Vapor Workshop

Please have Homebrew, and Xcode >= 14.3 installed

Let me know if you don't!

BROKENHANDS

Vapor 4 Workshop

Tim Condon

BROKENHANDS

Introduction

- Founder of Broken Hands 
- Vapor Core Team 
- SSWG and SWWG Member 
- Server-side Swift Team @ Kodeco 
- `@0xTim`    
- Organise ServerSide.swift, NSManchester, Vapor London

Getting Started

```
$ brew install vapor
```

```
OR
$ brew upgrade vapor
```

```
$ vapor new HelloVapor
```

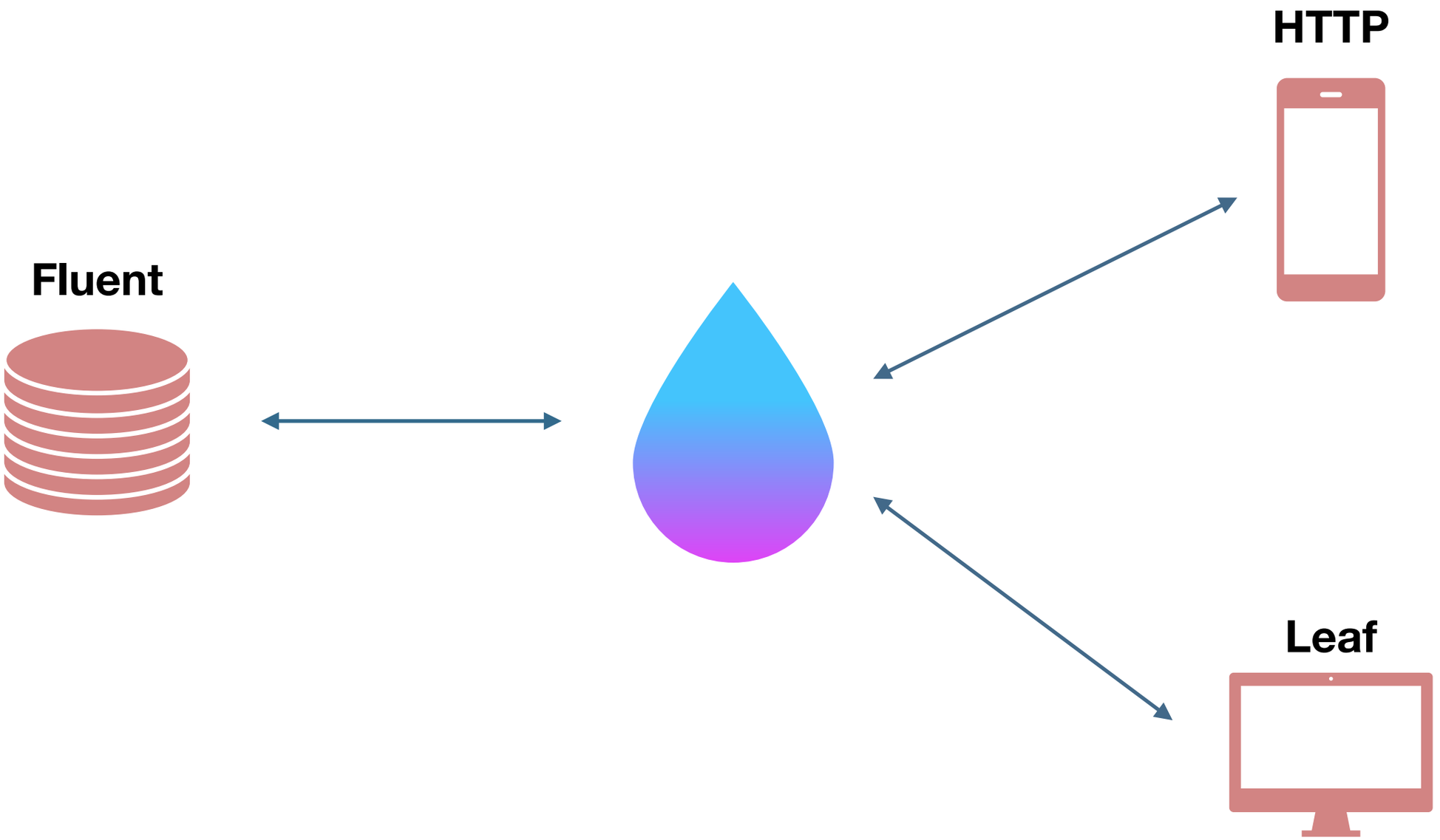
Yes to a Fluent, choose SQLite, no to Leaf

Xcode

```
$ cd HelloVapor
$ open Package.swift
```

It will take some time whilst the dependencies download before any schemes appear

What Is Vapor?



Vapor

- Open-source web framework created by Tanner Nelson and Logan Wright
- Built on top of SwiftNIO
- Makes use of latest Swift features and Swift-like API
- Most popular Swift framework
- Awesome community

Why Vapor?

- Happier developers
- You get a compiler! 🤯
- Same tooling as iOS
- Debugging support in Xcode

Why Vapor?



- Completely non-blocking (including the database drivers)
- More requests per second = less servers
- Small memory footprint ~ 6MB
- Less memory = less servers
- Safer

Hello Workshop

```swift
// routes.swift
//
// http://localhost:8080/hello/workshop

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

Make sure you select the Run scheme and My Mac destination

Parameters

```
// routes.swift
//
// http://localhost:8080/bottles/99

app.get("bottles", ":count") { req -> String in
  let count =
    try req.parameters.require("count", as: Int.self)
  return "There were \(count) bottles on the wall"
}
```


Challenge - Parameters

```
// Create a String parameter route to say hello to anyone
//
// http://localhost:8080/hello/Tim
//
// returns "Hello Tim"
```


Challenge - Parameters

```
// http://localhost:8080/hello/Tim

app.get("hello", ":name") { req -> String in
    let name = try req.parameters.require("name")
    return "Hello \(name)"
}
```


Returning JSON

```
// routes.swift
//
// http://localhost:8080/bottles/99

app.get("bottles", ":count") { req -> Bottles in
  let count =
    try req.parameters.require("count", as: Int.self)
  return Bottles(count: count)
}

struct Bottles: Content {
  let count: Int
}
```


Accepting JSON

```
// routes.swift
//
// POST http://localhost:8080/bottles/
//
// {
//    “count”: 99
// }
//

app.post("bottles") { req -> String in
    let bottles = try req.content.decode(Bottles.self)
    return "There were \(bottles.count) bottles"
}
```


Challenge - Accept and Return JSON

```
// POST http://localhost:8080/user-info
// {
//    “name”: “Tim”,
//    “age”: 99
// }
//
// Return
// {
//    “message”: “Hello Tim, you are 99”
// }
```


Challenge - Accept and Return JSON

```
struct UserInfo: Content {
    let name: String
    let age: Int
}

struct UserMessage: Content {
    let message: String
}

app.post("user-info") { req -> UserMessage in
    let userInfo = try req.content.decode(UserInfo.self)
    let message =
        "Hello \(userInfo.name), you are \(userInfo.age)"
    return UserMessage(message: message)
}
```


Async

BROKENHANDS

Async Motivations

- Vapor 4 built on `EventLoopFutures`
- `async/await` introduced with macOS 12
- Simplifies everything!

Fluent

BROKENHANDS

Create a Class

```swift
// Inside App/Models create User.swift

import Fluent
import Vapor

final class User: Model, Content {
    static let schema = "users"

    @ID var id: UUID?
    @Field(key: "name") var name: String
    @Field(key: "username") var username: String

    init() {}
    init(id: UUID? = nil, name: String, username: String) {
        self.id = id
        self.name = name
        self.username = username
    }
}
```


Migrations

```swift
// Inside App/Migrations create CreateUser.swift

import Fluent

struct CreateUser: AsyncMigration {
    func prepare(on database: Database) async throws {
        try await database.schema("users")
            .id()
            .field("name", .string, .required)
            .field("username", .string, .required)
            .create()
    }

    func revert(on database: Database) async throws {
        try await database.schema("users").delete()
    }
}
```


Registering Migrations

```
// configure.swift
app.migrations.add(CreateUser())

try await app.autoMigrate()
```


Create a Controller

```swift
// Inside App/Controllers create UsersController.swift

import Vapor
import Fluent

struct UserController: RouteCollection {

    func boot(routes: RoutesBuilder) throws {

    }
}

// In routes.swift routes(_:)
try app.register(collection: UserController())
```


Creating a User

```
// POST http://localhost:8080/api/users/
//
// {
//    “name”: “Tim”,
//    “username”: “timc”
// }
//
// In UsersController.swift

func boot(routes: RoutesBuilder) throws {
  routes.post("api", "users", use: createHandler)
}

func createHandler(req: Request) async throws -> User {
  let user = try req.content.decode(User.self)
  try await user.save(on: req.db)
  return user
}
```


Getting all of the users

```
// GET http://localhost:8080/api/users/

func boot(routes: RoutesBuilder) throws {
  // ...
  routes.get("api", "users", use: getAllHandler)
}

func getAllHandler(req: Request) async throws -> [User] {
  try await User.query(on: req.db).all()
}
```


Challenge - Get a Single User

```
// GET http://localhost:8080/api/users/<ID>
//
// HINT 1:

User.find(_:on:)

// HINT 2:
throw Abort(.notFound)
```


Challenge - Get a Single User

```
func boot(routes: RoutesBuilder) throws {
  // ...
  routes.get("api", "users", ":userID", use: getHandler)
}

func getHandler(req: Request) async throws -> User {
  guard let user = try await User.find(
    req.parameters.get("userID"),
    on: req.db) else {
        throw Abort(.notFound)
  }
  return user
}
```


Deleting a User

```
// DELETE http://localhost:8080/api/users/<ID>

func boot(routes: RoutesBuilder) throws {
  // ...
  routes.delete("api", "users", ":userID", use: deleteHandler)
}

func deleteHandler(req: Request) async throws -> HTTPStatus {
  guard let user = try await User.find(
    req.parameters.get("userID"),
    on: req.db) else {
      throw Abort(.notFound)
  }
  try await user.delete(on: req.db)
  return .ok
}
```


Updating a User

```
// PUT http://localhost:8080/api/users/<ID>
//
// {
//     “name”: “Tim”
//     “username”: “timc”
// }

func boot(routes: RoutesBuilder) throws {
    // ...
    routes.put("api", "users", ":userID", use: updateHandler)
}

func updateHandler(req: Request) async throws -> User {
    let updatedUser = try req.content.decode(User.self)
    guard let user = try await User.find(
      req.parameters.get("userID"),
      on: req.db) else {
        throw Abort(.notFound)
    }
    user.name = updatedUser.name
    user.username = updatedUser.username
    try await user.save(on: req.db)
    return user
}
```


Route Groups

```swift
// UsersController.swift

func boot(routes: RoutesBuilder) throws {
    let usersRoutes = routes.grouped("api", "users")
    usersRoutes.post(use: createHandler)
    usersRoutes.get(use: getAllHandler)
    usersRoutes.get(":userID", use: getHandler)
    usersRoutes.delete(":userID", use: deleteHandler)
    usersRoutes.put(":userID", use: updateHandler)
}
```


Where to Go From Here?

BROKENHANDS

Code is available here:
https://github.com/brokenhandsio/VaporWorkshops

Feedback
https://bit.ly/vapor-workshop

Questions/ Comments/Queries

tim@brokenhands.io
Twitter: @0xTim

BROKENHANDS