



THE UNIVERSITY OF  
CHICAGO



MPCS 51033 • SPRING 2017 • SESSION 7

---

# BACKENDS FOR MOBILE APPLICATIONS

# CLASS NEWS

## CLASS NEWS

- No Office hours Thursday
- Make-up class Saturday, May 20, 9AM-12PM in Young 101

## CLASS NEWS

- Week 6 - Assignment 4 (CloudKit)
- Week 7 - Finish assignment 4 and work on case study and final project
- Week 8 - Assignment 5 Assigned (Swift on Server)
- Week 9 - Case Studies in Class; Assignment 5 Due
- Week 10 -
- Week 11 - Final Project Presentations

# OPEN SOURCE SWIFT

# OPEN SOURCE SWIFT

Dec 3, 2015

## Swift is Open Source

Swift is now open source. Today Apple launched the open source Swift community, as well as amazing new tools and resources including:

- [Swift.org](#) – a site dedicated to the open source Swift community
- Public source code repositories at [github.com/apple](#)
- A new Swift package manager project for easily sharing and building code
- A Swift-native core libraries project with higher-level functionality above the standard library
- Platform support for all Apple platforms as well as Linux

Now anyone can download the code and in-development builds to see what the team is up to. More advanced developers interested in contributing to the project can file bugs, participate in the community, and contribute their own fixes and enhancements to make Swift even better. For production App Store development you should always use the stable releases of Swift included in Xcode, and this remains a requirement for app submission.

- It all started here

## Swift.org

Swift.org is an entirely new site dedicated to open source Swift. This site hosts resources for the community of developers that want to help evolve Swift, contribute fixes, and most importantly, interact with each other. Swift.org hosts:

- A bug reporting and tracking system
- Mailing lists
- A blog dedicated to the engineering of Swift
- Community guidelines
- Getting started tutorials
- Contributing instructions
- Documentation on Swift

© 2015 Apple Inc. All rights reserved.

## OPEN SOURCE SWIFT

- The promise
  - Same code runs in both places
  - Reduce development time by sharing code
  - Leverage (some) frameworks and APIs



# OPEN SOURCE SWIFT



- Something is missing...

# OPEN SOURCE SWIFT

- Linux and Mac Platform support
- Standard Library Foundation, Dispatch, and XCTest Compiler
- Command Line Tools



# Swift

ABOUT SWIFT

BLOG

DOWNLOAD

GETTING STARTED

DOCUMENTATION

SOURCE CODE

COMMUNITY

CONTRIBUTING

CONTINUOUS  
INTEGRATION

PROJECTS

COMPILER AND  
STANDARD LIBRARY

PACKAGE MANAGER

CORE LIBRARIES

REPL AND DEBUGGER

# Welcome to Swi

Swift is now open source!

We are excited by this new chapter in the story of Swift. When Apple unveiled the Swift programming language, it quickly became one of the fastest growing languages in history. Swift is now open source software that is incredibly fast and safe by design. As Swift becomes open source, you can help make the best general-purpose programming language available everywhere.

For students, learning Swift has been a great introduction to modern programming concepts and best practices. Now that Swift is open source, their Swift skills will be able to be applied to a broader range of platforms, from mobile devices to the web to the cloud.

Welcome to the Swift community. Together we can continue to build a better programming language for everyone.

– **The Swift Team**

# STATE OF SWIFT ON THE SERVER

## STATE OF SWIFT ON THE SERVER

- Many Swift web frameworks in development
- Most popular (15,000+ stars)



# STATE OF SWIFT ON THE SERVER

- Perfect
  - A complete framework
  - 11,448 ★
  - Rails inspired

The screenshot shows the homepage of perfect.org. At the top, there is a navigation bar with links for "WHAT IS PERFECT?", "DOCUMENTATION", "DEVELOPER RESOURCES", and "EVENTS". Social media icons for LinkedIn, Twitter, Facebook, YouTube, and GitHub are also present, along with a search icon. Below the navigation, there are four main sections: "Get Started" (with an orange bird icon), "Documentation" (with an orange open book icon), "Assistant" (with an orange laptop icon), and "Commercial" (with an orange hexagonal icon). Each section has a brief description and a call-to-action button at the bottom. A large orange banner at the bottom features a play button icon and the text "TAKE ME TO THE LEARNING CENTRE". Another banner to the right encourages users to "Show me Ray Wenderlich's video tutorials >>". The central part of the page contains the heading "What is Perfect?" and a brief description of the service.

perfect.org

WHAT IS PERFECT? DOCUMENTATION DEVELOPER RESOURCES EVENTS

Get Started Documentation Assistant Commercial

Access Perfect's feature set and start your project.

Get support from our comprehensive set of documentation.

Meet the Perfect Assistant to help you do more with Perfect and Swift, more easily.

Use Perfect to build apps for business.

Get Started >> Documentation >> Meet the Assistant >> Find Out How >>

TAKE ME TO THE LEARNING CENTRE

Show me Ray Wenderlich's video tutorials >>

## What is Perfect?

Perfect is a web server and toolkit for developers using the Swift programming language to build applications and other REST services. It lets developers build using only Swift to program both the client-facing and server-side of their

# STATE OF SWIFT ON THE SERVER

- Perfect
  - A complete framework
  - Runs its own server or as a fastCGI module for Apache or NGINX

```
import PerfectLib

public func PerfectServerModuleInit() {
    Routing.Handler.registerGlobally()

    Routing.Routes["/] = { _ in return HelloWorldHandler() }

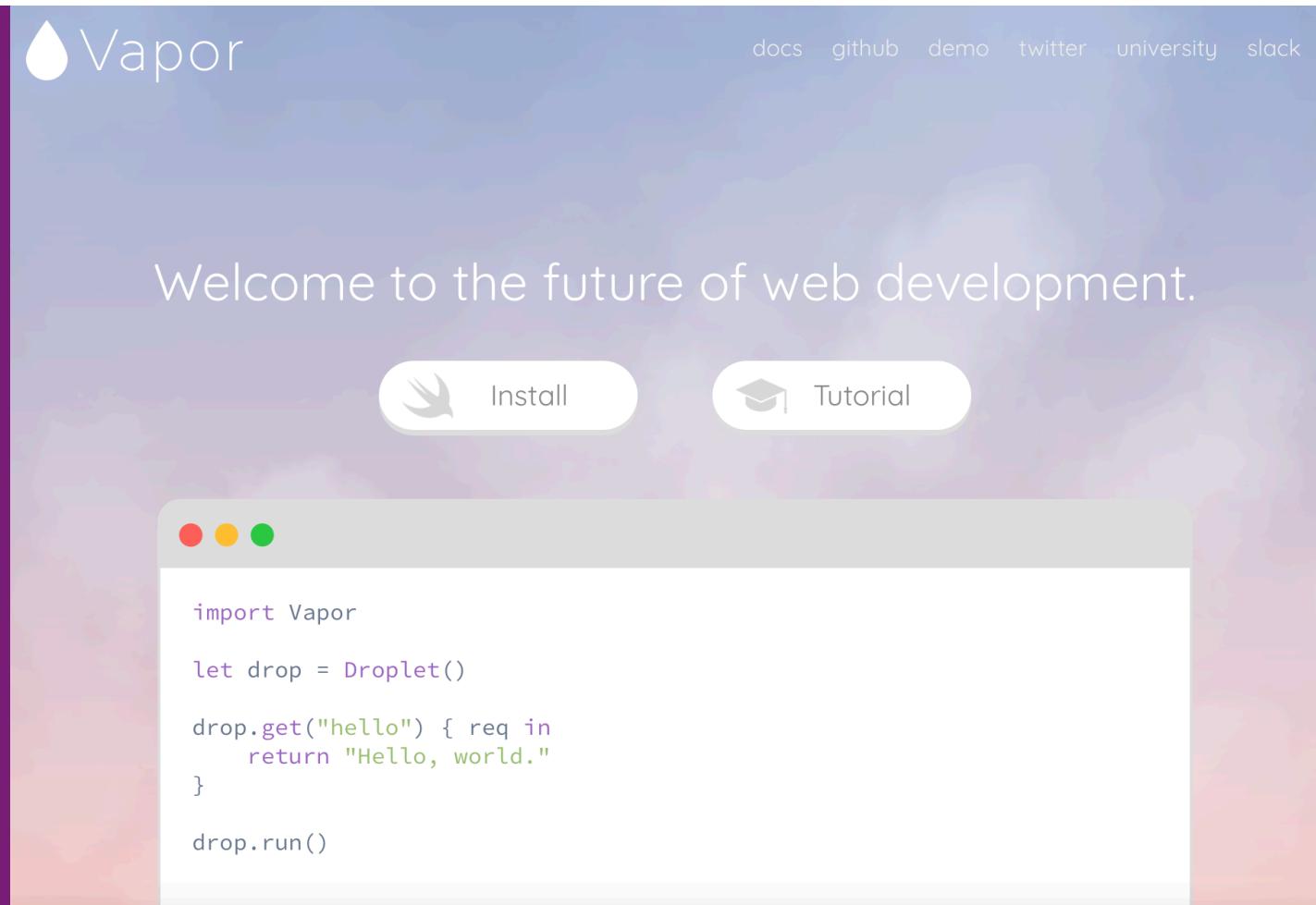
}

class HelloWorldHandler: RequestHandler {
    func handleRequest(request: WebRequest, response: WebResponse) {
        response.appendBodyString("Hello, World!")
        response.requestCompletedCallback()
    }
}
```

# STATE OF SWIFT ON THE SERVER

- Vapor

- 9,479 ★
- Laravel (php)  
inspired



The image shows the Vapor website homepage. At the top left is the Vapor logo (a white water droplet icon) followed by the word "Vapor". To the right are links for "docs", "github", "demo", "twitter", "university", and "slack". Below the header is a large, semi-transparent text box containing the slogan "Welcome to the future of web development." In the center of this box are two buttons: "Install" with a bird icon and "Tutorial" with a graduation cap icon. At the bottom of the page is a code editor window displaying Swift code:

```
import Vapor

let drop = Droplet()

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

drop.run()
```



## STATE OF SWIFT ON THE SERVER

- Perfect
  - A complete framework
  - Runs its own server or as a fastCGI module for Apache or NGINX

```
import Vapor

let app = Application()

app.get("/") { request in
    return "Hello, World!"
}

app.start()
```

# STATE OF SWIFT ON THE SERVER

- Kitura
  - Swift@IBM
  - 5,686 ★
  - Express.js inspired

The image shows a screenshot of the Kitura website. At the top left is the Kitura logo (a blue bird icon) and the word "Kitura". To its right are links for "Getting started", "Tutorials", "API reference", and "Support". Below this is a dark banner featuring a hand typing on a laptop keyboard. On the left side of the banner, there's a blurred view of a person working at a desk. The main content area displays a screenshot of a Mac OS X desktop with a "Cloud Runtimes" window open. The window lists three projects: "Kitura-Runtime" (status: Running), "Bluepic-Sample-Runtime" (status: Running), and "ToDo-Server" (status: Running). Each project entry includes a "Deploy" button and other deployment options. A large Kitura logo is overlaid on the right side of the screenshot. Below the screenshot, a blue button with white text reads "Get started with Kitura".

A high performance and simple to use web framework for building modern Swift applications.

Get started with Kitura

# STATE OF SWIFT ON THE SERVER

- Kitura
  - Swift@IBM
  - 5,686 ★

```
import Kitura

let router = Router()

router.get("/") { request, response, next in
    response.send("Hello, World!")
    next()
}

Kitura.addHTTPServer(onPort: 8090, with: router)
Kitura.run()
```

# STATE OF SWIFT ON THE SERVER

- Server Side Swift Standards (S4) project
- Protocols and standards

open-swift / S4

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs

HTTP standards for Swift

264 commits 2 branches 27 releases 12 contributors

Branch: master New pull request Create new file Upload files Find file Clone

noppoMan committed with paulofaria Updating to September 10, 2016 (#78) Latest commit a544ad7 on

File	Last Commit	Author(s)
Sources	Updating to September 10, 2016 (#78)	8
Tests	Updating to August 4, 2016	9
.gitignore	Update to C7 0.4.0	
.swift-version	Updating to September 10, 2016 (#78)	8
.travis.yml	Updated Swift to 07-25 (#74)	10
LICENSE	initial setup	
Package.swift	Updating to August 4, 2016	9
README.md	Updated Swift to 07-25 (#74)	10
S4.podspec	add cocoapods support	

README.md

## S4 - HTTP

## STATE OF SWIFT ON THE SERVER

- Server Side Swift Standards (S4) project
  - Protocols and standards
- Apple has a working group dedicated to swift on the server

VAPOR

# VAPOR

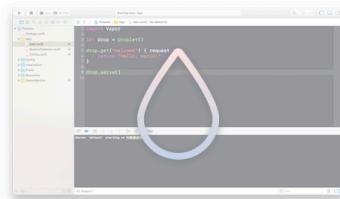
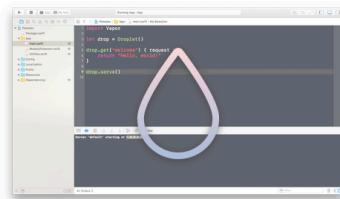
- Web framework and server for Swift
- Works on macOS and Ubuntu

The screenshot shows the GitHub repository page for 'vapor / vapor'. The repository has 3,196 commits, 6 branches, 172 releases, and 77 contributors. The latest commit was made 6 hours ago. The repository uses the MIT license.

File	Description	Time Ago
.Documents	Adding another break line	3 months ago
.Sources	Sessions Middleware doesn't require Foundation anymore	10 hours ago
.Tests	Merge branch 'master' into sessionsMiddlewareImprovements	8 hours ago
.Utilities	allow xcode 8.1	8 months ago
.codecov.yml	remove unnecessary files	a day ago
.gitignore	Make sure sessions cookie is HTTP only	14 hours ago
.travis.yml	prevent double provider boot	a month ago
ISSUE_TEMPLATE.md	Update ISSUE_TEMPLATE.md	a month ago
LICENSE	add MIT License	a year ago
Package.swift	remove perf target	15 days ago
README.md	text updates	13 hours ago
circle.yml	prevent double provider boot	a month ago

# VAPOR

- Great documentation and resources



vapor.university



## Look Maa! Server Side Swift Using Vapor

A detailed introduction to using Vapor and a SQLite database.

15m • Easy • Mohammad Azam

## Social Authentication

Learn how to implement user authentication using Facebook and Google.

8m • Intermediate • Caleb Kleveter

## User Authentication

Learn how to authenticate users using Vapor's Auth module and PostgreSQL.

10m • Intermediate • Caleb Kleveter

## GETTING STARTED

[Install Swift 3: macOS](#)[Install Swift 3: Ubuntu](#)[Install Toolbox](#)[Hello, World](#)[Manual](#)[Xcode](#)

## GUIDE

[Droplet](#)[Folder Structure](#)[JSON](#)[Config](#)[Views](#)[Leaf](#)

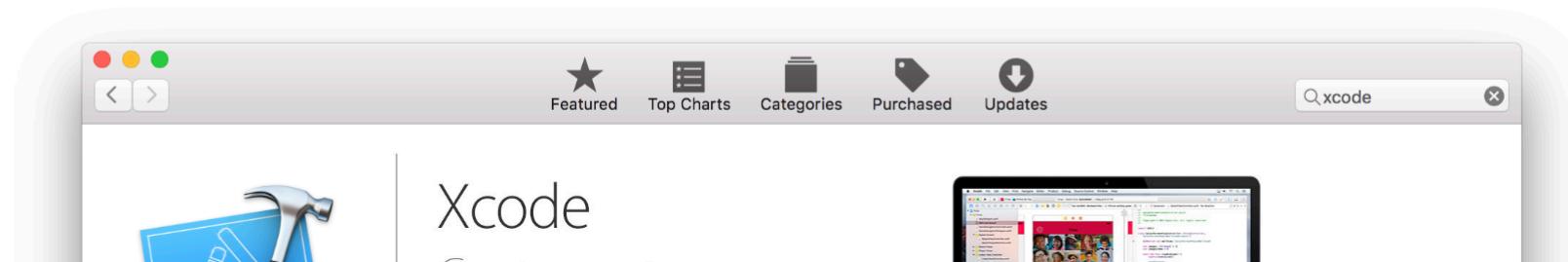
# Install Swift 3: macOS

[!\[\]\(6065b758eea3bdd3c03c46692ee2db49\_img.jpg\) Edit on GitHub](#)

To use Swift 3 on macOS, you just need to have Xcode 8 installed.

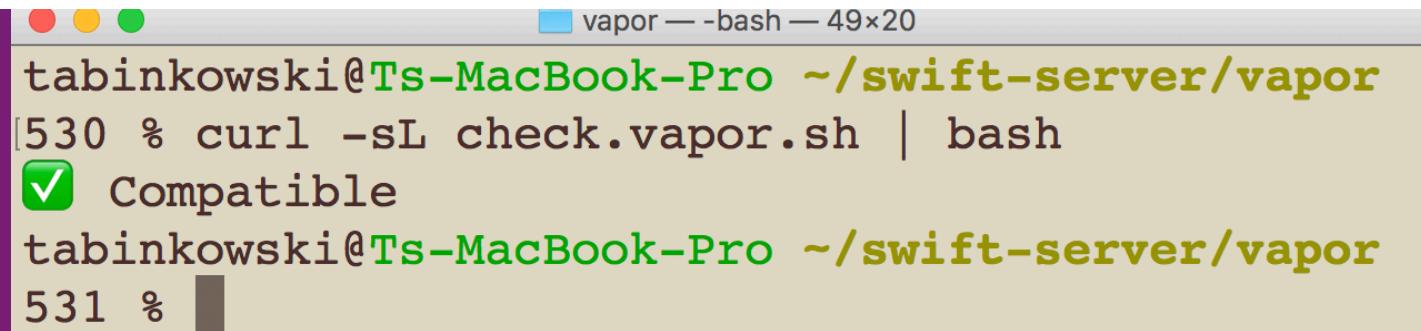
## Install Xcode

Install [Xcode 8](#) from the Mac App Store.



# VAPOR

- Install Xcode
- Check version for vapor



A screenshot of a macOS terminal window titled "vapor — -bash — 49x20". The window shows the following command and its output:

```
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
[530 % curl -sL check.vapor.sh | bash
✓ Compatible
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
531 % ]
```

The terminal uses color coding: green for host and path, grey for command, brown for output, and black for the prompt.

# VAPOR

- Toolbox provides command line and shortcuts for common tasks
- Install with homebrew 🍺

```
vapor — git-remote-https • brew install vapor/tap/vapor — 49×20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
534 % brew install vapor/tap/vapor
```

# VAPOR

- Toolbox provides command line and shortcuts for common tasks
- Install with homebrew 🍺

```
vapor -- bash -- 50x20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
542 % vapor --help
[Usage: vapor <new|build|update|run|fetch|clean|test|xcode|version|self|heroku>
Join our Slack if you have questions, need help,
or want to contribute: http://vapor.team
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
543 % ]
```

# VAPOR

```
vapor — -bash — 80x20
[ tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
542 % vapor --help
Usage: vapor <new|build|update|run|fetch|clean|test|xcode|version|self|heroku>
Join our Slack if you have questions, need help,
or want to contribute: http://vapor.team
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
543 % ]
```

# VAPOR

```
# The toolbox can update itself. This may be useful if # you
experience any issues in the future.
vapor self update

# Templates
# The toolbox can create a project from the Vapor basic-
template or any other git repo.
vapor new <name> [--template=<repo-url-or-github-path>]
```

# VAPOR

- `vapor new vapor-hello-world`
  - Build a new project from basic templates
  - Can specify other templates

```
vapor — bash — 50x20
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ * *
* * ~~~~~ +++++ * *
* * ~~~~~ ++++ * *
*** ~~~~~ +++ * ***
**** ~~~~~ ++++ * ***
***** ~~~~~ +--- * ***
***** ~*****
```

\ \ / / \ \ / \ [P) / \ \ / [R)  
a web framework for Swift

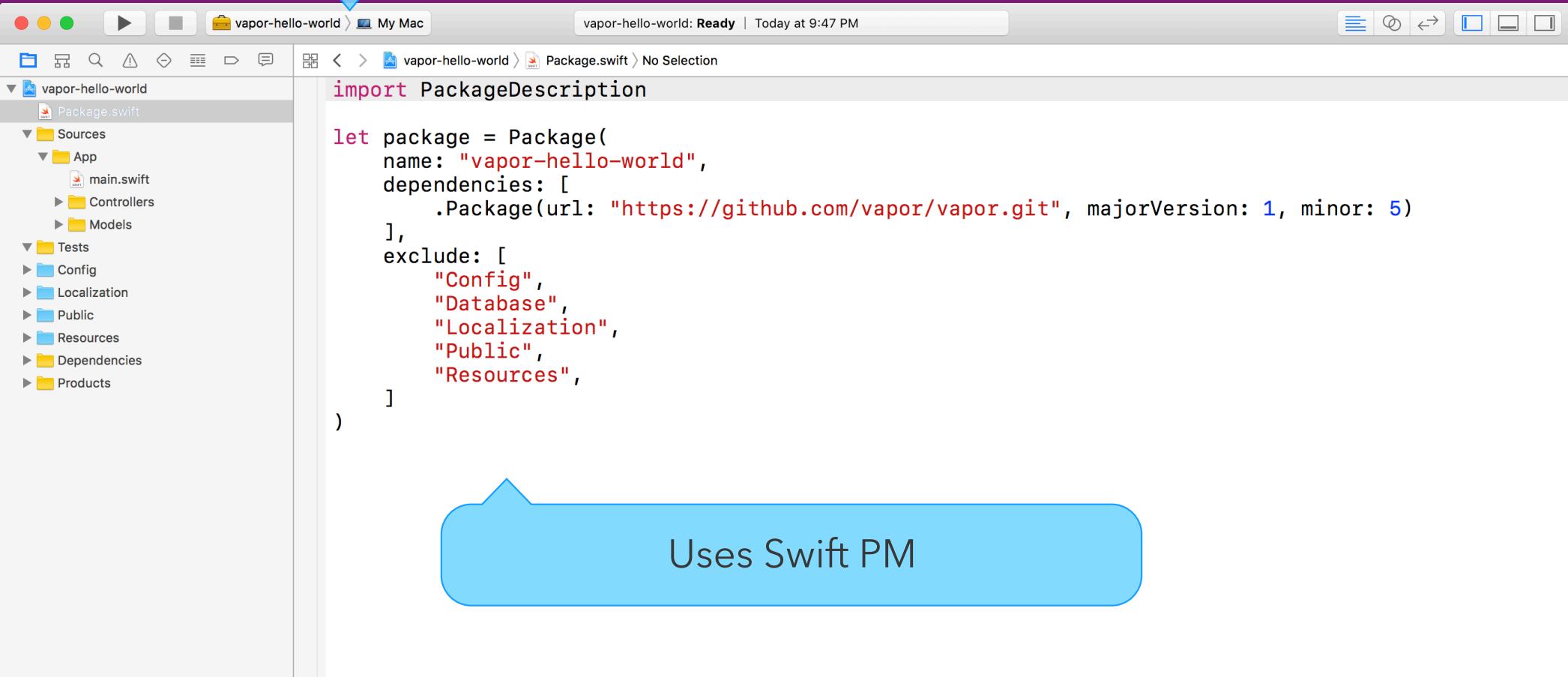
# VAPOR

- `vapor new vapor-hello-world`
- Managing vapor projects
  - Xcode
  - Swift Package Manager for other builds

```
vapor-hello-world — swift-package ▾ vapor xcode — 50×20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
[557 % cd vapor-hello-world/
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[558 % vapor xcode
No .build folder, fetch may take a while...
Fetching Dependencies [• ]
```

# VAPOR

Builds a Framework and App



The screenshot shows a Mac OS X desktop with an Xcode project window open. The title bar reads "vapor-hello-world: Ready | Today at 9:47 PM". The left sidebar shows the project structure:

- vapor-hello-world (selected)
- Package.swift
- Sources
  - App
    - main.swift
    - Controllers
    - Models
  - Tests
- Config
- Localization
- Public
- Resources
- Dependencies
- Products

The main editor area displays the contents of Package.swift:

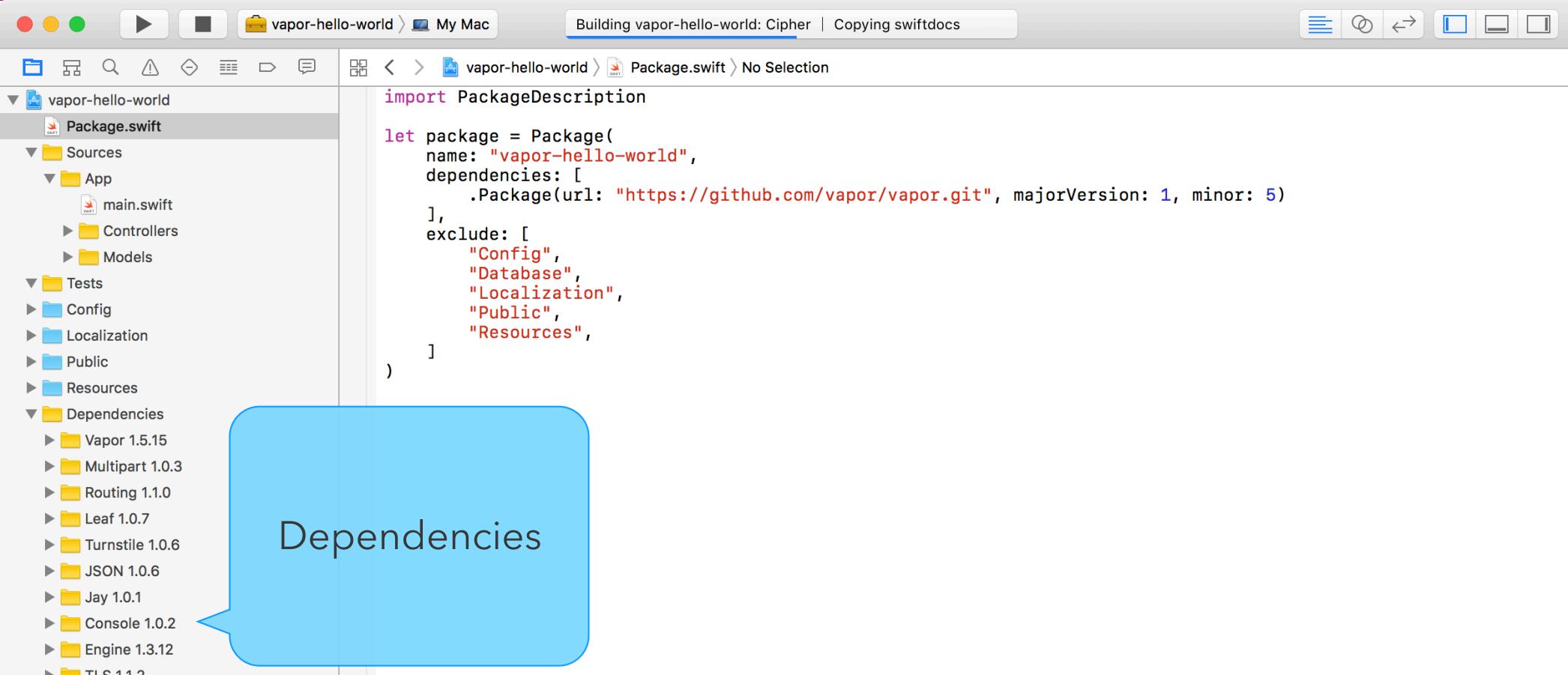
```
import PackageDescription

let package = Package(
    name: "vapor-hello-world",
    dependencies: [
        .Package(url: "https://github.com/vapor/vapor.git", majorVersion: 1, minor: 5)
    ],
    exclude: [
        "Config",
        "Database",
        "Localization",
        "Public",
        "Resources",
    ]
)
```

A blue speech bubble points from the text "Builds a Framework and App" to the "Sources" folder in the project structure.

A blue speech bubble points from the text "Uses Swift PM" to the "import PackageDescription" line in the code.

# VAPOR



The screenshot shows the Xcode interface with a Vapor project named "vapor-hello-world". The left sidebar displays the project structure, including the `Package.swift` file. The main editor window shows the contents of `Package.swift`:

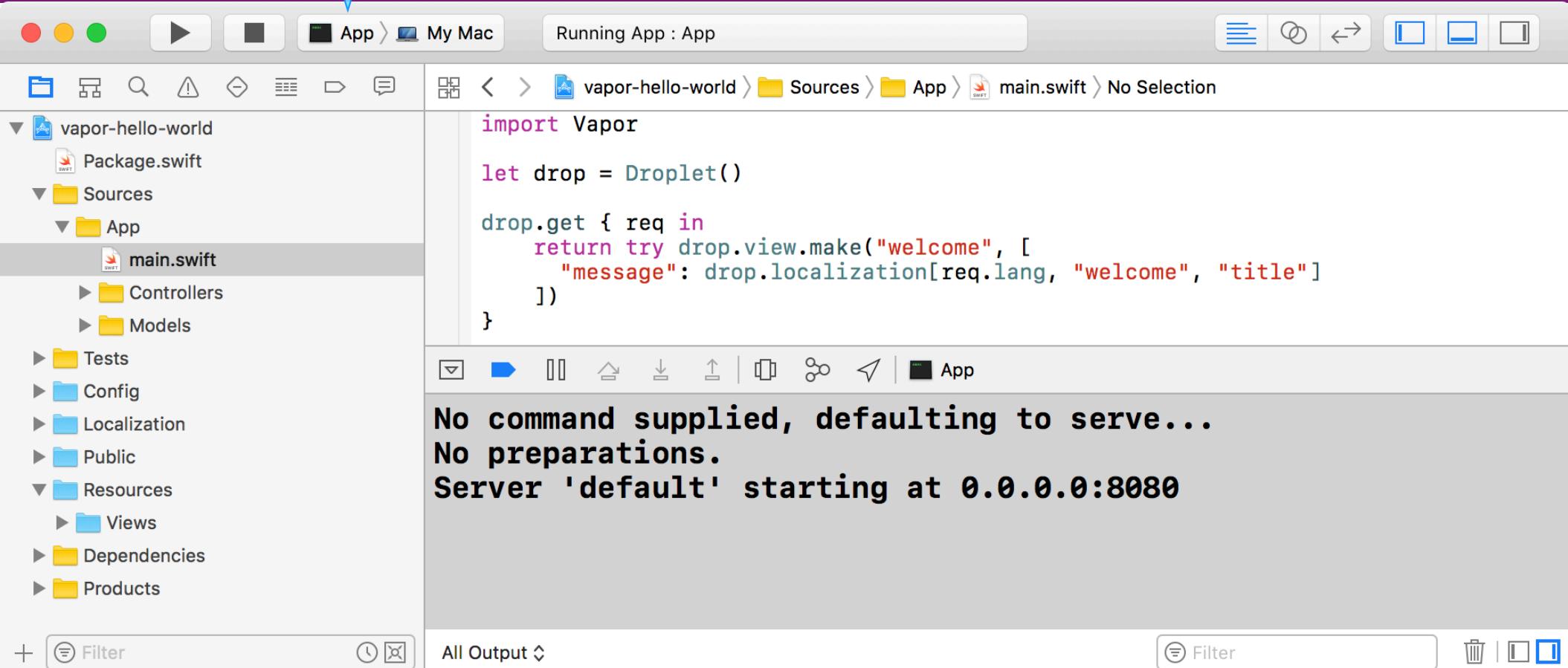
```
import PackageDescription

let package = Package(
    name: "vapor-hello-world",
    dependencies: [
        .Package(url: "https://github.com/vapor/vapor.git", majorVersion: 1, minor: 5)
    ],
    exclude: [
        "Config",
        "Database",
        "Localization",
        "Public",
        "Resources",
    ]
)
```

A blue callout bubble with the word "Dependencies" points to the line where the Vapor framework is imported: `.Package(url: "https://github.com/vapor/vapor.git", majorVersion: 1, minor: 5)`.

# VAPOR

Run App



A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "Running App : App". The terminal output is as follows:

```
import Vapor

let drop = Droplet()

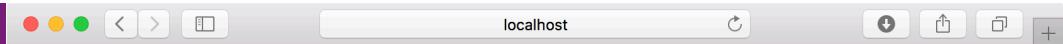
drop.get { req in
    return try drop.view.make("welcome", [
        "message": drop.localization[req.lang, "welcome", "title"]
    ])
}

No command supplied, defaulting to serve...
No preparations.
Server 'default' starting at 0.0.0.0:8080
```

The left sidebar shows the project structure of "vapor-hello-world" with files like Package.swift, main.swift, and various models/controllers.

# VAPOR

- Server runs on [http://localhost:  
8080](http://localhost:8080)



# VAPOR

The screenshot shows a Mac OS X desktop environment with an Xcode window open. The window title is "Running App : App". The file navigation bar shows the path: "vapor-hello-world > Sources > App > main.swift". The left sidebar displays the project structure:

- vapor-hello-world
  - Package.swift
  - Sources
    - App
      - main.swift
      - Controllers
      - Models
    - Tests
    - Config
    - Localization
    - Public
  - Resources
    - Views
  - Dependencies
  - Products

The main editor area contains the following Swift code:

```
import Vapor

let drop = Droplet()

drop.get { req in
    return "Hello Vapor"
}

drop.run()
```

Below the editor is a terminal-like interface showing the output of the application:

```
No command supplied,
No preparations.
Server 'default' started
GET /
```

A separate browser window is open at "localhost", displaying the text "Hello Vapor".

# VAPOR

The screenshot shows a Mac OS X desktop with an Xcode window open. The title bar says "Running App : App". The file browser sidebar shows a project structure:

- vapor-hello-world
  - Package.swift
  - Sources
    - App
      - main.swift
      - Controllers
      - Models
    - Tests
    - Config
    - Localization
    - Public
  - Resources
    - Views
    - Dependencies
    - Products

The main editor area contains the following Swift code in `main.swift`:

```
import Vapor

let drop = Droplet()

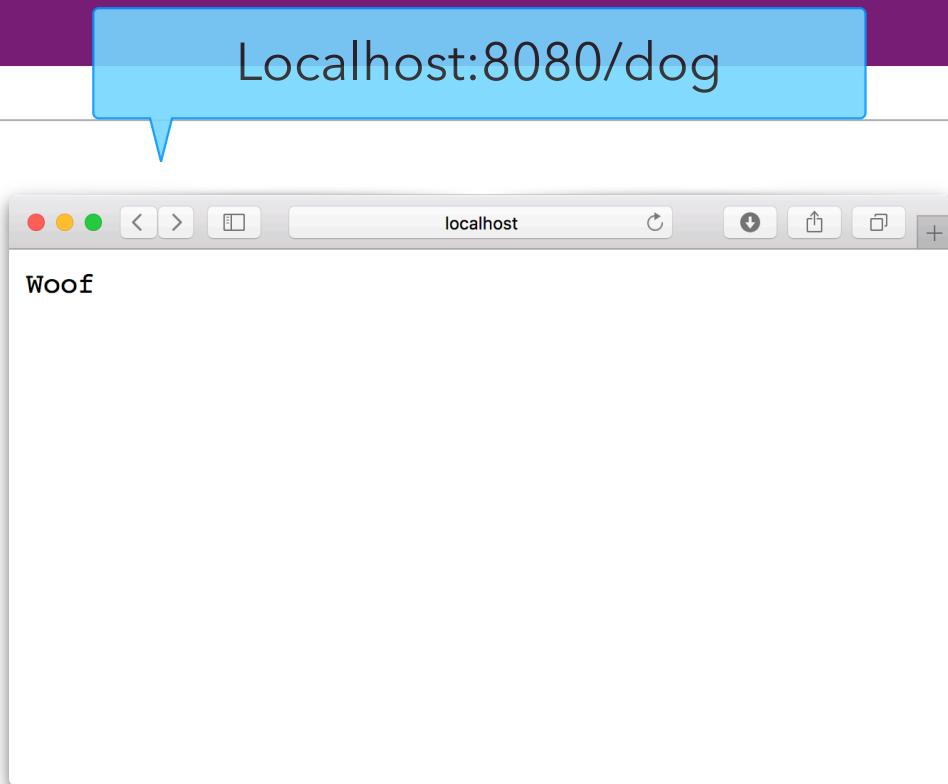
drop.get { req in
    return try JSON(node:
        ["message": "Hello Vapor"]
    )
}

drop.run()
```

A small preview window in the bottom right corner shows the JSON response: `{"message": "Hello Vapor"}`. The preview window has a title bar saying "localhost".

# VAPOR

```
vapor-hello-world > Sources > App > main.swift > No Selection  
import Vapor  
  
let drop = Droplet()  
  
drop.get { req in  
    return try JSON(node:  
        ["message": "Hello Vapor"]  
    )  
}  
  
drop.get("dog") { req in  
    return "Woof"  
}  
  
drop.get("dog", "speak") { req in  
    return "Bark! Bark!"  
}  
  
drop.run()
```



# VAPOR

```
drop.get { req in
    return try JSON(node:
        ["message": "Hello Vapor"]
    )
}

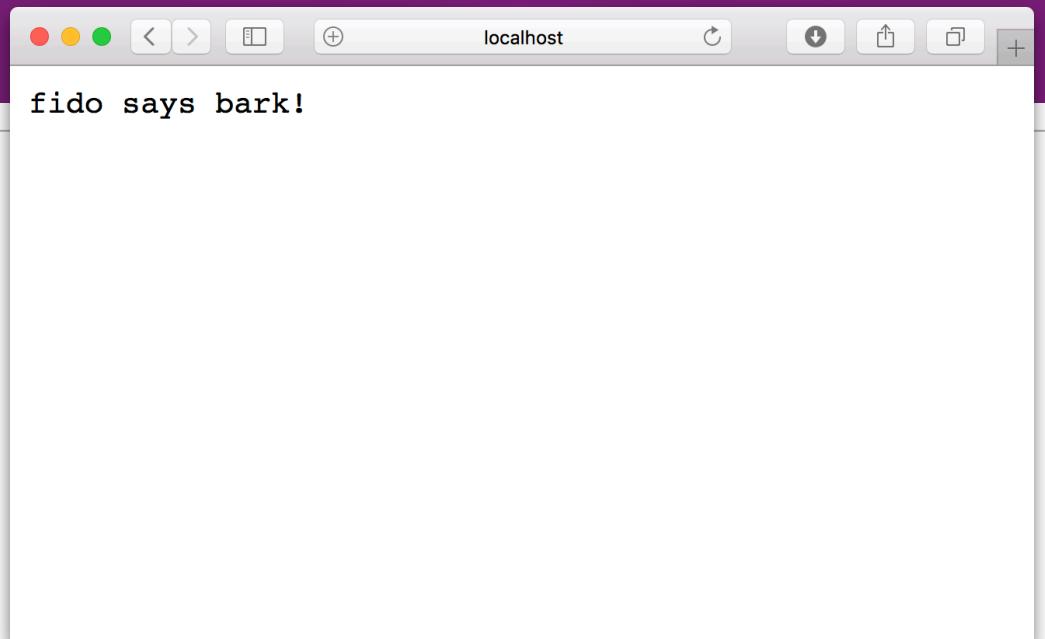
drop.get("dog") { req in
    return "Woof"
}

drop.get("dog", "speak") { req in
    return "Bark! Bark!"
}

drop.get("dog", "speak", String.self) { req, name in
    return "\(name) says bark!"
}

drop.run()
```

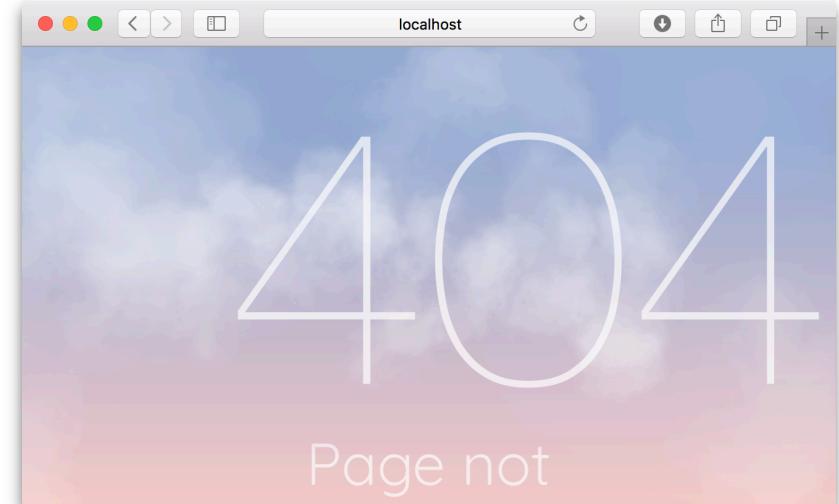
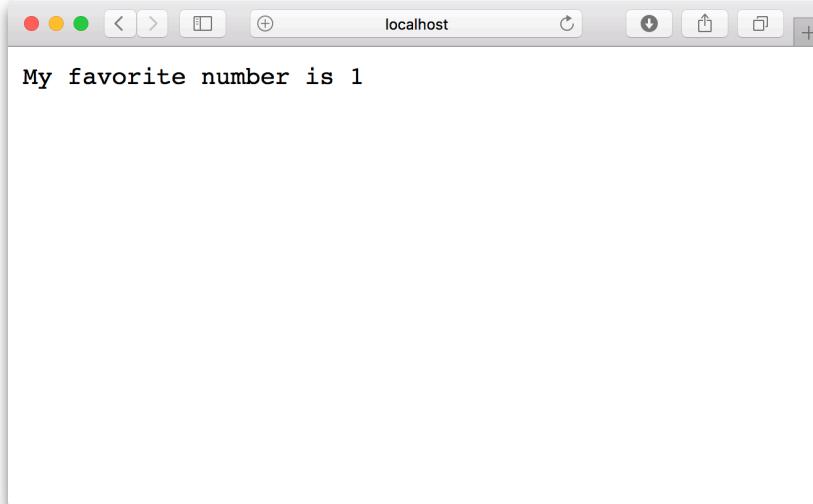
localhost:8080/dog/speak/fido/



VAPOR

Localhost:8080/dog/speak/fido/

Localhost:8080/dog/speak/1/



```
drop.get("dog", "speak", Int.self) { req, number in
    return "My favorite number is \(number)"
}
```

# DEPLOY VAPOR APP TO HEROKU

# DEPLOY VAPOR APP TO HEROKU

- Heroku is a platform as a service company
- Free 'hobby' tier to play around



Log in to your account

Email address

Password

Log In

New to Heroku? [Sign Up](#)

[Log in via SSO](#)    [Forgot your password?](#)

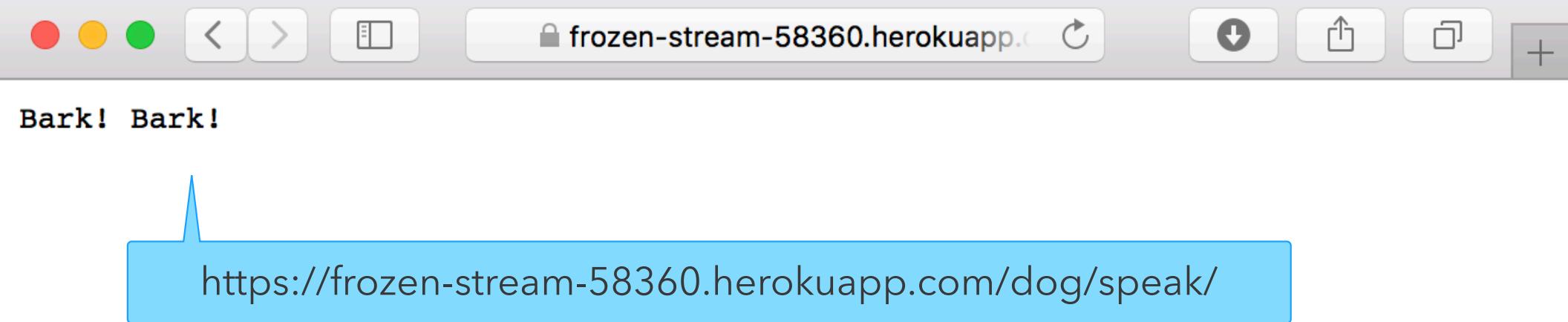
# DEPLOY VAPOR APP TO HEROKU

```
brew install heroku  
heroku login
```

# DEPLOY VAPOR APP TO HEROKU

```
vapor-hello-world — bash — 50x20  
Building on Heroku ... ~5-10 minutes [ •  
Building on Heroku ... ~5-10 minutes [ •  
Building on Heroku ... ~5-10 minutes [D •  
Building on Heroku ... ~5-10 minutes [Do •  
Building on Heroku ... ~5-10 minutes [Do •  
Building on Heroku ... ~5-10 minutes [Do •  
Building on Heroku ... ~5-10 minutes [Done] •  
Spinning up dynos [Done]  
Visit https://dashboard.heroku.com/apps/  
App is live on Heroku, visit  
https://frozen-stream-58360.herokuapp.com/ | https://git.heroku.com/frozen-stream-58360.git  
  
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)  
578 %
```

# DEPLOY VAPOR APP TO HEROKU



# VAPOR WITH POSTGRESQL

# VAPOR WITH POSTGRESQL

- Vapor support many different databases
  - MySQL
  - MongoDB
  - PostgreSQL
- Databases run independently of the web app

The world's most advanced open source database.

Home | About | Download | Documentation | Community | Developers | Support | Your account

**11<sup>th</sup> May 2017**

**PostgreSQL 9.6.3, 9.5.7, 9.4.12, 9.3.17 and 9.2.21 Released!**

The PostgreSQL Global Development Group is pleased to announce the availability of PostgreSQL 9.6.3, 9.5.7, 9.4.12, 9.3.17 and 9.2.21.

These new releases contain bug fixes over previous releases. All users should plan to upgrade their systems as soon as possible.

» [Release Announcement](#)  
» [Release Notes](#)  
» [Download](#)



**FEATURED USER**

... mission-critical technology tasks can continue to depend on the power, flexibility and robustness of PostgreSQL.

Ram Mohan, CTO, [Afiliias](#)  
» [Case Studies](#) | [More Quotes](#) | [Featured Users](#)

**LATEST NEWS**

2017-05-11  
[2017-05-11 Security Update Release](#)

2017-05-07  
[pg\\_chameleon 1.0 released](#)

2017-05-04  
[Announcing The Release Of pglogical 2.0](#)

2017-04-25  
[DB Doc 4.1 released](#)

2017-04-19  
[New version of MySQL-to-PostgreSQL has been released](#)

2017-04-07  
[PgComment for PostgreSQL released](#)

2017-03-20  
[Announcing AgensGraph v1.1 Release](#)

» [More](#) | [Submit News](#) | [RSS](#)

**UPCOMING EVENTS**

2017-05-18 – 2017-05-20  
[PostgreSQL Booth at PyCon 2017](#)  
(Portland, Oregon, United States)

2017-05-23 – 2017-05-26  
[PGCon 2017](#)  
(Ottawa, Ontario, Canada)

2017-06-08  
[Pg Day France 2017](#)  
(Toulouse, France)

2017-06-09  
[PgDay Argentina 2017](#)  
(Santa Fe, Santa Fe, Argentina)

2017-06-26 – 2017-06-28  
[Postgres Vision 2017](#)  
(Boston, MA, United States)

» [More](#) | [Submit Event](#) | [RSS](#)

**UPCOMING TRAINING**

There are 11 training events in 3 countries scheduled over the next six months from PostgresCourse.com,

**LATEST RELEASES**

9.6.3 · May 11, 2017 · [Notes](#)  
9.5.7 · May 11, 2017 · [Notes](#)  
9.4.12 · May 11, 2017 · [Notes](#)  
9.3.17 · May 11, 2017 · [Notes](#)  
9.2.21 · May 11, 2017 · [Notes](#)

[Download](#) | [RSS](#)  
Why should I upgrade?  
Upcoming releases

**SHORTCUTS**

» [Security](#)  
» [International Sites](#)  
» [Mailing Lists](#)  
» [Wiki](#)  
» [Report a Bug](#)  
» [FAQs](#)

**SUPPORT US**

PostgreSQL is free. Please support our work by making a [donation](#).

**PLANET POSTGRES**

2017-05-16  
[Joshua Drake: PGConf US Local: Seattle Call for Papers](#)

2017-05-16  
[Michael Paquier: Postgres 10 highlight - Incompatible changes](#)

2017-05-15  
[Bruce Momjian: RAID Controllers and SSDs](#)

2017-05-15  
[Holly Orr: DevOps: Can't we just all get along?](#)

2017-05-13  
[Federico Campoli: pg\\_chameleon v1.1 out](#)

2017-05-13  
[Leo Hsu and Regina Obe: PostgreSQL JQuery extension Windows binaries](#)

2017-05-12  
[Bruce Momjian: Postgres Window Magic](#)

» [More](#) | [RSS](#)

# VAPOR WITH POSTGRESQL

- Importing Vapor also imports the `Fluent` framework
- ORM tool for Swift
  - Works with different databases
- Can we be used on app for consistency

The screenshot shows the GitHub repository page for the project `vapor/fluent`. The repository has 608 commits, 3 branches, 97 releases, and 31 contributors. A recent commit by `tanner0101` is highlighted, showing changes made to various files like Sources, Tests, .gitignore, ISSUE\_TEMPLATE.md, LICENSE, Package.swift, README.md, and circle.yml. The commit message indicates fixes for case mismatch in custom keys tests, node updates, pins, model comments, and SQLite removal. The README.md file is also updated. The Fluent logo is visible in the bottom right corner.

vapor / fluent

Code Issues 0 Pull requests 0 Wiki Pulse Graphs

Swift models, relationships, and querying for NoSQL and SQL databases.

vapor orm swift database sql nosql

608 commits 3 branches 97 releases 31 c

Branch: master New pull request Create new file

tanner0101 committed on GitHub Merge pull request #255 from vapor/customkeys-test-fix-for-postgres

File	Description
Sources	Fix case mismatch in the custom keys test
Tests	node updates
.gitignore	pins
ISSUE_TEMPLATE.md	model comments written
LICENSE	MIT License
Package.swift	optionally remove sqlite
README.md	update readme
circle.yml	update readme

README.md

\* Fluent

# VAPOR WITH POSTGRESQL

- Import package
- Configure droplet
- Add configuration file

The world's most advanced open source database.

Home | About | Download | Documentation | Community | Developers | Support | Your account

**11<sup>th</sup> May 2017**

**PostgreSQL 9.6.3, 9.5.7, 9.4.12, 9.3.17 and 9.2.21 Released!**

The PostgreSQL Global Development Group is pleased to announce the availability of PostgreSQL 9.6.3, 9.5.7, 9.4.12, 9.3.17 and 9.2.21.

These new releases contain bug fixes over previous releases. All users should plan to upgrade their systems as soon as possible.

» [Release Announcement](#)  
» [Release Notes](#)  
» [Download](#)



**FEATURED USER**

... mission-critical technology tasks can continue to depend on the power, flexibility and robustness of PostgreSQL.

Ram Mohan, CTO, Afilias  
» [Case Studies](#) | [More Quotes](#) | [Featured Users](#)

**LATEST NEWS**

2017-05-11 [2017-05-11 Security Update Release](#)  
2017-05-07 [pg\\_chameleon 1.0 released](#)  
2017-05-04 [Announcing The Release Of pglogical 2.0](#)  
2017-04-25 [DB Doc 4.1 released](#)  
2017-04-19 [New version of MySQL-to-PostgreSQL has been released](#)  
2017-04-07 [PgComment for PostgreSQL released](#)  
2017-03-20 [Announcing AgensGraph v1.1 Release](#)  
» [More](#) | [Submit News](#) | [RSS](#)

**UPCOMING EVENTS**

2017-05-18 – 2017-05-20 [PostgreSQL Booth at PyCon 2017](#)  
(Portland, Oregon, United States)  
2017-05-23 – 2017-05-26 [PGCon 2017](#)  
(Ottawa, Ontario, Canada)  
2017-06-08 [PG Day France 2017](#)  
(Toulouse, France)  
2017-06-09 [PgDay Argentina 2017](#)  
(Santa Fe, Santa Fe, Argentina)  
2017-06-26 – 2017-06-28 [Postgres Vision 2017](#)  
(Boston, MA, United States)  
» [More](#) | [Submit Event](#) | [RSS](#)

**UPCOMING TRAINING**

There are 11 training events in 3 countries scheduled over the next six months from PostgresCourse.com,

**LATEST RELEASES**

9.6.3 · May 11, 2017 · [Notes](#)  
9.5.7 · May 11, 2017 · [Notes](#)  
9.4.12 · May 11, 2017 · [Notes](#)  
9.3.17 · May 11, 2017 · [Notes](#)  
9.2.21 · May 11, 2017 · [Notes](#)

[Download](#) | [RSS](#)  
[Why should I upgrade?](#)  
[Upcoming releases](#)

**SHORTCUTS**

» [Security](#)  
» [International Sites](#)  
» [Mailing Lists](#)  
» [Wiki](#)  
» [Report a Bug](#)  
» [FAQs](#)

**SUPPORT US**

PostgreSQL is free. Please support our work by making a [donation](#).

**PLANET POSTGRES**

2017-05-16 [Joshua Drake: PGConf US Local: Seattle Call for Papers](#)  
2017-05-16 [Michael Paquier: Postgres 10 highlight - Incompatible changes](#)  
2017-05-15 [Bruce Momjian: RAID Controllers and SSDs](#)  
2017-05-15 [Holly Orr: DevOps: Can't we just all get along?](#)  
2017-05-13 [Federico Campoli: pg\\_chameleon v1.1 out](#)  
2017-05-13 [Leo Hsu and Regina Obe: PostgreSQL JQuery extension Windows binaries](#)  
2017-05-12 [Bruce Momjian: Postgres Window Magic](#)  
» [More](#) | [RSS](#)

# VAPOR WITH POSTGRESQL

```
# Install postgres
brew install postgres

# Start postgres server
postgres -D /usr/local/var/postgres/

# Create a database
createdb dogs
```

# VAPOR WITH POSTGRESQL

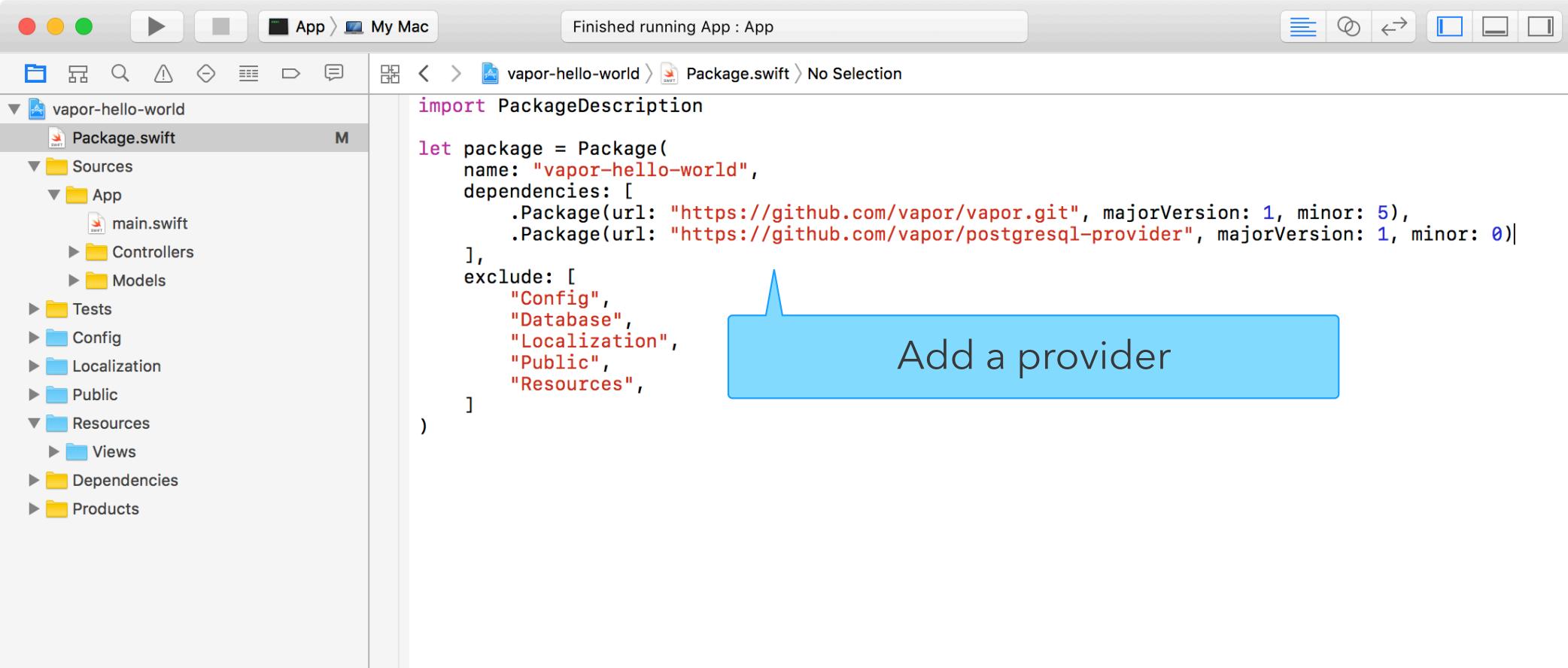
```
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[592 % psql
psql (9.6.3)
Type "help" for help.

[tabinkowski=# \l
                                         List of databases
  Name | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----+
dogs   | tabinkowski | UTF8    | en_US.UTF-8 | en_US.UTF-8 |
postgres | tabinkowski | UTF8    | en_US.UTF-8 | en_US.UTF-8 |
tabinkowski | tabinkowski | UTF8    | en_US.UTF-8 | en_US.UTF-8 |
template0 | tabinkowski | UTF8    | en_US.UTF-8 | en_US.UTF-8 | =c/tabinkowski          +
                                         |                   tabinkowski=CTc/tabinkowski
template1 | tabinkowski | UTF8    | en_US.UTF-8 | en_US.UTF-8 | =c/tabinkowski          +
                                         |                   tabinkowski=CTc/tabinkowski
(5 rows)

tabinkowski=# ]
```

- Vapor support many different databases
  - MySQL

# VAPOR WITH POSTGRESQL



Finished running App : App

vapor-hello-world

Package.swift

Sources

App

main.swift

Controllers

Models

Tests

Config

Localization

Public

Resources

Views

Dependencies

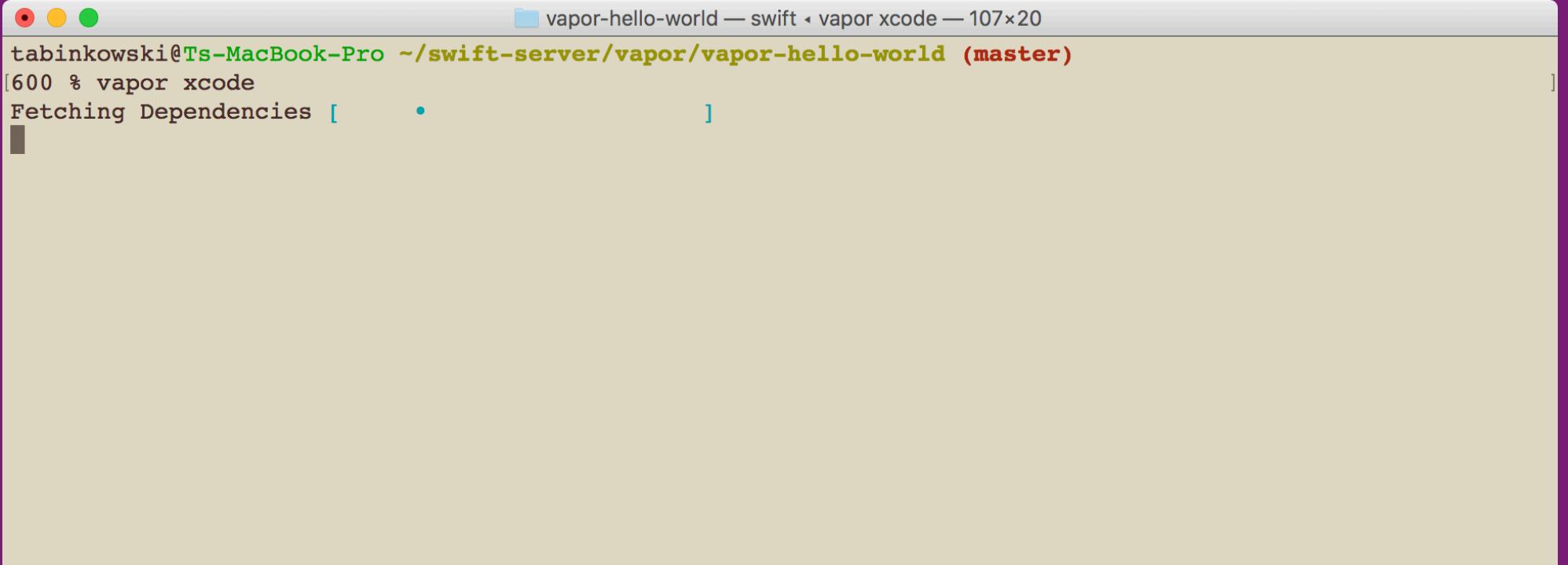
Products

```
import PackageDescription

let package = Package(
    name: "vapor-hello-world",
    dependencies: [
        .Package(url: "https://github.com/vapor/vapor.git", majorVersion: 1, minor: 5),
        .Package(url: "https://github.com/vapor/postgresql-provider", majorVersion: 1, minor: 0)
    ],
    exclude: [
        "Config",
        "Database",
        "Localization",
        "Public",
        "Resources",
    ]
)
```

Add a provider

# VAPOR WITH POSTGRESQL



A screenshot of a macOS terminal window titled "vapor-hello-world — swift ▾ vapor xcode — 107x20". The window shows the command "tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)" followed by "[600 % vapor xcode". Below this, the text "Fetching Dependencies [ • ]" is displayed, indicating a progress bar.

- Update the project and rebuild your framework in Xcode

# VAPOR WITH POSTGRESQL

< > vapor-hello-world > Sources > App > main.swift > No Selection

```
import Vapor
import VaporPostgreSQL

let drop = Droplet()

// Add provider
do {
    try drop.addProvider(VaporPostgreSQL.Provider.self)
} catch {
    assertionFailure("Error adding provider: \(error)")
}
```

Add a provider

# VAPOR WITH POSTGRESQL

vapor-hello-world | Build **Succeeded**

vapor-hello-world > Config > secrets > postgresql.json > No Selection

```
{  
    "host": "127.0.0.1",  
    "user": "tabinkowski",  
    "password": "",  
    "database": "dogs",  
    "port": 5432  
}
```

Database configuration file

Add a new file

vapor-hello-world

- Package.swift
- Sources
  - App
    - main.swift
  - Controllers
  - Models
- Tests
- Config
  - app.json
  - clients.json
  - crypto.json
  - droplet.json
  - production
  - secrets
    - postgresql.json
    - servers.json
- Localization
- Public
- Resources
- Dependencies

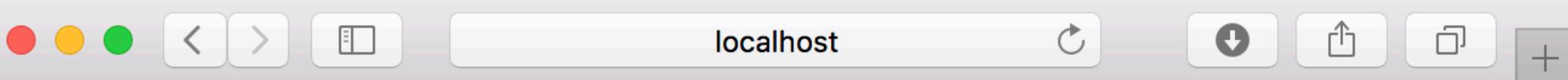
# VAPOR WITH POSTGRESQL

```
// Add provider
do {
    try drop.addProvider(VaporPostgreSQL.Provider.self)
} catch {
    assertionFailure("Error adding provider: \(error)")
}

// Route to print out the db version
drop.get("version") { req in
    if let db = drop.database?.driver as? PostgreSQLDriver {
        let version = try db.raw("SELECT version()")
        return try JSON(node: version)
    } else {
        return "No database connection"
    }
}
```

Print out database version

## VAPOR WITH POSTGRESQL



```
[{"version": "PostgreSQL 9.6.3 on x86_64-apple-darwin16.5.0, compiled by Apple LLVM version 8.1.0 (clang-802.0.42), 64-bit"}]
```

It's boring, but it works

# VAPOR WITH POSTGRESQL

The screenshot shows a Mac OS X desktop environment. In the center is a terminal window titled "Running App : App". The window displays Swift code for a Vapor application. A blue callout box points from the text "Drop.preparations.append(Dog.self)" to the right, labeled "Add Dog model". Another blue callout box points from the code block "let dog = [Dog(name:"Lulu", breed: "Boston Terrier"), Dog(name:"Snoopy", breed: "Beagle"), Dog(name:"Fifi", breed: "Poodle")]..." to the right, labeled "Create data". The terminal window also shows the file path "vapor-hello-world > Sources > App > main.swift > No Selection". The top of the screen features the Mac OS X dock with various icons, and the system menu bar is visible at the very top.

```
let drop = Droplet()
drop.preparations.append(Dog.self)                                Add Dog model

// Add provider
do {
    try drop.addProvider(VaporPostgreSQL.Provider.self)
} catch {
    assertionFailure("Error adding provider: \(error)")
}

drop.get("dog", "create") { req in
    let dog = [Dog(name:"Lulu", breed: "Boston Terrier"),
              Dog(name:"Snoopy", breed: "Beagle"),
              Dog(name:"Fifi", breed: "Poodle")]
    let dogNode = try dog.makeNode()
    let nodeDictionary = ["dogs": dogNode]
    return try JSON(node: nodeDictionary)
}
```

# VAPOR WITH POSTGRESQL

The screenshot shows a Mac OS X desktop with a terminal window titled "Running App : App". The path in the title bar is "vapor-hello-world > Sources > App > main.swift > No Selection". The terminal contains Swift code for a Vapor application:

```
let drop = Droplet()
drop.preparations.append(Dog.self)

// Add provider
do {
    try drop.addProvider(VaporPostgreSQL.Provider.self)
} catch {
    assertionFailure("Error adding provider: \(error)")
}

drop.get("dog", "create") { req in
    let dog = [Dog(name:"Lulu", breed: "Boston Terrier"),
              Dog(name:"Snoopy", breed: "Beagle"),
              Dog(name:"Fifi", breed: "Poodle")]
    let dogNode = try dog.makeNode()
    let nodeDictionary = ["dogs": dogNode]
    return try JSON(node: nodeDictionary)
}
```

A blue callout box points to the line `drop.preparations.append(Dog.self)` with the text "Add Dog model". Another blue callout box points to the line `let dog = [Dog(name:"Lulu", breed: "Boston Terrier"), Dog(name:"Snoopy", breed: "Beagle"), Dog(name:"Fifi", breed: "Poodle")]` with the text "Create data to test our model".

# VAPOR WITH POSTGRESQL

The screenshot shows a Mac OS X desktop with a terminal window titled "Running App : App". The path in the title bar is "vapor-hello-world > Sources > App > main.swift > No Selection". The terminal contains Swift code for a Vapor application:

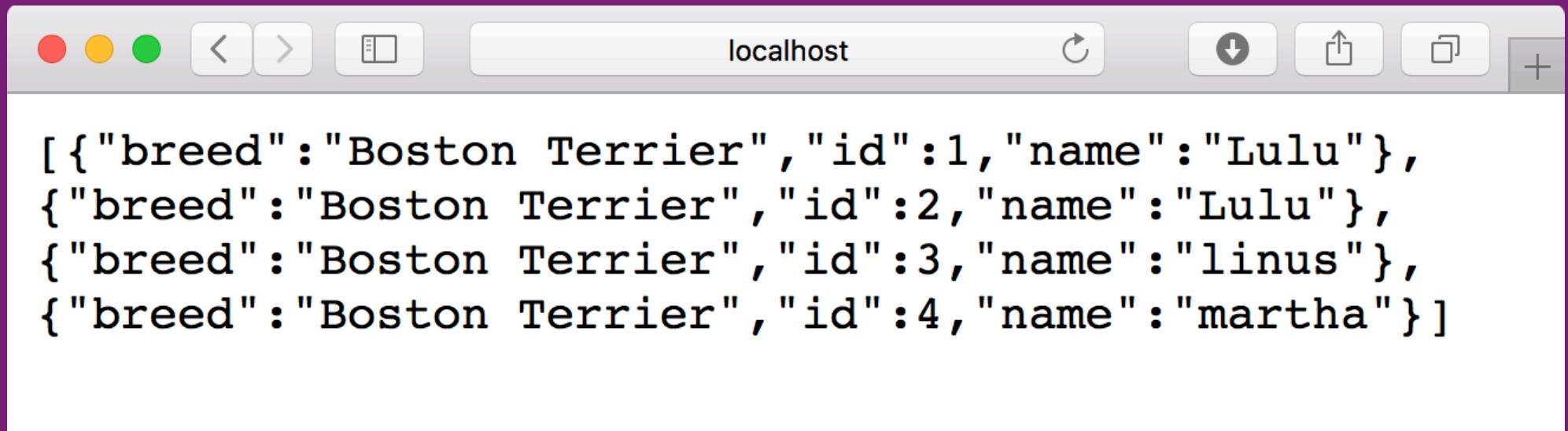
```
let drop = Droplet()
drop.preparations.append(Dog.self)

// Add provider
do {
    try drop.addProvider(VaporPostgreSQL.Provider.self)
} catch {
    assertionFailure("Error adding provider: \(error)")
}

drop.get("dog", "create") { req in
    let dog = [Dog(name:"Lulu", breed: "Boston Terrier"),
              Dog(name:"Snoopy", breed: "Beagle"),
              Dog(name:"Fifi", breed: "Poodle")]
    let dogNode = try dog.makeNode()
    let nodeDictionary = ["dogs": dogNode]
    return try JSON(node: nodeDictionary)
}
```

A blue callout box points to the line `drop.preparations.append(Dog.self)` with the text "Add Dog model". Another blue callout box points to the `Dog` type in the `makeNode()` call with the text "Create data to test our model".

## VAPOR WITH POSTGRESQL



```
// List all dogs
drop.get("dog", "pound") { req in
    return try JSON(node: Dog.all().makeNode())
}
```

# VAPOR WITH POSTGRESQL

- Check that data is being persisted using psql
  - \l - list dis
  - \c - switch db
  - \d - list tables

```
vapor-hello-world — psql — 60x20
(5 rows)

tabinkowski=# \dt
No relations found.
tabinkowski=# \c dogs
You are now connected to database "dogs" as user "ta
i".
dogs=# select * from dogs;
 id | name      |      breed
----+-----+-----
    1 | Lulu      | Boston Terrier
    2 | Lulu      | Boston Terrier
    3 | linus     | Boston Terrier
    4 | martha    | Boston Terrier
    5 | martha    | Boston Terrier
    6 | martha    | Boston Terrier
    7 | martha    | Boston Terrier
(7 rows)

dogs=#
```

# DEPLOY TO HEROKU WITH POSTGRESQL

## DEPLOY TO HEROKU WITH POSTGRESQL

- Set up postgresql on heroku
  - Built-in, just need to add it
- Tell Postgresql where the database is located
  - Do not upload the secrets/postgresql.json file

```
{  
  "host": "127.0.0.1",  
  "user": "tabinkowski",  
  "password": "",  
  "database": "dogs",  
  "port": 5432  
}
```

# DEPLOY TO HEROKU WITH POSTGRESQL

Add postgres to your heorku app

```
vapor-hello-world — -bash — 89x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
613 % heroku addons:create heroku-postgresql:hobby-dev
Creating heroku-postgresql:hobby-dev on ⚙ frozen-stream-58360... free
Database has been created and is available
! This database is empty. If upgrading, you can transfer
! data from another database with pg:copy
Created postgresql-vertical-49786 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
tabinkowski@Ts-MacBook-Pro /swift-server/vapor/vapor-hello-world (master)
614 %
```

New database

# DEPLOY TO HEROKU WITH POSTGRESQL

```
vapor-hello-world bash 89x15
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[613 % heroku addons:create heroku-postgresql:hobby-dev
Creating heroku-postgresql:hobby-dev on ⚙️ frozen-stream-58360... free
Database has been created and is available
! This database is empty. If upgrading, you can transfer
! data from another database with pg:copy
Created postgresql-vertical-49786 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[614 % heroku config
==== frozen-stream-58360 Config Vars
DATABASE_URL: postgres://hzgtyzqowaztwa:4ad9847249b30c97f1835744ca06b1bae2b9012422c21d32d
4dad5e5088def04@ec2-23-23-227-188.compute-1.amazonaws.com:5432/d9hapt1pulfgrf
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
615 %
```

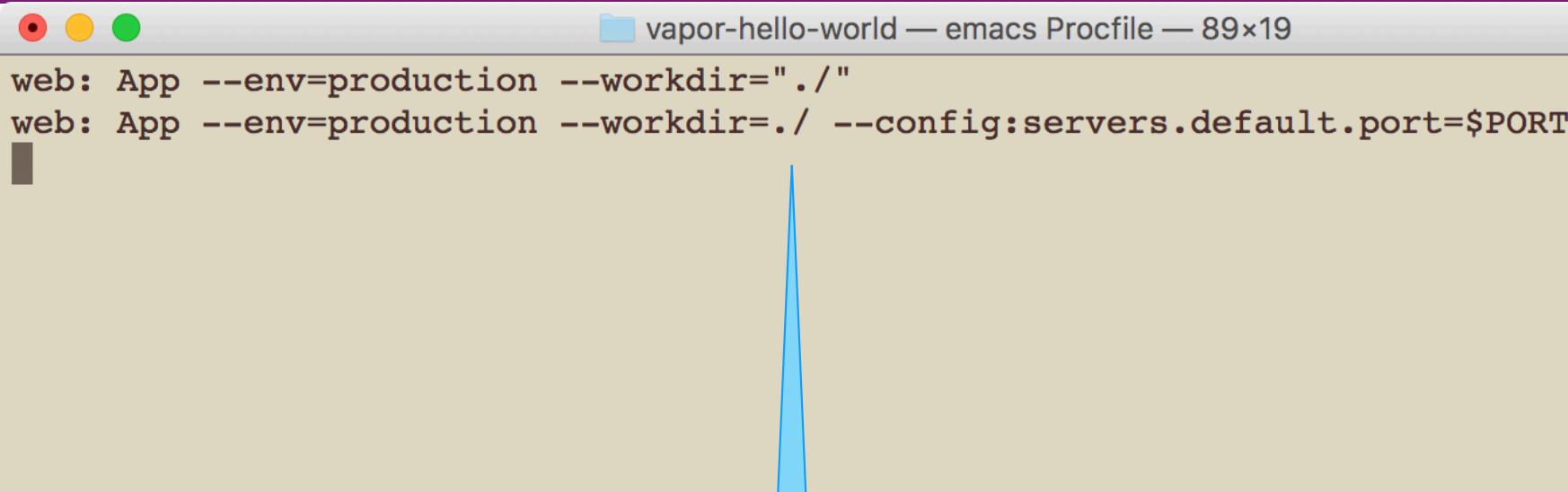
Database location

# DEPLOY TO HEROKU WITH POSTGRESQL

```
vapor-hello-world — -bash — 89x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[618 % ls
Config                               Procfile
Localization                         Public
Package.pins                          README.md
Package.swift                         Resources
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
619 % ]
```

Update the procfile with the database location

# DEPLOY TO HEROKU WITH POSTGRESQL



```
web: App --env=production --workdir=". /"
web: App --env=production --workdir= ./ --config:servers.default.port=$PORT
```

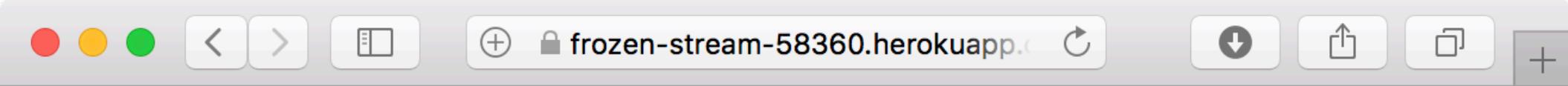
Update the procfile with the database location

# DEPLOY TO HEROKU WITH POSTGRESQL

```
vapor-hello-world — -bash — 89x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
629 % git push heroku master
```

Push the changes to heroku

# DEPLOY TO HEROKU WITH POSTGRESQL



```
[{"breed": "Boston Terrier", "id": 1, "name": "martha"}, {"breed": "Boston Terrier", "id": 2, "name": "martha"}]
```

<https://frozen-stream-58360.herokuapp.com/dog/pound/>

# VAPOR WITH POSTGRESQL

```
// List all dogs
drop.get("dog", "pound") { req in
    return try JSON(node: Dog.all().makeNode())
}
```

Get all entries

# VAPOR WITH POSTGRESQL

Filter

```
// Get all bostons
drop.get("query-boston") { req in
    return try JSON(node: Dog.query().filter("breed", .equals, "Boston Terrier").all().makeNode())
}
```

Query

# VAPOR WITH POSTGRESQL

Get the first entry

```
//  
drop.get("update-poodle") { req in  
    guard var dog = try Dog.query().first() else {  
        throw Abort.badRequest  
    }  
    dog.breed = "Poodle"  
    try dog.save()  
    return dog  
}
```

Change values

Save

PERFECT

# PERFECT

- Web framework and server for Swift server applications

The screenshot shows the homepage of perfect.org. At the top left is the Perfect logo (a stylized bird) and the text "perfect.org". At the top right is a three-line menu icon. Below the header are four main sections arranged in a grid:

- Get Started**: Features a large orange bird icon. Description: "Access Perfect's feature set and start your project." Call-to-action: "Get Started >>"
- Documentation**: Features an orange book icon. Description: "Get support from our comprehensive set of documentation." Call-to-action: "Documentation >>"
- Assistant**: Features an orange laptop icon. Description: "Meet the Perfect Assistant to help you do more." Call-to-action: "Get Started >>"
- Commercial**: Features an orange bar chart icon. Description: "Use Perfect to build apps for business." Call-to-action: "Get Started >>"

# PERFECT

- Requirements
  - Xcode 8 Mac
  - Linux toolchain

The screenshot shows the homepage of perfect.org. At the top left is the Perfect logo (a stylized bird) and the URL 'perfect.org'. At the top right is a three-line menu icon. The page features four main sections arranged in a 2x2 grid:

- Get Started**: Features a large orange bird icon. Below it is the text "Access Perfect's feature set and start your project." and a "Get Started >>" button.
- Documentation**: Features an orange book icon. Below it is the text "Get support from our comprehensive set of documentation." and a "Documentation >>" button.
- Assistant**: Features an orange laptop icon. Below it is the text "Meet the Perfect Assistant to help you do more...".
- Commercial**: Features an orange bar chart icon. Below it is the text "Use Perfect to build apps for business...".

# PERFECT

```
[637 % cd perfect/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[638 % ls
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[639 % mkdir perfect-hello-world
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[640 % cd perfect-hello-world/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
Swift package manager to create an executable package
[641 % ls
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[642 % swift package init --type executable
Creating executable package: perfect-hello-world
Creating Package.swift
Creating .gitignore
Creating Sources/
Creating Sources/main.swift
Creating Tests/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
643 %
```

# PERFECT

```
perfect-hello-world — -bash ▶ postgres: stats collector process — 70x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
645 %
```

Swift package manager to create an Xcode project

# PERFECT

```
perfect-hello-world — bash ▶ postgres: stats collector process — 70x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[645 % ls
Package.swift                               Tests
Sources                                     perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[646 % ]
```

# PERFECT

- Working command line application

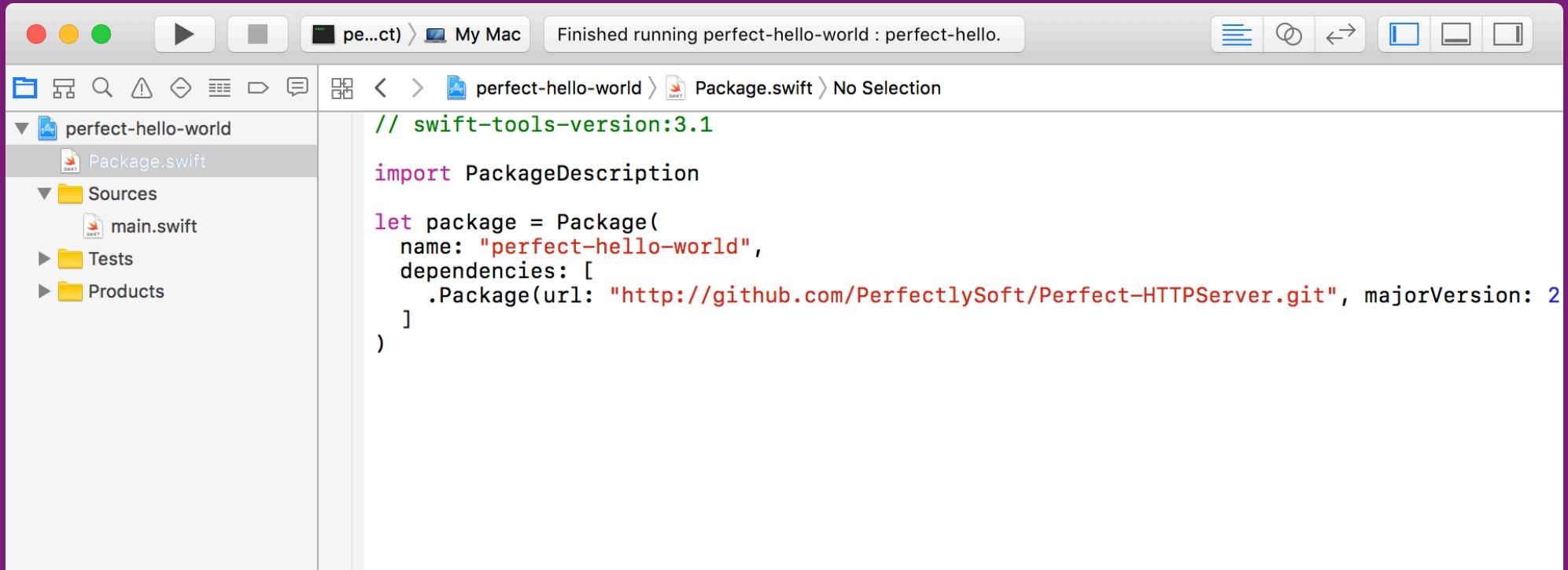
The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure for "perfect-hello-world". It includes a "Package.swift" file, a "Sources" folder containing "main.swift", a "Tests" folder, and a "Products" folder.
- Editor:** The code editor displays the content of "main.swift": 

```
print("Hello, world!")
```
- Output Navigator:** The output window shows the results of running the program:

```
Hello, world!
Program ended with exit code: 0
```
- Bottom Bar:** Includes a "Filter" field and a "All Output" dropdown.

# PERFECT



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with a selected "Package.swift" file.
- Editor:** Displays the contents of the "Package.swift" file:

```
// swift-tools-version:3.1
import PackageDescription

let package = Package(
    name: "perfect-hello-world",
    dependencies: [
        .Package(url: "http://github.com/PerfectlySoft/Perfect-HTTPServer.git", majorVersion: 2
    ]
)
```
- Toolbar:** Standard Xcode toolbar with icons for file operations, search, and navigation.
- StatusBar:** Shows the message "Finished running perfect-hello-world : perfect-hello."

- Add Perfect as a project dependency

# PERFECT



A screenshot of a macOS terminal window titled "perfect-hello-world — swift-package update ▶ git-remote-https — 70x19". The window shows the command "tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world [648 % swift package update" followed by the output "Fetching http://github.com/PerfectlySoft/Perfect-HTTPServer.git" and "Fetching https://github.com/PerfectlySoft/Perfect-HTTP.git". The terminal has a light beige background and a dark grey header bar.

```
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world [648 % swift package update
Fetching http://github.com/PerfectlySoft/Perfect-HTTPServer.git
Fetching https://github.com/PerfectlySoft/Perfect-HTTP.git
```

- Update the package manager

# PERFECT

```
perfect-hello-world — -bash ▶ postgres: stats collector process — 70x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[651 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
652 % ]
```

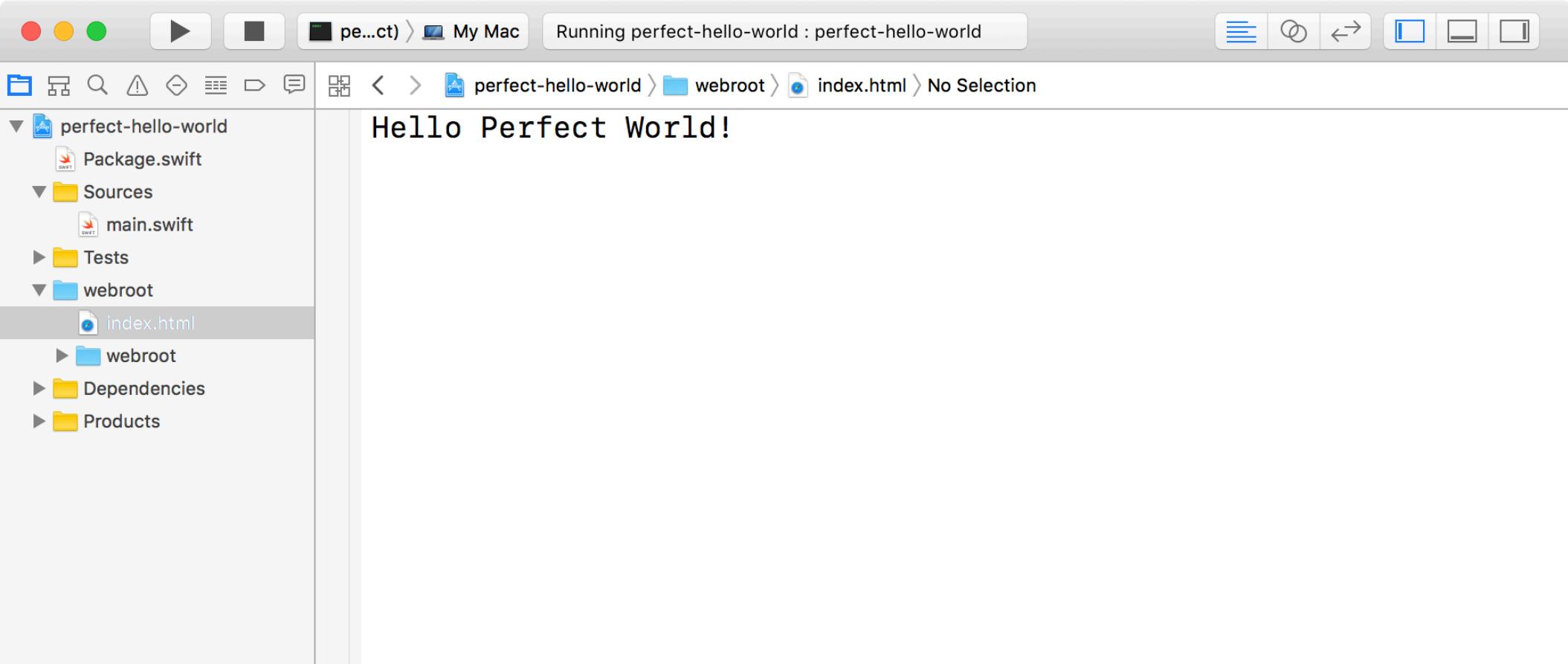
- Update the Xcode project

# PERFECT

```
perfect-hello-world — -bash ▶ postgres: stats collector process — 70×19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[653 % mkdir webroot
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[654 % touch webroot/hello.txt
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[655 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
656 %
```

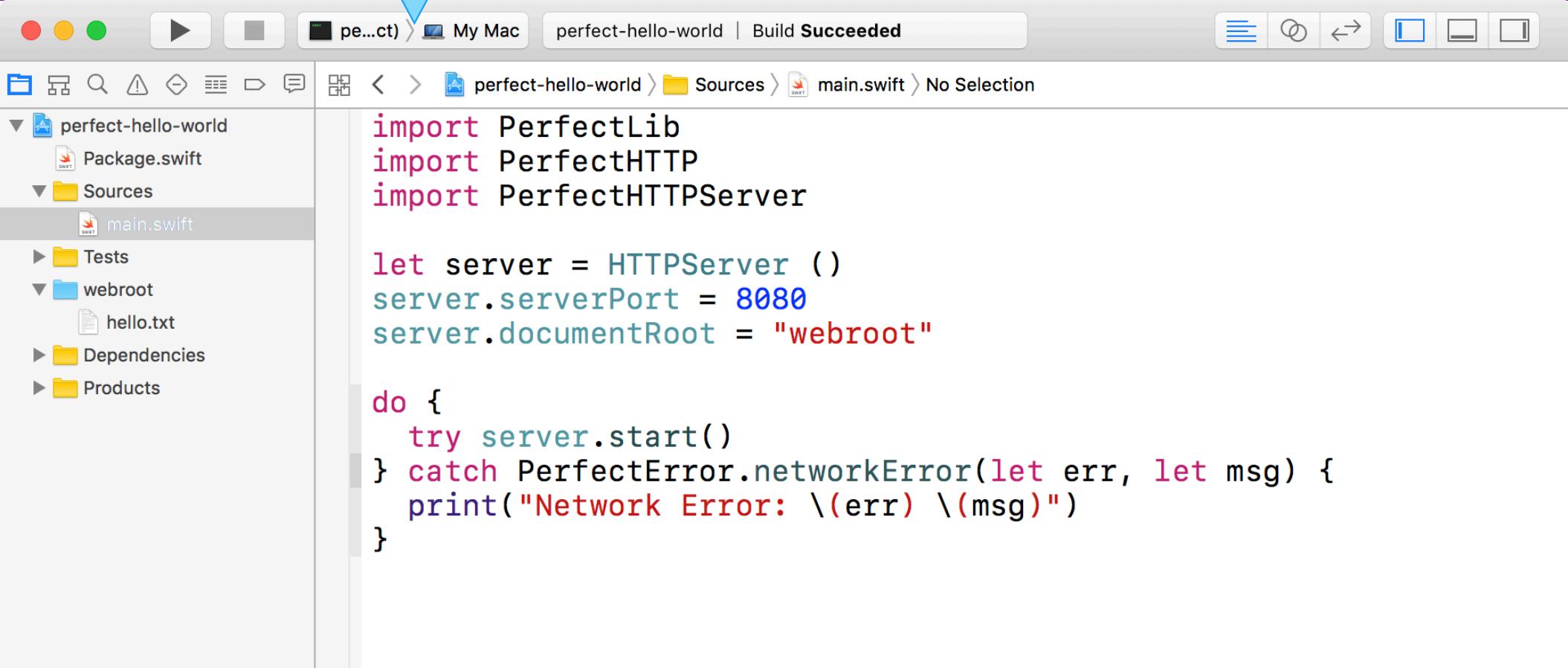
- Create a root directory for the web server

# PERFECT



# PERFECT

Create a server



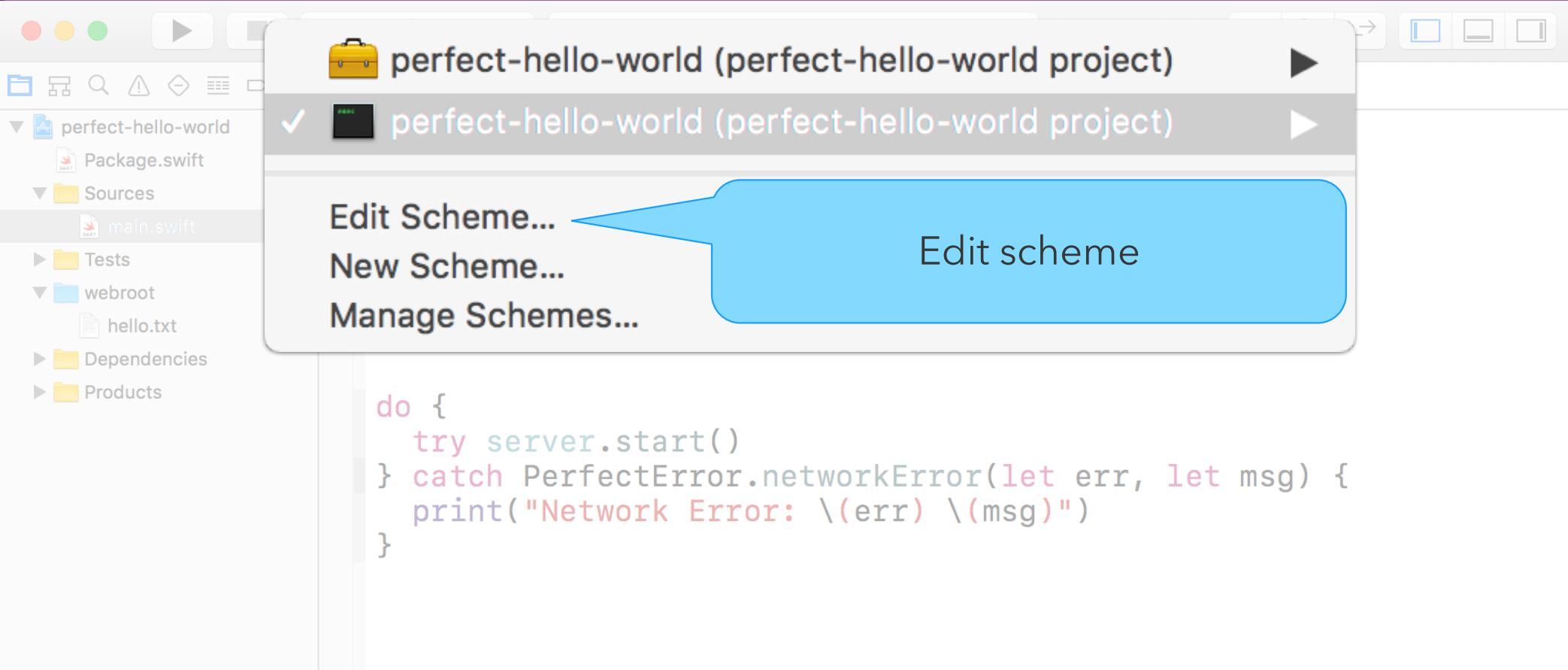
The screenshot shows a Mac OS X desktop with an Xcode interface. A blue callout bubble points from the text "Create a server" to the Xcode window. The Xcode window title bar says "pe...ct) My Mac perfect-hello-world | Build Succeeded". The left sidebar shows a project structure with a "perfect-hello-world" folder containing "Package.swift", "Sources" (with "main.swift" selected), "Tests", "webroot" (containing "hello.txt"), "Dependencies", and "Products". The main editor area displays the following Swift code:

```
import PerfectLib
import PerfectHTTP
import PerfectHTTPServer

let server = HTTPServer()
server.serverPort = 8080
server.documentRoot = "webroot"

do {
    try server.start()
} catch PerfectError.networkError(let err, let msg) {
    print("Network Error: \(err) \(msg)")
}
```

# PERFECT



# PERFECT

## Options

Build  
1 target

Run  
Debug

Test  
Debug

Profile  
Release

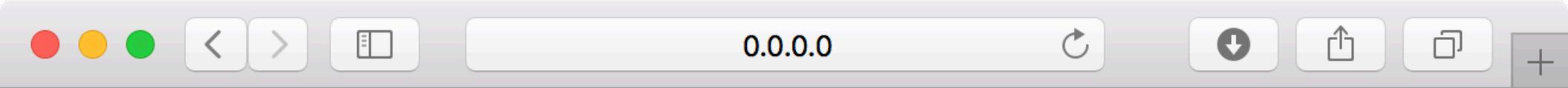
Analyze  
Debug

Archive  
Release

	Info	Arguments	Options	Diagnostics
Core Location	<input checked="" type="checkbox"/> Allow Location Simulation			
Default Location	<input type="button" value="None"/>			
Application Data	<input type="button" value="None"/>			
Routing App Coverage File	<input type="button" value="None"/>			
GPU Frame Capture	<input type="button" value="Automatically Enabled"/>			
Metal API Validation	<input type="button" value="Enabled"/>			
Persistent State	<input type="checkbox"/> Launch application without state restoration			
Document Versions	<input checked="" type="checkbox"/> Allow debugging when using document versions Browser			
Working Directory	<input checked="" type="checkbox"/> Use custom working directory:	<input type="text" value=""/>	<input type="button" value=""/>	
Localization Debugging	<input type="checkbox"/> Show non-localized strings			
Application Language	<input type="button" value="System Language"/>			

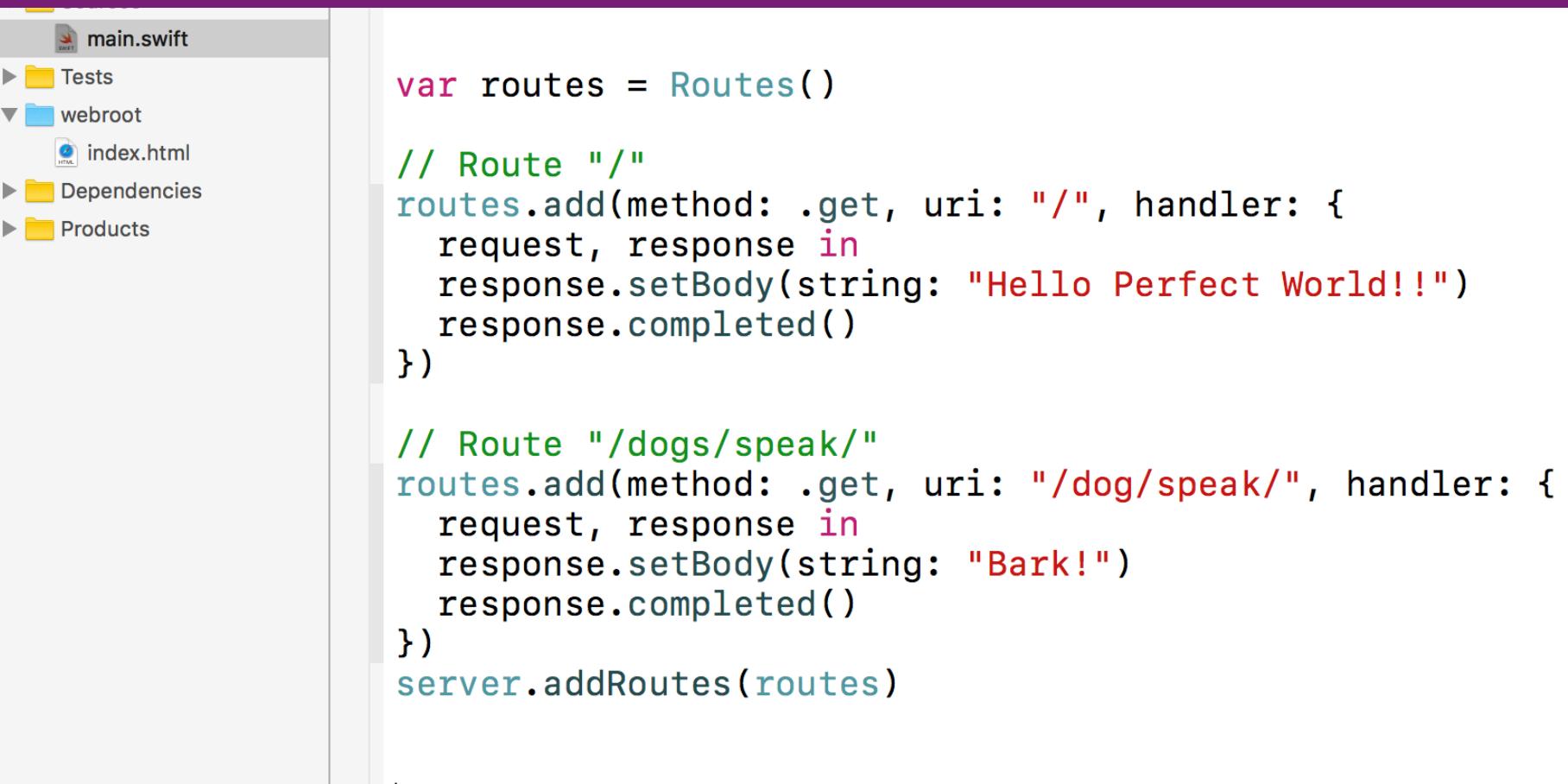
Working Directory

PERFECT



Hello Perfect World!

# PERFECT



The screenshot shows a code editor with a sidebar and a main editor area. The sidebar on the left lists project files: Tests, webroot (with index.html), Dependencies, and Products. The main editor area displays the following Swift code:

```
main.swift
Tests
webroot
index.html
Dependencies
Products

var routes = Routes()

// Route "/"
routes.add(method: .get, uri: "/", handler: {
    request, response in
    response.setBody(string: "Hello Perfect World!!")
    response.completed()
})

// Route "/dogs/speak/"
routes.add(method: .get, uri: "/dog/speak/", handler: {
    request, response in
    response.setBody(string: "Bark!")
    response.completed()
})
server.addRoutes(routes)
```

# PERFECT

The image shows a macOS desktop environment. On the left, there is a Xcode project sidebar with the following structure:

- main.swift (selected)
- Tests
- webroot
  - index.html
- Dependencies
- Products

The main editor area contains the following Swift code:

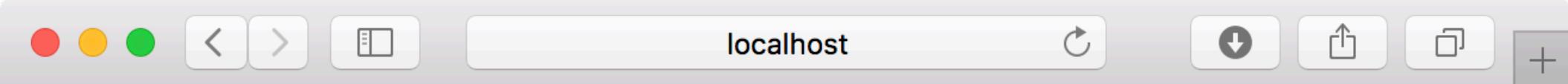
```
var routes = Route.Builder()  
// Route "/"  
routes.add(method: .get, uri: "/", handler: {  
    request, response in  
    response.setBody(string: "Bark!")  
    response.completed()  
})  
server.addRoutes(routes)
```

To the right of the editor, a web browser window is open at the URL `localhost`. The page title is "Bark!" and the content of the page is "Bark!".

# PERFECT

```
// Route "/dogs/{name}/"
routes.add(method: .get, uri: "/dog/speak/{name}/", handler: {
    request, response in
    guard let name = request.urlVariables["name"],
          let nameString = String(name) else {
        response.completed(status: .badRequest)
        return
    }
    response.setBody(string: "\(nameString) says Bark!")
    response.completed()
})
```

PERFECT



lassie says Bark!

<http://localhost:8088/dog/speak/lassie/>

# PERFECT ASSISTANT

## PERFECT ASSISTANT

- Set up new projects easily or download existing project templates
- Manage dependencies
- Create simultaneous macOS and Ubuntu builds on your local machine
- Configure Amazon deployment information
- Push projects up to EC2 servers



Perfect Assistant

Create, test and deploy your Xcode projects.

Manager to help you manage dependencies and Docker helps you build for Ubuntu, providing a smooth cross-platform build experience.

**Docker** is required to be installed and running to perform Xcode integrations and deployments.

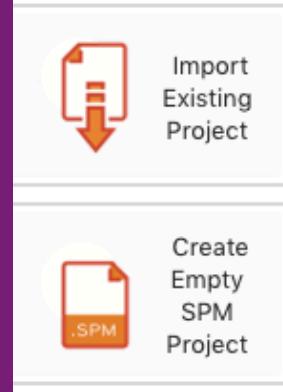
Docker Status: **Active**  
Docker is installed and running.

[Update Docker Image](#)

To deploy on EC2, ensure the AWS command line tools are installed and enter your AWS security credentials.

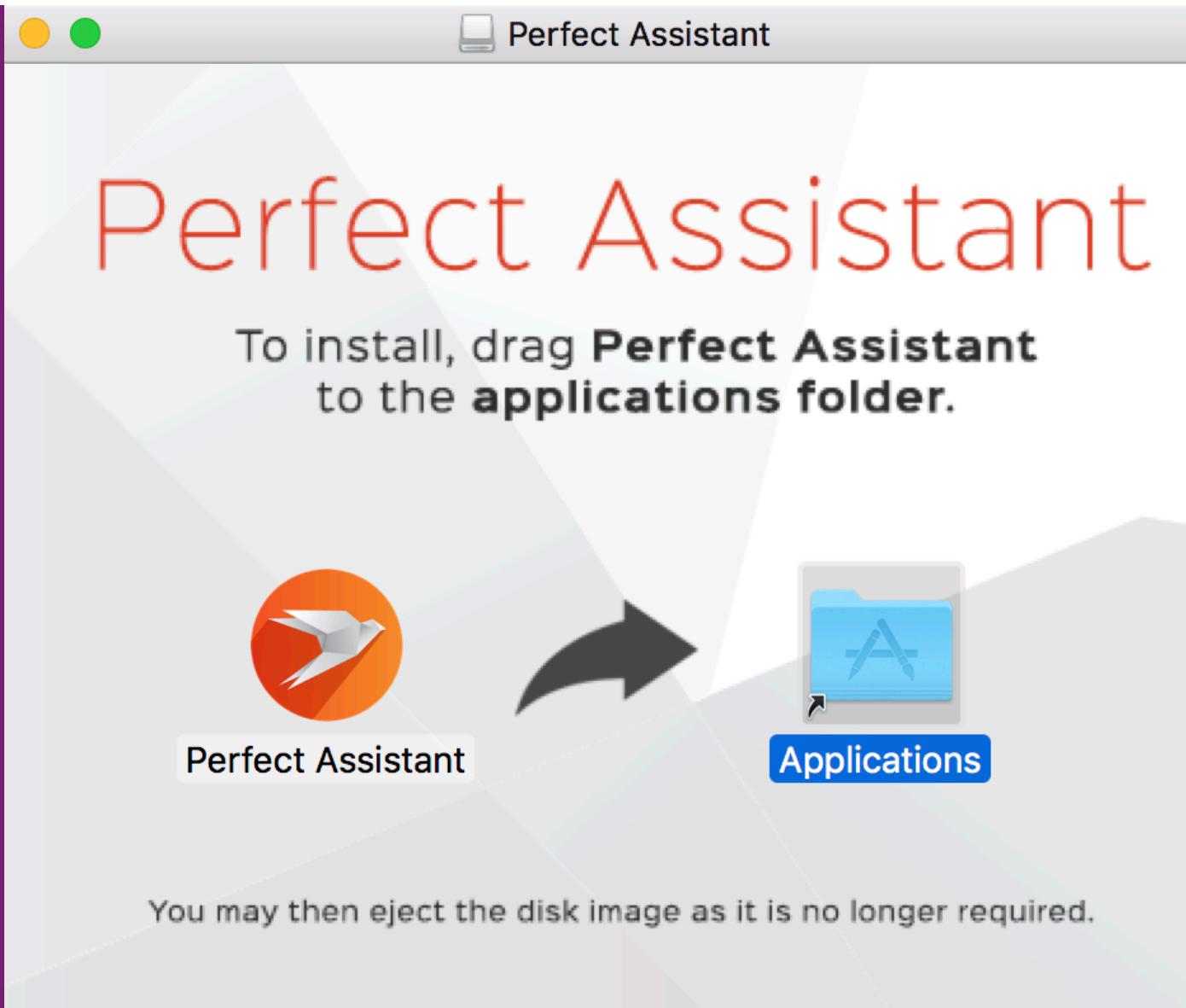
AWS Tools: **Active**  
AWS command line tools are installed.

[Configure EC2 Credentials...](#)



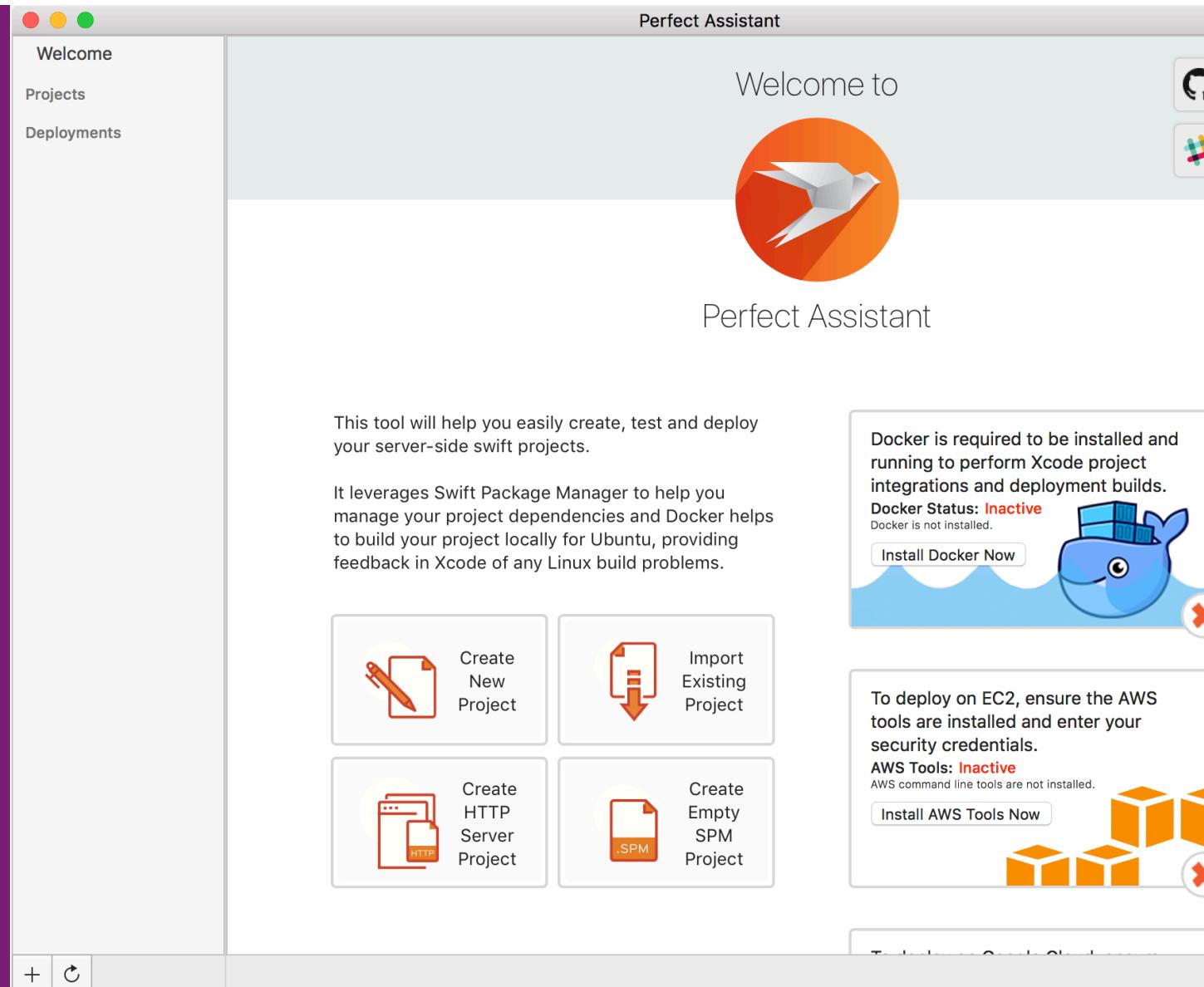
# PERFECT ASSISTANT

- Download and install



# PERFECT ASSISTANT

- Download and install



# PERFECT ASSISTANT

- Deploy using Docker

This tool will help you easily create, test and deploy your server-side swift projects.

It leverages Swift Package Manager to help you manage your project dependencies and Docker helps to build your project locally for Ubuntu, providing feedback in Xcode of any Linux build problems.



Create  
New  
Project



Import  
Existing  
Project



Create  
HTTP  
Server  
Project



Create  
Empty  
.SPM  
Project

## Perfect Assistant

Docker is required to be installed and running to perform Xcode project integrations and deployment builds.

**Docker Status: Inactive**  
Docker is not installed.

[Install Docker Now](#)



To deploy on EC2, ensure the AWS tools are installed and enter your security credentials.

**AWS Tools: Inactive**  
AWS command line tools are not installed.

[Install AWS Tools Now](#)



To deploy on Google Cloud, ensure the cloud tools are installed and enter your security credentials.

**Cloud Tools: Active**  
Cloud command line tools are installed.

[Open Cloud Console...](#)



# PERFECT ASSISTANT

- Deploy using Docker

This tool will help you easily create, test and deploy your server-side swift projects.

It leverages Swift Package Manager to help you manage your project dependencies and Docker helps to build your project locally for Ubuntu, providing feedback in Xcode of any Linux build problems.



Create  
New  
Project



Import  
Existing  
Project



Create  
HTTP  
Server  
Project



Create  
Empty  
.SPM  
Project

## Perfect Assistant

Docker is required to be installed and running to perform Xcode project integrations and deployment builds.

**Docker Status: Inactive**  
Docker is not installed.

[Install Docker Now](#)



To deploy on EC2, ensure the AWS tools are installed and enter your security credentials.

**AWS Tools: Inactive**  
AWS command line tools are not installed.

[Install AWS Tools Now](#)



To deploy on Google Cloud, ensure the cloud tools are installed and enter your security credentials.

**Cloud Tools: Active**  
Cloud command line tools are installed.

[Open Cloud Console...](#)



# PERFECT ASSISTANT

Perfect Assistant

Welcome Projects Deployments

## New Project

Project Deployment

Choose one of the starter project templates below.

**Basic**

- Perfect Template App
- Empty Executable Project
- Empty Library Project
- Custom Repository URL
- Perfect Template App Engine

**Examples**

- Perfect: Upload
- Perfect: URL
- Perfect:
- Perfect:
- Perfect:

What we just did.

# PERFECT ASSISTANT

Perfect Assistant

Welcome Projects Deployments

## New Project

Project Deployment

Choose one of the starter project templates below.

**Basic**

- Perfect Template App
- Empty Executable Project
- Empty Library Project
- Custom Repository URL
- Perfect Template App Engine

**Examples**

- Perfect: Upload
- Perfect: URL
- Perfect:
- Perfect:
- Perfect:

Create a project

# PERFECT ASSISTANT

Perfect Assistant

Welcome Projects Deployments

## New Project

Project Deployment

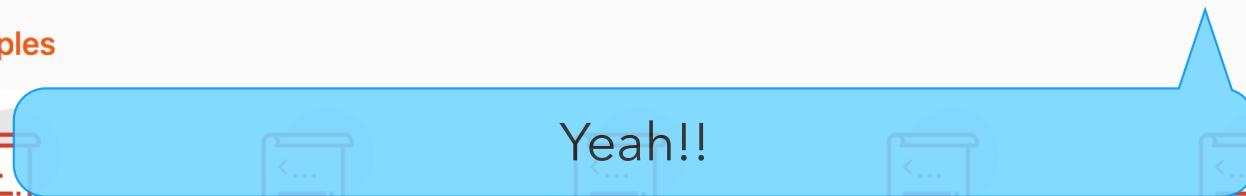
Choose one of the starter project templates below.

**Basic**

- Perfect Template App
- Empty Executable Project
- Empty Library Project
- Custom Repository URL
- Perfect Template App Engine

**Examples**

- Perfect: Upload
- Perfect: URL
- Perfect: Yeah!!
- Perfect:
- Perfect:



# PERFECT ASSISTANT

 Perfect Template App	 Empty Executable Project	 Empty Library Project	 Custom Repository URL	 Perfect Template App Engine
<b>Examples</b>				
 Perfect: Upload Enumerator	 Perfect: URL Routing	 Perfect: WebSockets Server	 Perfect: Turnstile with SQLite	 Perfect: Turnstile with PostgreSQL
 Perfect: Weather	 Perfect: Polling	 Perfect: Yeah!!	 Perfect: Blog Mustache	 Perfect: JSON API

[Cancel](#) [Previous](#) [Next](#)

# PERFECT ASSISTANT

Feedback in Xcode or any Linux build problems.

The screenshot shows the Perfect Assistant interface. At the top, there are two large buttons: "Create New Project" (with a pencil icon) and "Import Existing Project" (with a downward arrow icon). Below these are three smaller buttons: "Create HTTP Server Project" (with a folder icon labeled ".HTTP"), "Create SPM Project" (with a folder icon labeled ".SPM"), and "Empty SPM Project" (with a folder icon labeled ".SPM"). A blue callout bubble points to the "Create HTTP Server Project" button with the text "Create a new project". To the right of this callout, there is a note about deploying to EC2: "To deploy on EC2, ensure the AWS tools are installed and enter your security credentials." Below this note is a button labeled "Install AWS Tools Now". In the bottom left corner, there is a status message: "Creating /Users/tabinkowski/swift-server/perfect/perfect-template • ⚡". In the bottom right corner, there is a "Clear" button. The background features a cartoon character of a blue fish swimming in water.

Create New Project

Import Existing Project

Create a new project

To deploy on EC2, ensure the AWS tools are installed and enter your security credentials.

Install AWS Tools Now

Creating /Users/tabinkowski/swift-server/perfect/perfect-template • ⚡

Clear

Project saved.  
Creating working directory and files  
Backed up existing package file  
Created package file

# PERFECT ASSISTANT

Perfect Assistant

Welcome

Projects perfect-template

Deployments

project: perfect-template

location: /Users/tabinkowski/swift-server/perfect/perfect-template

Add dependencies for your project

OPEN

- Project Directory
- Project Terminal
- Xcode Project

BUILD

- Local
- Linux
- Deploy
- Clean

DEPENDENCIES

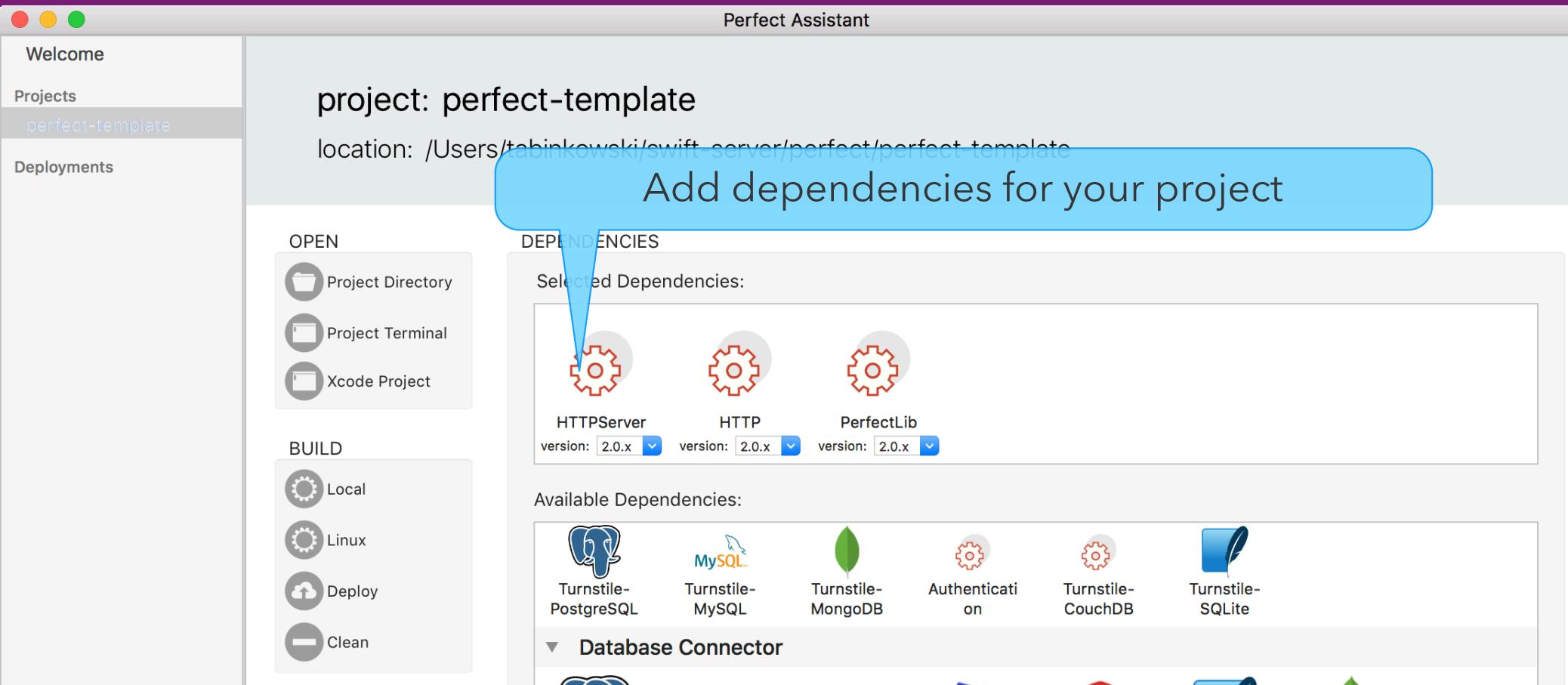
Selected Dependencies:

- HTTPServer version: 2.0.x
- HTTP version: 2.0.x
- PerfectLib version: 2.0.x

Available Dependencies:

- Turnstile-PostgreSQL
- Turnstile-MYSQL
- Turnstile-MongoDB
- Authentication
- Turnstile-CouchDB
- Turnstile-SQLite

Database Connector

The screenshot shows the Perfect Assistant application window. On the left is a sidebar with tabs for Welcome, Projects (selected), Deployments, OPEN (with Project Directory, Project Terminal, Xcode Project), and BUILD (with Local, Linux, Deploy, Clean). The main area displays project information: 'project: perfect-template' and 'location: /Users/tabinkowski/swift-server/perfect/perfect-template'. A large blue callout bubble points to the 'DEPENDENCIES' section with the text 'Add dependencies for your project'. Below this, under 'DEPENDENCIES', there's a 'Selected Dependencies:' section showing three selected components: 'HTTPServer' (version 2.0.x), 'HTTP' (version 2.0.x), and 'PerfectLib' (version 2.0.x). Under 'Available Dependencies:', there are six listed: Turnstile-PostgreSQL, Turnstile-MYSQL, Turnstile-MongoDB, Authentication, Turnstile-CouchDB, and Turnstile-SQLite. A section titled 'Database Connector' is partially visible at the bottom. The overall interface is clean with a light gray background and blue highlights.

# PERFECT ASSISTANT

The screenshot shows the Perfect Assistant application interface. On the left, there are two vertical panels: 'BUILD' and 'RUN'. The 'BUILD' panel contains icons for Local, Linux, Deploy, and Clean. The 'RUN' panel contains icons for Local Tests, Linux Tests, Local Exe, and Linux Exe. At the top, there are five dropdown menus labeled 'HTTPServer', 'HTTP', 'PerfectLIB', 'PostgreSQL', and 'Postgres-STORM', each with a version selector (e.g., 'version: 2.0.x'). A large blue callout box is positioned over the center of the screen, containing the text 'Available Dependencies:' and a grid of dependency icons. The grid includes MySQL-StORM, CouchDB-StORM, SQLite-StORM, Postgres-StORM, MongoDB-StORM, WebRedirects, HTTPServer, WebSockets, HTTP, FastCGI, Session-CouchDB, Session-SQLite, Session-MYSQL, Session-PostgreSQL, Session-MongoDB, and a general Session icon.

BUILD

- Local
- Linux
- Deploy
- Clean

RUN

- Local Tests
- Linux Tests
- Local Exe
- Linux Exe

HTTPServer version: 2.0.x HTTP version: 2.0.x PerfectLIB version: 2.0.x PostgreSQL version: 2.x.x Postgres-STORM version: 1.x.x

Available Dependencies:

- MySQL-StORM
- CouchDB-StORM
- SQLite-StORM
- Postgres-StORM
- MongoDB-StORM
- WebRedirects
- HTTPServer
- WebSockets
- HTTP
- FastCGI
- Session-CouchDB
- Session-SQLite
- Session-MYSQL
- Session-PostgreSQL
- Session-MongoDB
- Session

Build and test (mac and linux)

# PERFECT ASSISTANT



Local



Linux



Deploy



Clean

## RUN



Local Tests



Linux Tests



Local Exe



Linux Exe

## XCODE PROJECT



Regenerate

## Available Dependencies:

### ▼ Authentication

Turnstile-  
PostgreSQLTurnstile-  
MySQLTurnstile-  
MongoDBAuthenticati  
onTurnstile-  
CouchDBTurnstile-  
SQLite

### ▼ Database Connector



PostgreSQL



MySQL



CouchDB



FileMaker



Redis



SQLite



MongoDB

### ▼ ORM

MySQL-  
StORMCouchDB-  
StORMSQLite-  
StORMPostgres-  
StORMMongoDB-  
StORM

### ▼ Server



# PERFECT ASSISTANT

Perfect Assistant

project: perfect-template Postgresql setup for swift server  
location: /Users/tabinkowski/swift-server/perfect/perfect-template

OPEN

- Project Directory
- Project Terminal
- Xcode Project

BUILD

- Local
- Linux
- Deploy
- Clean

DEPENDENCIES

Selected Dependencies:

 HTTPServer	 HTTP	 PerfectLib	 PostgreSQL	 Postgres-StORM
version: 2.0.x	version: 2.0.x	version: 2.0.x	version: 2.x.x	version: 1.x.x

Available Dependencies:

▼ Authentication

 Turnstile-PostgreSQL	 Turnstile-MYSQL	 Turnstile-MongoDB	 Authentication	 Turnstile-CouchDB	 Turnstile-SQLite
--	--	---	--	---	--

# PERFECT ASSISTANT

The screenshot shows the Perfect Assistant interface with several components:

- Session Drivers:** Session-touchDB, Session-SQLite, Session-MySQL, Session-PostgreSQL, Session-MongoDB, Session.
- Utility Components:** questLogger, Logger, Notifications, OpenSSL, Logger, Repeater, XML, PerfectLib.
- Network Components:** Net, Zip, SMTP, Thread, Mustache, CURL.

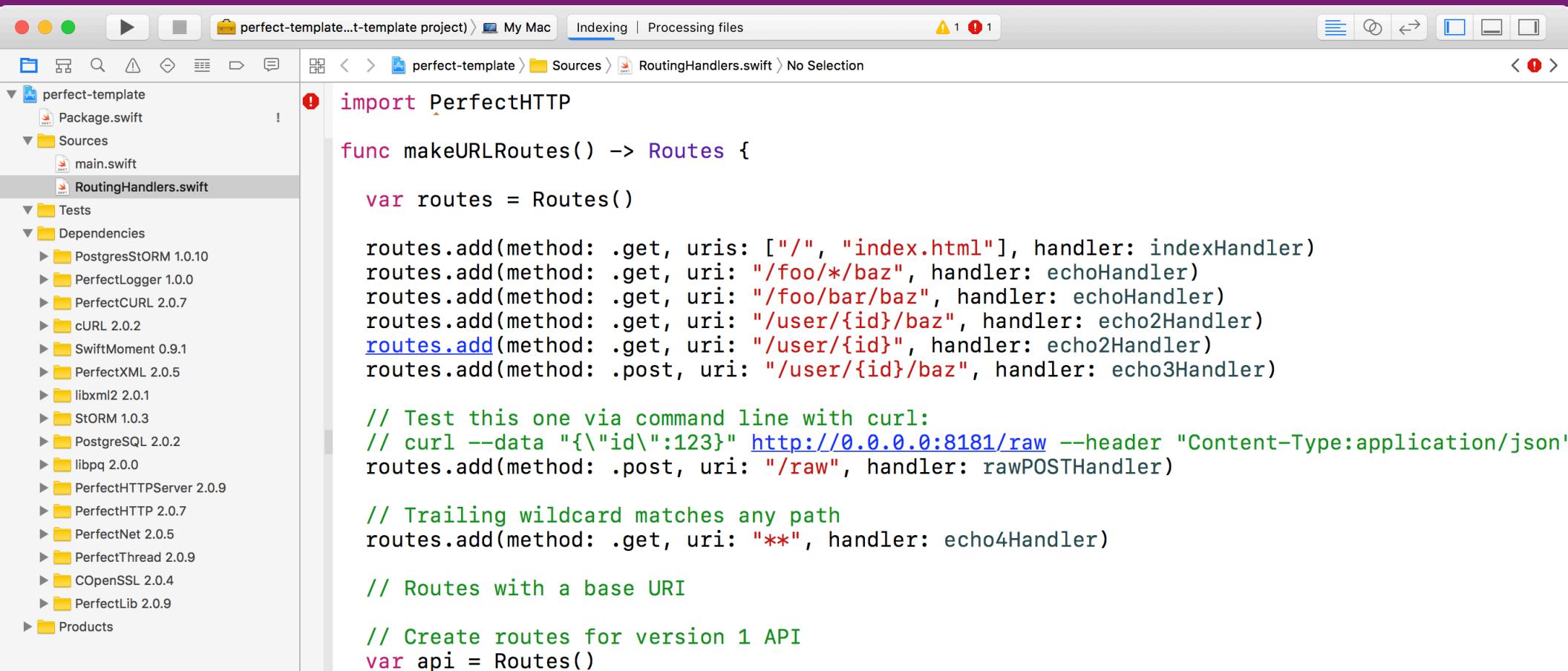
A blue callout bubble points to the "Logger" component under "Utility". The text inside the bubble reads: "Save and update the project".

Add Dependency...

Automatically integrate Xcode  
when regenerating project

Save Changes

# PERFECT ASSISTANT



The screenshot shows a Mac OS X desktop with an Xcode interface. The title bar reads "perfect-template...t-template project > My Mac". The status bar shows "Indexing | Processing files" and "1 1 1". The main window displays a Swift file named "RoutingHandlers.swift" under the "Sources" folder of the "perfect-template" project. The code implements a routing system using the PerfectHTTP framework.

```
! import PerfectHTTP

func makeURLRoutes() -> Routes {
    var routes = Routes()

    routes.add(method: .get, uris: ["/", "index.html"], handler: indexHandler)
    routes.add(method: .get, uri: "/foo/*baz", handler: echoHandler)
    routes.add(method: .get, uri: "/foo/bar/baz", handler: echoHandler)
    routes.add(method: .get, uri: "/user/{id}/baz", handler: echo2Handler)
    routes.add(method: .get, uri: "/user/{id}", handler: echo2Handler)
    routes.add(method: .post, uri: "/user/{id}/baz", handler: echo3Handler)

    // Test this one via command line with curl:
    // curl --data "{\"id\":123}" http://0.0.0.0:8181/raw --header "Content-Type:application/json"
    routes.add(method: .post, uri: "/raw", handler: rawPOSTHandler)

    // Trailing wildcard matches any path
    routes.add(method: .get, uri: "**", handler: echo4Handler)

    // Routes with a base URI

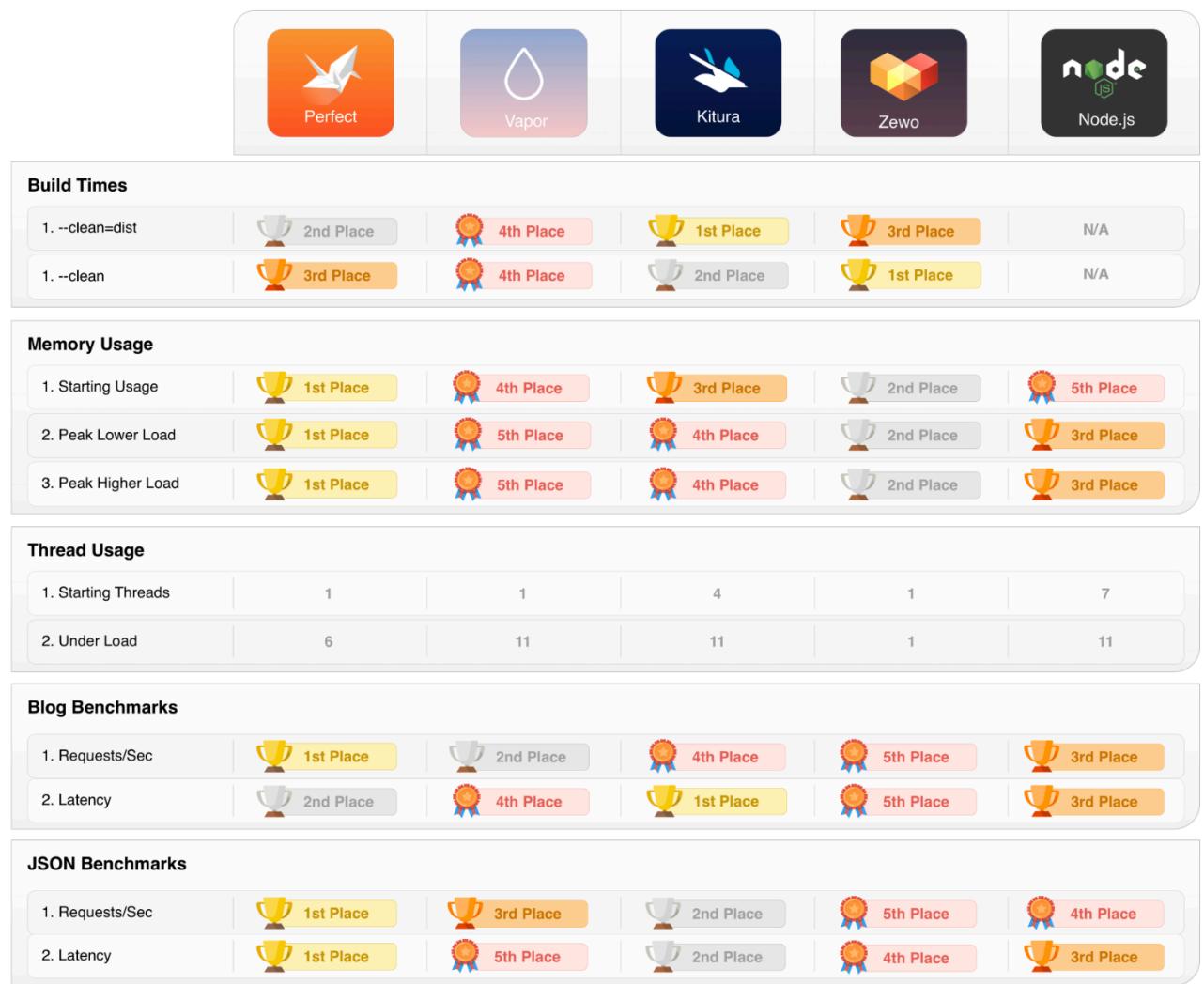
    // Create routes for version 1 API
    var api = Routes()
```

# THE FUTURE OF SWIFT ON THE SERVER

# PERFORMANCE

- <https://medium.com/@rymcol/benchmarks-for-the-top-server-side-swift-frameworks-vs-node-js-24460cfe0beb>

## Results Summary



## FUTURE OF SWIFT ON THE SERVER

- Great interest, tools, support
- Many projects that are highly active (and with industry support)
  - Apple
  - IBM
  - Prefect

## PRODUCTION READY

- Is it ready for prime time? Probably not yet.
- Missing frameworks
  - No URLSession
- The frameworks will make the shared code promise hard to keep
  - Custom models still need to be converted

# ASSIGNMENT 5

# ASSIGNMENT 4

- Create a simple address book backend using a Swift Framework
  - Create
  - Retrieve
  - Update
  - Delete
- Deploy to a Paas



THE UNIVERSITY OF  
CHICAGO



MPCS 51033 • SPRING 2017 • SESSION 6

---

# BACKENDS FOR MOBILE APPLICATIONS