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**

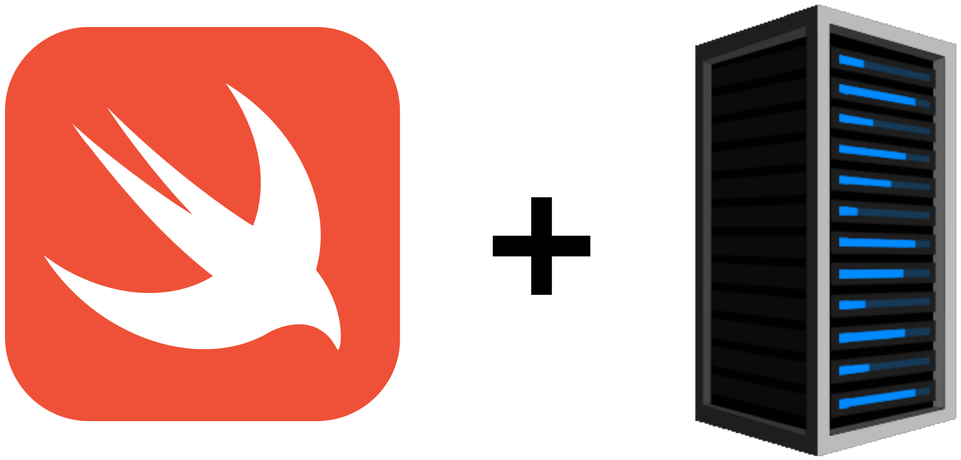
written in Swift

Brief History

0.1	1.0	2.0	3.0
Jan, 2016	Sep, 2016	May, 2017	April, 2018

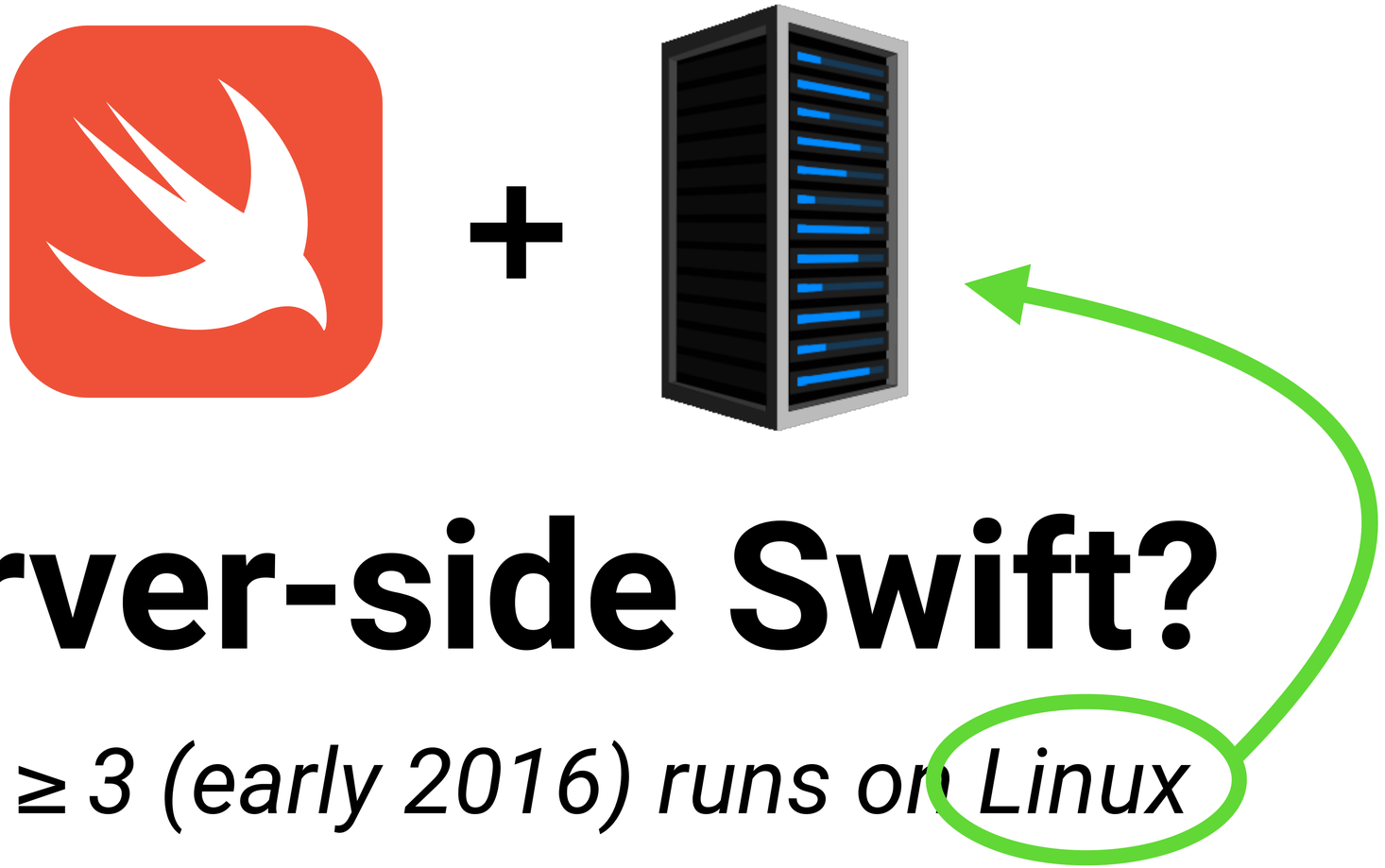
+

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

Frontend

Static Website

Dynamic Website

VAPOR

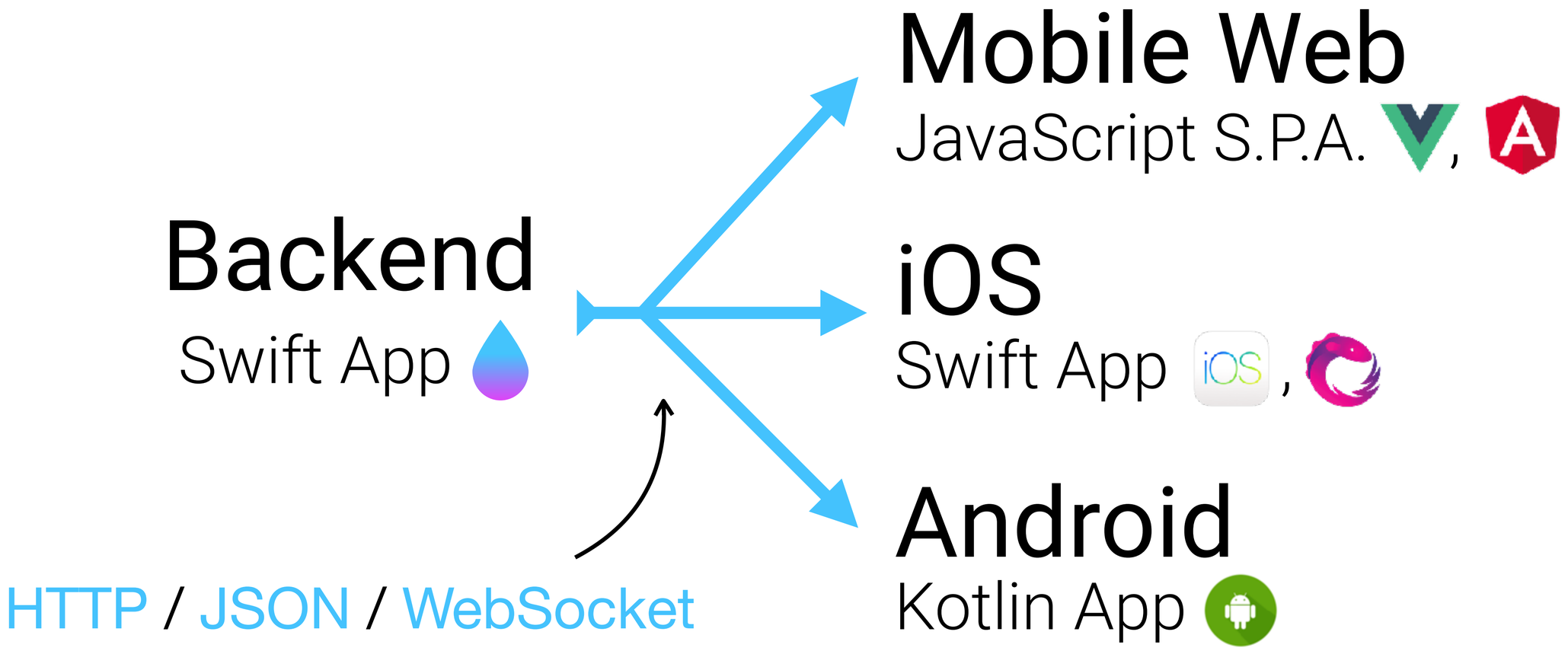
Backend

JSON API

WebSockets

CLI App

Swift Backend



⭐ Most popular on GitHub!

Ideology

Safety

Performance

Beautiful APIs

Ideology

Safety

Fail during development, not production

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

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

```swift
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()
    }
}
```

```swift
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()
    }
}
```

```swift
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

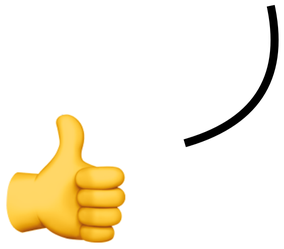
```swift
struct User: SQLiteModel {
    static let idKey = \User.id
    var id: UUID?
    var name: String
}

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

```swift
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()
    }
}
```

```swift
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

```swift
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.

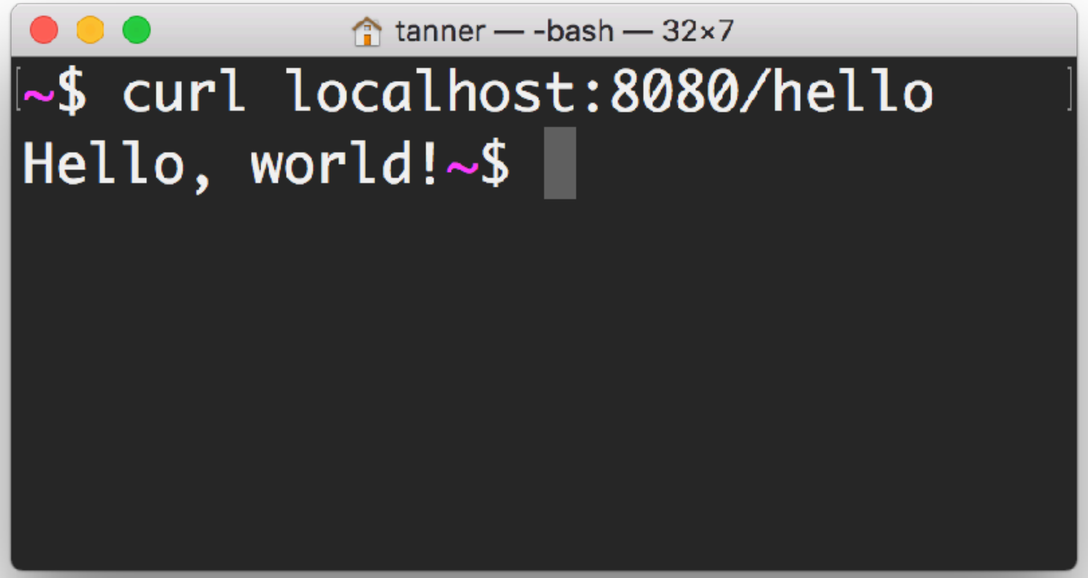
Vapor 3 (pre-NIO)	120k RPS	8MB
Gin (Go)	99k RPS	18MB
Vapor 3	95k RPS	6MB
Node	60k RPS	256MB
Laravel	~1k RPS	~2GB

💧 Ideology

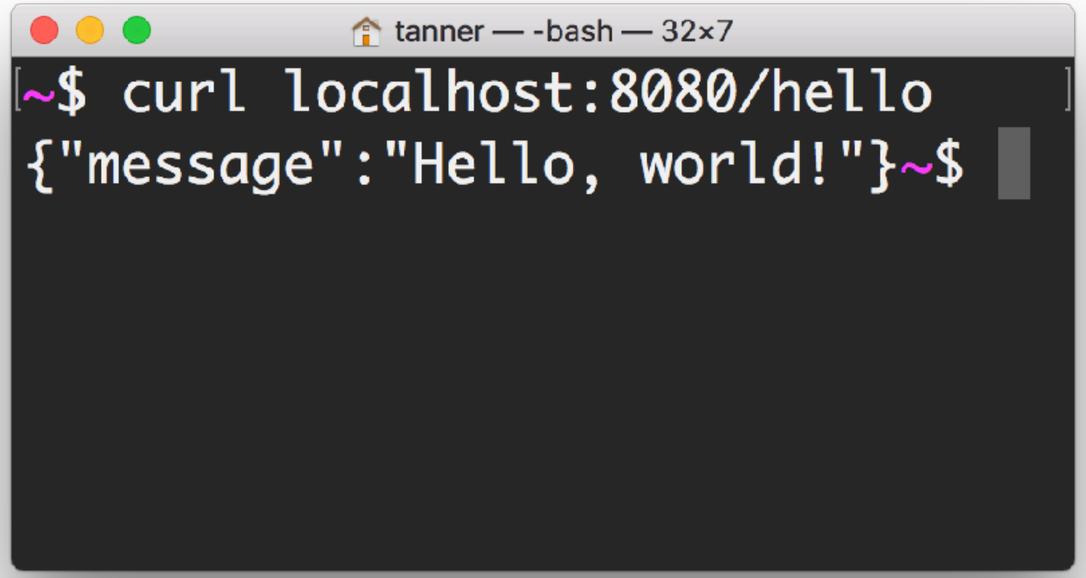
💖 Beautiful APIs

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

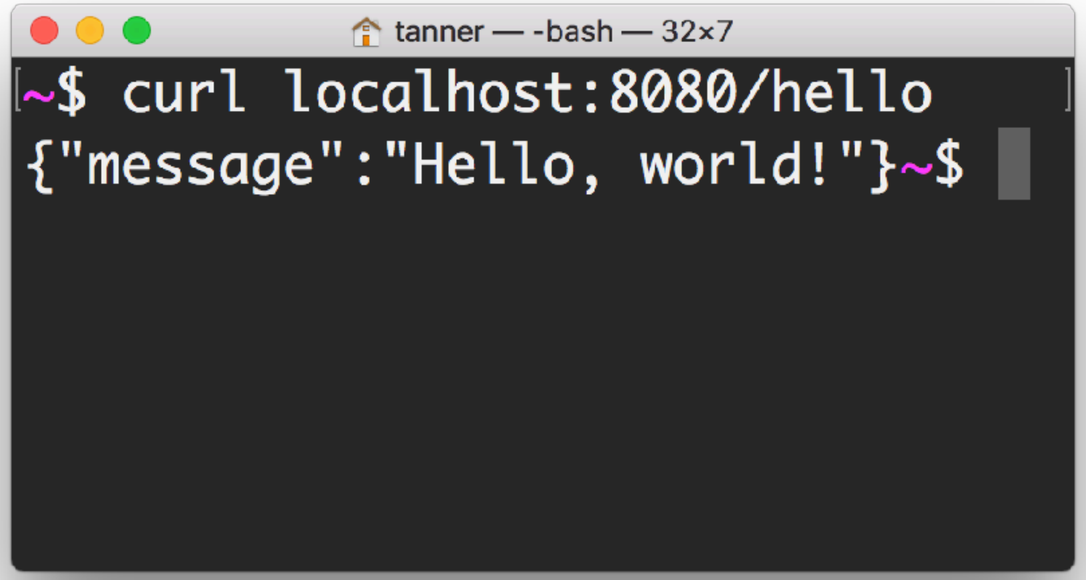
```
router.get("hello") { req in
    return "Hello, world!"
}
```



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



```swift
struct HelloResponse: Content {
    var message: String
}

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

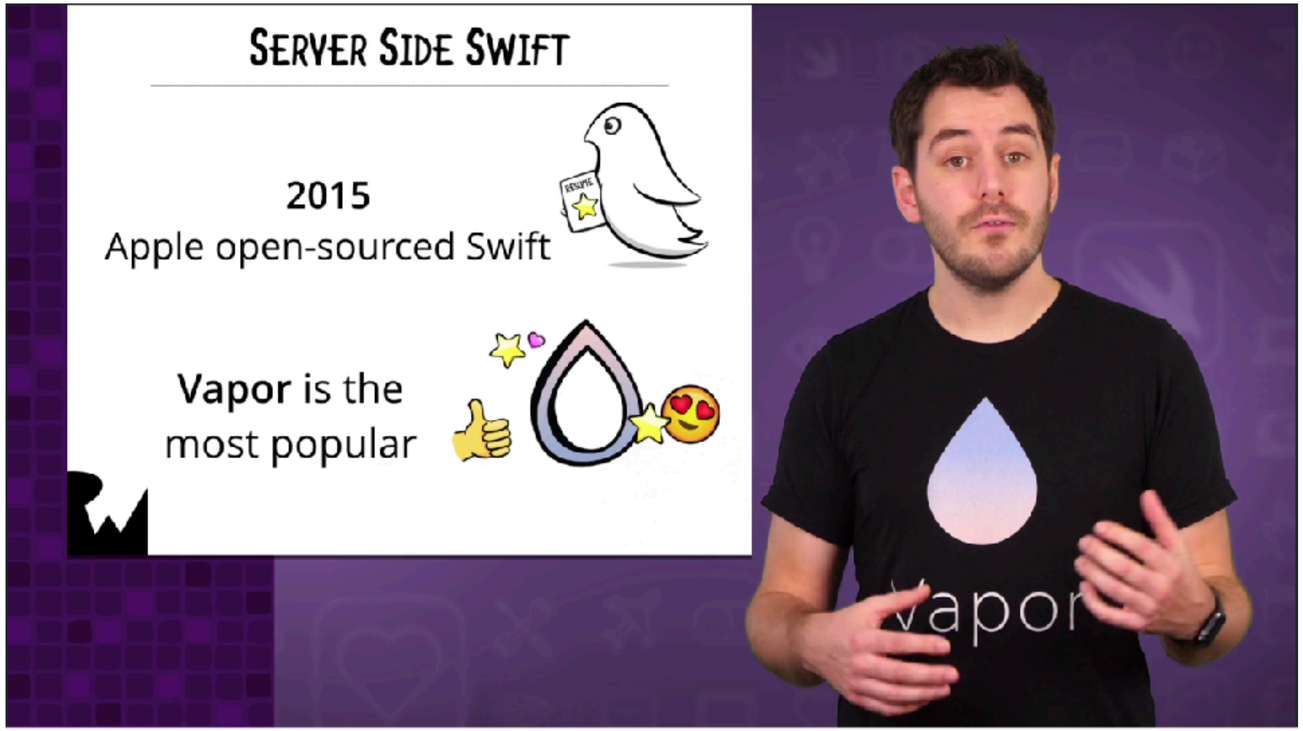


```
router.get("hello", String.parameter) { req in
    let name = try req.parameter(String.self)
    return ["message": "Hello, \(name)!"]
}
```


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

```swift
/// Controlers 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)
    }
}
```

```swift
import Vapor

let app = try Application()

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

try app.run()
```



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	HOBBY	SMALL	MEDIUM	LARGE	X LARGE
$0 /MO	$6 /MO	$30 /MO	$65 /MO	$225 /MO	$375 /MO
Try the power of Vapor Cloud, free forever.	Small hobby/personal projects.	Startup with low amount of requests.	Start getting some real traffic.	Now we're talking!	Serious performance.
20,000† requests/month	**Unlimited** requests	**Unlimited** requests	**Unlimited** requests	**Unlimited** requests	**Unlimited** requests
32MB memory	**32MB** memory	**256MB** memory	**512MB** memory	**1GB** memory	**3GB** memory
128 millicore CPU‡	**128 millicore** CPU‡	**256 millicore** CPU‡	**512 millicore** CPU‡	**1 core** CPU‡	**2 core** CPU‡

Thank you!

vapor.codes
vapor.team
github.com/vapor/vapor

@codevapor