💧 Vapor 3

What is Vapor?

- 👨‍👩‍👧‍👦 Communal
- 🚀 Performant
- ❤️ Developer Friendly
- 💪 Feature set
- 🔒 Secure
- ☁️ Cloud

Inspirations

reactive-streams.org

nodejs.org
dartlang.org

Swift 4 codable

🚀 Performance

- Async
- Reactive
- Lazy
- Copy on Write
- Buffer Reuse
- Custom EventLoop

Ecosystem	Async	Reactive	Lazy	CoW	Buffer Reuse
Vapor 3	X	X	X	X	X
Vapor 2			~	X	
ExpressJS	X				?
FastHTTP	X				X
Gin	X				X
Laravel					

You're only as good as your weakest link

Remove all weak links!

Estimated performance (Rounded down)

Name	Type	Req/sec	Avg. Responsetime	Memory
Vapor Engine	WebServer	103'000	1.03ms	6.4MB
Vapor 3	Framework	80'000	1.82ms	6.4MB
FastHTTP	WebServer	111'000	0.89ms	6.5MB
Gin	Framework	79'000	1.24ms	6.7MB

Tested in unprofessional environments

Performance Consistency

Reactiveness increases consistency by lowering CPU idle time and memory usage

- Custom EventLoops

❤️ Developer Friendly

Developer benchmarks

- Less code
- Less bugs
- More developer productivity

Proof (181 loc)

```swift
final class Tutorial: Model, NodeConvertible {
    var name: String
    var author: String
    var medium: Medium
    var image: String
    var url: String
    var description: String
    var duration: Int
    var difficulty: Difficulty
    var version: String
    var storage: Storage

    init(node: Node) throws {
        name = try node.get("name")
        author = try node.get("author")
        medium = try Medium(node: node.get("medium"))
        image = node["image"]?.string ?? "sample-tile.png"
        url = try node.get("url")
        description = try node.get("description")
        duration = try node.get("duration")
        difficulty = try Difficulty(node: node.get("difficulty"))
        version = try node.get("version")
        self.storage = Storage()
        id = try node.get("id")
    }
}

enum Difficulty: NodeConvertible {
    case easy, intermediate, advanced

    init(node: Node) throws {
        guard let difficulty = node.string else {
            throw Error.databaseParseError("Difficulty was not a string.")
        }

        switch difficulty {
        case "easy":
            self = .easy
        case "intermediate":
            self = .intermediate
        case "advanced":
            self = .advanced
        default:
            throw Error.databaseParseError("Difficulty was an invalid type: \(difficulty).")
        }
    }

    func makeNode(in context: Context?) throws -> Node {
        switch self {
        case .easy:
            return "easy"
        case .intermediate:
            return "intermediate"
        case .advanced:
            return "advanced"
        }
    }
}

enum Medium: NodeConvertible {
    case video, article

    init(_ string: String) throws {
        switch string {
        case "video":
            self = .video
        case "article":
            self = .article
        default:
            throw Error.databaseParseError("Medium was an invalid type: \(string).")
        }
    }

    init(node: Node) throws {
        guard let string = node.string else {
            throw Error.databaseParseError("Medium was not a string.")
        }

        self = try .init(string)
    }

    func makeNode(in context: Context?) throws -> Node {
        switch self {
        case .article:
            return "article"
        case .video:
            return "video"
        }
    }

    class Validator: Validation.Validator {
        enum Error: Swift.Error {
            case invalid
        }

        func validate(_ input: String) throws {
            do {
                _ = try Medium(input)
            } catch {
                throw Error.invalid
            }
        }

        class Middleware: HTTP.Middleware {
            func respond(to request: Request, chainingTo next: Responder) throws -> Response {
                do {
                    return try next.respond(to: request)
                } catch Error.invalid {
                    throw Abort(.badRequest, reason: "Invalid medium. Must be either video or article.")
                }
            }
        }
    }
}

extension Tutorial {
    convenience init(row: Row) throws {
        try self.init(node: row.makeNode(in: Row.defaultContext))
    }

    func makeRow() throws -> Row {
        return try makeNode(in: Row.defaultContext).converted()
    }
}

// MARK: Fluent Serialization
extension Tutorial {
    func makeNode(in context: Context?) throws -> Node {
        return try Node(node: [
            "id": id ?? nil,
            "name": name,
            "author": author,
            "medium": medium,
            "image": image,
            "url": url,
            "description": description,
            "duration": duration,
            "difficulty": difficulty,
            "version": version
        ])
    }
}

// MARK: Preparations
extension Tutorial: Preparation {
    static func prepare(_ database: Database) throws {
        try database.create(self) { tutorials in
            tutorials.id()
            tutorials.string("name")
            tutorials.string("medium")
            tutorials.string("image")
            tutorials.string("url")
            tutorials.string("description")
            tutorials.int("duration")
            tutorials.string("difficulty")
        }
    }

    static func revert(_ database: Database) throws {
        try database.delete(self)
    }
}

extension Tutorial: JSONRepresentable {
    func makeJSON() throws -> JSON {
        return try makeNode(in: JSON.defaultContext).converted()
    }
}

extension Tutorial: ResponseRepresentable {

}
```

Improvements (25 loc)

```
enum Difficulty: String, Codable, KeyStringDecodable {
    case easy, intermediate, advanced

    static var keyStringTrue: Difficulty { return .easy }
    static var keyStringFalse: Difficulty { return .advanced }
}

enum Medium: NodeConvertible: String, Codable, KeyStringDecodable {
    case video, article

    static var keyStringTrue: Difficulty { return .video }
    static var keyStringFalse: Difficulty { return .article }
}

final class Tutorial: Model, Content {
    var name: String
    var author: String
    var medium: Medium
    var image: String
    var url: String
    var description: String
    var duration: Int
    var difficulty: Difficulty
    var version: String
}
```

Services (Configurable)

- Dependency inversion
- Protocols over concrete types
- Register factories
- Request instances by conformance
- Inject custom implementations

```
router.get("users") { req in
    return User.query(on: req).all()
}
```

More thorough testing of criticial systems

```
struct TestClient: Client {
    let container: Container
    let client: Client

    init(container: Container) throws {
        self.container = container
        self.client = try container.make()
    }

    /// Returns a future response for the supplied request.
    func respond(to req: Request) throws -> Future<Response> {
        return try client.respond(to: req).map(to: Response.self) { response in
            XCTAssert(response.status == 200)

            return response
        }
    }
}
```

Flexible, Integrated Streams

- Most components are streams
- Streams are chained easily
- One implementation for all use cases

```
// Connect to the SOCKS 5 host
let tcpClient = try TCPClient(socket: TCPSocket())
try tcpClient.connect(hostname: "socks-host", port: 5050)

// Connect TCP to SOCKS
let tcpStream = tcpClient.socket.stream(on: self.container),
let proxyStream = SocksProxy(over: tcpStream)

// Connect to MongoDB over the socks proxy
let mongodb = MongoKitten.DatabaseConnection(stream: proxyStream)
```

💪 Feature set

HTTP support

- HTTP/1.1
- WebSockets
- SSL (HTTPS)

🎉 In development

- HTTP/2
- GZIP

📝 Next level templating

Leaf & Mustache

```
router.get("users") { req in
    return req.view().render("users.leaf", [
        "users": User.query(on: req)
    ])
}
```

📖 Wide Database support

- MongoDB
- MySQL
- PostgreSQL
- Redis
- SQLite
- Kafka

🔋 Batteries included

- JWT
- Auth
- Codable ORM
- SMTP
- UDP
- Certificate Management

🔒 Secure

Strong DDoS resistance

Attack	Solution
DDoS	Hooks for custom protections
DoS	Rate limit IPs
Slow Read	Reactive streams
Slow Write	"
Memory Attack	"

☁️ Easy deployment

```
$ vapor cloud deploy
```

- Databases
- Upcoming integrations with Vapor 3
- Free to test out
- Powerful CLI

Framework Roadmap

- Reactive FileSystem APIs
- HTTP/2
- Certificate Management
- Security Modules
- Better MongoDB 3.6 support
- Better PostgreSQL support
- GZIP compression

👨‍👩‍👧‍👦 Community

- http://vapor.team
- 5000 slack members
- Friendly discussions
- Active members