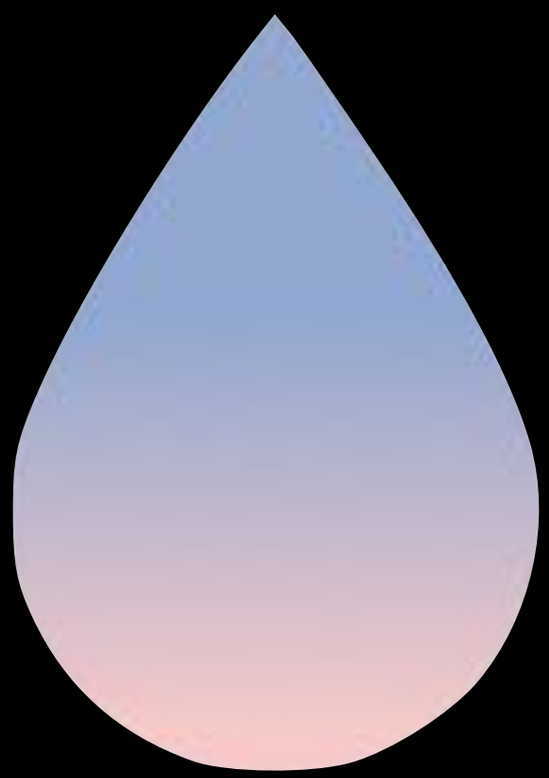
Making web apps in swift using Vapor 2



About me

Life: Alternate Reality Productivity

LimeTray TAP white-label B2C products on iOS
LimeTray Pulse B2B dashboard on iOS

Vapor

- Most used web framework in swift
- Incredibly swifty
- By Tanner Nelson + Logan Wright + community.
- v1.0 in sept’16 and v2.0 in may’17
- Absolutely awesome community in terms of support and contribution

A web framework?! What?

Respond to requests using **Vapor** and for routing, sessions, cookies, middleware. **Engine** works on network protocols and URI. Manage databases using **Fluent** including **NoSQL** & **SQL** databases. Interface **MySQL.** Exhaustive data-type handling using **Node.** **Leaf** web page templating. Pure swift libraries for **Validation**, **JWT**, **Redis**, **JSON, multipart, crypto**, **bcrypt**. **Console** handles commands and arguments. SSL and TLS using **TLS. auth** for authentication. Extend functionalities using **Providers**. + awesome **community providers**.

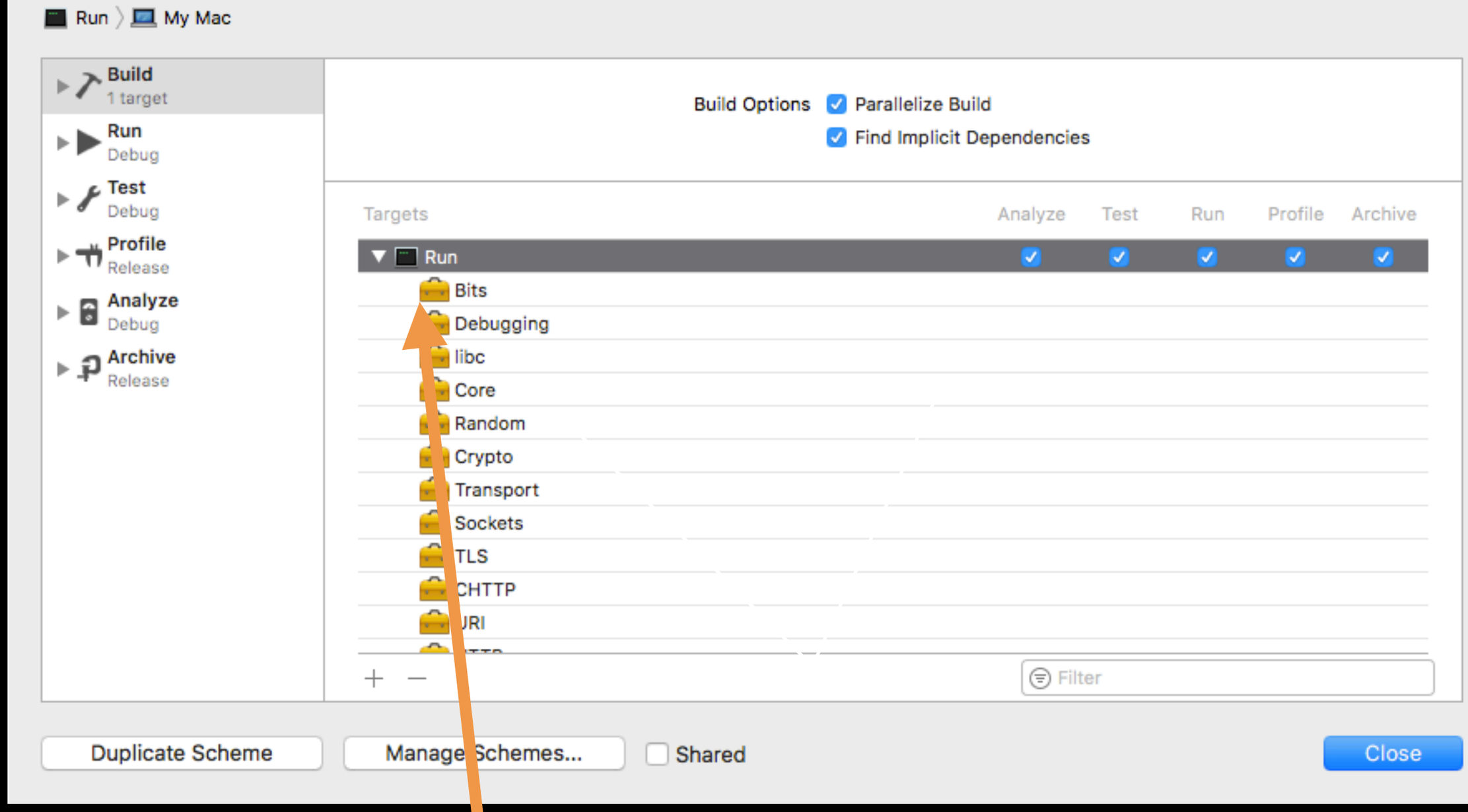
Swift and command line can be best friends

macOS/Linux(Ubuntu)

Access to Foundation, libdispatch, XCTest.

Swift Package manager

- Create packages and set interdependencies.
- Create libraries and executable binaries.
- Add dependencies to a `package.swift` manifest file in root of your repo.
- Simply call `swift build` to resolve dependencies in the manifest file and build packages.



Your vapor app is just an SPM executable backed by SPM libraries.

A droplet of Vapor

- Initialise your server with an instance of `droplet`.
- Provide custom config in code or through static jsons or command line arguments
- Configs are chosen priority wise. In decreasing order of priority: CLI > Config/secrets/ > Config/name-of-environment/ > Config/

- Optionally provide the details like host address, port, `Server`, `Client`, `Router`, `Middleware`s so it is customised and ready for your specific use cases.
- Calling `setup()` on `Config` is last chance to make changes and provide providers to your config

Setting your routes

- Set routes a droplet is going to observe in the implementation of **`setup()`** function
- Example: to listen to /hello route in your URL set your HelloController to listen to ‘hello’ route

```
/// GET /hello/...
builder.resource("hello", HelloController(view))
```

- Provide closure handler to droplet for specific routes.

- Basic closures that handle routes can have the signature of: `typealias RouteHandler = (Request) throws -> ResponseRepresentable`

- When you are ready, just `run()`

```
let config = try Config()
try config.setup()

let drop = try Droplet(config)
try drop.setup()
```

Template your response HTML

- Data needs to be provided by this controller to your view following the MVC pattern.
- Leaf builds HTML using the template and escaped variables.
- `droplet`'s `viewrenderer` can build views using base template and controller data

- Leaf uses special DSL made of tags to access the data passed by the controller

- to access a key named ‘name’ simply call #(name) in leaf file

```
#extend("base")

#export("title") { Hello, #(name)! }

#export("content") {
    <h1>Hello, #(name)!</h1>
}
```

Demo

Community contributions:
postgresql driver https://github.com/vapor-community/postgresql
MongoDB driver https://github.com/OpenKitten/MongoKitten
matthijs2704/Vapor-apns https://github.com/matthijs2704/vapor-apns
swiftybeaver-provider https://github.com/SwiftyBeaver/SwiftyBeaver-Vapor

Official
https://github.com/vapor/vapor
https://docs.vapor.codes/

Today's demo:
https://github.com/ishaanSejwal/vaporMeetupDemo

Ishaan Sejwal
ishaan.sejwal@gmail.com

Thanks for the attention 😀