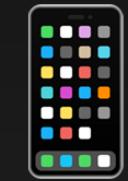


# Secure Your API Keys

## Harnessing Swift on Server

# About me

Michael Freiwald



iOS Developer



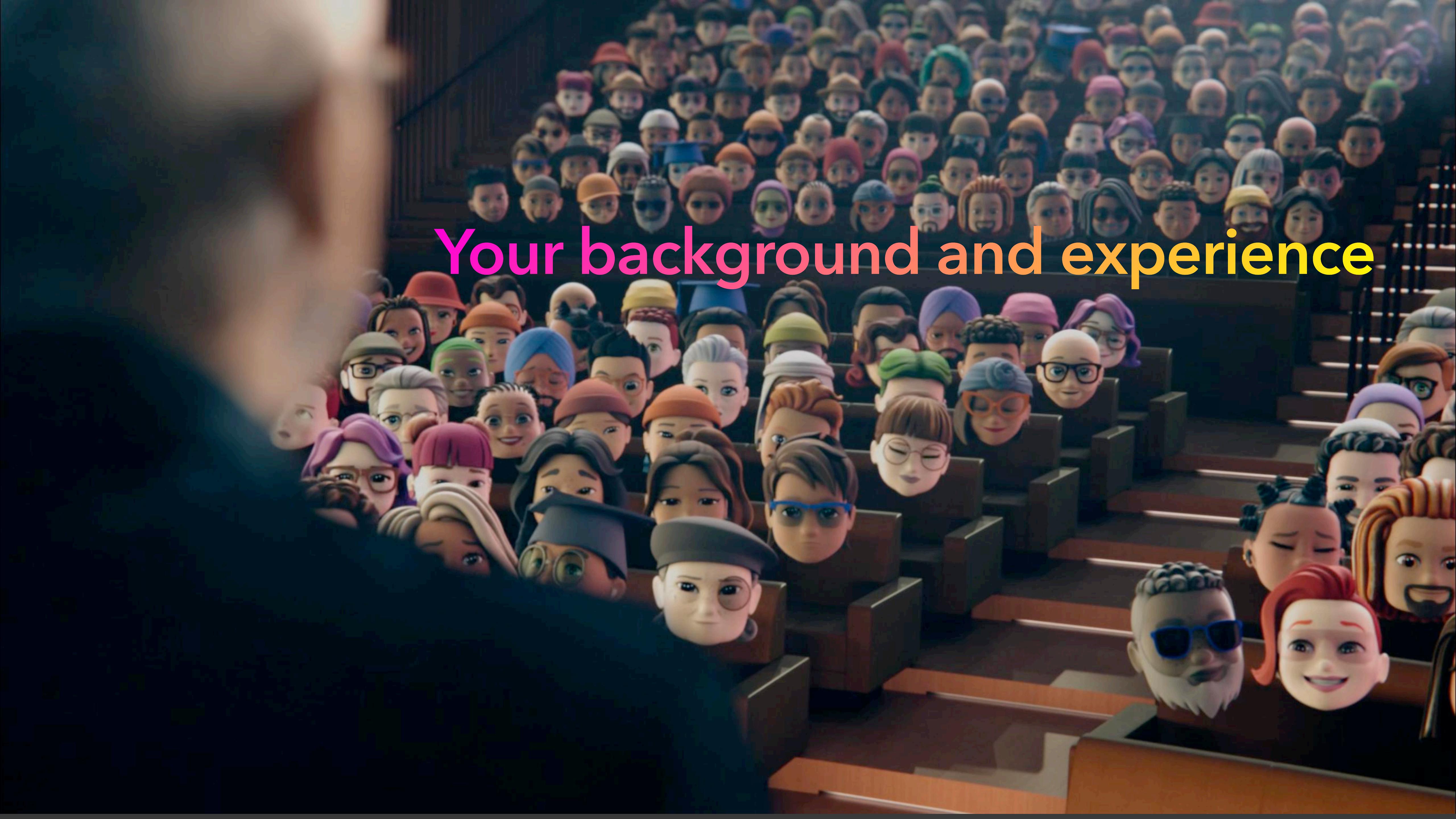
IT-Consultant @ Netlight



Love writing Code with Swift



**Not** a backend developer

A large, diverse crowd of 3D avatars is seated in rows of wooden theater-style seats, viewed from behind. The avatars are highly detailed, showing various ethnicities, ages, and styles of clothing. They are all looking towards the right side of the frame. The lighting is dramatic, with strong highlights on the faces and the wood grain of the seats.

Your background and experience



Atlas



Atlas

09:41

● ● ●

## Articles

**The Power of Custom ShapeStyle for SwiftUI...** >

In SwiftUI, customizing app themes can enhanc...

**How to Build a Proxy Server with Humminbird...** >

Hummingbird is a versatile system for creating...

**Calculating the semantic distance between two...** >

The Natural Language framework can calculate...

URL

Load

A mobile application interface for the "Atlas" app. The screen title is "Articles". It displays a list of three articles with titles, subtitles, and expandable arrows. At the bottom, there is a search bar labeled "URL" and a button labeled "Load". The status bar at the top shows the time as 09:41 and signal strength.



```
var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)
request.httpMethod = "PUT"
request
    .setValue(
        "application/json",
        forHTTPHeaderField: "Content-Type"
    )
request
    .setValue(
        "Bearer jina_a1b2c3d4e5f6g7h8i9j",
        forHTTPHeaderField: "Authorization"
    )
request.httpBody = httpBody
```

```
var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)
request.httpMethod = "PUT"
request
    .setValue(
        "application/json",
        forHTTPHeaderField: "Content-Type"
    )
request
    .setValue(
        "Bearer jina_a1b2c3d4e5f6g7h8i9j"
        forHTTPHeaderField: "Authorization"
    )
request.httpBody = httpBody
```

```
enum Constants {  
    static let apiKey = "jina_a1b2c3d4e5f6g7h8i9j"  
}
```

```
var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)  
request.httpMethod = "PUT"  
request  
    .setValue(  
        "application/json",  
        forHTTPHeaderField: "Content-Type"  
    )  
request  
    .setValue(  
        "Bearer \(\apiKey)",  
        forHTTPHeaderField: "Authorization"  
    )  
request.httpBody = httpBody
```

```
let apiKey = Bundle.main.object(forInfoDictionaryKey: "API_KEY") as? String ?? "  
  
var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)  
request.httpMethod = "PUT"  
request  
    .setValue(  
        "application/json",  
        forHTTPHeaderField: "Content-Type"  
    )  
request  
    .setValue(  
        "Bearer \(apiKey)",  
        forHTTPHeaderField: "Authorization"  
    )  
request.httpBody = httpBody
```

# On-Demand Resources

The screenshot shows the Xcode project settings for a target named "Jarvis". The "Resource Tags" tab is selected. Under the "Initial Install Tags" section, there is one entry: "APIKey (Zero KB)" which points to "APIKey.json ...in Jarvis/Resources". This indicates that the file "APIKey.json" is an on-demand resource. Other sections like "Prefetched Tag Order" and "Download Only On Demand" are also visible.

PROJECT

Jarvis

TARGETS

Jarvis

General Signing & Capabilities Resource Tags Info Build Settings Build Phases Build Rules

+ All Prefetched Filter

Initial Install Tags (not built)

APIKey (Zero KB)

APIKey.json ...in Jarvis/Resources

Prefetched Tag Order (not built)

Download Only On Demand (not built)

# On-Demand Resources

The screenshot shows the Xcode project settings interface, specifically the "Resource Tags" tab. The tab bar includes General, Signing & Capabilities, Resource Tags (which is selected and highlighted in blue), Info, Build Settings, Build Phases, and Build Rules. Below the tab bar, there are two tabs: "All" and "Prefetched", with "Prefetched" being the active tab. A yellow box highlights the "Initial Install Tags (not built)" section, which contains an entry for "APIKey (Zero KB)". This entry is associated with a file icon and the path "APIKey.json ...in Larvis/Resources". There are also "+" and "-" buttons for managing tags at the bottom of this list. Below this, another section titled "Prefetched Tag Order (not built)" is visible.

General    Signing & Capabilities    **Resource Tags**    Info    Build Settings    Build Phases    Build Rules

All    Prefetched

+    Initial Install Tags (not built)

▼ APIKey (Zero KB)  
APIKey.json ...in Larvis/Resources

+ -

▼ Prefetched Tag Order (not built)

```
let apiKey = Bundle.main.object(forInfoDictionaryKey: "API_KEY") as? String ?? ""

var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)
request.httpMethod = "PUT"
request
    .setValue(
        "application/json",
        forHTTPHeaderField: "Content-Type"
    )
request
    .setValue(
        "Bearer \(apiKey)",
        forHTTPHeaderField: "Authorization"
    )
request.httpBody = httpBody
```

```
func loadApiKey() async throws -> String {
    let request = NSBundleResourceRequest(tags: ["APIKey"])
    try await request.beginAccessingResources()

    let url = Bundle.main.url(forResource: "APIKey", withExtension: "json")!
    let data = try Data(contentsOf: url)

    let decoder = JSONDecoder()
    let apiKeyData = try decoder.decode(ApiKeyData.self, from: data)

    request.endAccessingResources()
    return apiKeyData.apiKey
}
```

```
let apiKey = try await loadApiKey()

var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)
request.httpMethod = "PUT"
request
    .setValue(
        "application/json",
        forHTTPHeaderField: "Content-Type"
)
request
```

```
func loadApiKey() async throws -> String {
    let request = NSBundleResourceRequest(tags: ["APIKey"])
    try await request.beginAccessingResources()

    let url = Bundle.main.url(forResource: "APIKey", withExtension: "json")!
    let data = try Data(contentsOf: url)

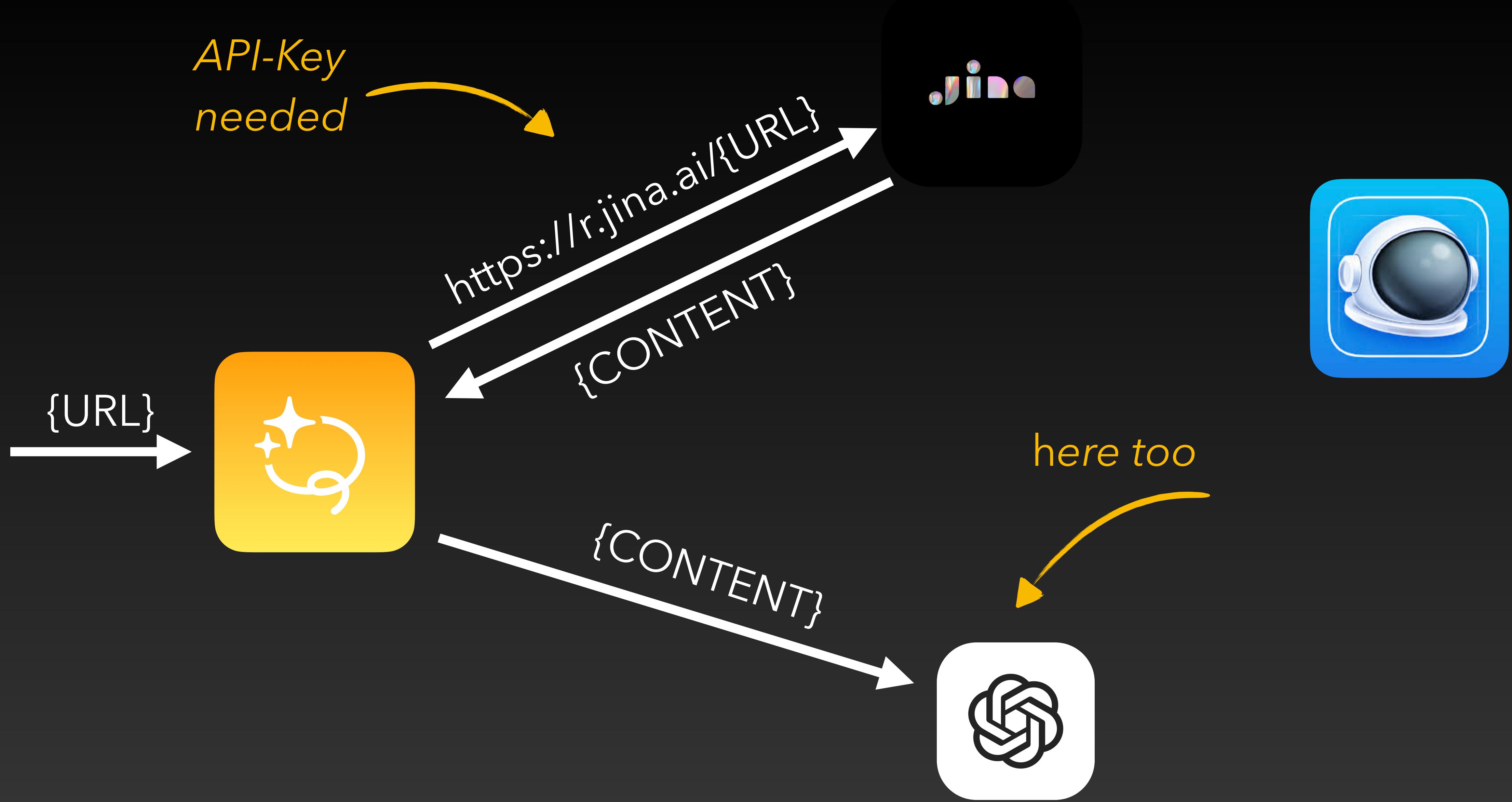
    let decoder = JSONDecoder()
    let apiKeyData = try decoder.decode(ApiKeyData.self, from: data)

    // TODO: Store in Keychain

    request.endAccessingResources()
    return apiKeyData.apiKey
}
```

```
let apiKey = try await loadApiKey()

var request = URLRequest(url: URL(string: "https://r.jina.ai/")!)
request.httpMethod = "PUT"
request
    .setValue(
        "application/json",
        forHTTPHeaderField: "Content-Type"
```



# Certificate Pinning

1. Client has public key of Server
2. Handshake between Client & Server (Server sends Certificate to Client)
3. Client check public keys
4. Allows or disallows connection to server

# Backend Development

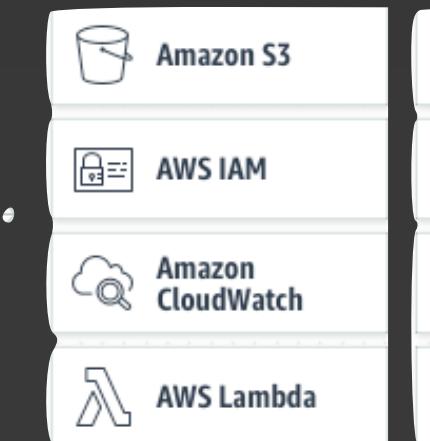
- No API keys on client-side
- Own request & response models
- Fast backend deployment (compared to app store process)
- Certificate owner
- App Store authorization
- Additional layers (rate limit, input validation etc.)



**Swift on Server**



Vapor



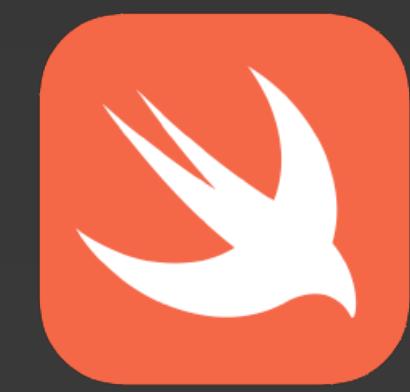
Swift Server Workgroup  
(SSWG)



ServerSide.swift



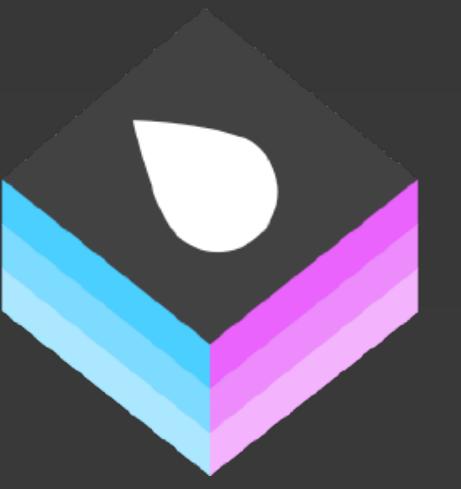
Swift on Server



Your next  
project



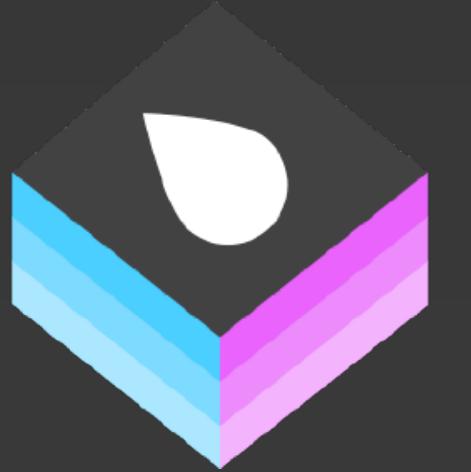
Hummingbird



Vapor



Hummingbird



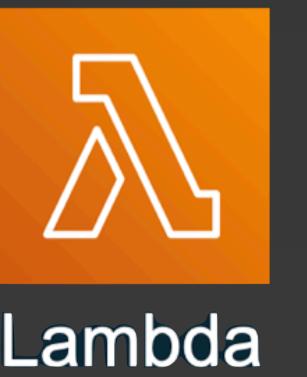
Vapor



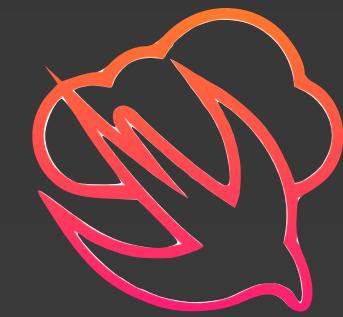
Hummingbird



Swift



Lambda



Swift Cloud  
x  
SwiftWasm





Hummingbird



# Hummingbird 2

Swift 6

Modular Architecture

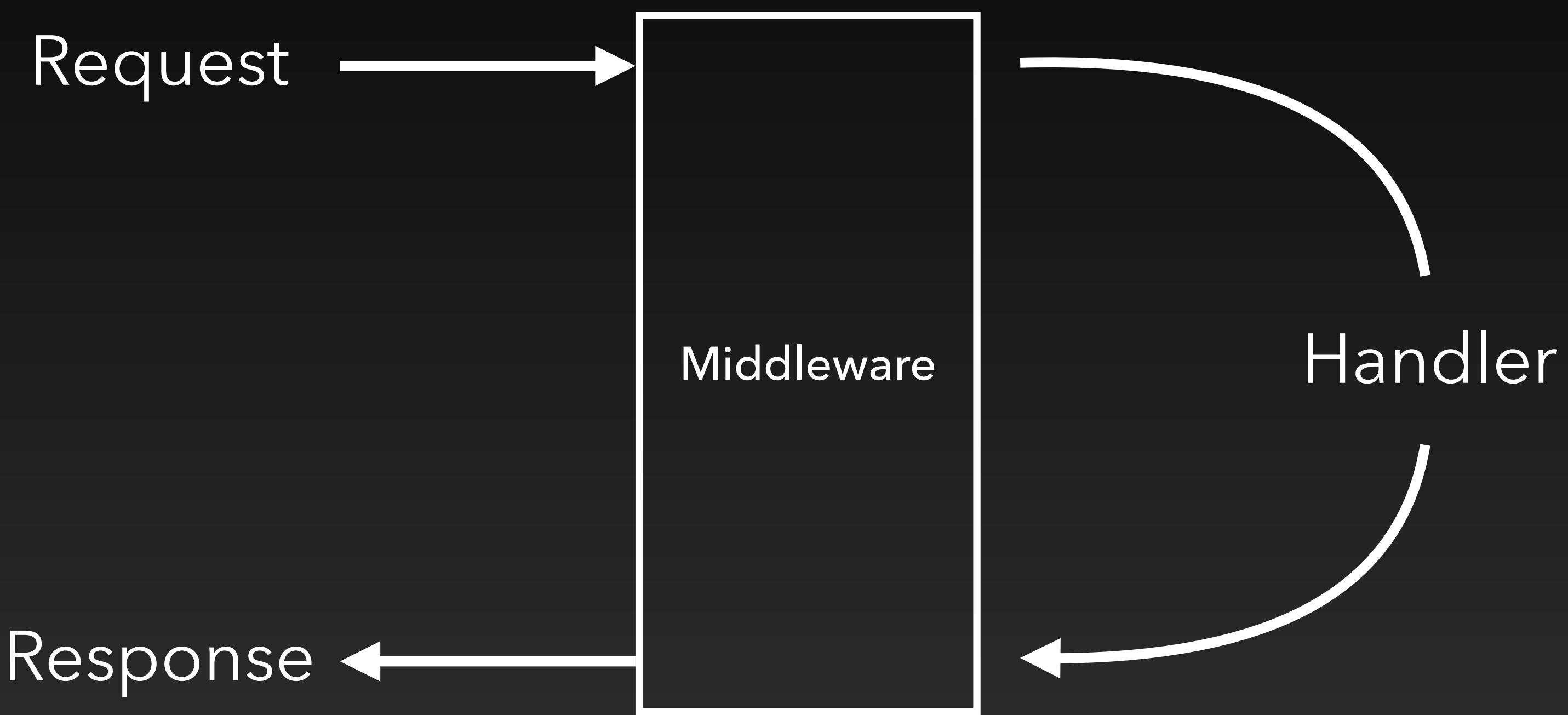
Redesigned APIs

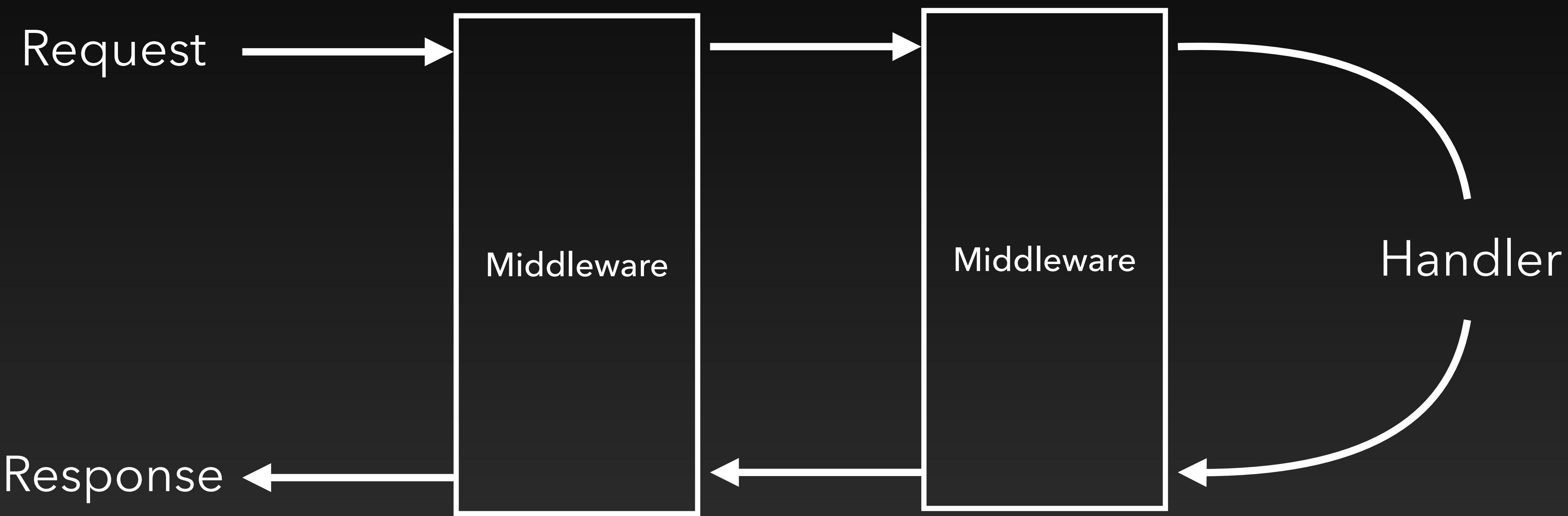
Request

Handler

Response







# Creating a Hummingbird Project

```
~/workspace/freiwald/Atlas main > mkdir AtlasBackend
~/workspace/freiwald/Atlas main > cd AtlasBackend
~/workspace/freiwald/Atlas/AtlasBackend main > swift package init --type executable
Creating executable package: AtlasBackend
Creating Package.swift
Creating .gitignore
Creating Sources/
Creating Sources/main.swift
~/workspace/freiwald/Atlas/AtlasBackend main > swift package add-dependency https://
github.com/hummingbird-project/hummingbird --from 2.3.0
Updating package manifest at Package.swift... done.
~/workspace/freiwald/Atlas/AtlasBackend main > swift package add-target-dependency
Hummingbird AtlasBackend
Updating package manifest at Package.swift... done.
~/workspace/freiwald/Atlas/AtlasBackend main > open Package.swift
```

AtlasBackend

main

AtlasBackend > My Mac

AtlasBackend: Ready | Today at 11:36

+ □

Package App

AtlasBackend > Package > No Selection

```
1 // swift-tools-version: 6.0
2 // The swift-tools-version declares the minimum version of Swift required to build this package.
3
4 import PackageDescription
5
6 let package = Package(
7   name: "AtlasBackend",
8   platforms: [.iOS(.v18), .macOS(.v15)],
9   products: [
10     .executable(name: "AtlasBackend", targets: ["AtlasBackend"])
11   ],
12   dependencies: [
13     .package(url: "https://github.com/hummingbird-project/hummingbird", from: "2.3.0"),
14   ],
15   targets: [
16     .executableTarget(
17       name: "AtlasBackend",
18       dependencies: [
19         .product(name: "Hummingbird", package: "Hummingbird")
20       ],
21     )
22   ]
23 )
```

+

Filter

Line: 5 Col: 1

The screenshot shows the Xcode interface with the project `AtlasBackend` selected. In the Project Navigator on the left, the `AtlasBackend` package is expanded, showing its `Sources` folder and an `AtlasBackend` target. The `Package.resolved` file is highlighted with a yellow box. In the main editor area, the `Package.swift` file is open, displaying the Swift Package Configuration code. The code defines a package named `AtlasBackend` for iOS and macOS, containing an executable target named `AtlasBackend` and dependencies on various Swift packages.

```
// swift-tools-version: 6.0
// The swift-tools-version declares the minimum
// version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "AtlasBackend",
    platforms: [.iOS(.v18), .macOS(.v15)],
    products: [
        .executable(name: "AtlasBackend", target: "AtlasBackend")
    ],
    dependencies: [
        .package(url: "https://github.com/hummingbird-project/hummingbird.git", from: "2.3.0")
    ],
    targets: [
        .executableTarget(
            name: "AtlasBackend",
            dependencies: []
        )
    ]
)
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")

        let router = Router(context: BasicRequestContext.self)

        router.get("/") { request, context in
            return "👋 Hello Swift Meetup Munich! 🚀"
        }

        let app = Application(
            router: router,
            configuration: .init(
                address: .hostname(hostname, port: port),
                serverName: "AtlasBackend"
            ),
            eventLoopGroupProvider: .shared(eventLoopGroup),
            logger: logger
        )

        try await app.runService()
    }
}
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")

        let router = Router(context: BasicRequestContext.self)

        router.get("/") { request, context in
            return "👋 Hello Swift Meetup Munich! 🚀"
        }

        let app = Application(
            router: router,
            configuration: .init(
                address: .hostname(hostname, port: port),
                serverName: "AtlasBackend"
            ),
            eventLoopGroupProvider: .shared(eventLoopGroup),
            logger: logger
        )

        try await app.runService()
    }
}
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")
    }

    let router = Router(context: BasicRequestContext.self)

    router.get("/") { request, context in
        return "👋 Hello Swift Meetup Munich! 🚀"
    }

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        ),
        eventLoopGroupProvider: .shared(eventLoopGroup),
        logger: logger
    )

    try await app.runService()
}
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")

        let router = Router(context: BasicRequestContext.self)

        router.get("/") { request, context in
            return "👋 Hello Swift Meetup Munich! 🚀"
        }

        let app = Application(
            router: router,
            configuration: .init(
                address: .hostname(hostname, port: port),
                serverName: "AtlasBackend"
            ),
            eventLoopGroupProvider: .shared(eventLoopGroup),
            logger: logger
        )

        try await app.runService()
    }
}
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")

        let router = Router(context: BasicRequestContext.self)

        router.get("/") { request, context in
            return "👋 Hello Swift Meetup Munich! 🚀"
        }
    }

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        ),
        eventLoopGroupProvider: .shared(eventLoopGroup),
        logger: logger
    )

    try await app.runService()
}
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")

        let router = Router(context: BasicRequestContext.self)

        router.get("/") { request, context in
            return "👋 Hello Swift Meetup Munich! 🚀"
        }
    }

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        ),
        eventLoopGroupProvider: .shared(eventLoopGroup),
        logger: logger
    )

    try await app.runService()
}
```

```
@main
struct App: AsyncParsableCommand {
    @Option(name: .shortAndLong)
    var hostname: String = "127.0.0.1"

    @Option(name: .shortAndLong)
    var port: Int = 8080

    func run() async throws {
        let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
        let logger = Logger(label: "AtlasBackend")

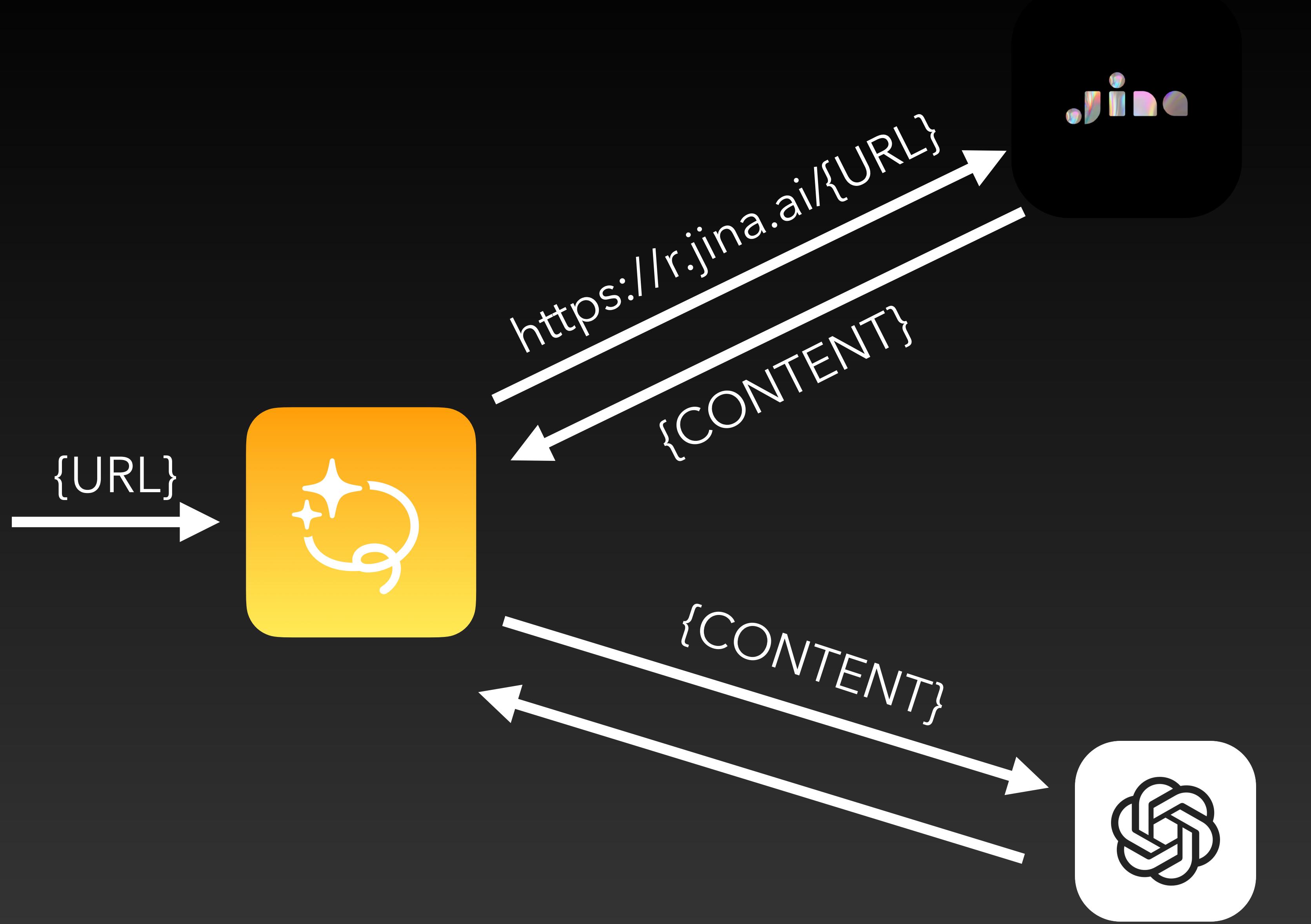
        let router = Router(context: BasicRequestContext.self)

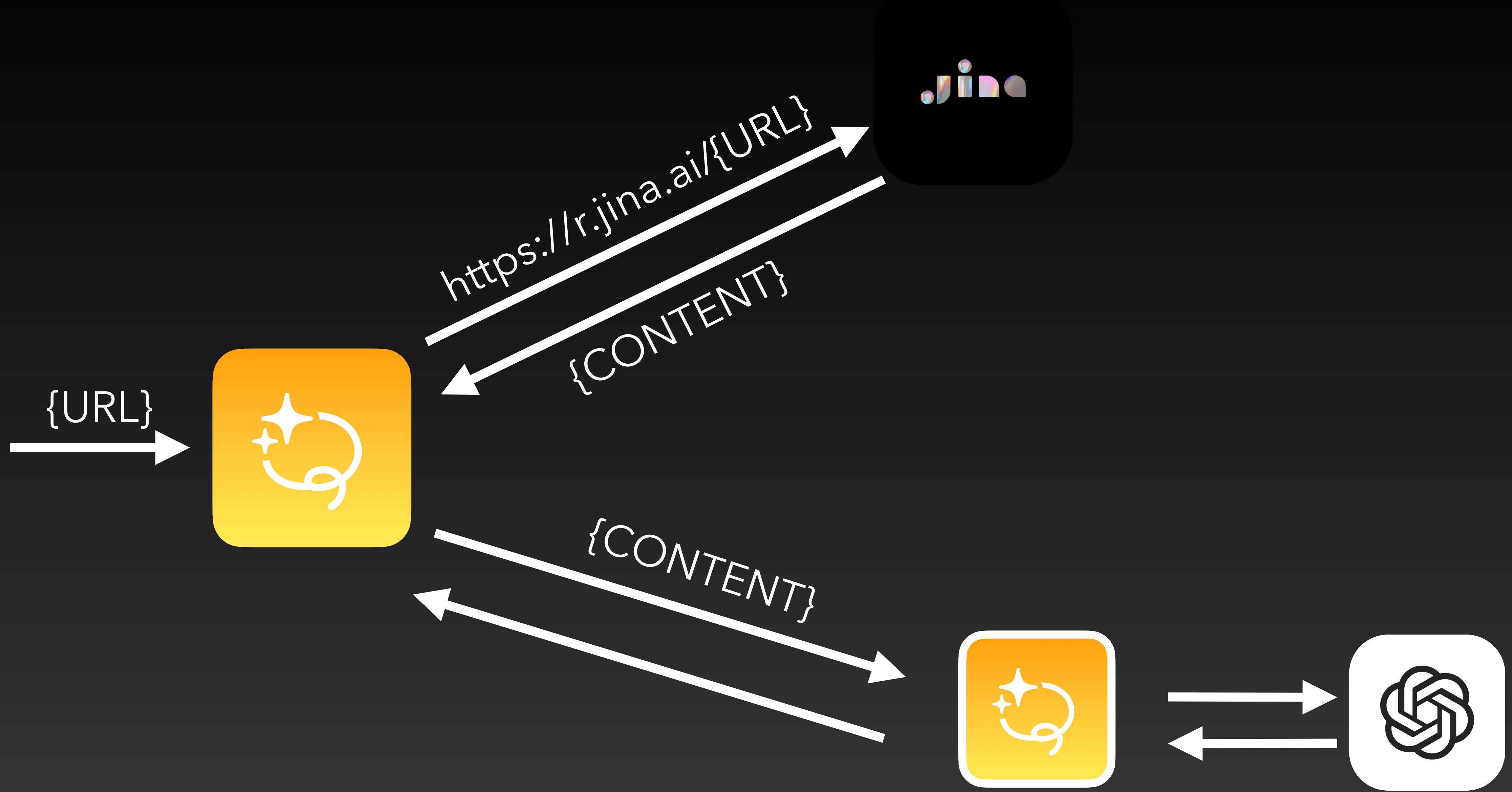
        router.get("/") { request, context in
            return "👋 Hello Swift Meetup Munich! 🚀"
        }

        let app = Application(
            router: router,
            configuration: .init(
                address: .hostname(hostname, port: port),
                serverName: "AtlasBackend"
            ),
            eventLoopGroupProvider: .shared(eventLoopGroup),
            logger: logger
        )

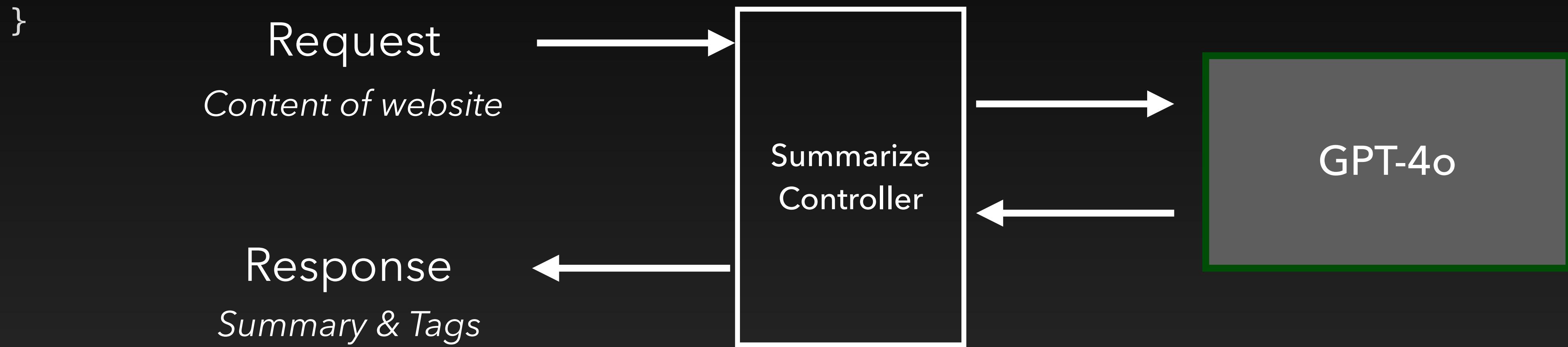
        try await app.runService()
    }
}
```

# Coding





```
public struct SummaryRequest: Sendable, Codable {  
    public let content: String  
  
    public init(content: String) {  
        self.content = content  
    }  
}
```



```
public struct SummaryResponse: Sendable, Codable {  
    public let summary: String  
    public let tags: [String]  
  
    package init(summary: String, tags: [String]) {  
        self.summary = summary  
        self.tags = tags  
    }  
}
```

```
struct SummaryController<Context: RequestContext> {  
}
```

```
struct SummaryController<Context: RequestContext> {
    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }
}
```

```
struct SummaryController<Context: RequestContext> {
    func addRoutes(to group: RouterGroup<Context>) {
        group post(use: summarize)
    }
}
```

```
struct SummaryController<Context: RequestContext> {
    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }

    @Sendable private func summarize(
        _ request: Request,
        context: Context
    ) async throws -> SummaryResponse {
        let input = try await request.decode(
            as: SummaryRequest.self,
            context: context
        )

        context.logger.info("Receive content: \(input.content.prefix(50))")

        let response = SummaryResponse(summary: "TODO: Summary", tags: [])
        return response
    }
}
```

```
struct SummaryController<Context: RequestContext> {
    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }

    @Sendable private func summarize(
        _ request: Request,
        context: Context
    ) async throws -> SummaryResponse {
        let input = try await request.decode(
            as: SummaryRequest.self,
            context: context
        )

        context.logger.info("Receive content: \(input.content.prefix(50))")

        let response = SummaryResponse(summary: "TODO: Summary", tags: [])
        return response
    }
}
```

```
struct SummaryController<Context: RequestContext> {
    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }

    @Sendable private func summarize(
        _ request: Request,
        context: Context
    ) async throws -> SummaryResponse {
        let input = try await request.decode(
            as: SummaryRequest.self,
            context: context
        )

        context.logger.info("Receive content: \(input.content.prefix(50))")

        let response = SummaryResponse(summary: "TODO: Summary", tags: [])
    }
}
```

```
protocol SummarizserService: Sendable {  
    func summarizse(_ request: SummaryRequest) async throws -> SummaryResponse  
}
```

```
protocol SummarizerService: Sendable {
    func summarize(_ request: SummaryRequest) async throws -> SummaryResponse
}

struct OpenAISummarizer: SummarizerService {
    private let service: any OpenAIService
    private let schema: JSONSchemaResponseFormat = { ... }()

    init(resourceName: String, apiKey: String) {
        service = OpenAIServiceFactory.service(
            azureConfiguration: AzureOpenAIConfiguration(
                resourceName: resourceName,
                openAIAPICKey: .apiKey(apiKey),
                apiVersion: "2024-10-01-preview"
            )
        )
    }

    let systemMessage = { ... }()

    func summarize(_ request: SummaryRequest) async throws -> SummaryResponse { ... }
}
```

```
}
```

```
let systemMessage = { ... }()
```

```
func summarizse(_ request: SummaryRequest) async throws -> SummaryResponse {
```

```
}
```

```
}
```

```
        }

let systemMessage = { ... }()

func summarizse(_ request: SummaryRequest) async throws -> SummaryResponse {
    let result = try await service.startChat(
        parameters: .init(
            messages: [
                .init(role: .system, content: .text(systemMessage)),
                .init(role: .user, content: .text(request.content))
            ],
            model: model,
            responseFormat: .jsonSchema(schema)
        )
    )

guard
    let content = result.choices.first?.message.content,
    let data = content.data(using: .utf8)
else { throw SummarizerError.noChatOutput }

let decoded = try JSONDecoder().decode(SummaryResponse.self, from: data)
return decoded
}
```

```
}

let systemMessage = { ... }()

func summarizse(_ request: SummaryRequest) async throws -> SummaryResponse {
    let result = try await service.startChat(
        parameters: .init(
            messages: [
                .init(role: .system, content: .text(systemMessage)),
                .init(role: .user, content: .text(request.content))
            ],
            model: model,
            responseFormat: .jsonSchema(schema)
        )
    )

    guard
        let content = result.choices.first?.message.content,
        let data = content.data(using: .utf8)
    else { throw SummarizerError.noChatOutput }

    let decoded = try JSONDecoder().decode(SummaryResponse.self, from: data)
    return decoded
}
```

```
        }

let systemMessage = { ... }()

func summarizse(_ request: SummaryRequest) async throws -> SummaryResponse {
    let result = try await service.startChat(
        parameters: .init(
            messages: [
                .init(role: .system, content: .text(systemMessage)),
                .init(role: .user, content: .text(request.content))
            ],
            model: model,
            responseFormat: .jsonSchema(schema)
        )
    )

guard
    let content = result.choices.first?.message.content,
    let data = content.data(using: .utf8)
else { throw SummarizerError.noChatOutput }

let decoded = try JSONDecoder().decode(SummaryResponse.self, from: data)
return decoded
}
```

```
    }

let systemMessage = { ... }()

func summarizse(_ request: SummaryRequest) async throws -> SummaryResponse {
    let result = try await service.startChat(
        parameters: .init(
            messages: [
                .init(role: .system, content: .text(systemMessage)),
                .init(role: .user, content: .text(request.content))
            ],
            model: model,
            responseFormat: .jsonSchema(schema)
        )
    )

    guard
        let content = result.choices.first?.message.content,
        let data = content.data(using: .utf8)
    else { throw SummarizerError.noChatOutput }

    let decoded = try JSONDecoder().decode(SummaryResponse.self, from: data)
    return decoded
}
```

```
struct SummaryController<Context: RequestContext> {
    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }

    @Sendable private func summarize(
        _ request: Request,
        context: Context
    ) async throws -> SummaryResponse {
        let input = try await request.decode(
            as: SummaryRequest.self,
            context: context
        )

        context.logger.info("Receive content: \(input.content.prefix(50))")

        let response = SummaryResponse(summary: "TODO: Summary", tags: [])
        return response
    }
}
```

```
struct SummaryController<Context: RequestContext> {
    let service: any SummarizerService

    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }

    @Sendable private func summarize(
        _ request: Request,
        _ context: Context
    ) async throws -> SummaryResponse {
        let input = try await request.decode(
            as: SummaryRequest.self,
            context: context
        )

        context.logger.info("Receive content: \(input.content.prefix(50))")

        let response = SummaryResponse(summary: "TODO: Summary", tags: [])
        return response
    }
}
```

```
struct SummaryController<Context: RequestContext> {
    let service: any SummarizerService

    func addRoutes(to group: RouterGroup<Context>) {
        group.post(use: summarize)
    }

    @Sendable private func summarize(
        _ request: Request,
        _ context: Context
    ) async throws -> SummaryResponse {
        let input = try await request.decode(
            as: SummaryRequest.self,
            context: context
        )

        context.logger.info("Receive content: \(input.content.prefix(50))")

        let response = try await service.summarize(input)

        return response
    }
}
```

```
func run() async throws {
    let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
    let logger = Logger(label: "AtlasBackend")

    let router = Router(context: BasicRequestContext.self)

    router.get("/") { request, context in
        let name = request.uri.queryParameters.get("name") ?? "Swift Meetup Munich"
        return "👋 Hello \(name)! 🚀 \n\nsummarizse Endpoint is not available anymore 😊"
    }

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        ),
        eventLoopGroupProvider: .shared(eventLoopGroup),
        logger: logger
    )

    try await app.runService()
}
```

```
func run() async throws {
    let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
    let logger = Logger(label: "AtlasBackend")

    let router = Router(context: BasicRequestContext.self)

    router.get("/") { request, context in
        let name = request.uri.queryParameters.get("name") ?? "Swift Meetup Munich"
        return "👋 Hello \(name)! 🚀 \n\nsummarizse Endpoint is not available anymore 😊"
    }

    SummaryController(service: any SummarizserService)
        .addRoutes(to: router.group("summarizse"))

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        ),
        eventLoopGroupProvider: .shared(eventLoopGroup),
        logger: logger
    )

    try await app.runService()
```

```
func run() async throws {
    let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
    let logger = Logger(label: "AtlasBackend")
```

```
let router = Router(context: BasicRequestContext.self)
```

```
router.get("/") { request, context in
    let name = request.uri.queryParameters.get("name") ?? "Swift Meetup Munich"
    return "👋 Hello \(name)! 🚀 \n\nsummarize Endpoint is not available anymore 😞"
}
```

```
SummaryController(service: any SummarizerService)
    .addRoutes(to: router.group("summarize"))
```

```
let app = Application(
    router: router,
    configuration: .init(
        address: .hostname(hostname, port: port),
        serverName: "AtlasBackend"
    ),
    eventLoopGroupProvider: .shared(eventLoopGroup),
    logger: logger
)
```



e.g. <https://atlas-backend.dev/summarize>

```
+ try await app.runService()
```

```
func run() async throws {
    let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
    let logger = Logger(label: "AtlasBackend")

    let router = Router(context: BasicRequestContext.self)

    router.get("/") { request, context in
        let name = request.uri.queryParameters.get("name") ?? "Swift Meetup Munich"
        return "👋 Hello \(name)! 🚀 \n\nsummarizse Endpoint is not available anymore 😊"
    }

    let summarizserService: SummarizserService = OpenAISummarizser(
        resourceName: "",
        apiKey: ""
    )

    SummaryController(service: any SummarizserService)
        .addRoutes(to: router.group("summarizse"))

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        )
    )
```

```
func run() async throws {
    let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
    let logger = Logger(label: "AtlasBackend")

    let router = Router(context: BasicRequestContext.self)

    router.get("/") { request, context in
        let name = request.uri.queryParameters.get("name") ?? "Swift Meetup Munich"
        return "👋 Hello \(name)! 🚀 \n\nsummarizse Endpoint is not available anymore 😊"
    }

    let summarizserService: SummarizserService = OpenAISummarizser(
        resourceName: "",
        apiKey: ""
    )

    SummaryController(service: summarizserService)
        .addRoutes(to: router.group("summarizse"))

    let app = Application(
        router: router,
        configuration: .init(
            address: .hostname(hostname, port: port),
            serverName: "AtlasBackend"
        )
    )
```

```
func run() async throws {
    let eventLoopGroup = MultiThreadedEventLoopGroup.singleton
    let logger = Logger(label: "AtlasBackend")

    let router = Router(context: BasicRequestContext.self)

    router.get("/") { request, context in
        let name = request.uri.queryParameters.get("name") ?? "Swift Meetup Munich"
        return "👋 Hello \(name)! 🚀 \n\nsummarize Endpoint is not available anymore 😊"
    }

    let (apiKey, resourceName) = try await loadEnv(logger)

    let summarizerService: SummarizerService = OpenAISummarizer(
        resourceName: resourceName,
        apiKey: apiKey
    )

    SummaryController(service: summarizerService)
        .addRoutes(to: router.group("summarize"))

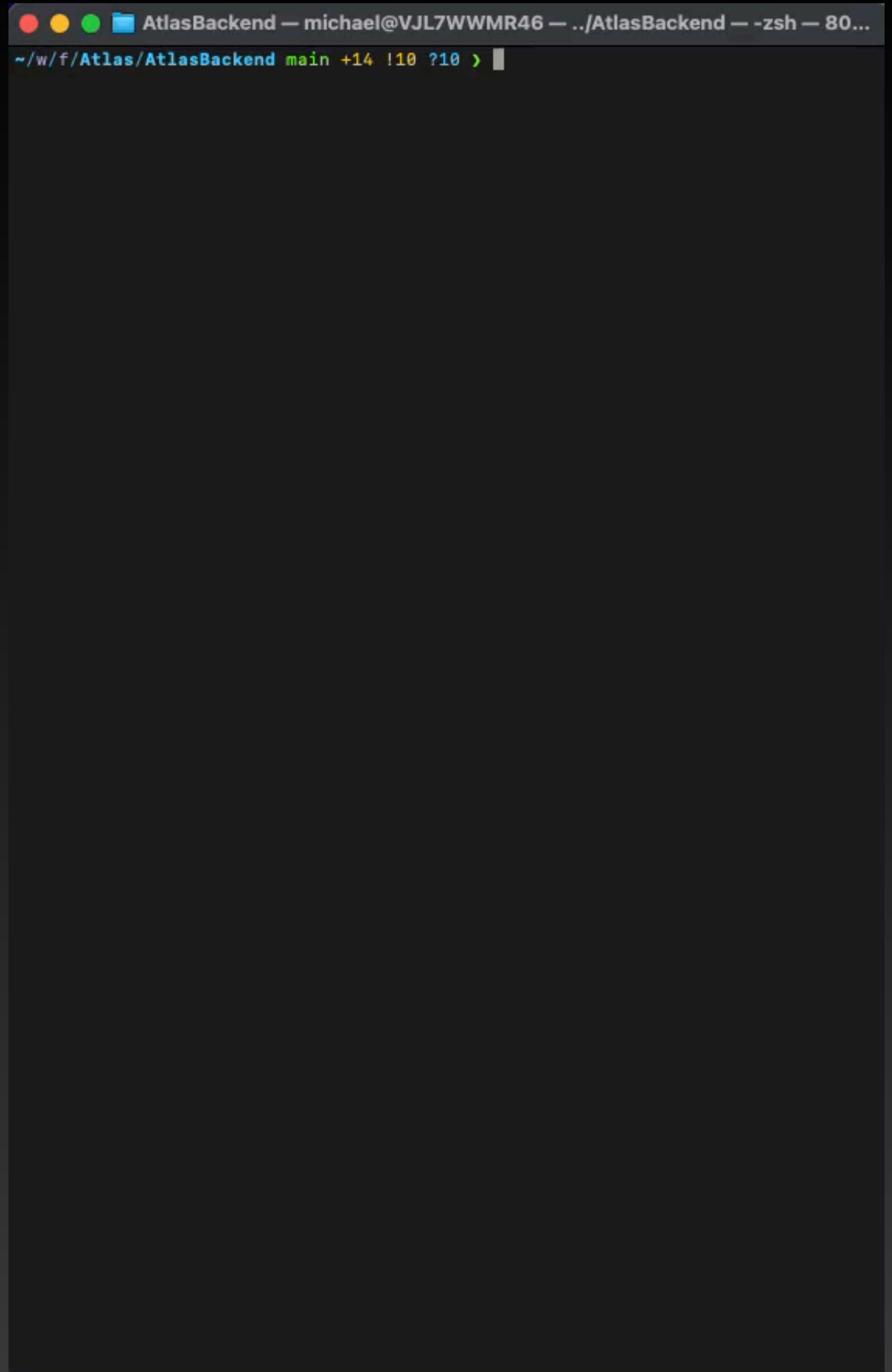
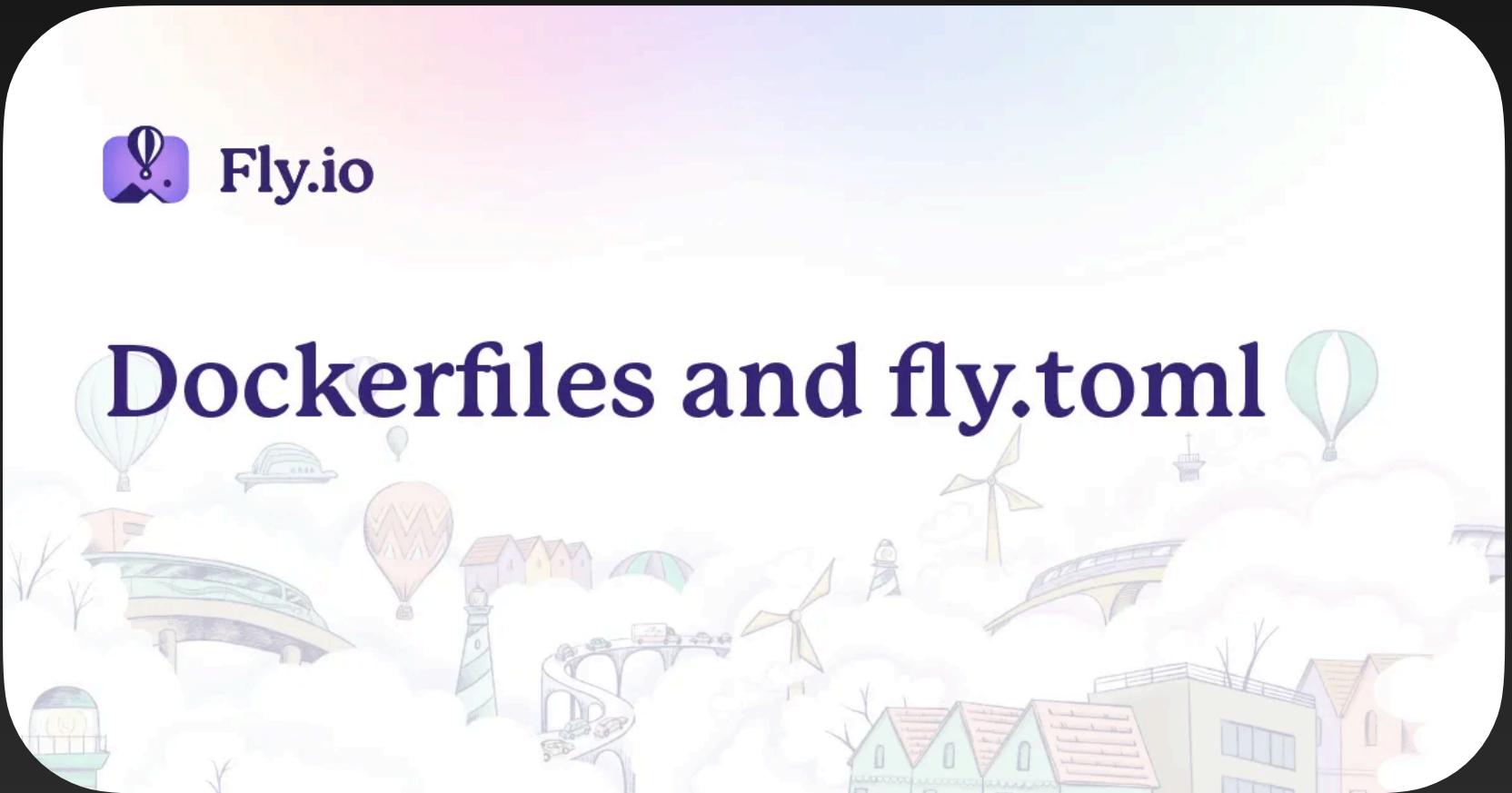
    let app = Application(
        router: router,
        configuration: .init(
            address: hostname(hostname: "localhost", port: port),
            port: port
        )
    )
}
```

# Demo

And now?



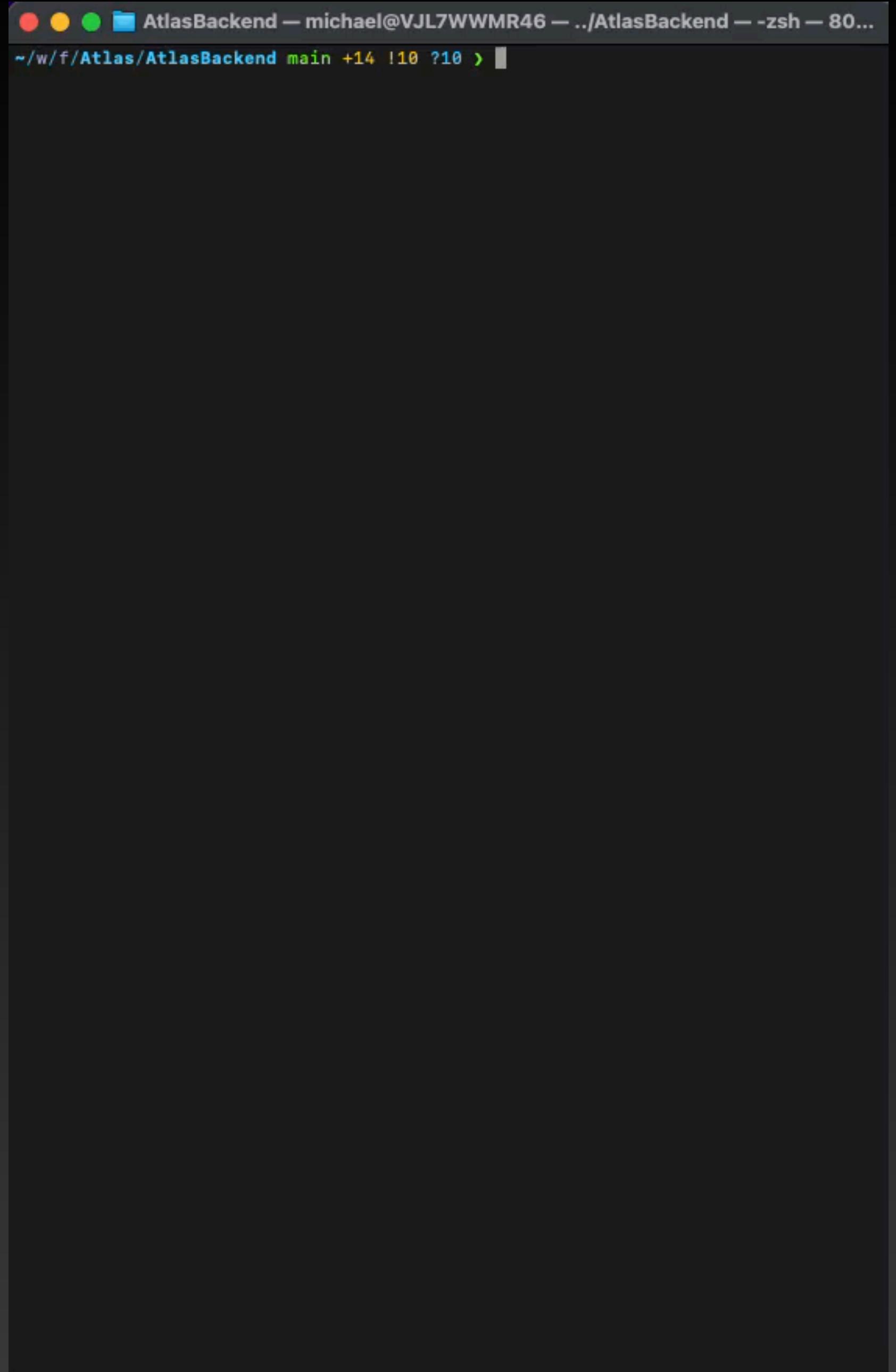
Fly.io



A screenshot of a terminal window titled "AtlasBackend". The window shows a file tree with "AtlasBackend" at the root, and command history with "main +14 !10 ?10 >". The terminal has a dark theme with colored icons for files and folders.

# Dockerfile

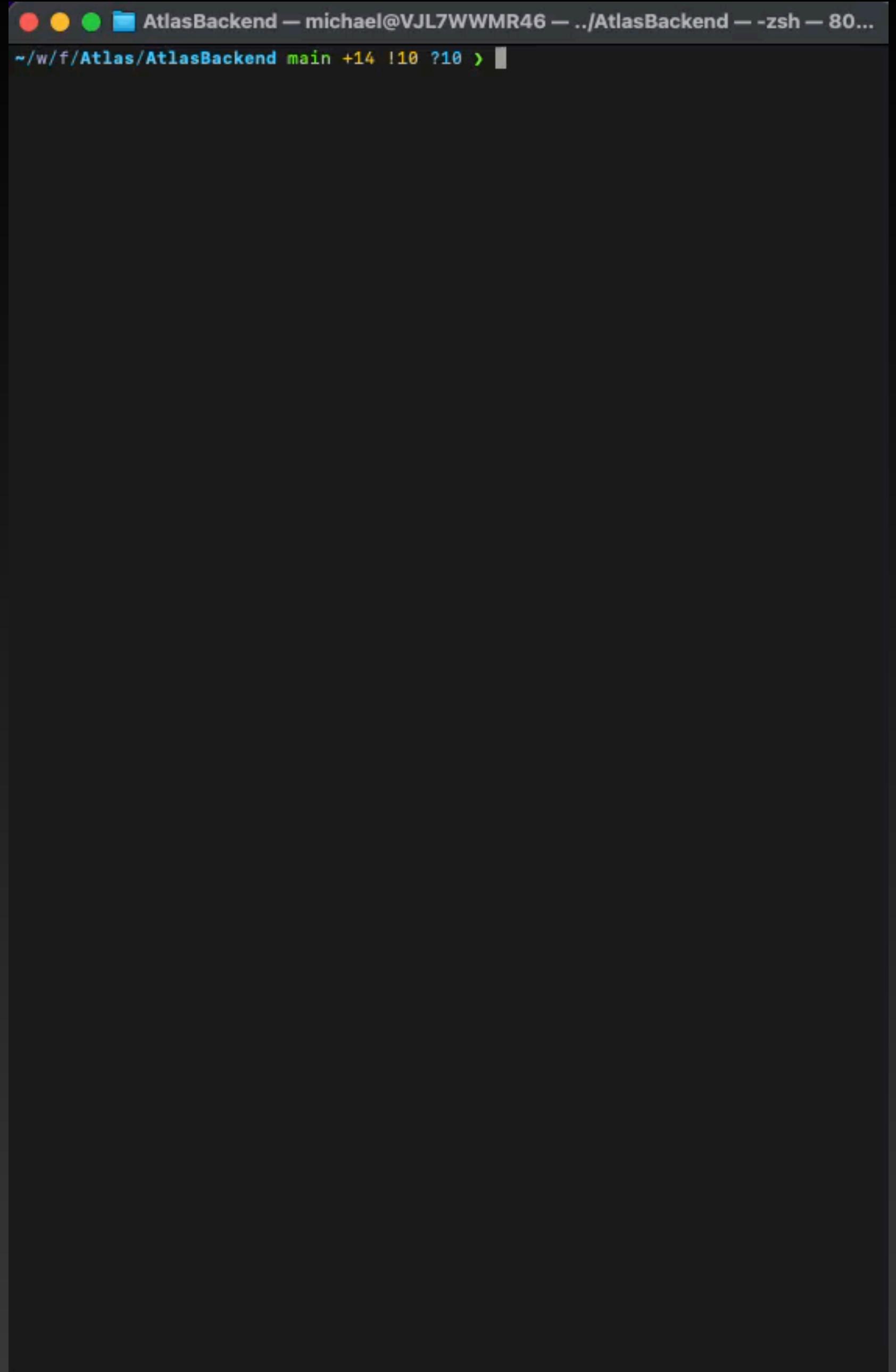
- Isolated container with running Linux
- Image with Swift compiler installed
- Builds our backend application
- Access to environment variables
- Runs application



A screenshot of a terminal window titled "AtlasBackend". The title bar also shows "michael@VJL7WWMR46" and "zsh - 80...". The terminal path is "~/.w/f/Atlas/AtlasBackend". The command being run is "main +14 !10 ?10 >". The window has a dark theme with light-colored text and icons.

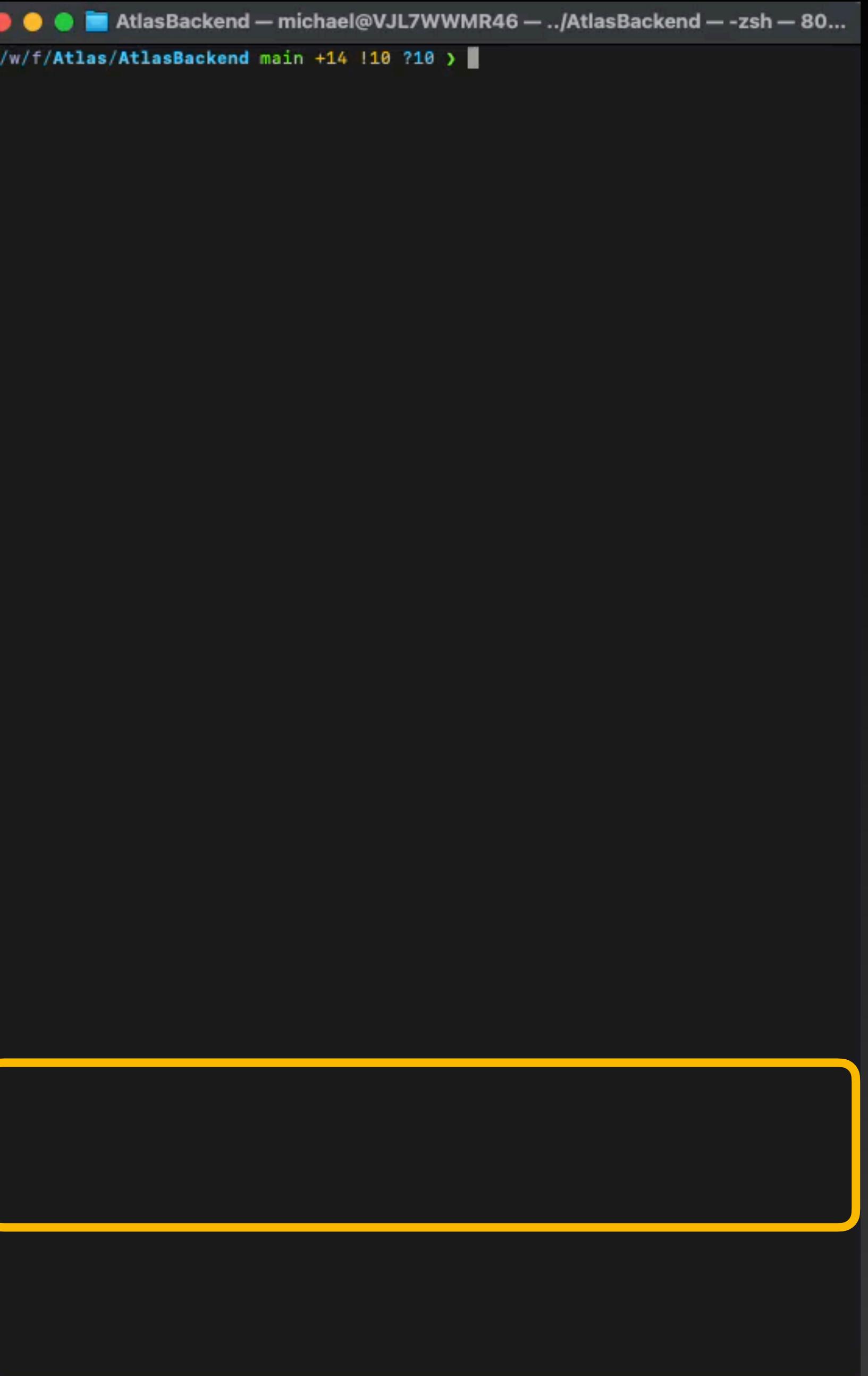
# fly.toml

- Configures the fly app
- Defines the number of fly machines
- Defines Resources



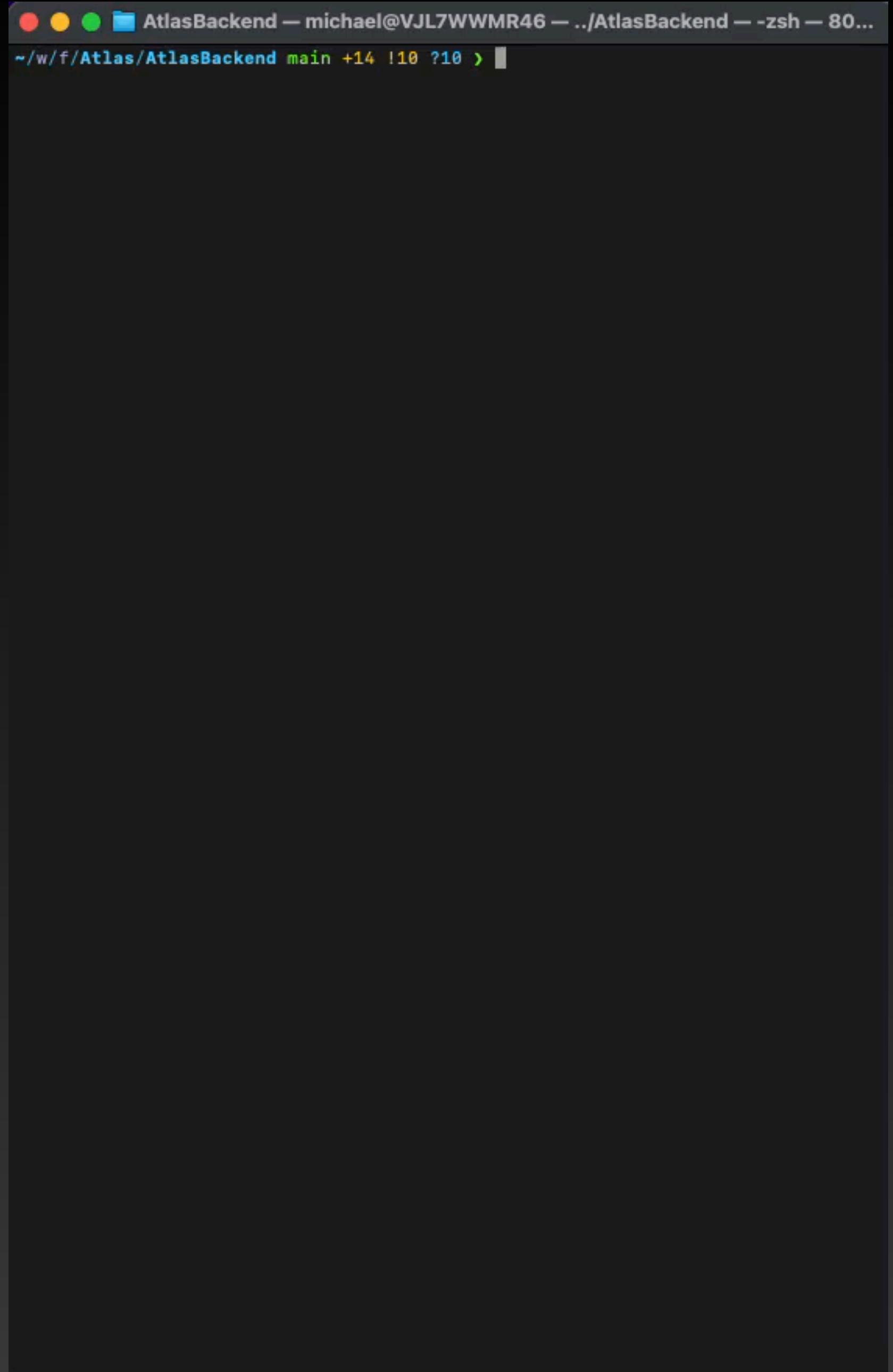
A screenshot of a terminal window with a dark background. At the top, there's a header bar with three colored dots (red, yellow, green) and a folder icon. The text in the header reads "AtlasBackend — michael@VJL7WWMR46 — .. /AtlasBackend — -zsh — 80...". Below the header, the terminal shows a file tree: "~ /w /f /Atlas /AtlasBackend". The current directory is "main" and the command "ls" was run, as indicated by the output "+14 !10 ?10 >". The bottom part of the terminal is a standard zsh command line interface.

✖ Error: no such module 'OSLog'



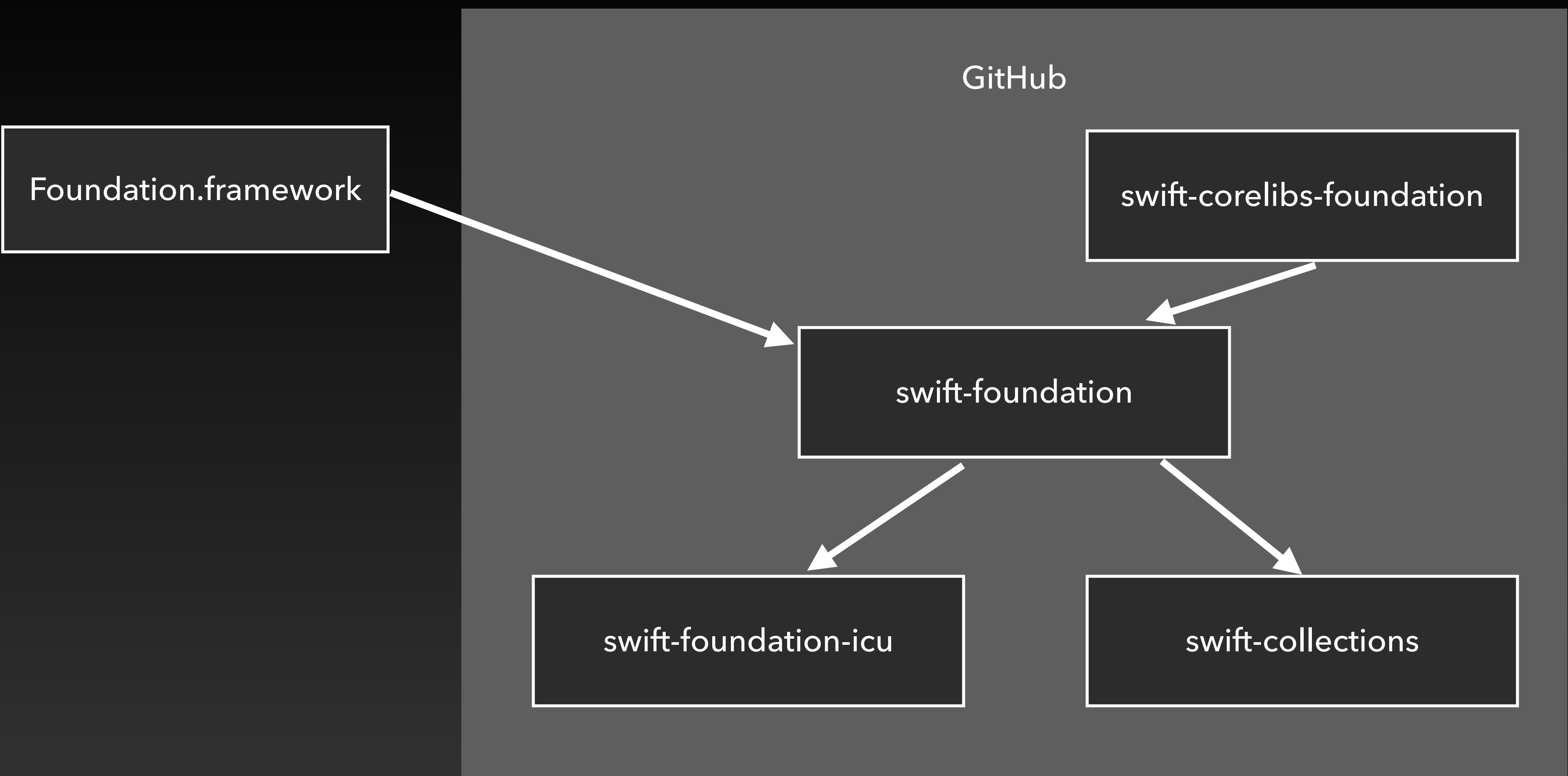
A screenshot of a terminal window titled "AtlasBackend". The title bar includes the path ".../AtlasBackend", the user "michael@VJL7WWMR46", the command "zsh", and the process ID "80...". The terminal window shows the command "cd .." followed by the output "w/f/Atlas/AtlasBackend main +14 !10 ?10 >". A yellow rectangular box highlights the error message "Error: no such module 'OSLog'" which appears to be cut off at the bottom of the terminal window.

✖ Error: 'URLSessionConfiguration'  
is unavailable: This type has moved  
to the **FoundationNetworking**  
module. Import that module to use  
it.



A screenshot of a terminal window titled "AtlasBackend". The title bar also shows "michael@VJL7WWMR46" and "zsh". The terminal window displays the following text:  
~/w/f/Atlas/AtlasBackend main +14 !10 ?10 >  
This is a standard macOS terminal interface with a dark background and light-colored text.

# FoundationNetworking...?



<https://github.com/swiftlang/swift-foundation>

URL

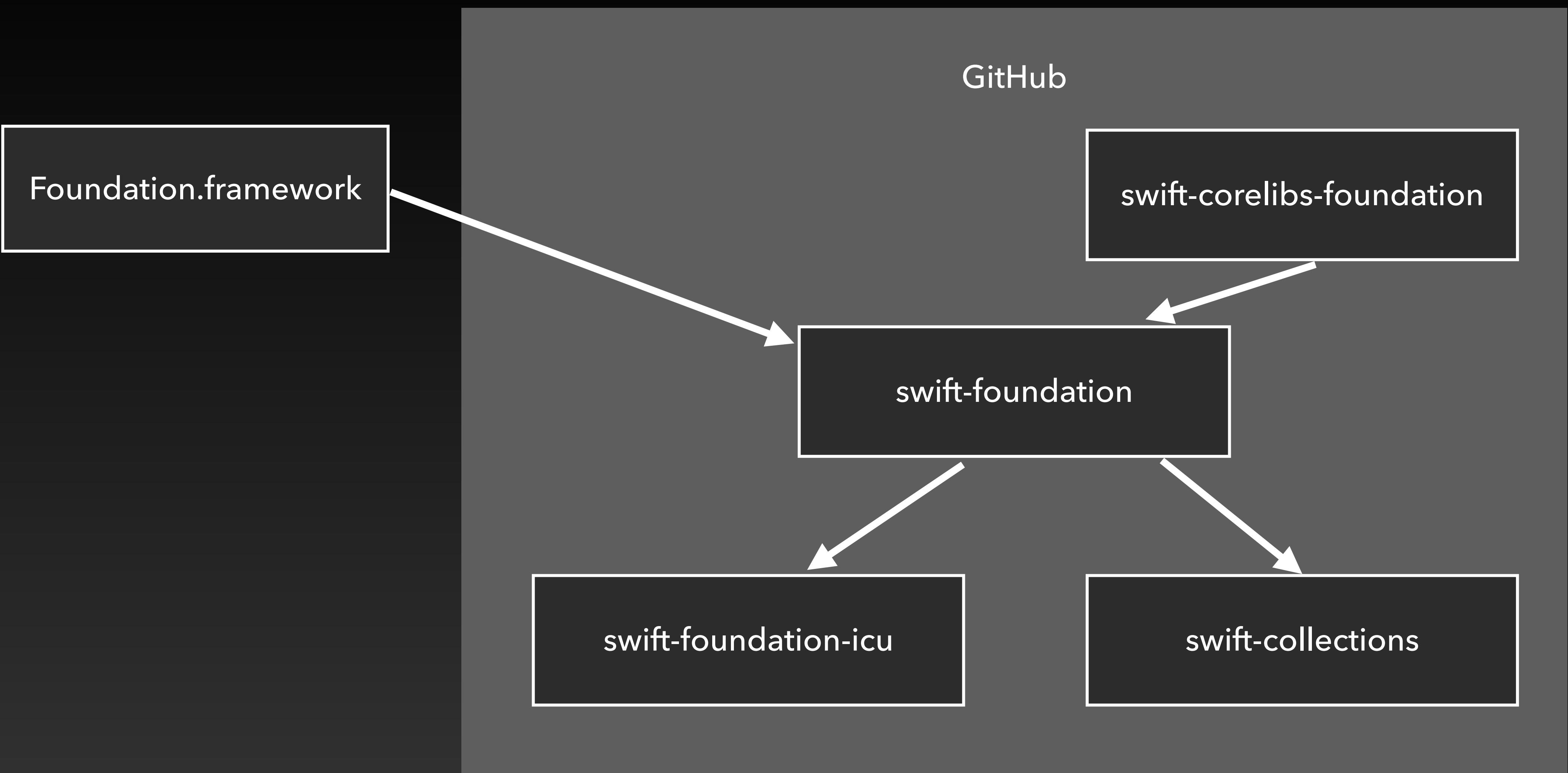
Data

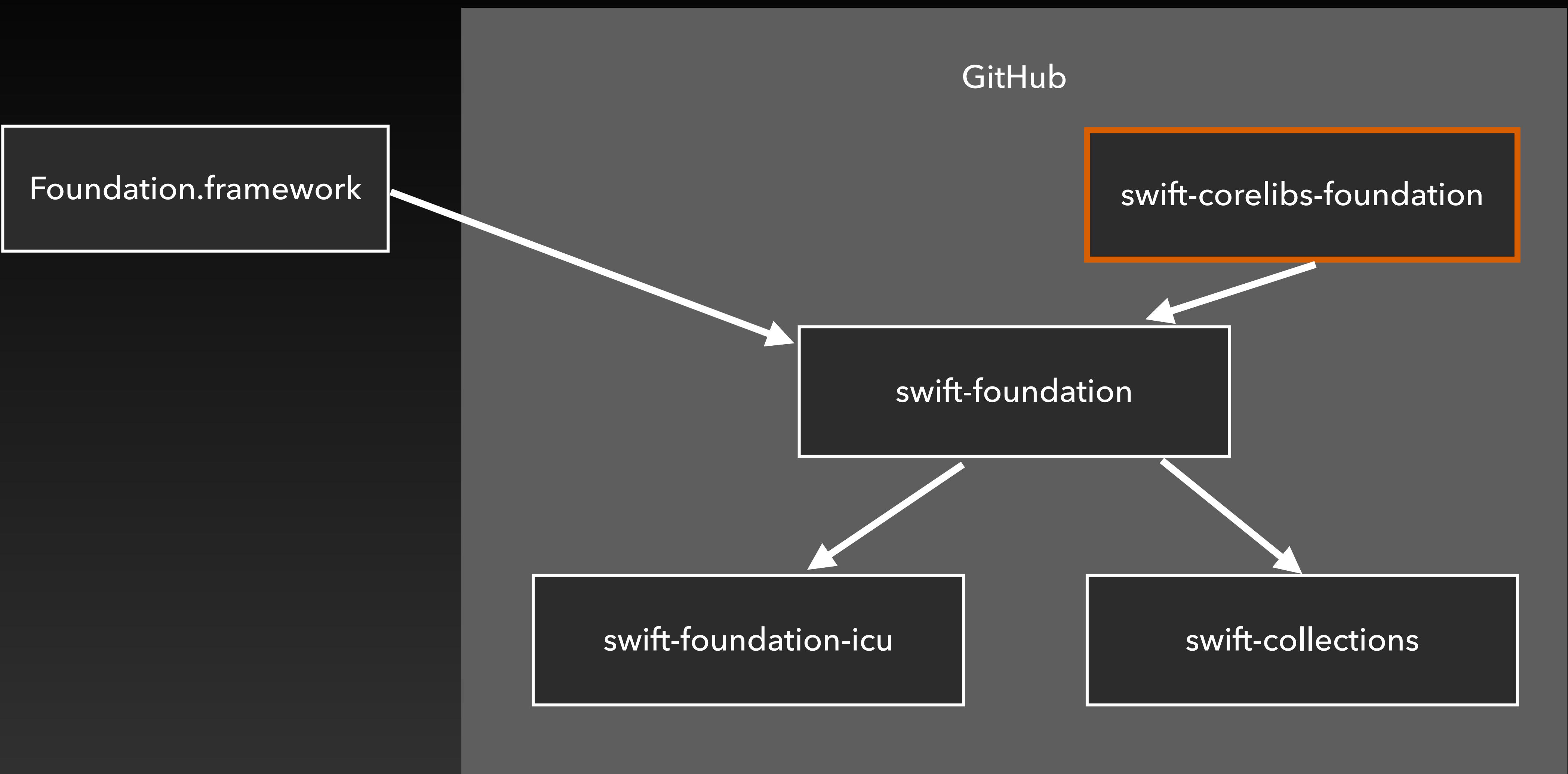
JSONDecoder

swift-foundation

Calendar

Locale





<https://github.com/swiftlang/swift-corelibs-foundation>

NSFormatter

NSKeyedArchiver

swift-corelibs-foundation

URLSession

URLRequest

*swift-corelibs-foundation builds for non-Darwin platforms only.*

*It installs the Foundation umbrella module, FoundationXML, and FoundationNetworking.*

<https://github.com/swiftlang/swift-corelibs-foundation>

NSFormatter

NSKeyedArchiver

swift-corelibs-foundation

URLSession

URLRequest

*swift-corelibs-foundation builds for **non-Darwin platforms only**.*

*It installs the **Foundation** umbrella module, **FoundationXML**, and **FoundationNetworking**.*

URLSession & Linux?  
this is needed

```
import Foundation
#if canImport(FoundationNetworking)
import FoundationNetworking
#endif
```



Everything fixed now?

✖ Error: value of type 'URLSession' has no member 'bytes'

✖ Error: value of type 'URLSession' has no member 'bytes'

```
/// Returns a byte stream that conforms to AsyncSequence protocol.  
///  
/// - Parameter request: The URLRequest for which to load data.  
/// - Parameter delegate: Task-specific delegate.  
/// - Returns: Data stream and response.  
public func bytes(  
    for request: URLRequest,  
    delegate: (any URLSessionTaskDelegate)? = nil  
) async throws -> (  
    URLSession.AsyncBytes,  
    URLResponse  
)
```

# How to solve?

<https://github.com/swift-server/async-http-client>

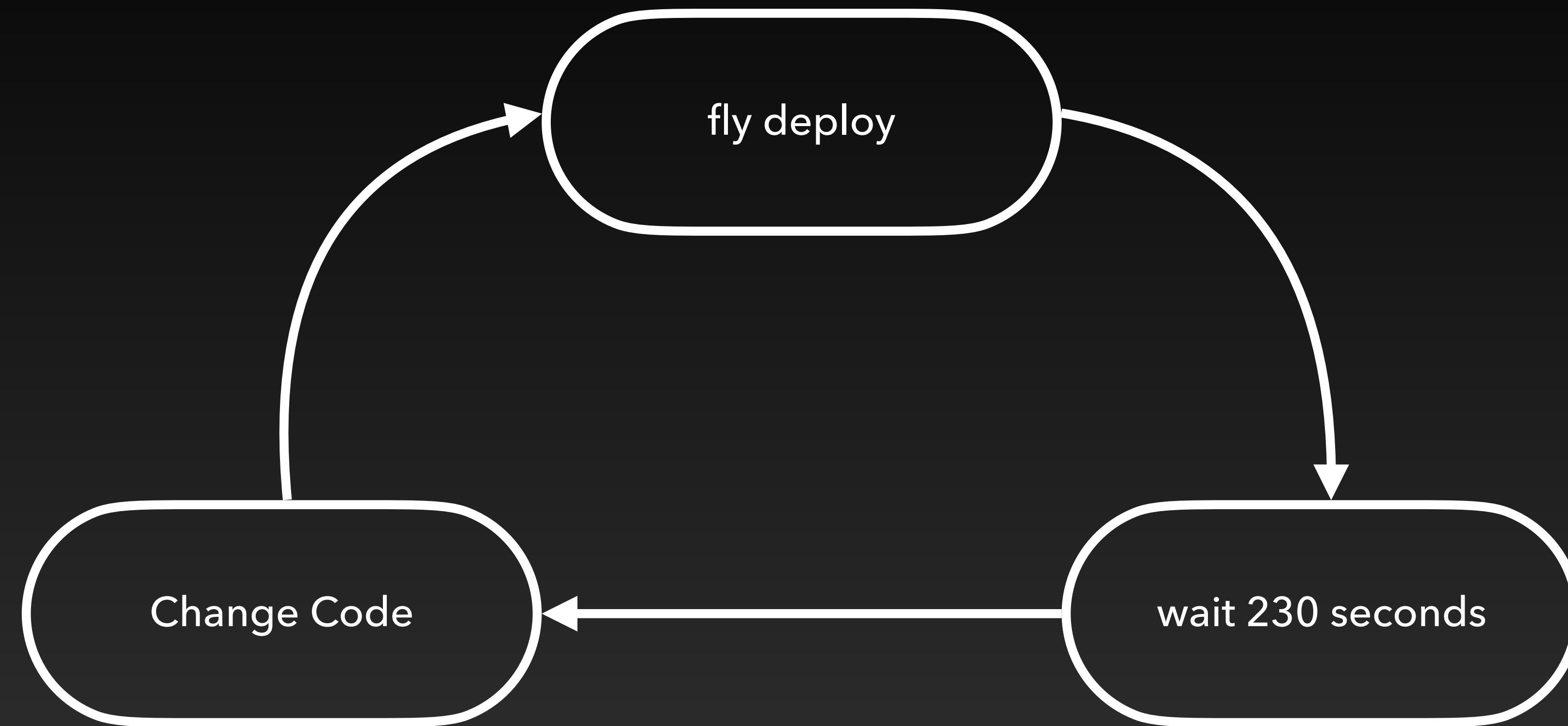
# AsyncHttpClient

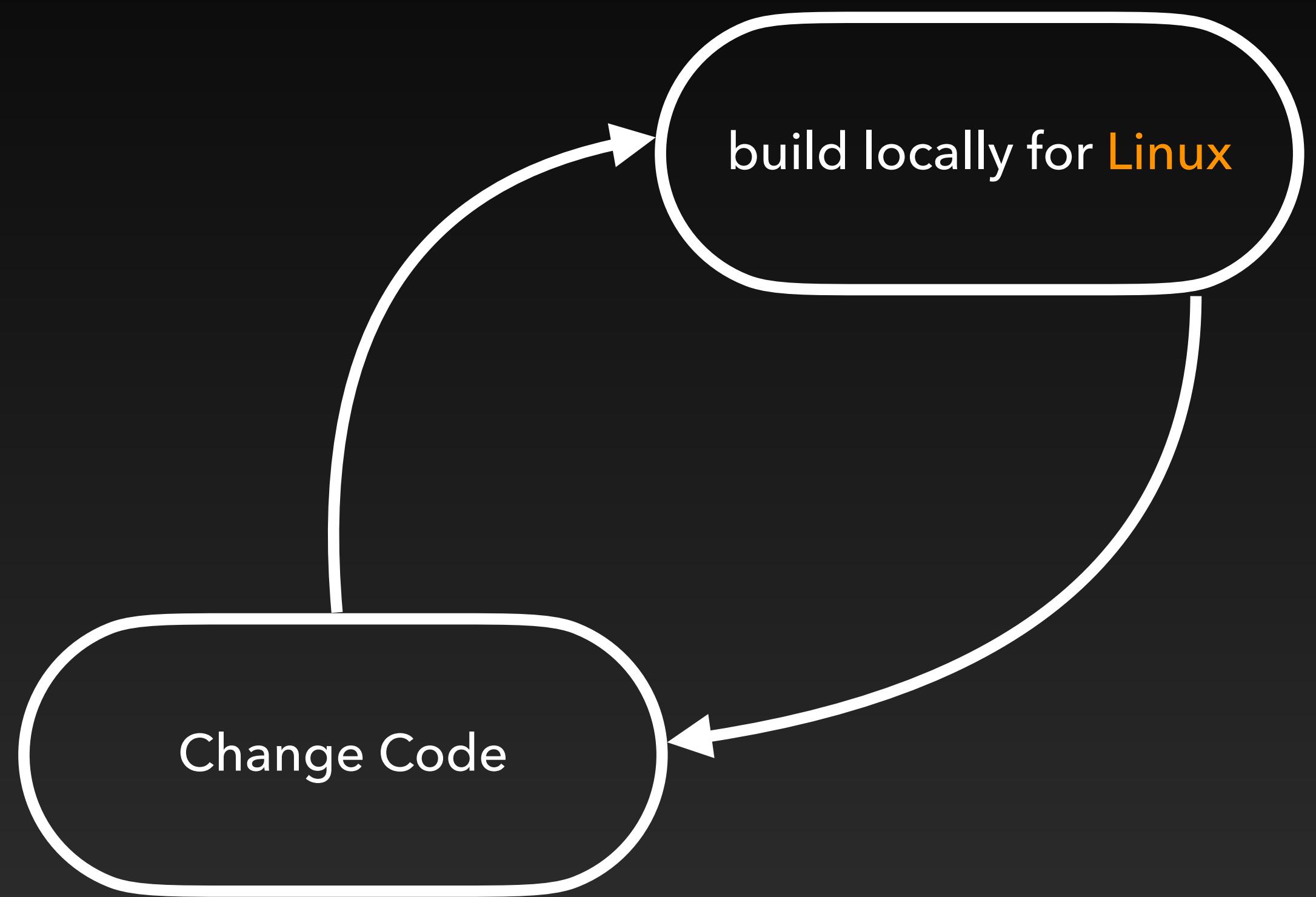
<https://github.com/swift-server/async-http-client>

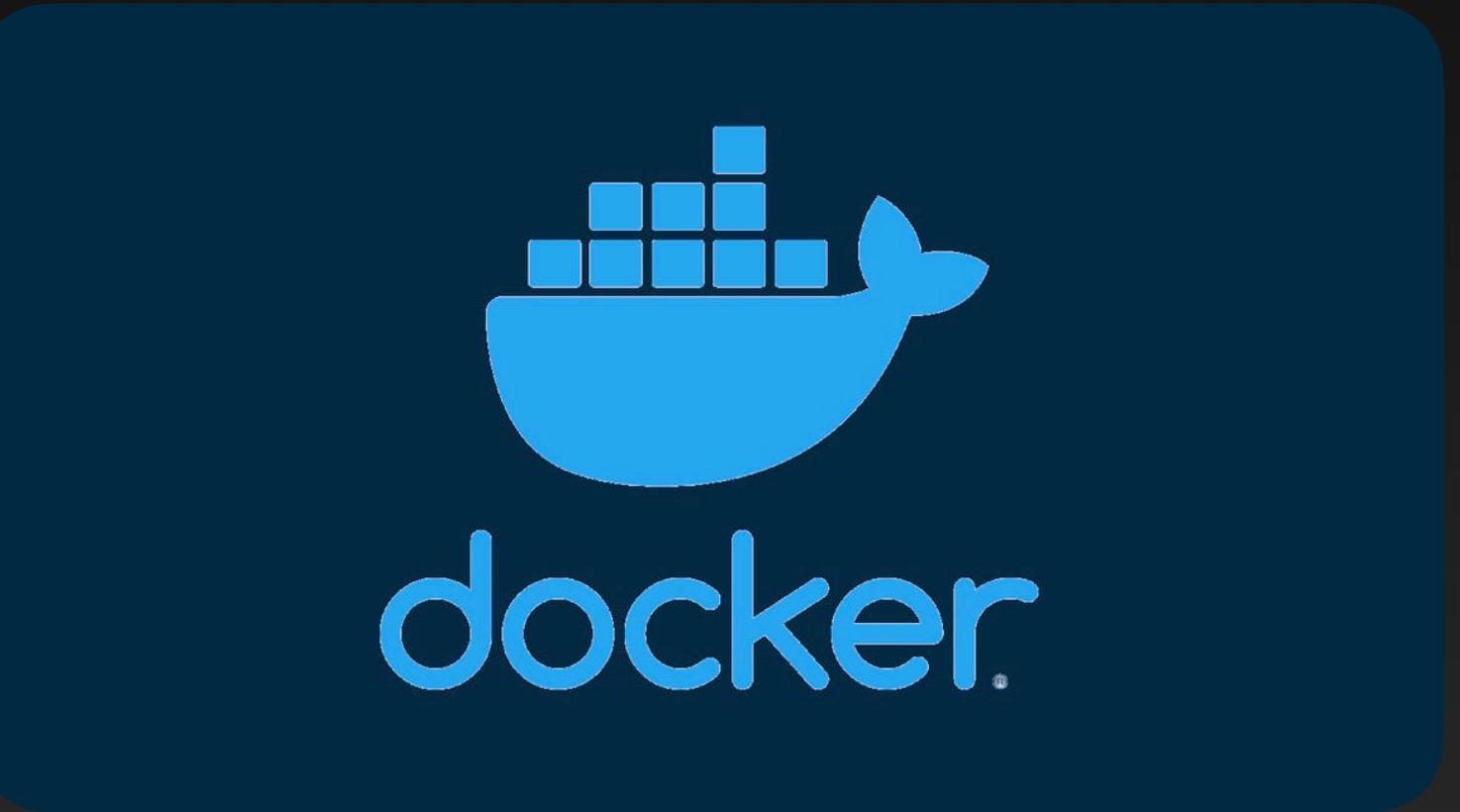
# AsyncHttpClient

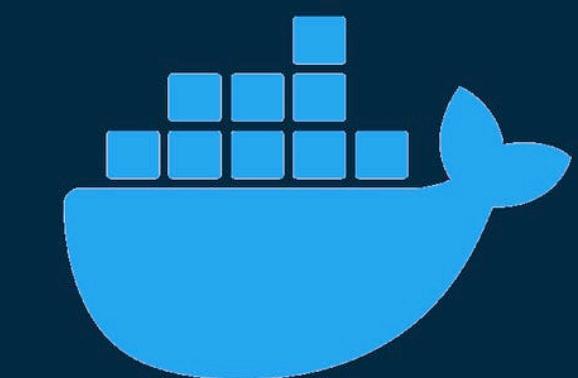
HTTP Client library built on top of **SwiftNIO**

- First class support for **Swift Concurrency**
- Asynchronous and non-blocking request methods
- Simple follow-redirects (cookie headers are dropped)
- **Streaming body download**
- TLS support
- Automatic HTTP/2 over HTTPS
- Cookie parsing (but not storage)





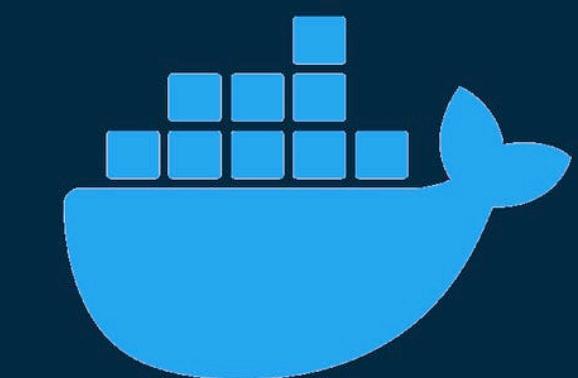




docker

AtlasBackend — root@255e4873c7d1: /src — docker run --rm -it -v ~/workspace/freiwald/Atlas/AtlasBackend:/src...

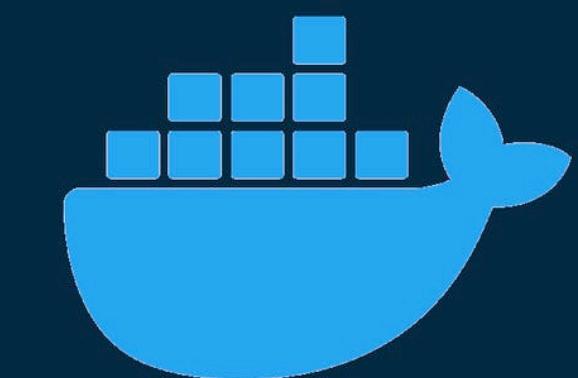
```
[~/workspace/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > docker run --rm -it -v "$(pwd)":"/src" -w "/src" swift ]  
[root@255e4873c7d1:/src# uname -a  
Linux 255e4873c7d1 6.6.32-linuxkit #1 SMP Thu Jun 13 14:13:01 UTC 2024 aarch64 aarch64 aarch64 GNU/Linux ]  
[root@255e4873c7d1:/src# swift --version  
Swift version 6.0.2 (swift-6.0.2-RELEASE)  
Target: aarch64-unknown-linux-gnu  
root@255e4873c7d1:/src# ]
```



docker

AtlasBackend — root@255e4873c7d1: /src — docker run --rm -it -v ~/workspace/freiwald/Atlas/AtlasBackend:/src...

```
[~/workspace/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > docker run --rm -it -v "$(pwd)":/src" -w "/src" swift]
[root@255e4873c7d1:/src# uname -a
Linux 255e4873c7d1 6.6.32-linuxkit #1 SMP Thu Jun 13 14:13:01 UTC 2024 aarch64 aarch64 aarch64 GNU/Linux
[root@255e4873c7d1:/src# swift --version
Swift version 6.0.2 (swift-6.0.2-RELEASE)
Target: aarch64-unknown-linux-gnu
root@255e4873c7d1:/src#
```

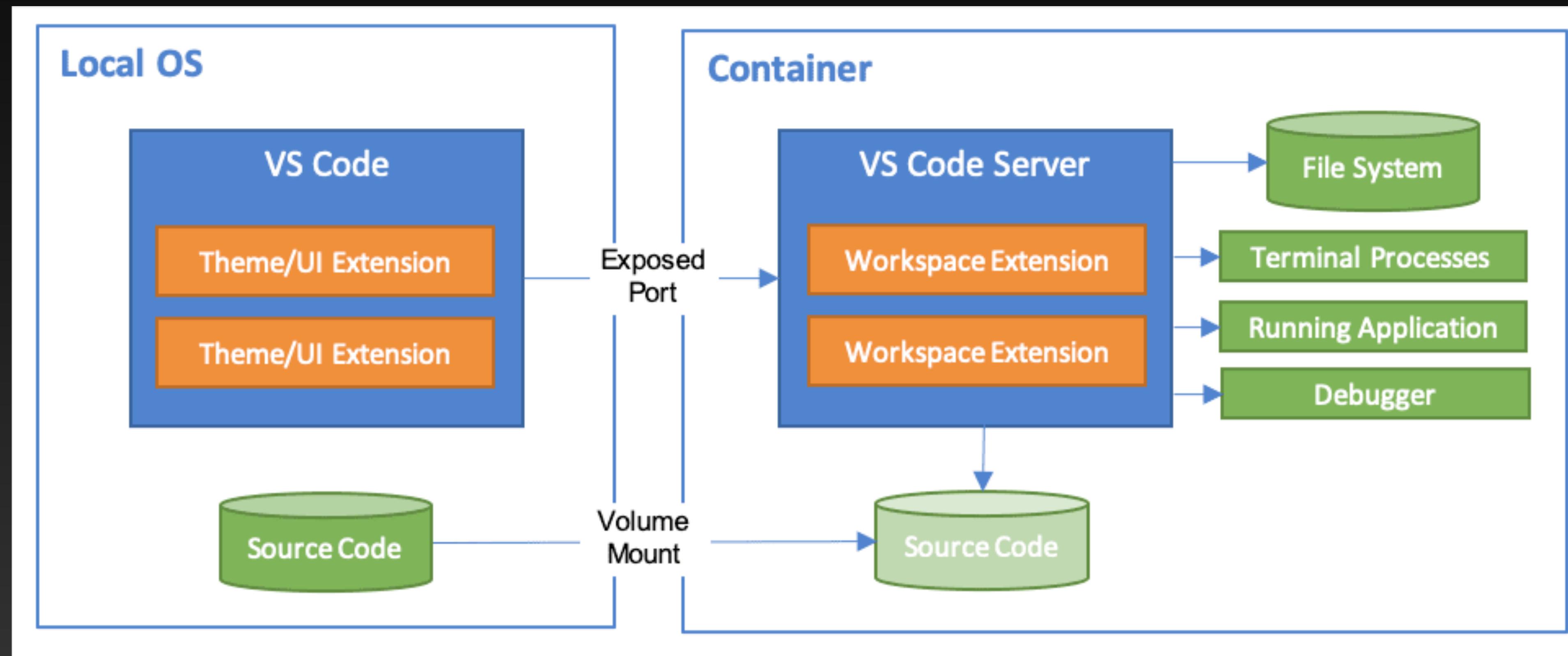


docker

AtlasBackend — root@255e4873c7d1: /src — docker run --rm -it -v ~/workspace/freiwald/Atlas/AtlasBackend:/src...

```
[~/workspace/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > docker run --rm -it -v "$(pwd):/src" -w "/src" swift ]  
root@255e4873c7d1:/src# uname -a  
Linux 255e4873c7d1 6.6.32-linuxkit #1 SMP Thu Jun 13 14:13:01 UTC 2024 aarch64 aarch64 aarch64 GNU/Linux  
root@255e4873c7d1:/src# swift --version  
Swift version 6.0.2 (swift-6.0.2-RELEASE)  
Target: aarch64-unknown-linux-gnu  
root@255e4873c7d1:/src#
```

# Visual Studio Code Dev Containers



There is a better way

# Swift Static Linux SDK

# Swift Static Linux SDK

- Install Open Source Toolchain of Swift (Xcode is not enough)
- Ensure to use this toolchain when building with swift
- Install the Static Linux SDK (same version as the toolchain)

<https://www.swift.org/documentation/articles/static-linux-getting-started.html>



AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23

```
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build && file .build/debug/AtlasBackend
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude fr
om the target
/Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Docume
ntation.docc
Building for debugging...
[2/2] Write swift-version--4A847ED0836F2485.txt
Build complete! (8.61s)
.build/debug/AtlasBackend: Mach-O 64-bit executable arm64
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```



AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23

```
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build && file .build/debug/AtlasBackend
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude fr
om the target
  /Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Docume
ntation.docc
Building for debugging...
[2/2] Write swift version: 4A947ED0836F2495.txt
Build complete! (8.61s)
.build/debug/AtlasBackend: Mach-O 64-bit executable arm64
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```



AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23

```
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build && file .build/debug/AtlasBackend
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude fr
om the target
  /Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Docume
ntation.docc
Building for debugging...
[2/2] Write swift-version--4A847ED0836F2485.txt
Build complete! (0.61s)
.build/debug/AtlasBackend: Mach-O 64-bit executable arm64
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```

```
AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > export TOOLCHAINS=$(plutil -extract CFBundleIdentifier raw /Library/Developer/Toolchains/swift-6.0.1-RELEASE.xctoolchain/Info.plist)
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build --swift-sdk x86_64-swift-linux-musl | (head -n 2 && tail -n 2)
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude from the target
    /Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Documentation.docc
[0/1] Planning build
[1/1] Compiling plugin GenerateManual
<unknown>:0: warning: libc not found for 'x86_64-swift-linux-musl'; C stdlib may be unavailable
Build complete! (14.01s)
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > file .build/x86_64-swift-linux-musl/debug/AtlasBackend
.build/x86_64-swift-linux-musl/debug/AtlasBackend: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, with debug_info, not stripped
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```



AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23

```
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > export TOOLCHAINS=$(plutil -extract CFBundleIdentifier raw /Library/Developer/Toolchains/swift-6.0.1-RELEASE.xctoolchain/Info.plist)
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build --swift-sdk x86_64-swift-linux-musl | (head -n 2 && tail -n 2)
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude from the target
    /Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Documentation.docc
[0/1] Planning build
[1/1] Compiling plugin GenerateManual
<unknown>:0: warning: libc not found for 'x86_64-swift-linux-musl'; C stdlib may be unavailable
Build complete! (14.01s)
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > file .build/x86_64-swift-linux-musl/debug/AtlasBackend
.build/x86_64-swift-linux-musl/debug/AtlasBackend: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, with debug_info, not stripped
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```



AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23

```
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > export TOOLCHAINS=$(plutil -extract CFBundleIdentifier raw /Library/Developer/Toolchains/swift-6.0.1-RELEASE.xctoolchain/Info.plist)
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build --swift-sdk x86_64-swift-linux-musl | (head -n 2 && tail -n 2)
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude from the target
    /Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Documentation.docc
[0/1] Planning build
[1/1] Compiling plugin GenerateManual
<unknown>:0: warning: libc not found for 'x86_64-swift-linux-musl'; C stdlib may be unavailable
Build complete! (14.01s)
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > file .build/x86_64-swift-linux-musl/debug/AtlasBackend
.build/x86_64-swift-linux-musl/debug/AtlasBackend: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, with debug_info, not stripped
~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```



AtlasBackend — michael@VJL7WWMR46 — ../AtlasBackend — -zsh — 116x23

```
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > export TOOLCHAINS=$(plutil -extract CFBundleIdentifier raw /Library/Developer/Toolchains/swift-6.0.1-RELEASE.xctoolchain/Info.plist)
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > swift build --swift-sdk x86_64-swift-linux-musl | (head -n 2 && tail -n 2)
warning: 'swift-algorithms': found 1 file(s) which are unhandled; explicitly declare them as resources or exclude from the target
    /Users/michael/workspace/freiwald/Atlas/AtlasBackend/.build/checkouts/swift-algorithms/Sources/Algorithms/Documentation.docc
[0/1] Planning build
[1/1] Compiling plugin GenerateManual
<unknown>:0: warning: libc not found for 'x86_64-swift-linux-musl'; C stdlib may be unavailable
Build complete! (14.01s)
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 > file .build/x86_64-swift-linux-musl/debug/AtlasBackend
.build/x86_64-swift-linux-musl/debug/AtlasBackend: ELF 64-bit LSB executable, x86-64 version 1 (SYSV), statically linked, with debug_info, not stripped
[~/w/freiwald/Atlas/AtlasBackend main +14 !10 ?10 >
```

# Upload to a Linux machine

Build on Mac  
Run anywhere

# Build on Mac

## Run anywhere

- Xcode can only change Toolchain, no build arguments for SPM
- VSCode for server development
- Use static linux sdk to check compatibility

# Build on Mac

## Run anywhere

- Build Linux binary on macOS
- Uploaded binary to a Linux machine
- Run app on Linux
- Make a request

# Summary

- Keep our API-Keys out of the client app
- Swift on Server works
- Be careful when selection 3rd-party-SDKs
- check Linux compatability if needed
- This is just the beginning



Swift Cloud  
x  
SwiftWasm



Lambda



# One more thing

**How to secure your own Backend?**

# One more thing

- API-Keys (on client-side 😜)
- Certificate Pinning (with your own server certificate)
- App Attest (to ensure that requests your server receives come from legitimate instances of your app)
- Validate app receipts from App Store
- Hope for Apple Intelligence SDK 🙏

# Thank you



<https://freiwald.dev>