



an introduction to

VAPOR 3

by Jonas Schwartz



Who am I?

Jonas Schwartz

Partner @ Vapor

Author @ raywenderlich.com

Past as backend developer and sysadmin

GitHub/Twitter: @joscdk

Slack/Discord: jonas



VAPOR

=

Web Framework

written in Swift

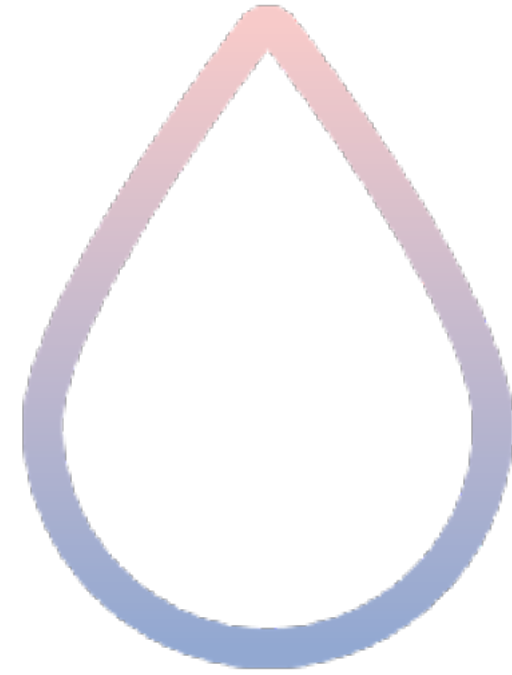


Brief History



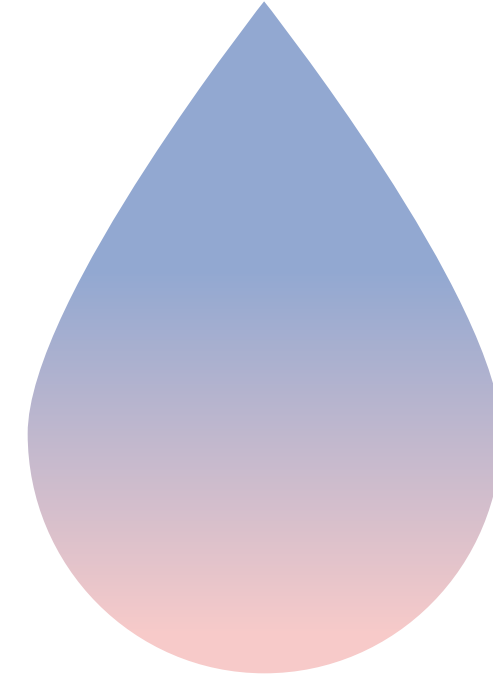
0.1

Jan, 2016



1.0

Sep, 2016



2.0

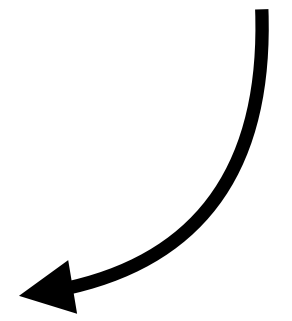
May, 2017

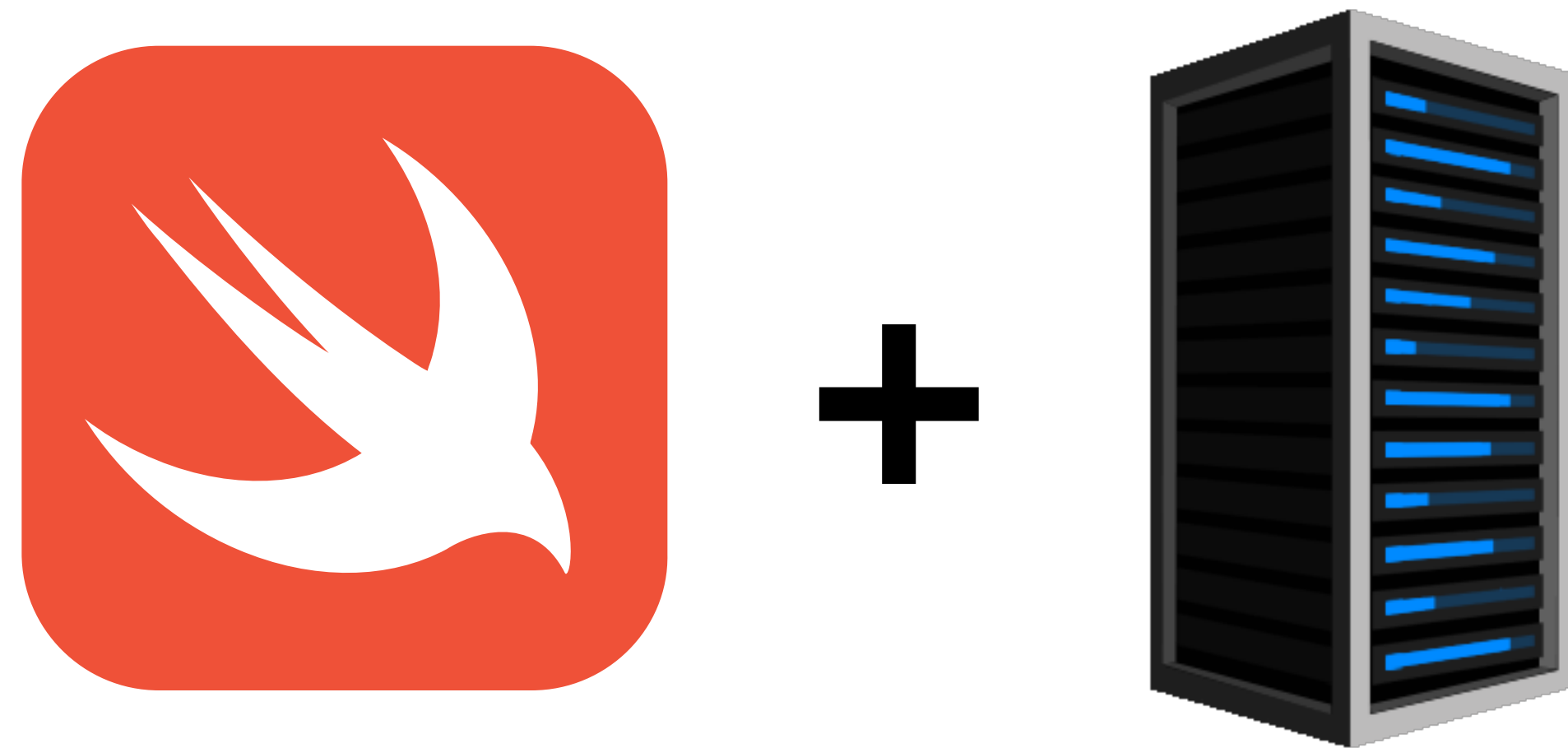


3.0

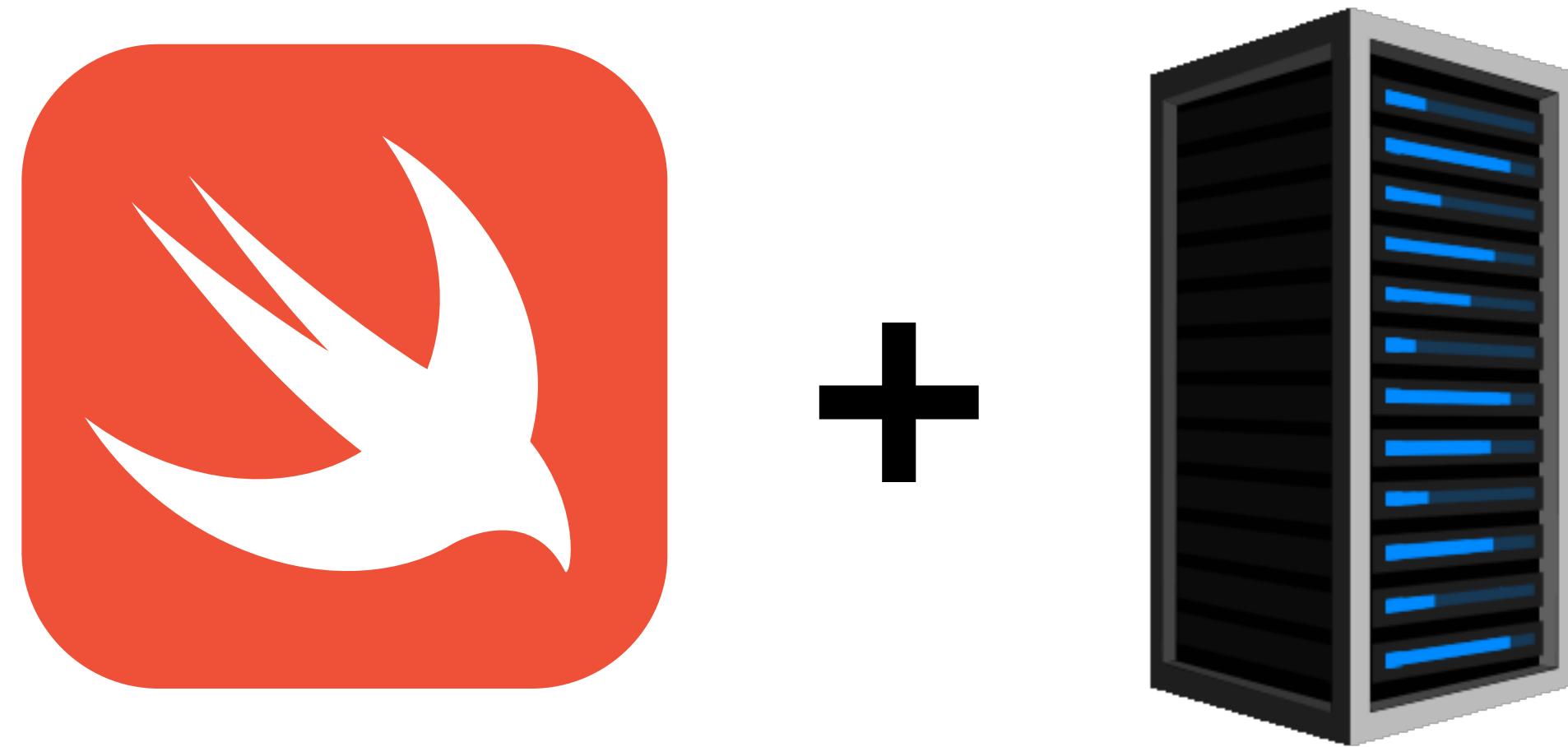
April, 2018

Rewrite



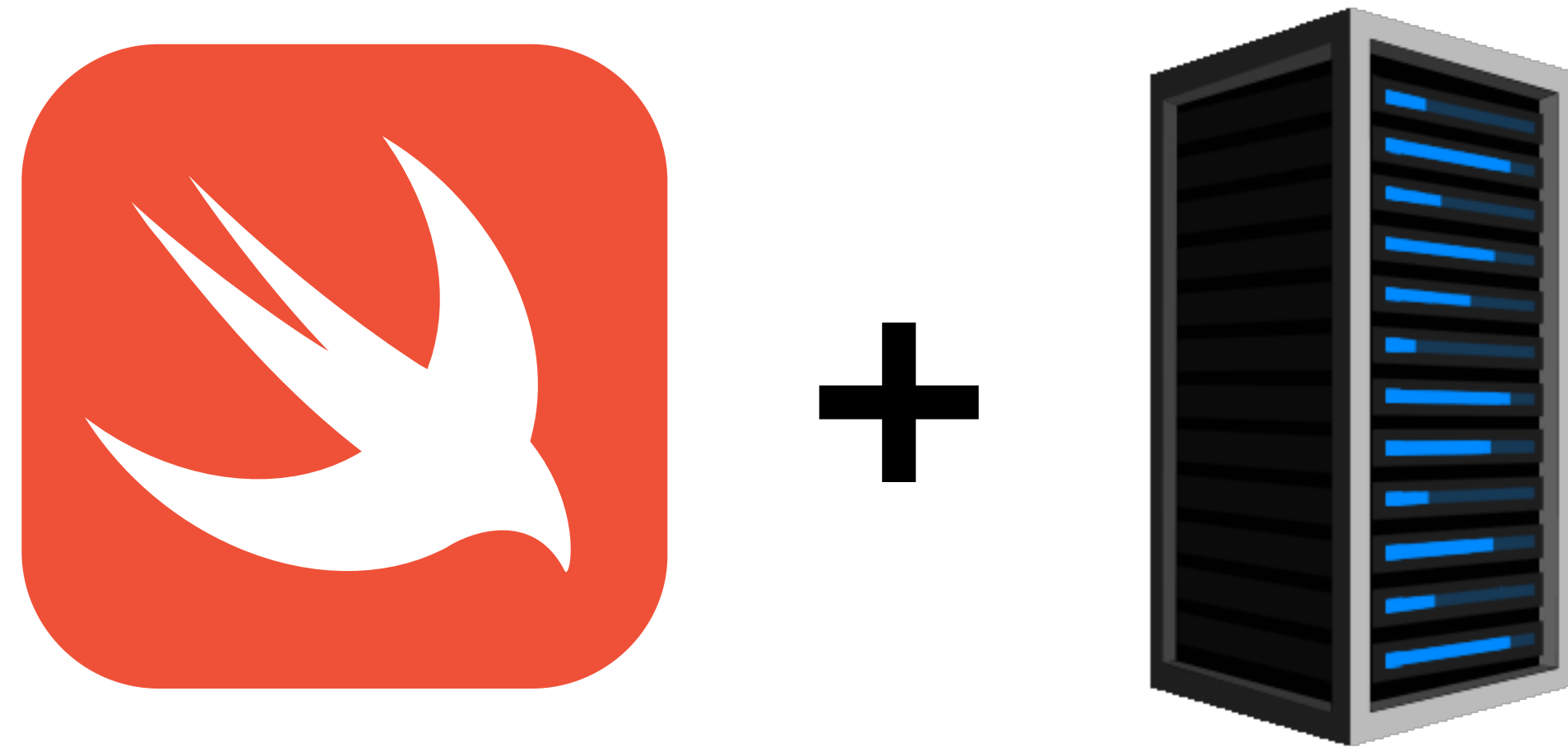


Server-side Swift?



Server-side Swift?

Swift ≥ 3 (early 2016) runs on Linux



Server-side Swift?

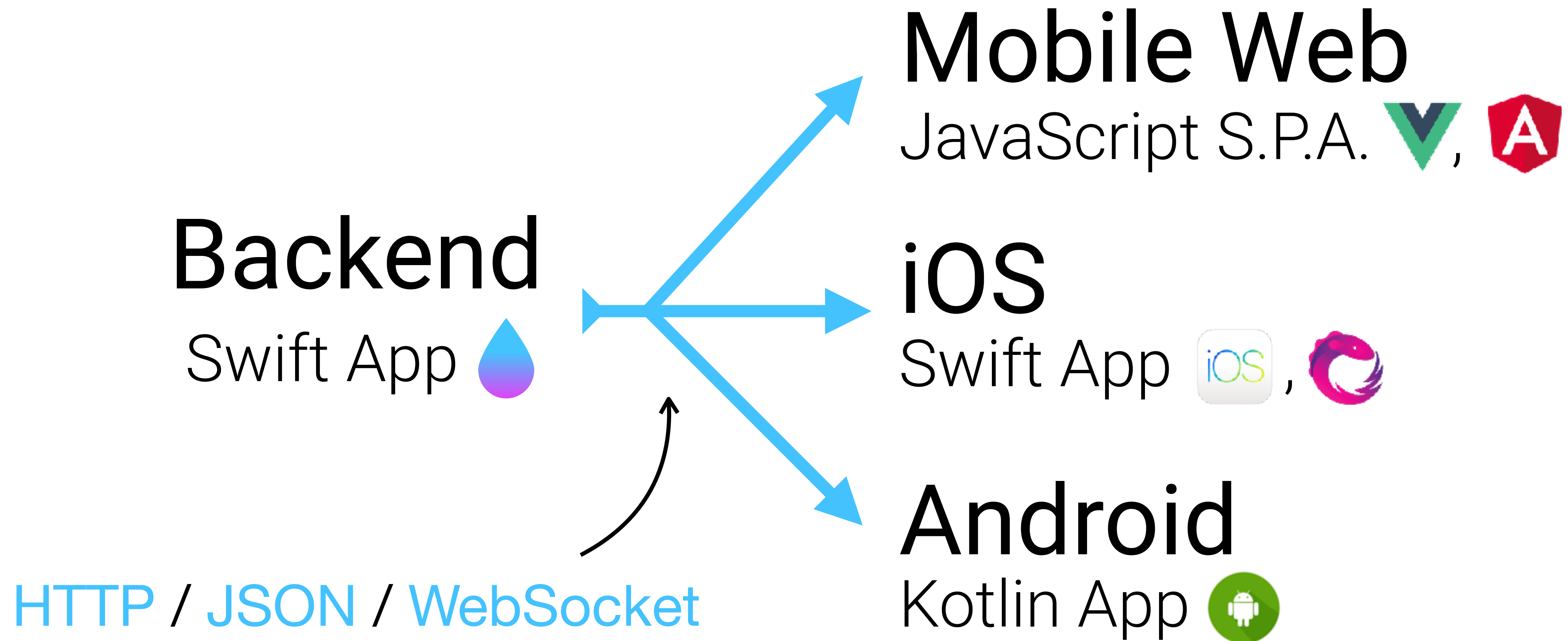
Swift ≥ 3 (early 2016) runs on Linux



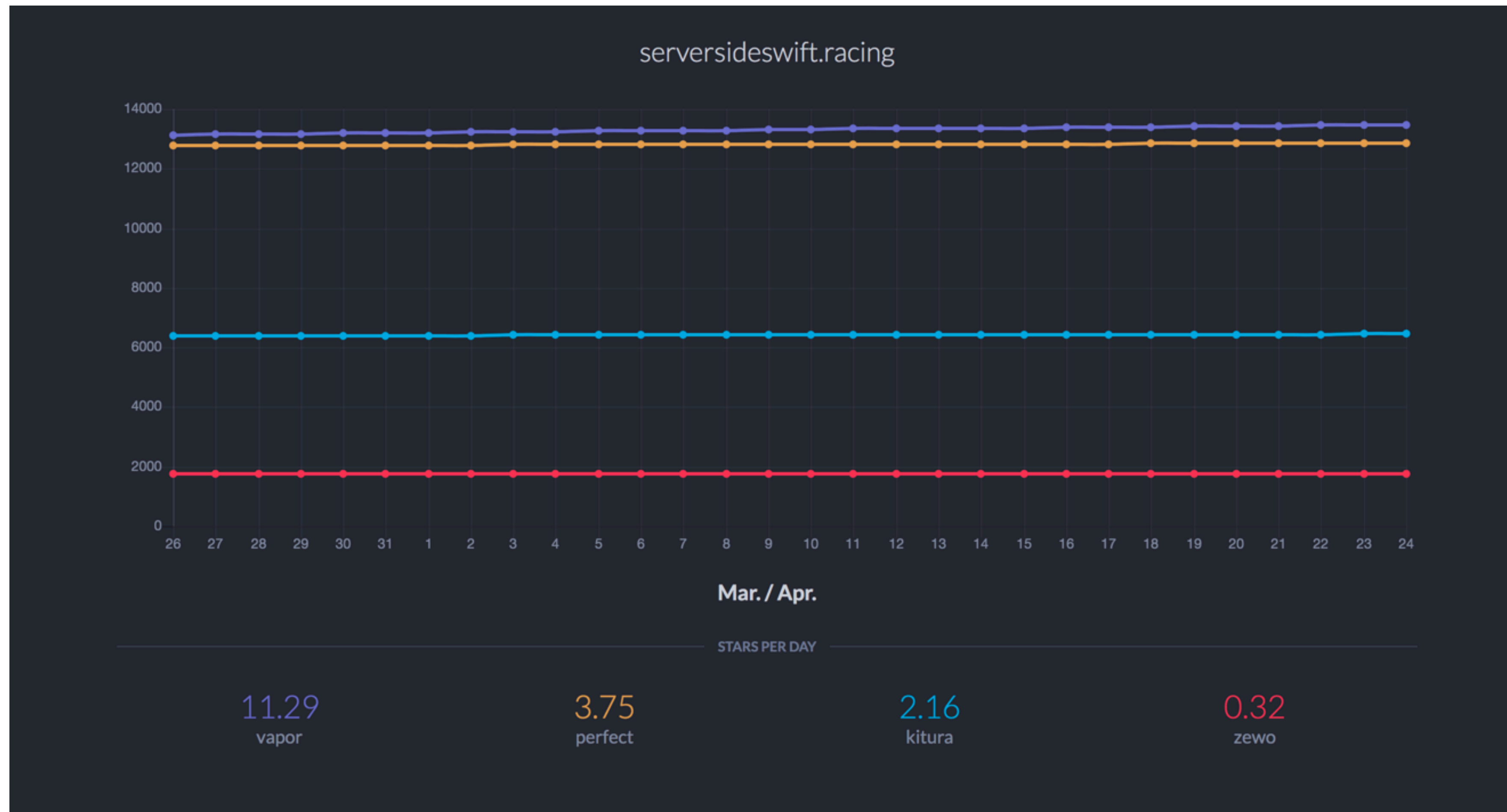
Web Framework



Swift Backend



★ Most popular on GitHub!







Ideology



Safety



Performance



Beautiful APIs



Ideology



Safety

Fail during development, not production

```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var name: String  
}
```

```
extension DatabaseConnectable {  
    func findUser(named name: String) -> Future<User?> {  
        return User.query(on: self).filter(\.name == name).first()  
    }  
}
```

```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var fullName: String { return firstName + " " + lastName }  
    var firstName: String  
    var lastName: String  
}
```

```
extension DatabaseConnectable {  
    func findUser(named name: String) -> Future<User?> {  
        return User.query(on: self).filter(\.name == name).first()  
    }  
}
```



```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var fullName: String { return firstName + " " + lastName }  
    var firstName: String  
    var lastName: String  
}
```

```
extension DatabaseConnectable {  
    func findUser(named name: String) -> Future<User?> {  
        return User.query(on: self).filter(\.name == name).first()  
    }  
}
```




```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var fullName: String { return firstName + " " + lastName }  
    var firstName: String  
    var lastName: String  
}
```

```
extension DatabaseConnectable {  
    func findUser(named name: String) -> Future<User?> {  
        return User.query(on: self).filter(\.name == name).first()  
    }  
}
```

❗ Type of expression is ambiguous without more context ✕



```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var name: String  
}
```

```
extension DatabaseConnectable {  
    func findUser(named names: [String]) -> Future<User?> {  
        return User.query(on: self).filter(\.name == names).first()  
    }  
}
```

```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var name: String  
}  
  
extension DatabaseConnectable {  
    func findUser(named names: [String]) -> Future<User?> {  
        return User.query(on: self).filter(\.name == names).first()  
    }  
}
```



```
struct User: SQLiteModel {  
    static let idKey = \User.id  
    var id: UUID?  
    var name: String  
}
```

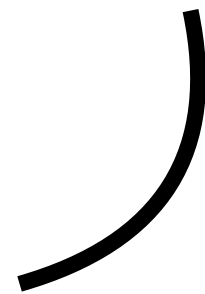
```
extension DatabaseConnectable {  
    func findUser(named names: [String]) -> Future<User?> {  
        return User.query(on: self).filter(\.name == names).first()  
    }  
}
```

❗ Type of expression is ambiguous without more context ✕



```
struct User: SQLiteModel {
    static let idKey = \User.id
    var id: UUID?
    var name: String
}

extension DatabaseConnectable {
    func findUser(named names: [String]) -> Future<User?> {
        return User.query(on: self).filter(\.name, in: names).first()
    }
}
```





Ideology



Performance

*Make a high-level framework that
is incredibly fast.*

Vapor 3 (pre-NIO)

120k RPS

8MB

Gin (Go)

99k RPS

18MB

Vapor 3

95k RPS

6MB

Node

60k RPS

256MB

Laravel

~1k RPS

~2GB



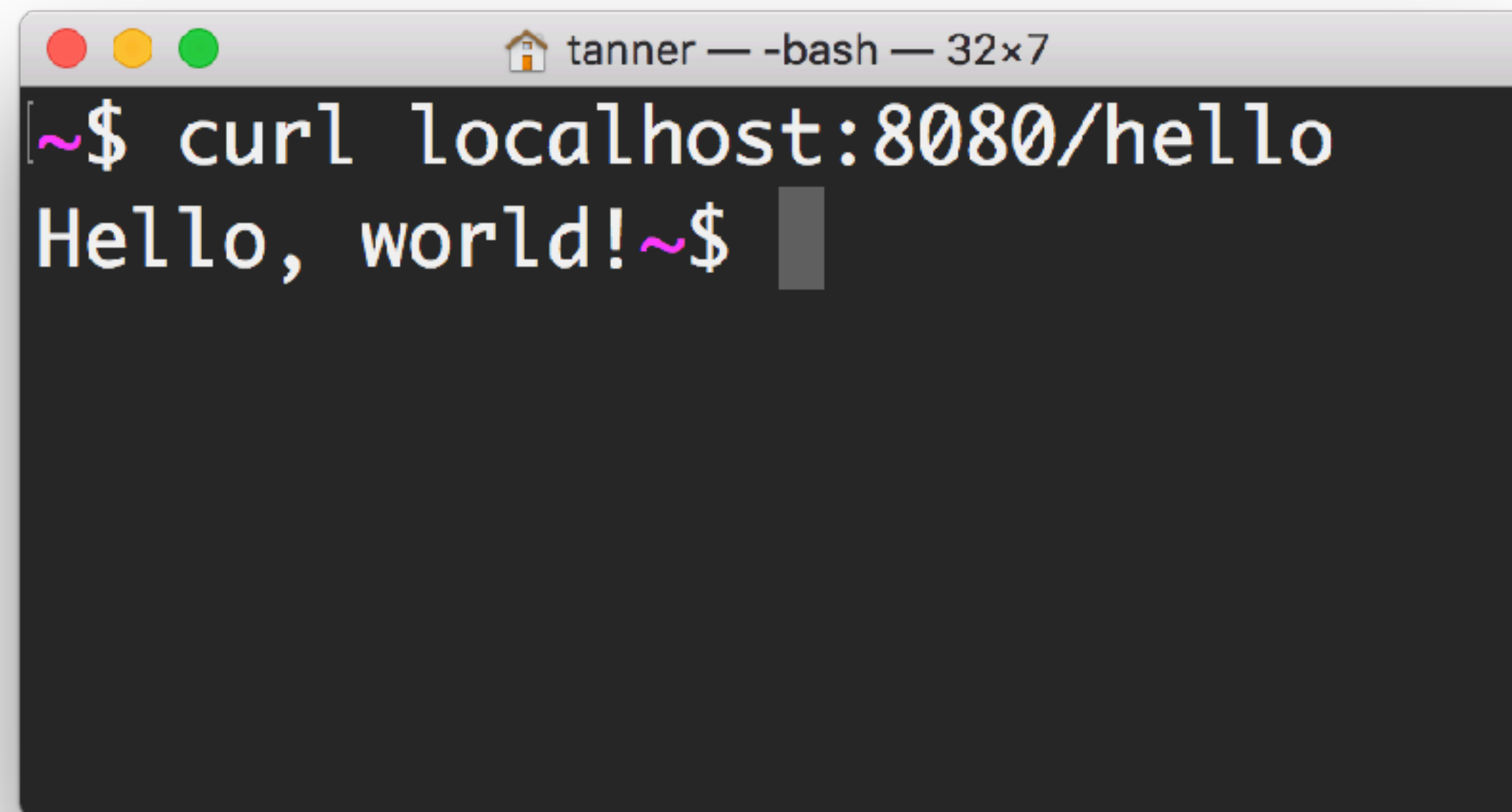
Ideology



Beautiful APIs

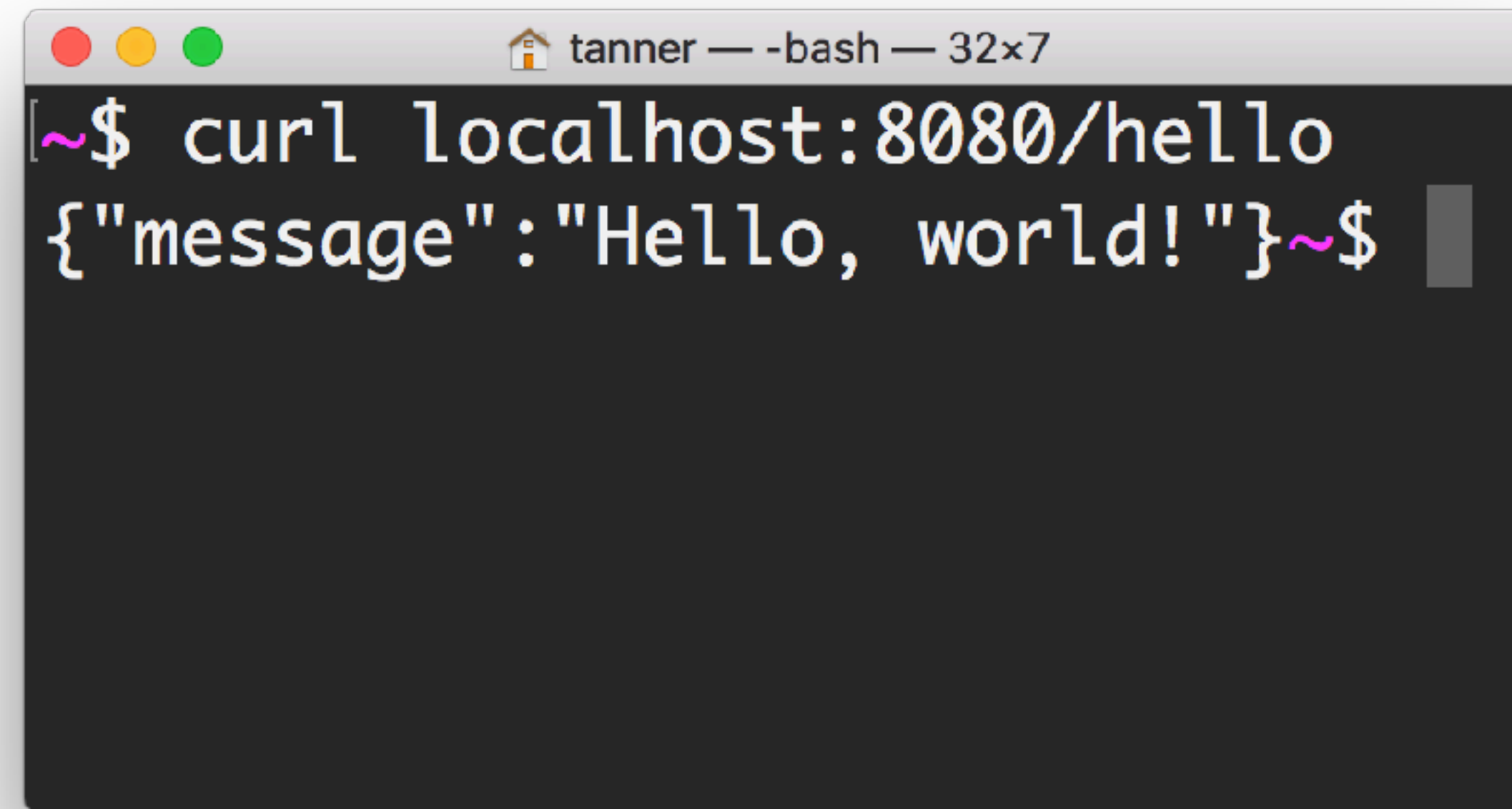
*Create APIs that are simple, expressive,
and maybe even fun.*


```
router.get("hello") { req in  
    return "Hello, world!"  
}
```

A terminal window with a dark background and a light gray title bar. The title bar contains three colored window control buttons (red, yellow, green) on the left, a home icon, and the text "tanner — -bash — 32x7". The terminal content shows a prompt "~\$" followed by the command "curl localhost:8080/hello". The output "Hello, world!" is displayed on the next line, followed by another prompt "~\$".

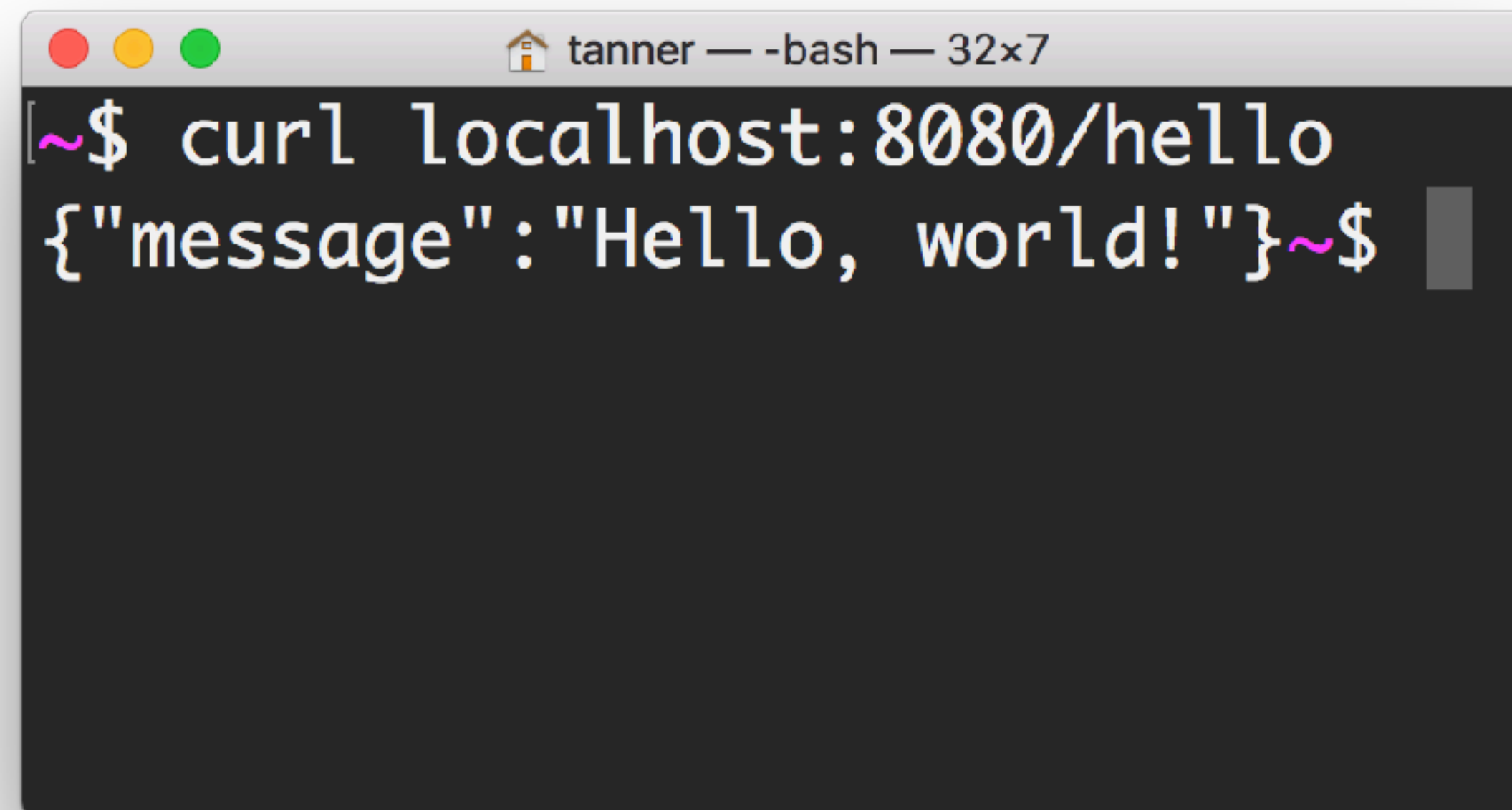
```
tanner — -bash — 32x7  
[~$ curl localhost:8080/hello  
Hello, world!~$
```

```
router.get("hello") { req in  
    return ["message": "Hello, world!"]  
}
```

A terminal window with a dark background and a light gray title bar. The title bar contains three colored window control buttons (red, yellow, green) on the left, a home icon followed by the text 'tanner — -bash — 32x7' in the center, and a close button on the right. The terminal content shows a command prompt '~\$' followed by the command 'curl localhost:8080/hello'. The output of the command is '{"message": "Hello, world!"}' followed by another '~\$' prompt. A gray cursor block is visible at the end of the output line.

```
tanner — -bash — 32x7  
[~$ curl localhost:8080/hello  
{"message": "Hello, world!"}~$
```

```
struct HelloResponse: Content {  
    var message: String  
}  
  
router.get("hello") { req in  
    return HelloResponse(message: "Hello, world!")  
}
```

A terminal window with a dark background and light text. The title bar shows a home icon, the name 'tanner', and the shell '-bash' with a window size of '32x7'. The prompt is '~\$'. The command 'curl localhost:8080/hello' has been executed, and the output is '{"message": "Hello, world!"}' followed by the prompt '~\$'.

```
tanner — -bash — 32x7  
[~$ curl localhost:8080/hello  
{"message": "Hello, world!"}~$
```

```
router.get("hello", String.parameter) { req in  
    let name = try req.parameter(String.self)  
    return ["message": "Hello, \(name)!"]  
}
```



```
app.get('/hello/:string', function (req, res) {  
    res.send({"message": "Hello, " + req.params["string"] + "!"})  
})
```

```
/// Controllers basic CRUD operations on `Todo`s.
final class TodoController {
    /// Returns a list of all `Todo`s.
    func index(_ req: Request) throws -> Future<[Todo]> {
        return Todo.query(on: req).all()
    }

    /// Saves a decoded `Todo` to the database.
    func create(_ req: Request) throws -> Future<Todo> {
        return try req.content.decode(Todo.self).flatMap(to: Todo.self) { todo in
            return todo.save(on: req)
        }
    }

    /// Deletes a parameterized `Todo`.
    func delete(_ req: Request) throws -> Future<HTTPStatus> {
        return try req.parameter(Todo.self).flatMap(to: Void.self) { todo in
            return todo.delete(on: req)
        }.transform(to: .ok)
    }
}
```

```
import Vapor
```

```
let app = try Application()
```

```
app.get("hello") { req in  
    return "Hello, world!"  
}
```



```
try app.run()
```

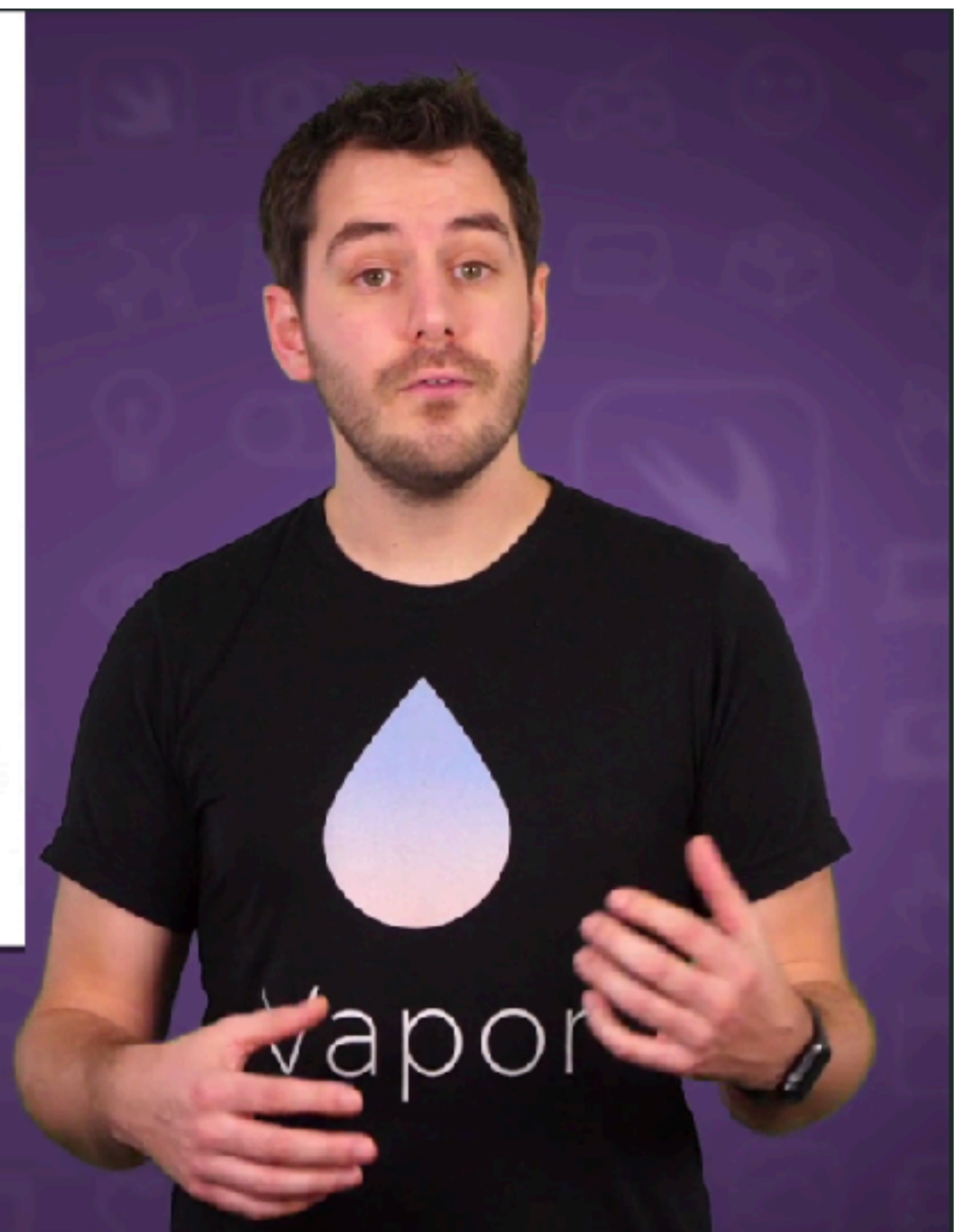



SERVER SIDE SWIFT

2015
Apple open-sourced Swift

Vapor is the
most popular





raywenderlich.com

Works With...



macOS



Ubuntu



Heroku



Digital Ocean



AWS



Docker



MySQL



SQLite



PostgreSQL



MongoDB



Redis



And More...



vapor
CLOUD

vapor.cloud



```
~ $ vapor new Hello
Creating Project "Hello" [Done]
~ $ cd Hello
~/Hello $ vapor cloud deploy --env=staging
Creating deployment [Done]
Building vapor [Done]
Creating container [Done]
Updating replicas [Done]
Deploy successful https://hello-staging.vapor.cloud
~ $
```

FREE

\$0 /MO

Try the power of Vapor Cloud,
free forever.

20,000[†] requests/month

32MB memory

128 millicore CPU[‡]

HOBBY

\$6 /MO

Small hobby/personal projects.

Unlimited requests

32MB memory

128 millicore CPU[‡]

SMALL

\$30 /MO

Startup with low amount of
requests.

Unlimited requests

256MB memory

256 millicore CPU[‡]

MEDIUM

\$65 /MO

Start getting some real traffic.

Unlimited requests

512MB memory

512 millicore CPU[‡]

LARGE

\$225 /MO

Now we're talking!

Unlimited requests

1GB memory

1 core CPU[‡]

X LARGE

\$375 /MO

Serious performance.

Unlimited requests

3GB memory

2 core CPU[‡]

Thank you!

vapor.codes

vapor.team

github.com/vapor/vapor

@codevapor