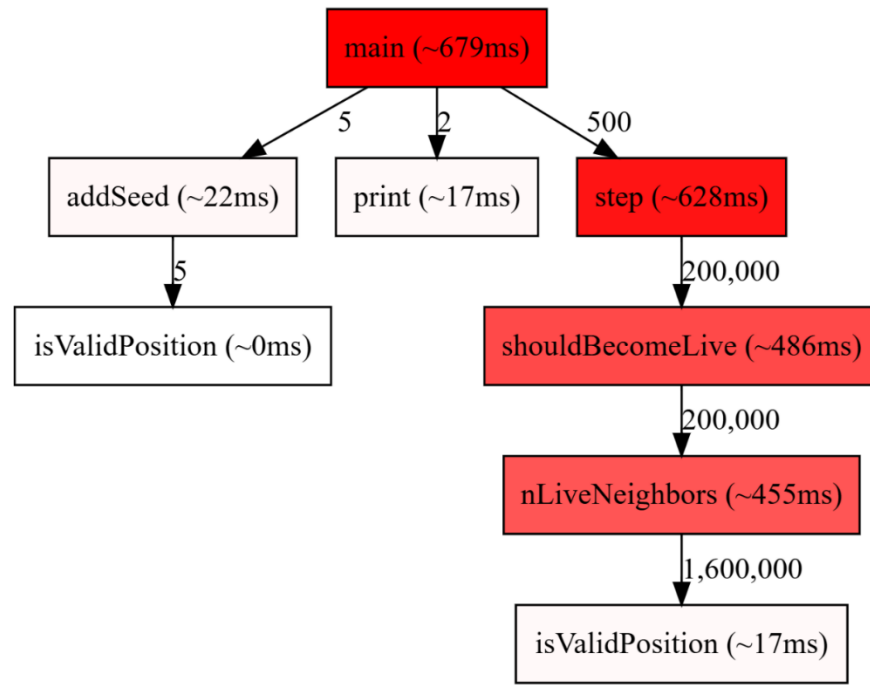
Results from last year (kperf with custom format)

```
>;1      // main entered  
>;2      // sayHello entered  
<;230    // sayHello left after 0.23ms  
>;2      // sayHello entered  
<;250    // sayHello left after 0.25ms  
<;7500   // main left after 7.5ms
```



Results from last year (kperf with custom format)

```
>;1      // main entered
>;2      // sayHello entered
<;230    // sayHello left after 0.23ms
>;2      // sayHello entered
<;250    // sayHello left after 0.25ms
<;7500   // main left after 7.5ms
```

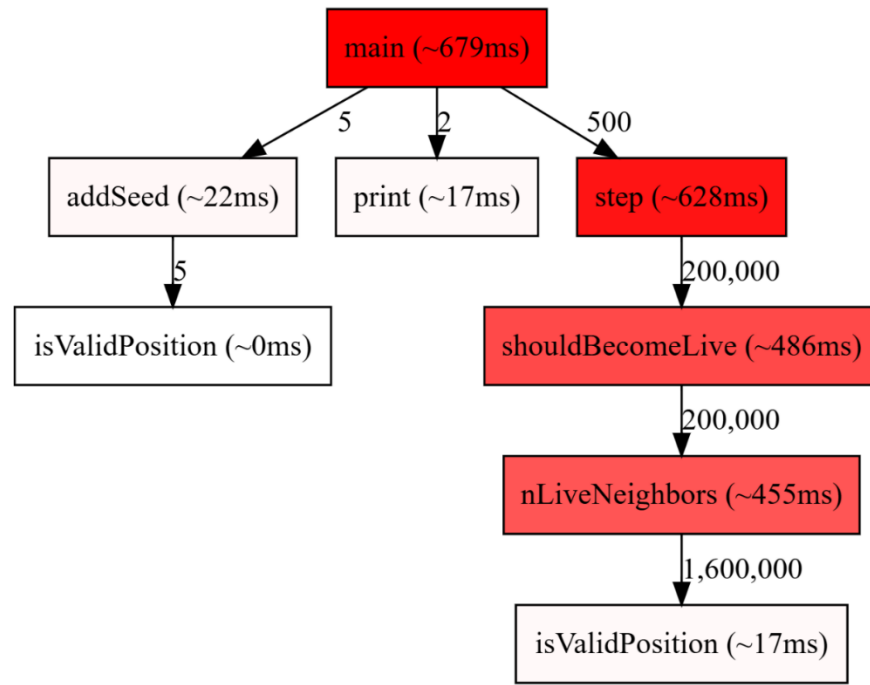


<i>KMP</i>	JVM	JS	native (.exe)
<i>Game of Life</i>			
Build Size w/o plugin	1753.2 kB	106.9 kB	741.9 kB
Build Size w/ plugin	1912.5 kB	167.9 kB	851.5 kB
Run Time w/o plugin	83.0 - 85.3 ms	127.5 - 130.8 ms	242.9 - 248.0 ms
Run Time w/ plugin	411.2 - 429.8 ms	9364.8 - 9771.0 ms	5322.4 - 5497.0 ms

Table 1: Build sizes (incl. Kotlin runtime + dependencies) and run times (95% conf. int. of 100 runs).

Results from last year (kperf with custom format)

```
>;1      // main entered
>;2      // sayHello entered
<;230    // sayHello left after 0.23ms
>;2      // sayHello entered
<;250    // sayHello left after 0.25ms
<;7500   // main left after 7.5ms
```



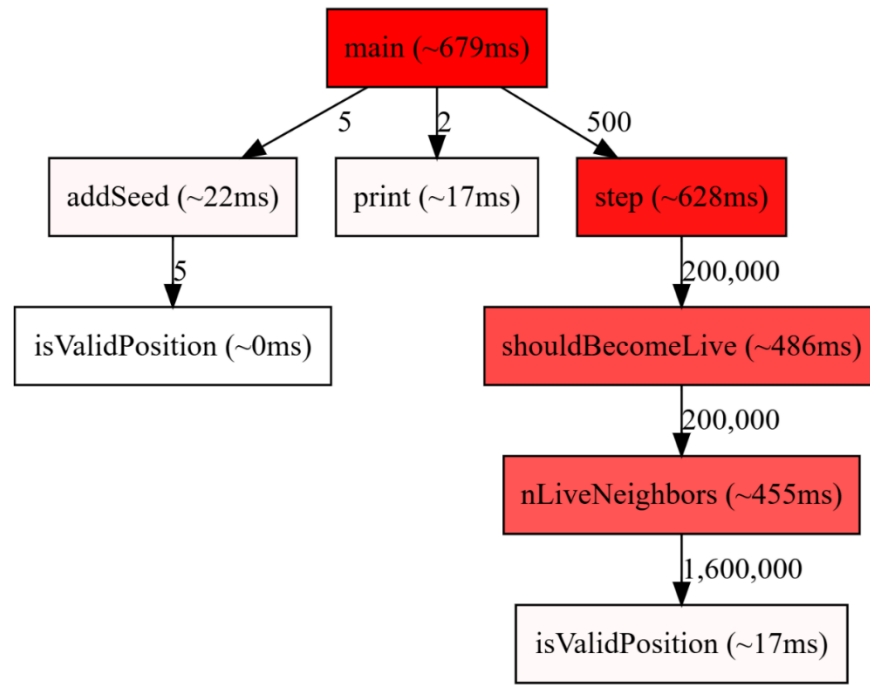
<i>KMP</i> <i>Game of Life</i>	JVM	JS	native (.exe)
Build Size w/o plugin	1753.2 kB	106.9 kB	741.9 kB
Build Size w/ plugin	1912.5 kB	167.9 kB	851.5 kB
Run Time w/o plugin	83.0 - 85.3 ms	127.5 - 130.8 ms	242.9 - 248.0 ms
Run Time w/ plugin	411.2 - 429.8 ms	9364.8 - 9771.0 ms	5322.4 - 5497.0 ms



Table 1: Build sizes (incl. Kotlin runtime + dependencies) and run times (95% conf. int. of 100 runs).

Results from last year (kperf with custom format)

```
>;1      // main entered
>;2      // sayHello entered
<;230    // sayHello left after 0.23ms
>;2      // sayHello entered
<;250    // sayHello left after 0.25ms
<;7500   // main left after 7.5ms
```



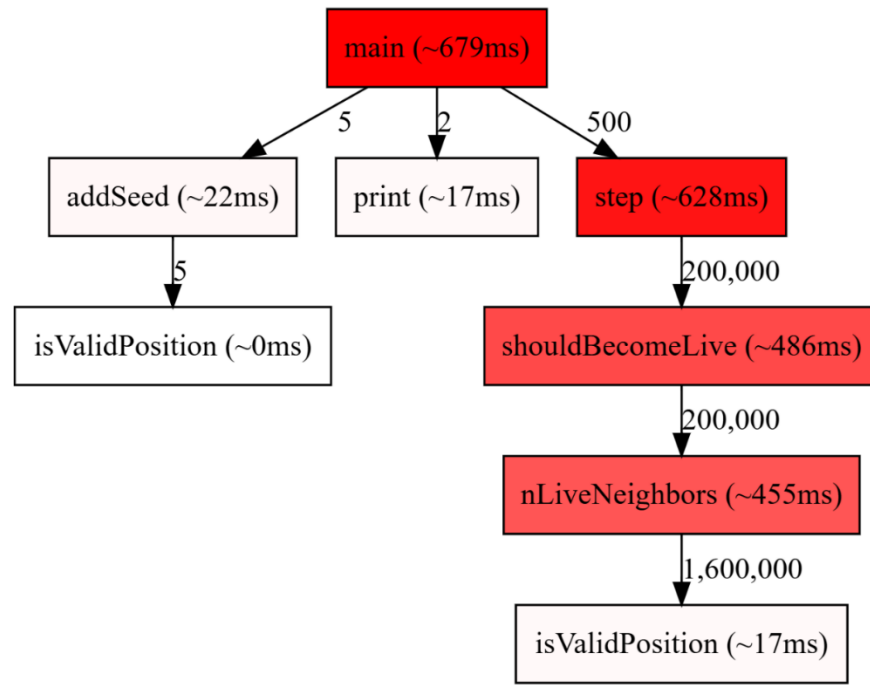
<i>KMP</i>	JVM	JS	native (.exe)
<i>Game of Life</i>			
Build Size w/o plugin	1753.2 kB	106.9 kB	741.9 kB
Build Size w/ plugin	1912.5 kB	167.9 kB	851.5 kB
Run Time w/o plugin	83.0 - 85.3 ms	127.5 - 130.8 ms	242.9 - 248.0 ms
Run Time w/ plugin	411.2 - 429.8 ms	9364.8 - 9771.0 ms	5322.4 - 5497.0 ms

x5

Table 1: Build sizes (incl. Kotlin runtime + dependencies) and run times (95% conf. int. of 100 runs).

Results from last year (kperf with custom format)

```
>;1      // main entered
>;2      // sayHello entered
<;230    // sayHello left after 0.23ms
>;2      // sayHello entered
<;250    // sayHello left after 0.25ms
<;7500   // main left after 7.5ms
```



<i>KMP</i>	JVM	JS	native (.exe)
<i>Game of Life</i>			
Build Size w/o plugin	1753.2 kB	106.9 kB	741.9 kB
Build Size w/ plugin	1912.5 kB	167.9 kB	851.5 kB
Run Time w/o plugin	83.0 - 85.3 ms	127.5 - 130.8 ms	242.9 - 248.0 ms
Run Time w/ plugin	411.2 - 429.8 ms	9364.8 - 9771.0 ms	5322.4 - 5497.0 ms

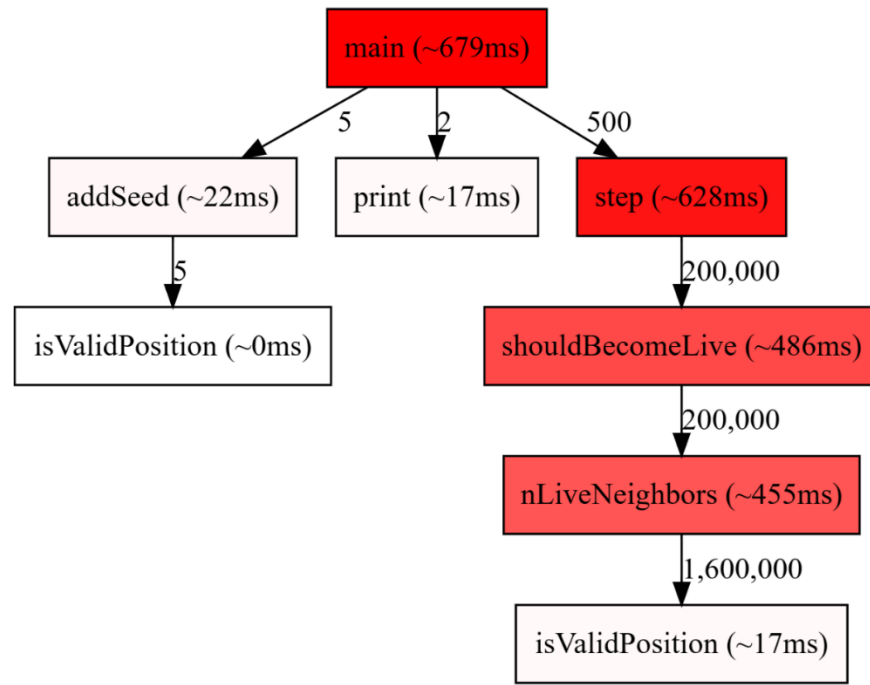
x5

x74

Table 1: Build sizes (incl. Kotlin runtime + dependencies) and run times (95% conf. int. of 100 runs).

Results from last year (kperf with custom format)

```
>;1      // main entered
>;2      // sayHello entered
<;230    // sayHello left after 0.23ms
>;2      // sayHello entered
<;250    // sayHello left after 0.25ms
<;7500   // main left after 7.5ms
```



<i>KMP</i>	JVM	JS	native (.exe)
<i>Game of Life</i>			
Build Size w/o plugin	1753.2 kB	106.9 kB	741.9 kB
Build Size w/ plugin	1912.5 kB	167.9 kB	851.5 kB
Run Time w/o plugin	83.0 - 85.3 ms	127.5 - 130.8 ms	242.9 - 248.0 ms
Run Time w/ plugin	411.2 - 429.8 ms	9364.8 - 9771.0 ms	5322.4 - 5497.0 ms

x5

x74

x22

Table 1: Build sizes (incl. Kotlin runtime + dependencies) and run times (95% conf. int. of 100 runs).

Results from kperf-otel

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

x255

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

x255

4.3 ns/func

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

x255

4.3 ns/func

x5,273

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

x255

4.3 ns/func

x5,273

54.8 ns/func

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

x255

4.3 ns/func

x5,273

54.8 ns/func

x19,300

Results from kperf-otel

		JVM	JavaScript	Native (Linux)
Reference	execution	$[33.84 \pm 0.59]$	$[20.82 \pm 0.37]$	$[3.68 \pm 0.18]$
Manual	execution	$[9, 269.59 \pm 240.36]$	$[19, 742.54 \pm 118.23]$	$[72, 831.72 \pm 957.3]$
	total	$[9, 602.45 \pm 243.77]$	$[110, 749.40 \pm 533.53]$	$[73, 270.53 \pm 956.53]$
Automatic	execution	$[8, 352.11 \pm 126.42]$	$[19, 584.33 \pm 124.95]$	$[70, 581.19 \pm 700.64]$
	total	$[8, 644.29 \pm 142.92]$	$[109, 786.73 \pm 420.45]$	$[71, 018.52 \pm 697.57]$

x255

4.3 ns/func

x5,273

54.8 ns/func

x19,300

35.5 ns/func

Future Work: Target-specific Performance Impact Analysis of Kotlin IR Instrumentation

Future Work: Target-specific Performance Impact Analysis of Kotlin IR Instrumentation

StringBuilder

File Writing

HTTP POST

...

Future Work: Target-specific Performance Impact Analysis of Kotlin IR Instrumentation

StringBuilder	JVM
File Writing	JVM
HTTP POST	JVM
...	

Future Work: Target-specific Performance Impact Analysis of Kotlin IR Instrumentation

StringBuilder	JVM	JS
File Writing	JVM	JS
HTTP POST	JVM	JS
...		

Future Work: Target-specific Performance Impact Analysis of Kotlin IR Instrumentation

StringBuilder	JVM	JS	Native/Windows
File Writing	JVM	JS	Native/Windows
HTTP POST	JVM	JS	Native/Windows
...			

Future Work: Target-specific Performance Impact Analysis of Kotlin IR Instrumentation

StringBuilder	JVM	JS	Native/Windows	Native/Linux
File Writing	JVM	JS	Native/Windows	Native/Linux
HTTP POST	JVM	JS	Native/Windows	Native/Linux
...				

Future Work: Multi-Threading

Listing 11: Instrumented OpenTelemetry functions

```
1 fun _startSpan(name: String, parent: Context): Span {
2     val span = tracer.spanBuilder(name)
3         .setParent(parent)
4         .setStartTimestamp(Clock.System.now())
5         .startSpan()
6     parent.with(span).makeCurrent() // create new span-context
7     return span
8 }
9
10 fun _endSpan(span: Span, parent: Context) {
11     span.end(Clock.System.now())
12     parent.makeCurrent()
13 }
```

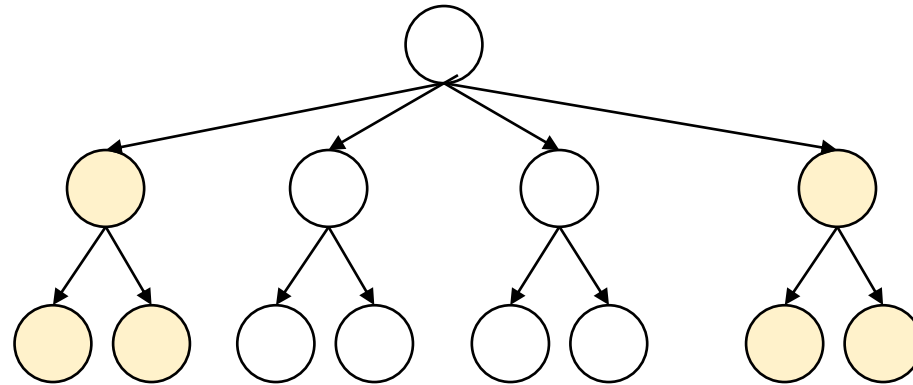
Future Work: Multi-Threading

Listing 11: Instrumented Op

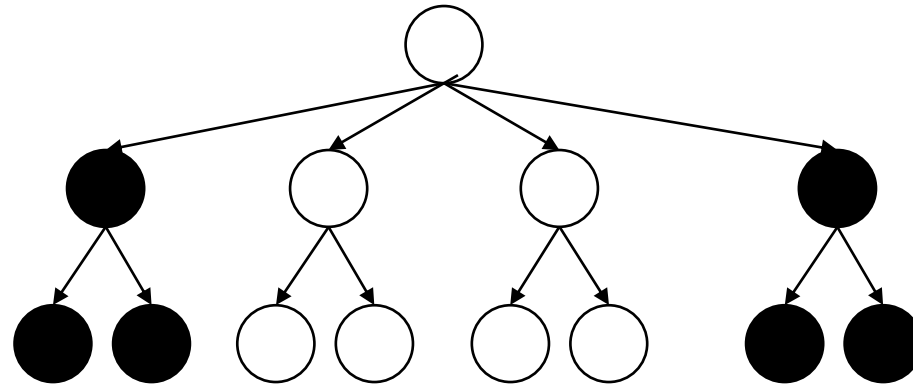
Currently: One
Context for all threads

```
1 fun _startSpan(name: String, parent: Context): Span {
2     val span = tracer.spanBuilder(name)
3         .setParent(parent)
4         .setStartTimestamp(Clock.System.now())
5         .startSpan()
6     parent.with(span).makeCurrent() // create new span-context
7     return span
8 }
9
10 fun _endSpan(span: Span, parent: Context) {
11     span.end(Clock.System.now())
12     parent.makeCurrent()
13 }
```

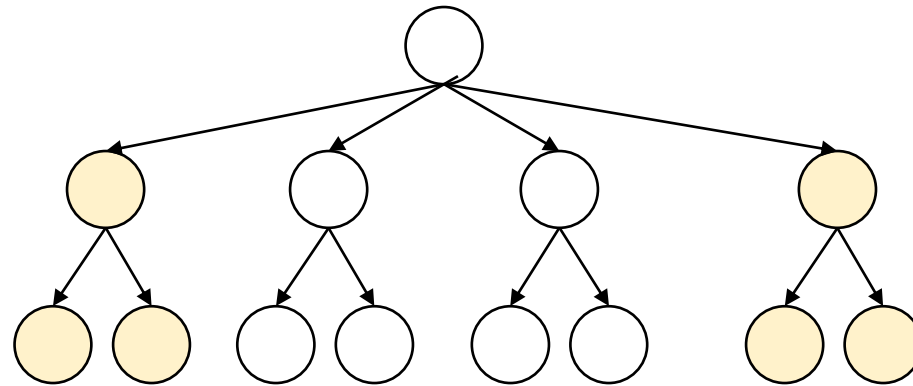
Future Work: Adaptive Tracing



Future Work: Adaptive Tracing



Future Work: Adaptive Tracing

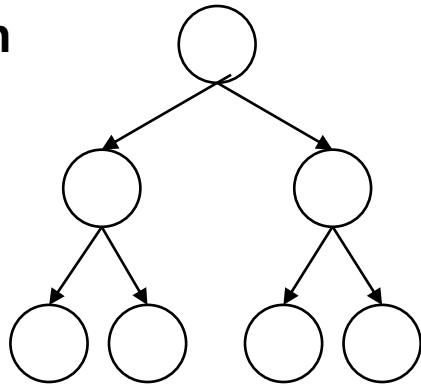


Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin

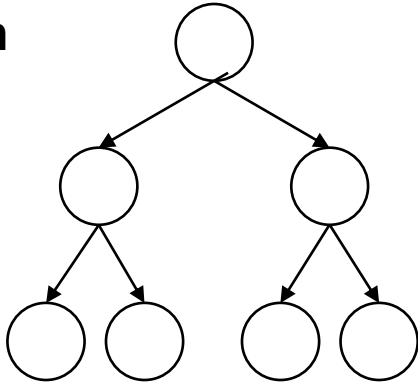
Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin



Future Work: Compiler-in-a-Compiler Instrumentation

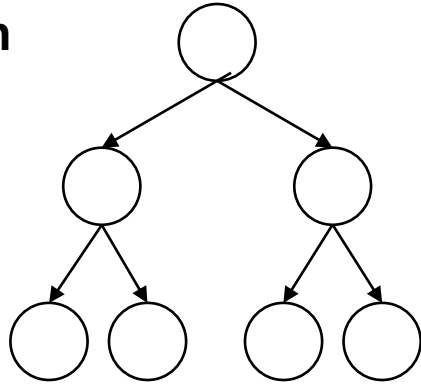
Compiler Plugin



```
myFunc.insertFirst("val span = openSpan()")
```


Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin

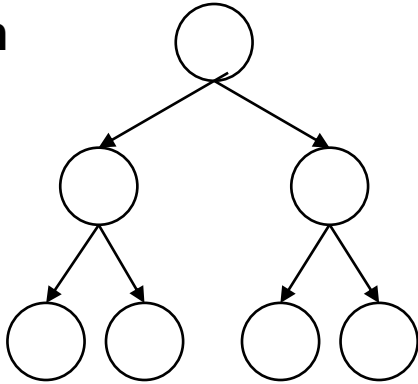


```
myFunc.insertFirst("val span = openSpan()")
```

```
myFunc.insertFirst(myCodeAsString);
```

Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin



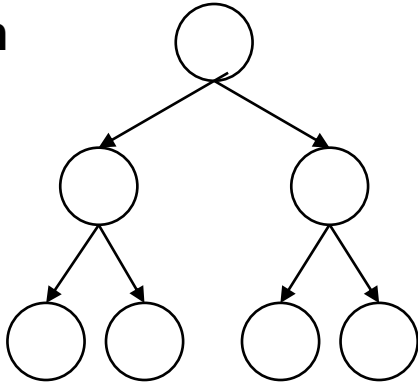
```
myFunc.insertFirst("val span = openSpan()")
```

```
myFunc.insertFirst(myCodeAsString);
```

Embedded Compiler

Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin



```
myFunc.insertFirst("val span = openSpan()")
```

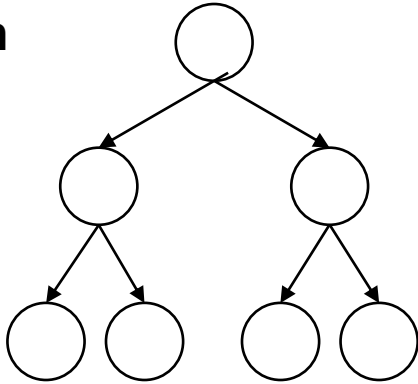
```
myFunc.insertFirst(myCodeAsString);
```

Embedded Compiler

```
fun dummy() {  
    val span = openSpan()  
}
```

Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin

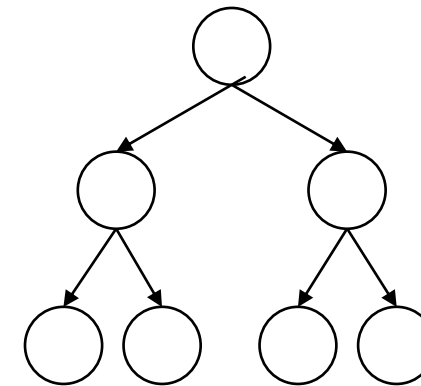


```
myFunc.insertFirst("val span = openSpan()")
```

```
myFunc.insertFirst(myCodeAsString);
```

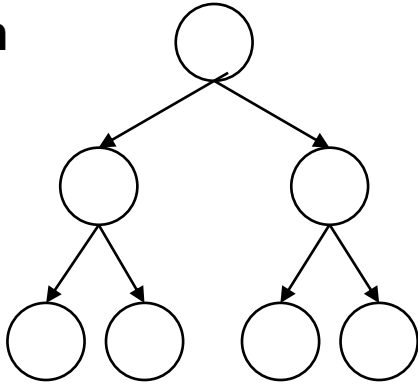
Embedded Compiler

```
fun dummy() {  
    val span = openSpan()  
}
```



Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin

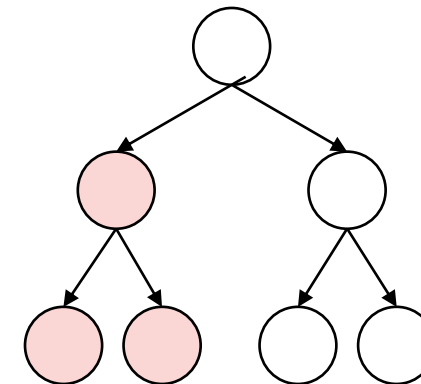


```
myFunc.insertFirst("val span = openSpan()")
```

```
myFunc.insertFirst(myCodeAsString);
```

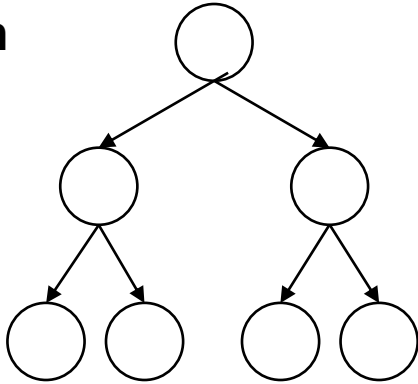
Embedded Compiler

```
fun dummy() {  
    val span = openSpan()  
}
```



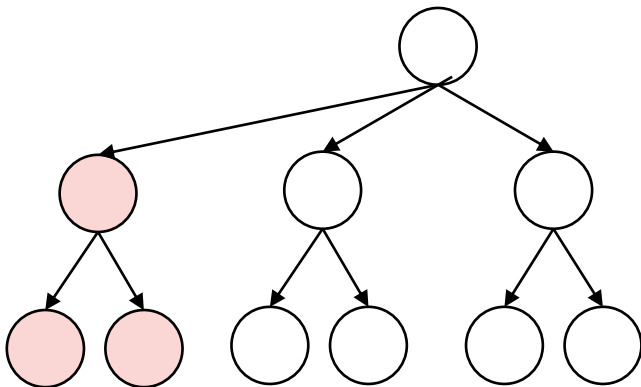
Future Work: Compiler-in-a-Compiler Instrumentation

Compiler Plugin



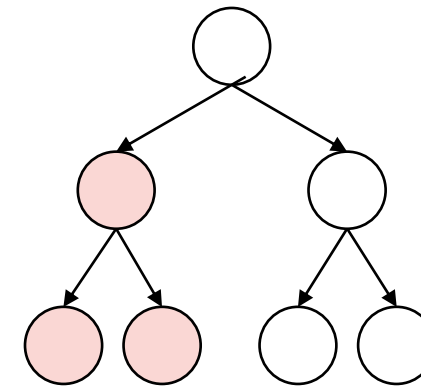
```
myFunc.insertFirst("val span = openSpan()")
```

```
myFunc.insertFirst(myCodeAsString);
```



Embedded Compiler

```
fun dummy() {  
    val span = openSpan()  
}
```



Take Aways

Take Aways

OpenTelemetry

**Collect, transport and
store:**

Traces

Metrics

Logs

Take Aways

OpenTelemetry

Collect, transport and store:

Traces

Metrics

Logs

Collect

Instrumentation

At **Compile-Time** on
Kotlin IR (AST)

Using a **Multiplatform**
OTEL Trace API / SDK
port

Take Aways

OpenTelemetry

Collect, transport and store:

Traces

Metrics

Logs

Collect

Instrumentation

**At Compile-Time on
Kotlin IR (AST)**

**Using a Multiplatform
OTEL Trace API / SDK
port**

Transport

**Using a custom
multiplatform exporter**

HTTP + JSON

**in combination with a
batch processor**

Take Aways

OpenTelemetry

Collect, transport and store:

Traces

Metrics

Logs

Collect

Instrumentation

At **Compile-Time** on
Kotlin IR (AST)

Using a **Multiplatform**
OTEL Trace API / SDK
port

Transport

Using a **custom**
multiplatform exporter

HTTP + JSON

in combination with a
batch processor

Future Work

Target-specific
performance impact
analysis

Multi-Threading

Adaptive Tracing

Compiler-in-Compiler



**JOHANNES KEPLER
UNIVERSITÄT LINZ**