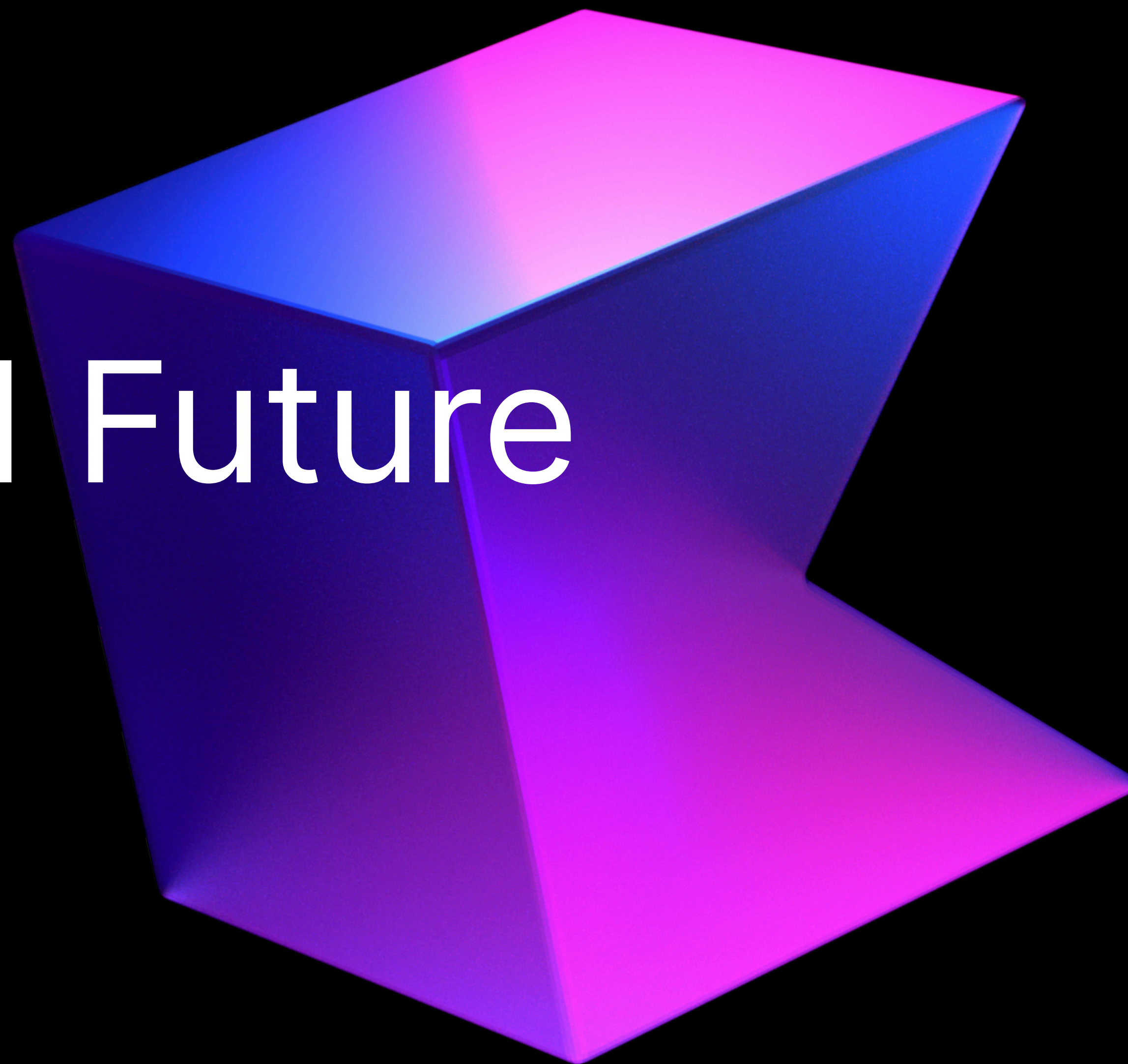


Ktor – Past, Present, and Future

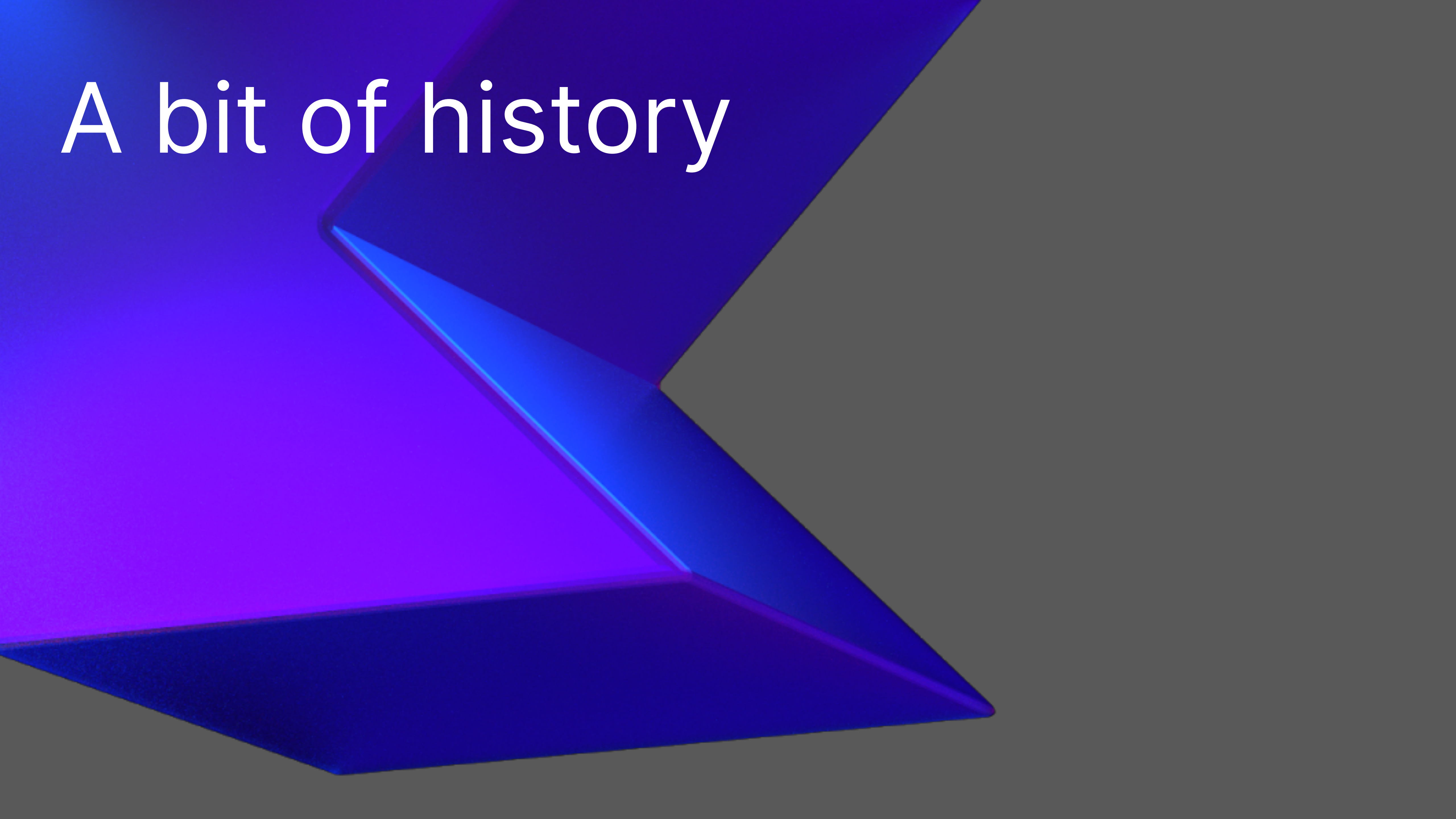
Hadi Hariri



What is Ktor?

- An OSS Framework
- Creating Server and Client applications
- Multiplatform

A bit of history



Frameworks for Kotlin

- Kara - A Ruby on Rails for Kotlin
- Wasabi - An Express.js for Kotlin

Ktor

- Initiated by Ilya Ryzhenkov
- Inspired by Kara and Wasabi
- Apart from being a framework, it served to
 - Explore Kotlin as a DSL tool
 - Base for library usage
 - Coroutines to the max

It caught on...

- Started being used externally
- Started being used internally
 - Space decided to go with Ktor
 - Multiple internal projects started using it
- It started getting external contributions (Pull Requests)

But there was a problem...

<> Code

! Issues 527

🔗 Pull requests 70

Questions tagged [ktor]

Ask Question

Ktor is a Kotlin Web framework developed by JetBrains

[Learn more...](#) [Top users](#) [Synonyms](#)

333 questions

Newest

Active

Bountied


Unanswered

More ▾



⚙️ Filter

#ktor ★

🚩 15 | Web Framework | <https://ktor.io/>



2,632

And there were just two
“part-time” developers

Brings us to today



We have quadrupled our team size!

Changes we've already made

- New site, new blog. And new Twitter account (@JetBrainsKtor).
- Moved to Semantic Versioning
- Commitment to 3 major/minor releases a year and monthly patch releases
- Introducing deprecation cycle

Ktor on the Server

The background features a series of overlapping, semi-transparent geometric planes in shades of blue and purple. These planes are arranged in a way that creates a sense of depth and perspective, with some planes appearing to recede into the distance. The right side of the image is a solid, dark grey color, which contrasts with the vibrant, multi-colored geometric shapes on the left.

```
fun main() {  
    val server = embeddedServer(Netty, 8080) {  
        routing {  
            get("/") {  
                call.respondText("Hello Ktor!", ContentType.Text.Plain)  
            }  
        }  
    }  
    server.start(wait = true)  
}
```

```
fun main() {  
    val server = embeddedServer(Netty, 8080) {  
        routing {  
            get("/") {  
                call.respondText("Hello Ktor!", ContentType.Text.Plain)  
            }  
        }  
    }  
    server.start(wait = true)  
}
```

```
fun main() {  
    val server = embeddedServer(Netty, 8080) {  
        routing {  
            get("/") {  
                call.respondText("Hello Ktor!", ContentType.Text.Plain)  
            }  
        }  
    }  
    server.start(wait = true)  
}
```



```
fun main() {  
    val server = embeddedServer(Netty, 8080) {  
        routing {  
            get("/") {  
                call.respondText("Hello Ktor!", ContentType.Text.Plain)  
            }  
        }  
    }  
    server.start(wait = true)  
}
```

```
fun main() {  
    val server = embeddedServer(Netty, 8080) {  
        routing {  
            get("/") {  
                call.respondText("Hello Ktor!", ContentType.Text.Plain)  
            }  
        }  
    }  
    server.start(wait = true)  
}
```

A Ktor Application



Module

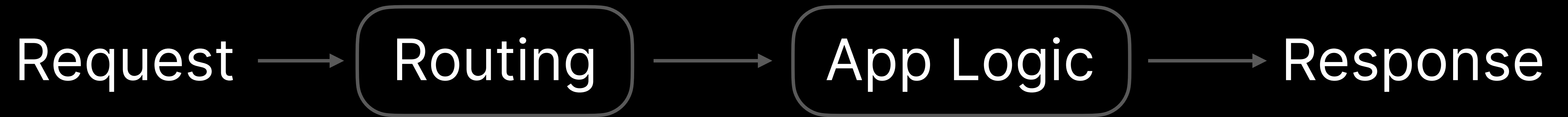
The diagram illustrates the modular architecture of a Ktor application. It features three rounded rectangular boxes, each labeled 'Module', arranged horizontally. The first two boxes are connected by a thin line, and the third box is preceded by an ellipsis (...), indicating that there can be more than two modules in the application.

Module

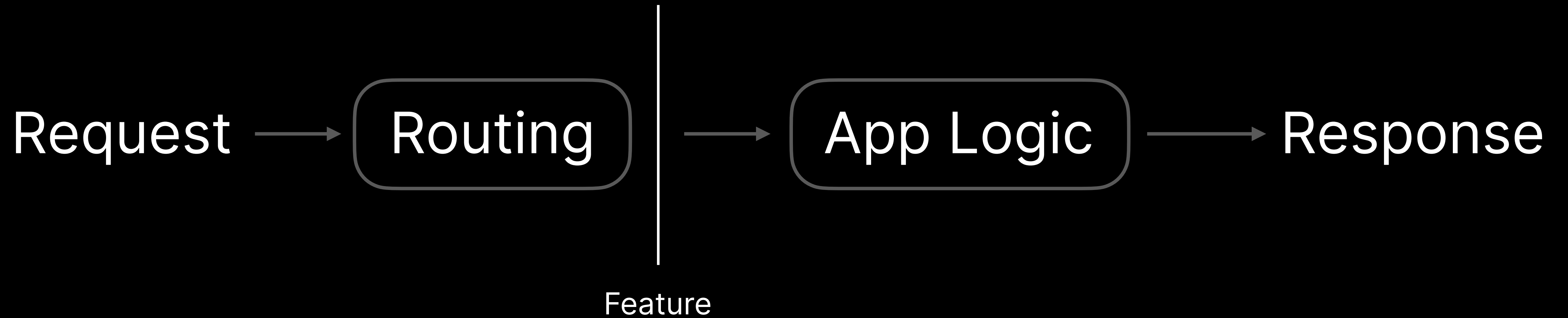
...

Module

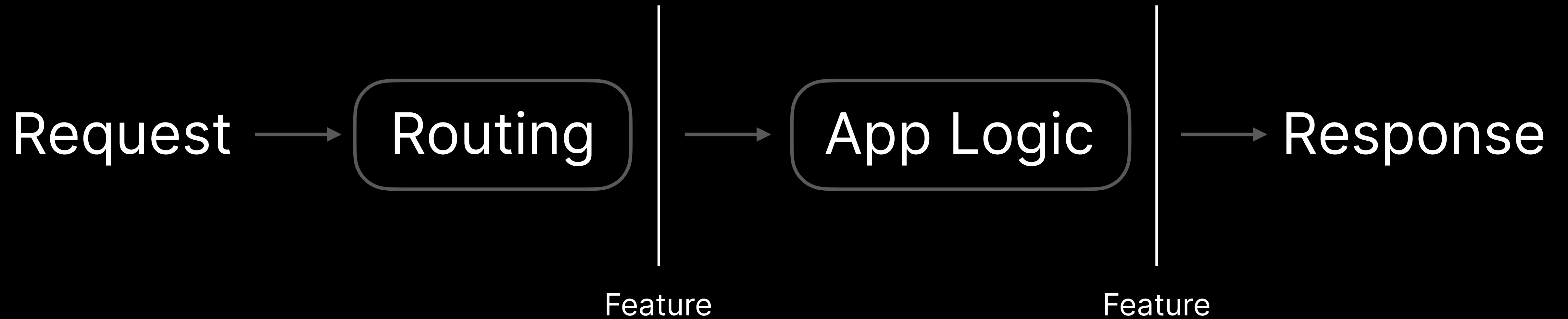
A Module



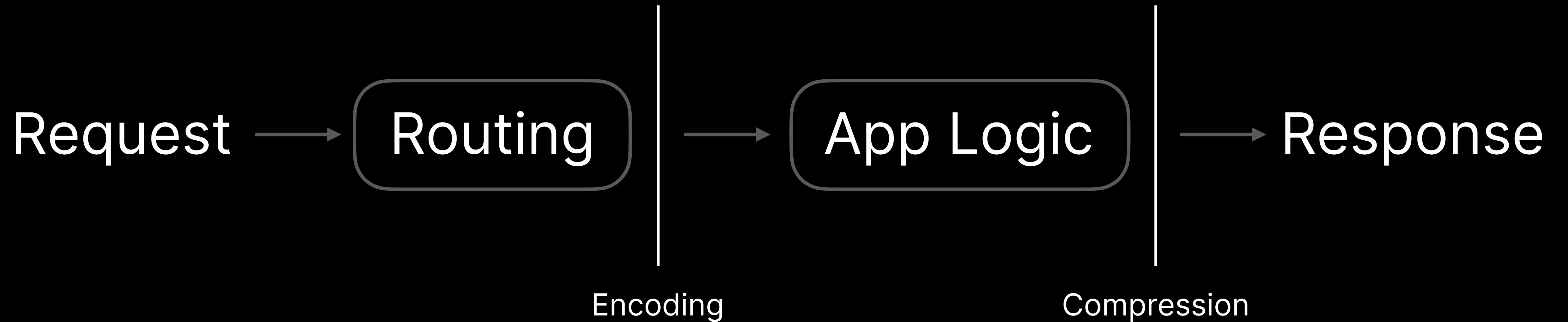
A Module



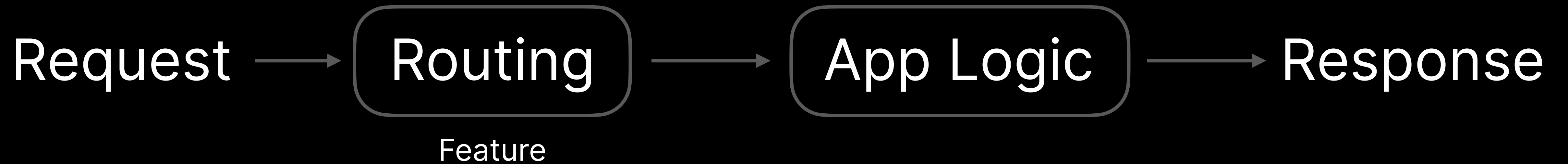
A Module



A Module



A Module



Demos

What lies ahead

What's in store?

- Improving of onboarding experience
- Improving development lifecycle experience (testing, deployment, etc.)
- Improving client/server parity
- Improving extensibility
- Being attentive to performance
- Revamping documentation (with compilable samples)
- Improving tooling

More at ktor.io

Thanks!
Have a nice Kotlin!



@hhariri