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

- It all started here

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.

Swift.org

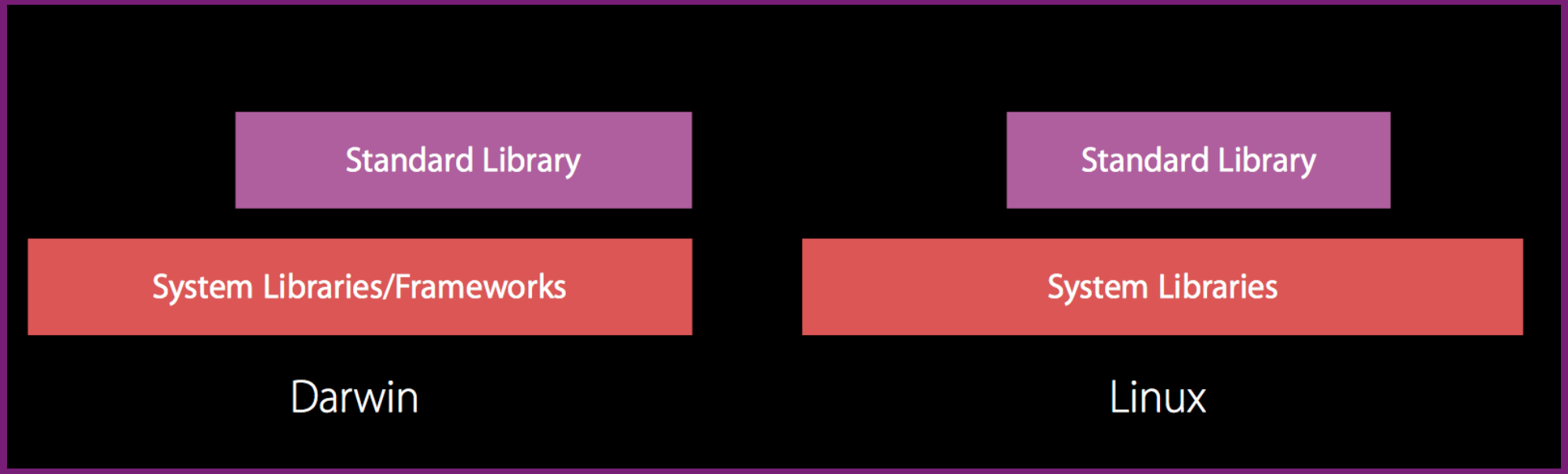
Swift.org is an entirely new site dedicated to open source Swift. This site hosts resources for the community of developers that want to help evolve Swift, contribute fixes, and most importantly, interact with each other. Swift.org hosts:

- 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

OPEN SOURCE SWIFT

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

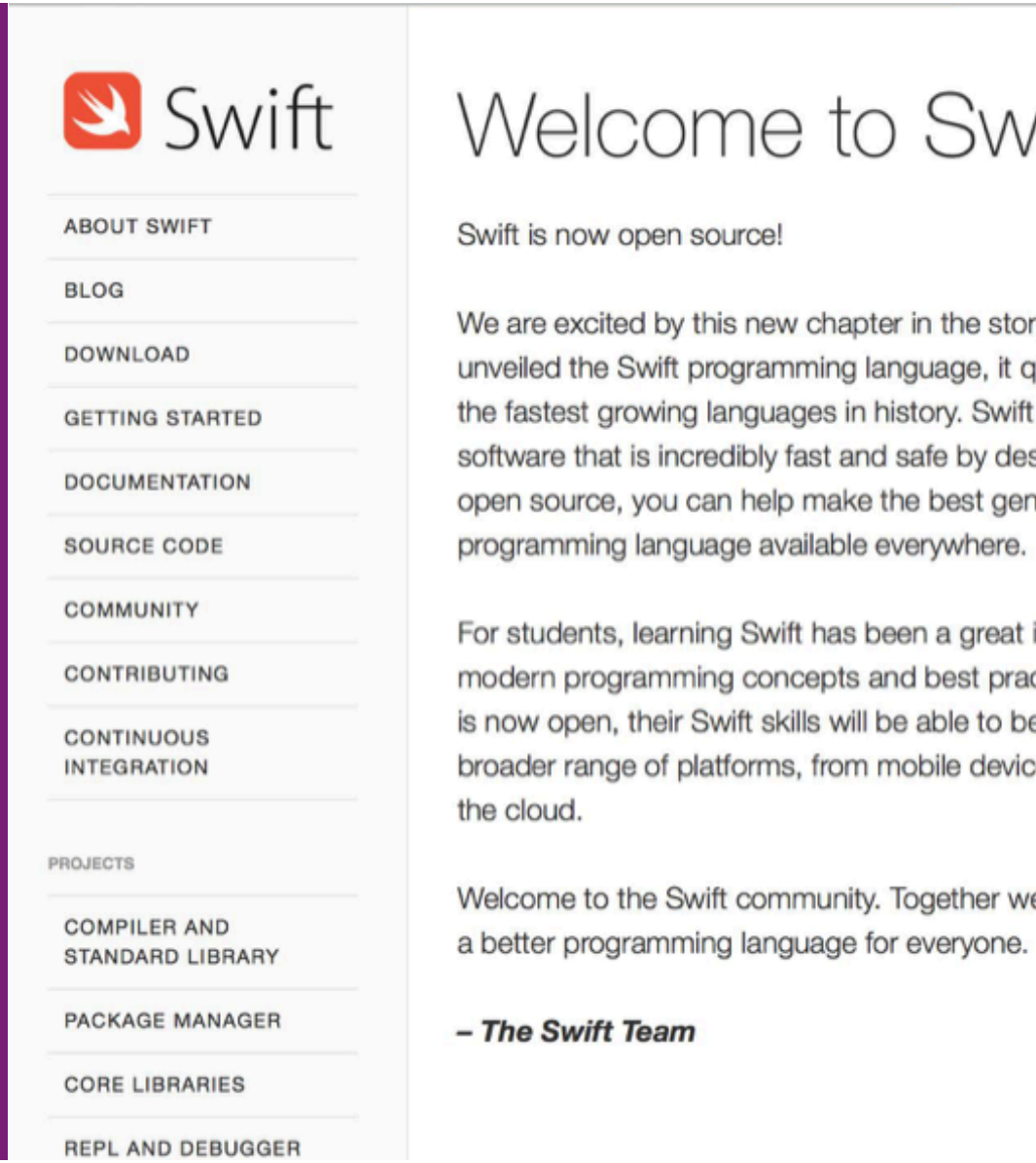
OPEN SOURCE SWIFT



- Something is missing...

OPEN SOURCE SWIFT

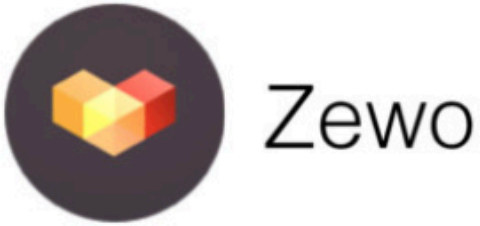
- Linux and Mac Platform support
- Standard Library Foundation, Dispatch, and XCTest Compiler
- Command Line Tools



STATE OF SWIFT ON THE SERVER

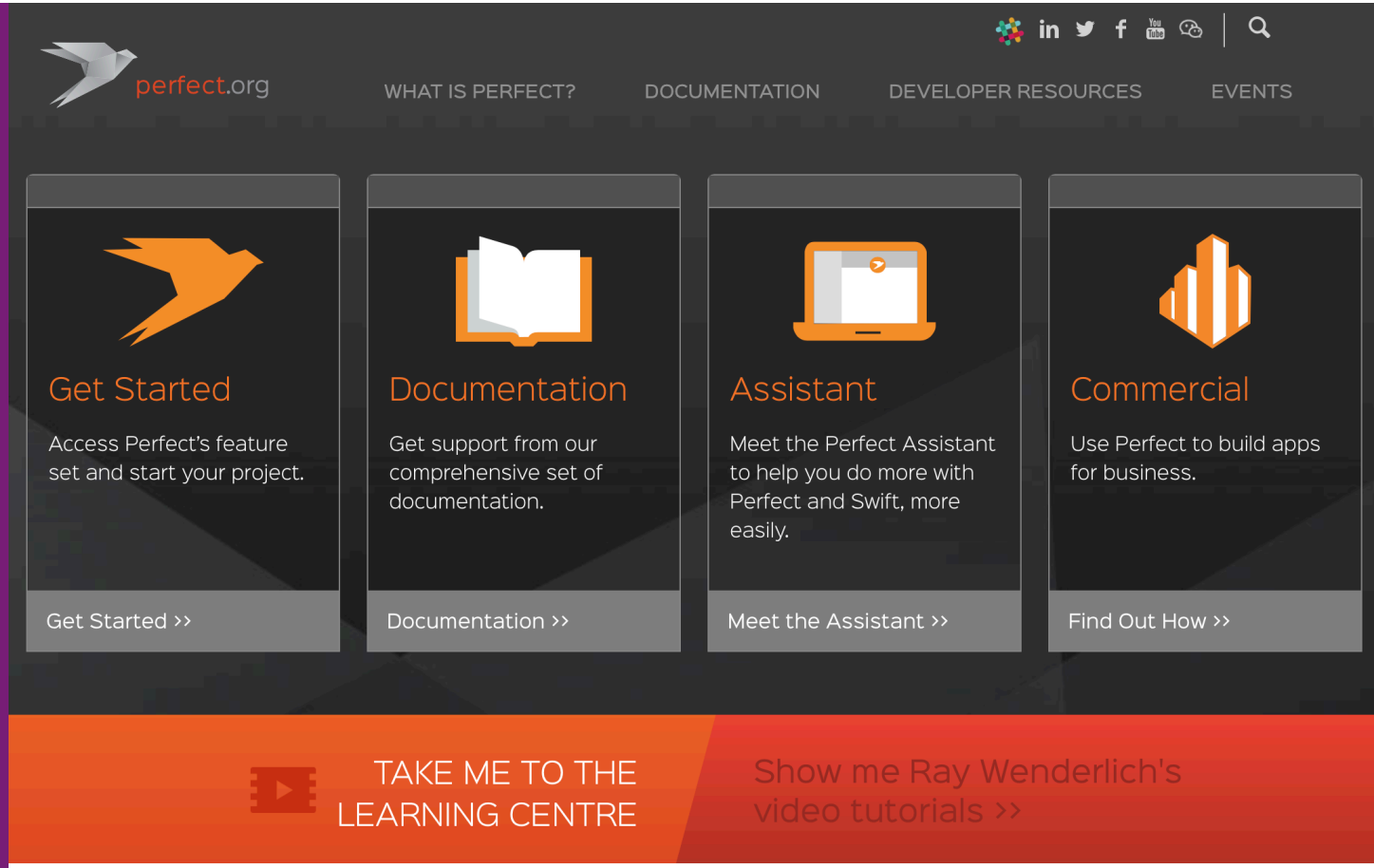
STATE OF SWIFT ON THE SERVER

- 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



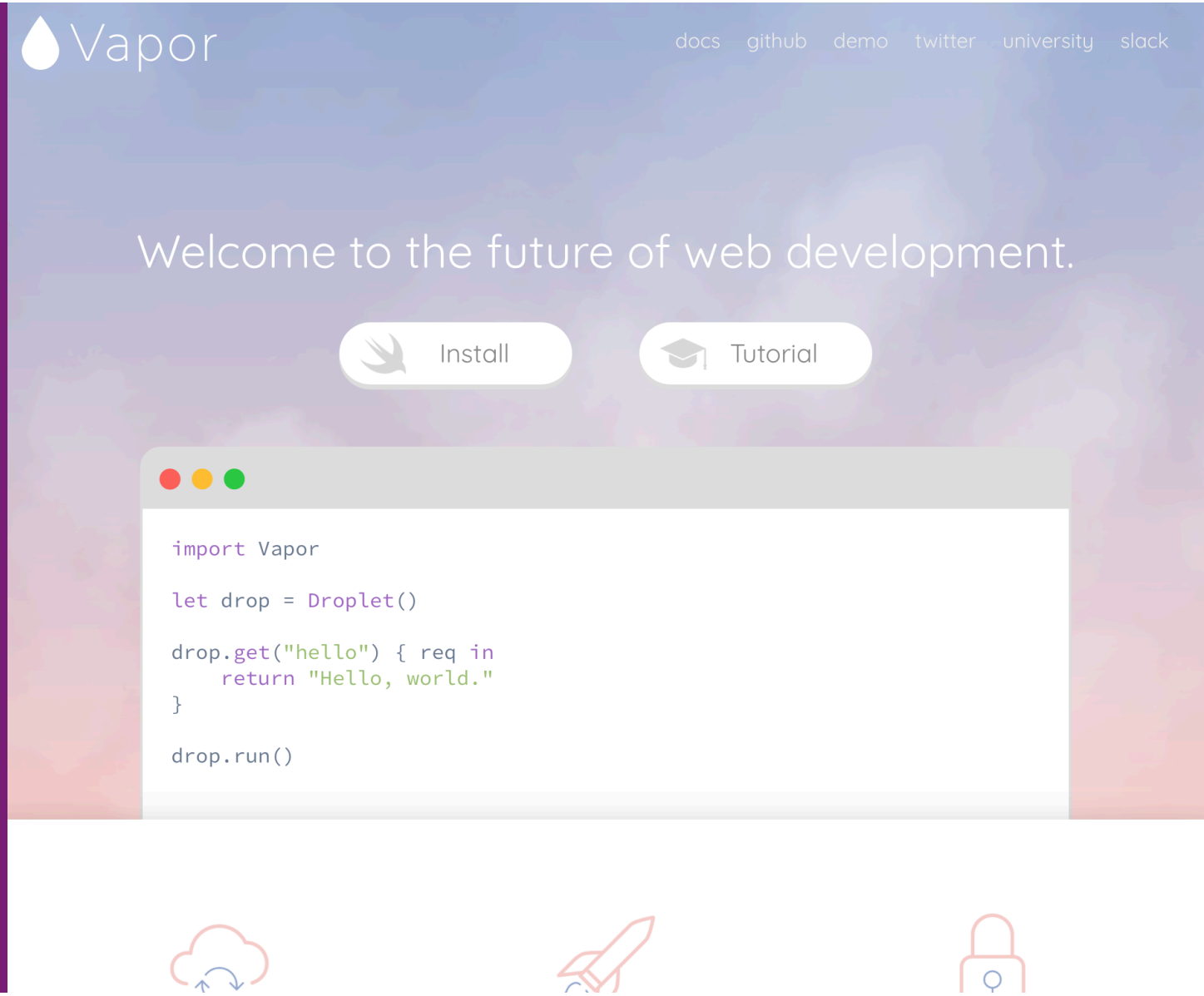
STATE OF SWIFT ON THE SERVER

- Perfect
  - A complete framework
  - Runs it 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



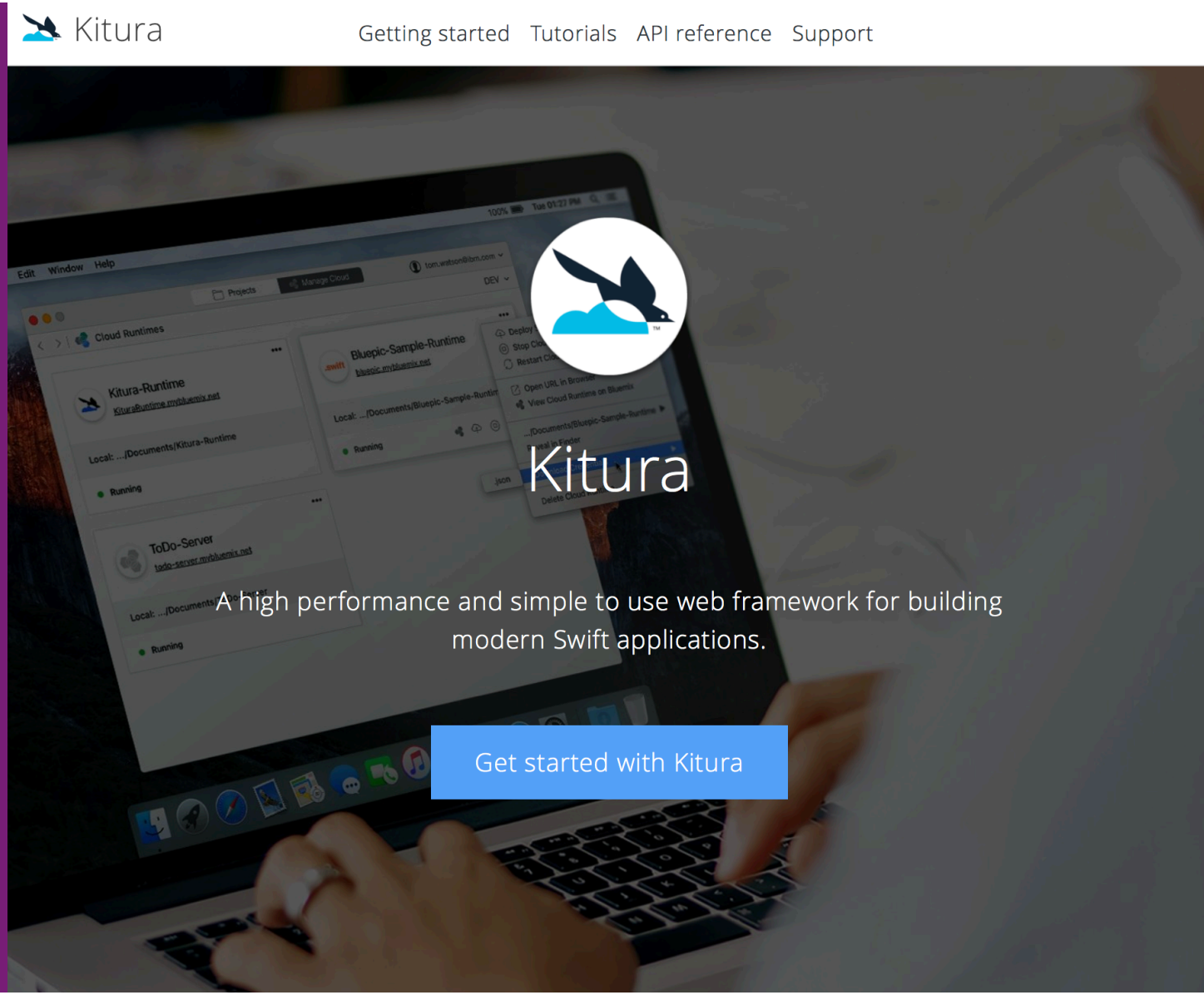
STATE OF SWIFT ON THE SERVER

- Perfect
  - A complete framework
  - Runs it 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



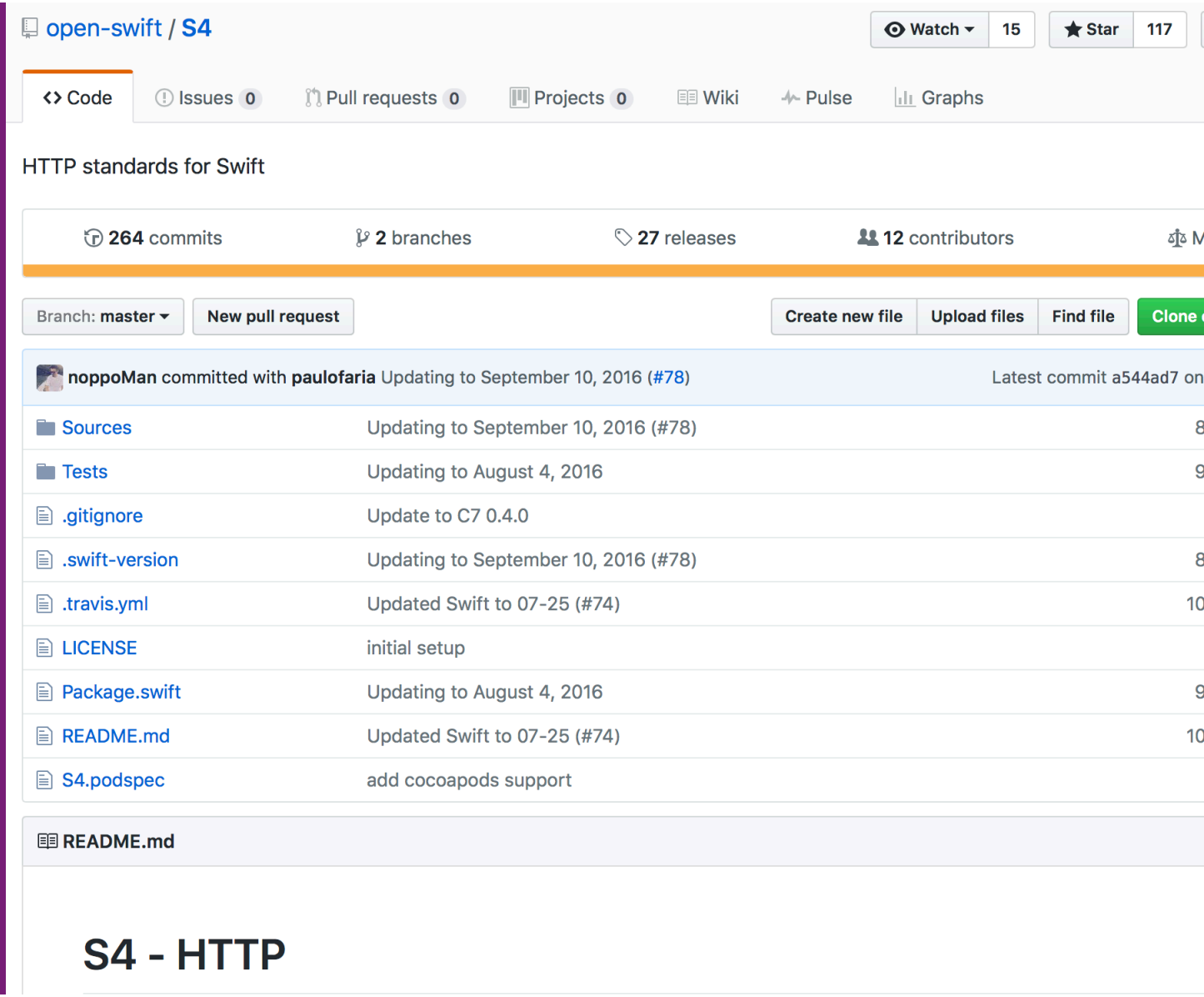
STATE OF SWIFT ON THE SERVER

- 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



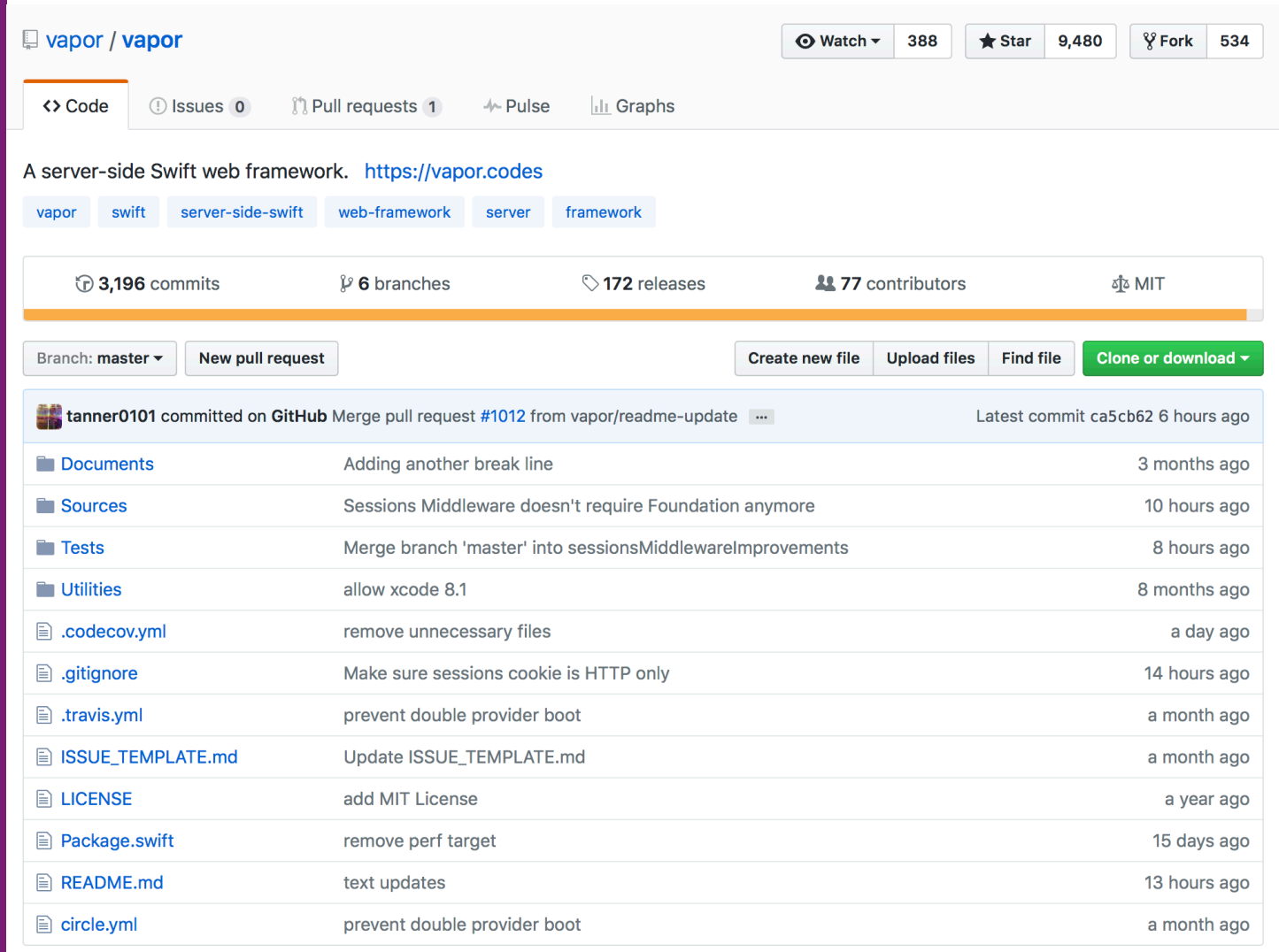
STATE OF SWIFT ON THE SERVER

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

VAPOR

VAPOR

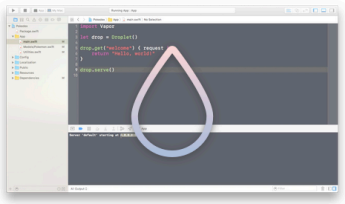
- Web framework and server for Swift
- Works on macOS and Ubuntu



VAPOR

- Great documentation and resources

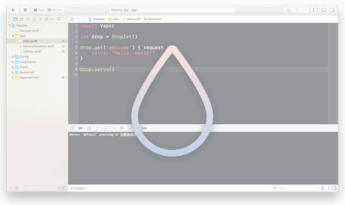
vapor.university



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

VAPOR

VAPOR Docs

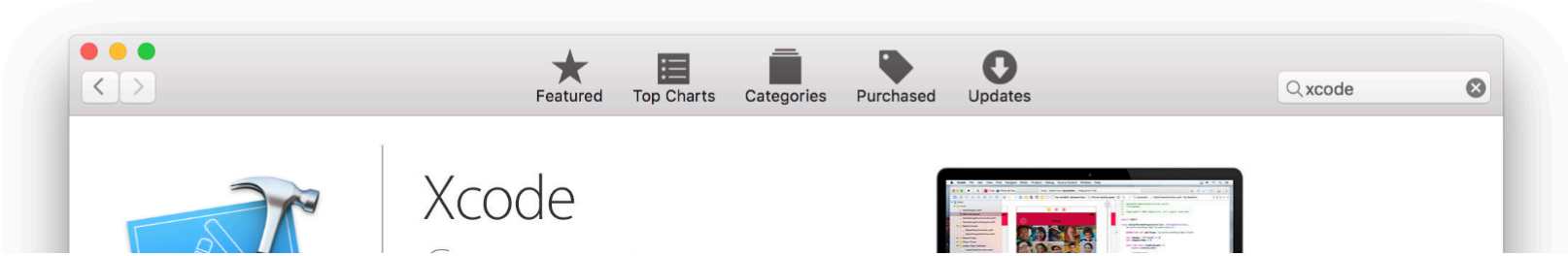
✎ Edit on GitHub

Install Swift 3: macOS

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

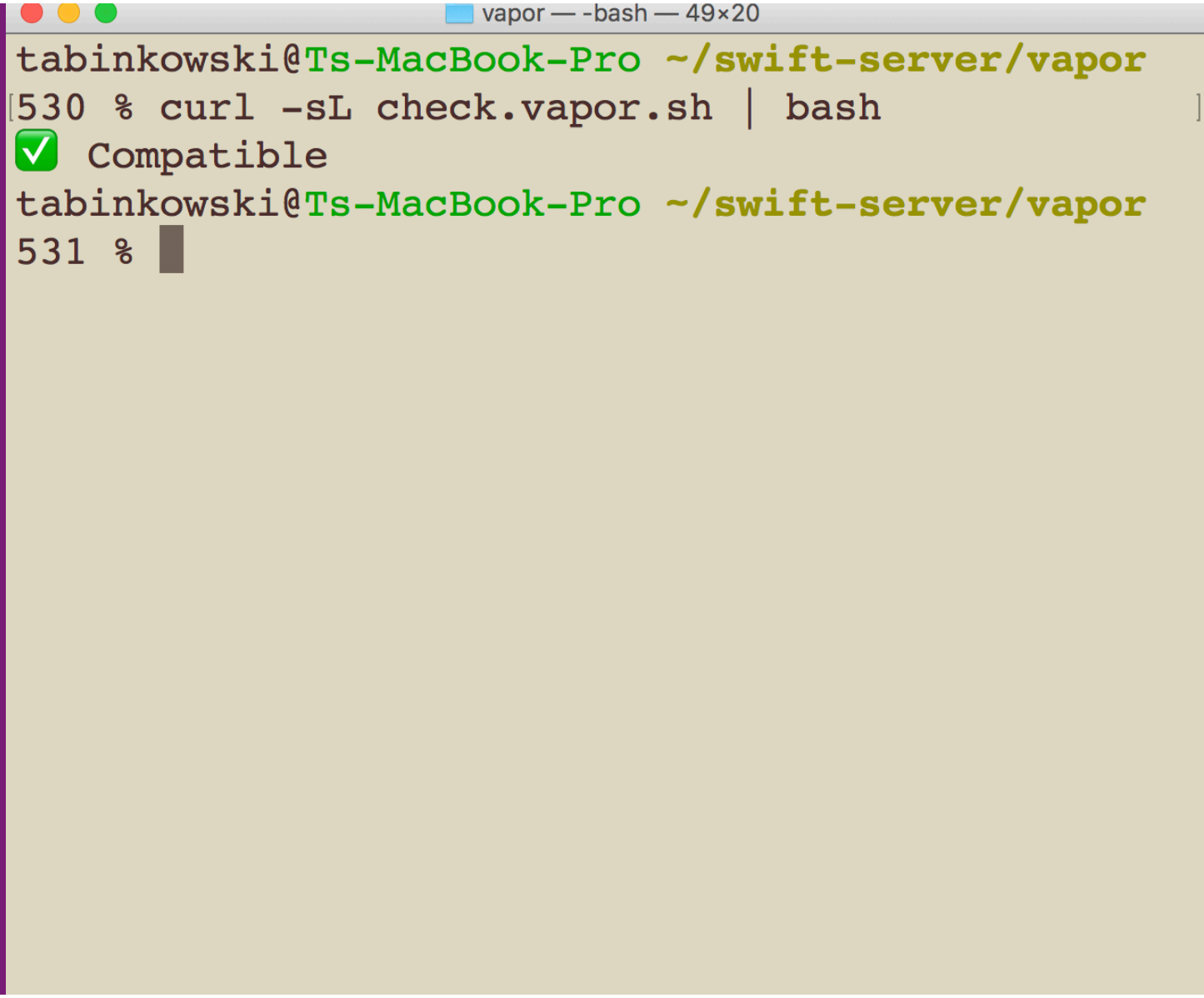
Install Xcode

Install Xcode 8 from the Mac App Store.



VAPOR

- Install Xcode
- Check version for vapor

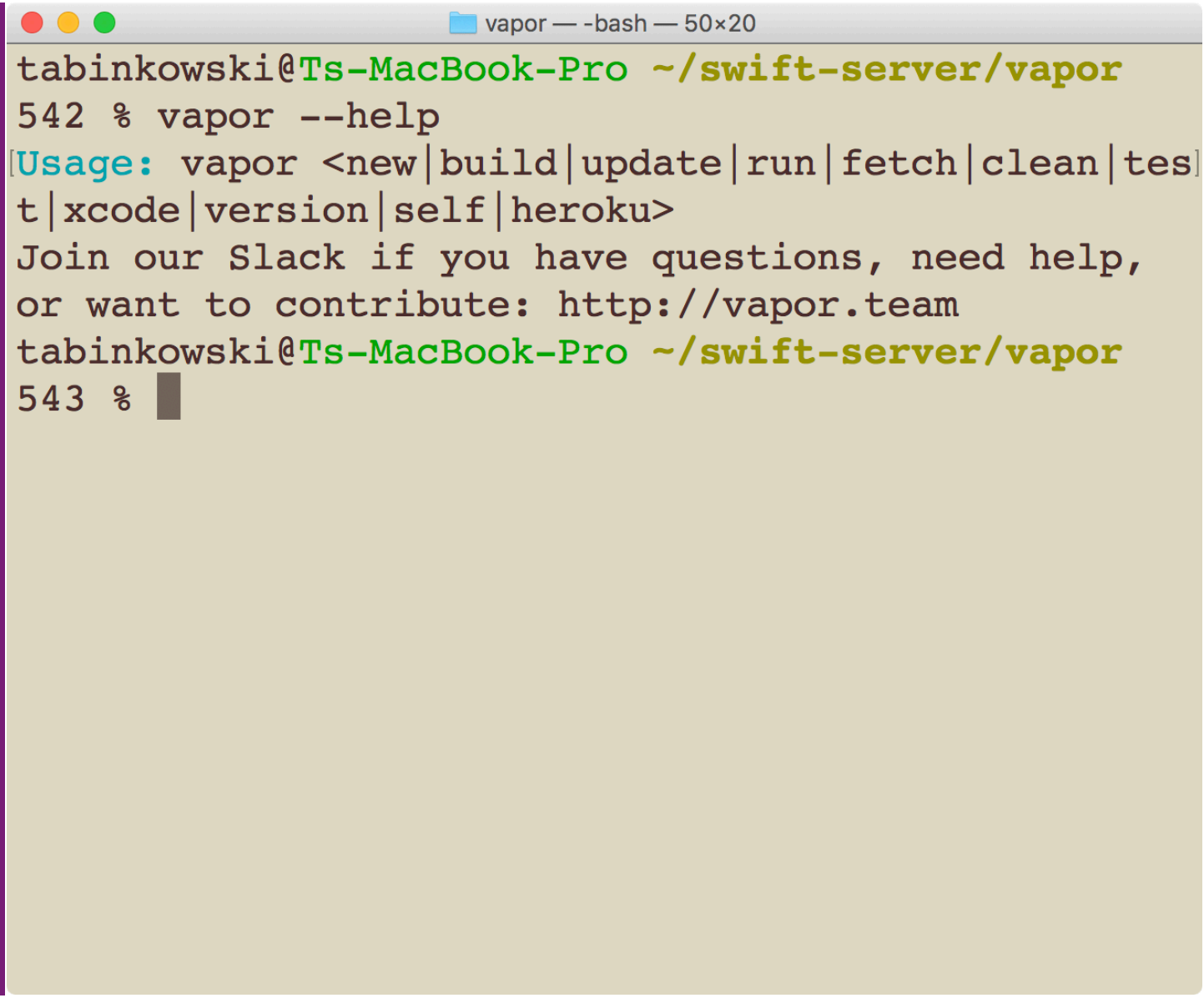


VAPOR

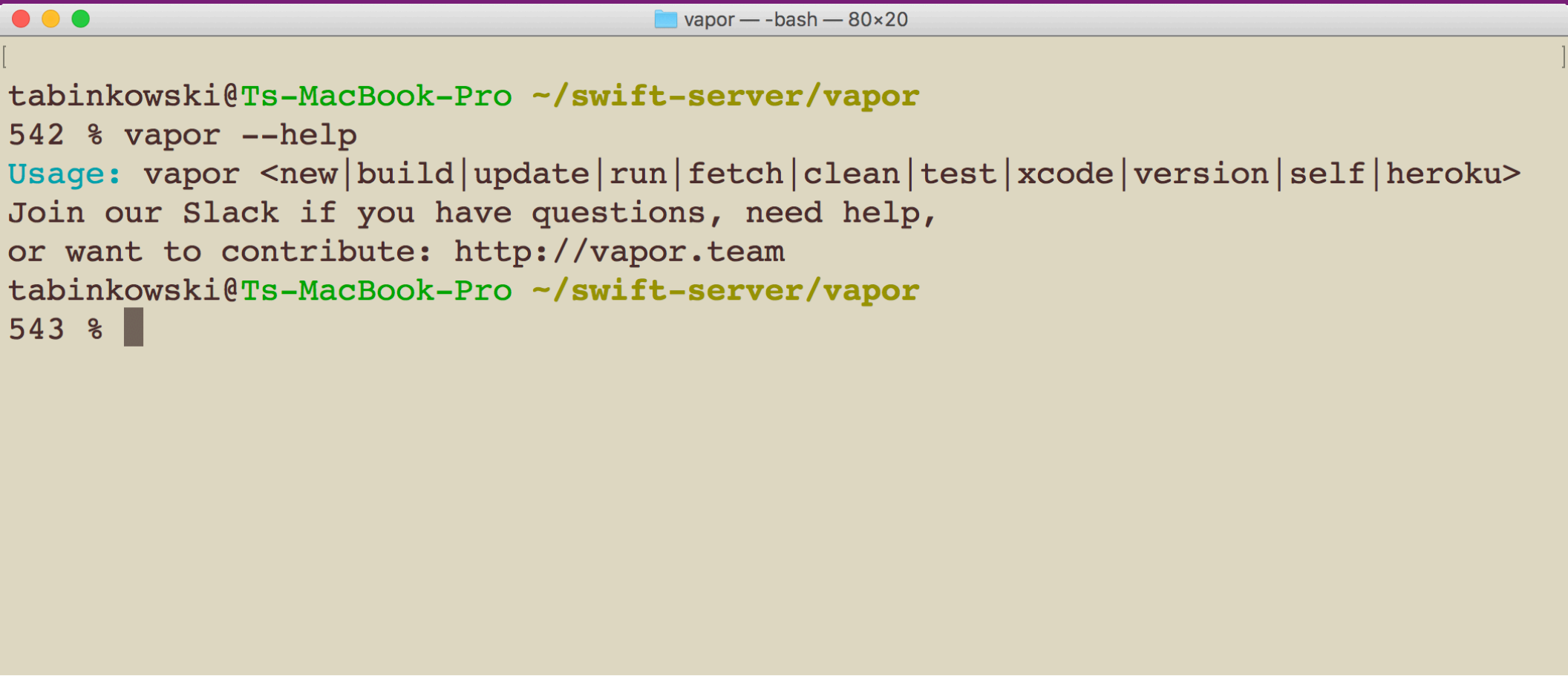
- Toolbox provides command line and shortcuts for common tasks
- Install with homebrew 🍺

VAPOR

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



VAPOR

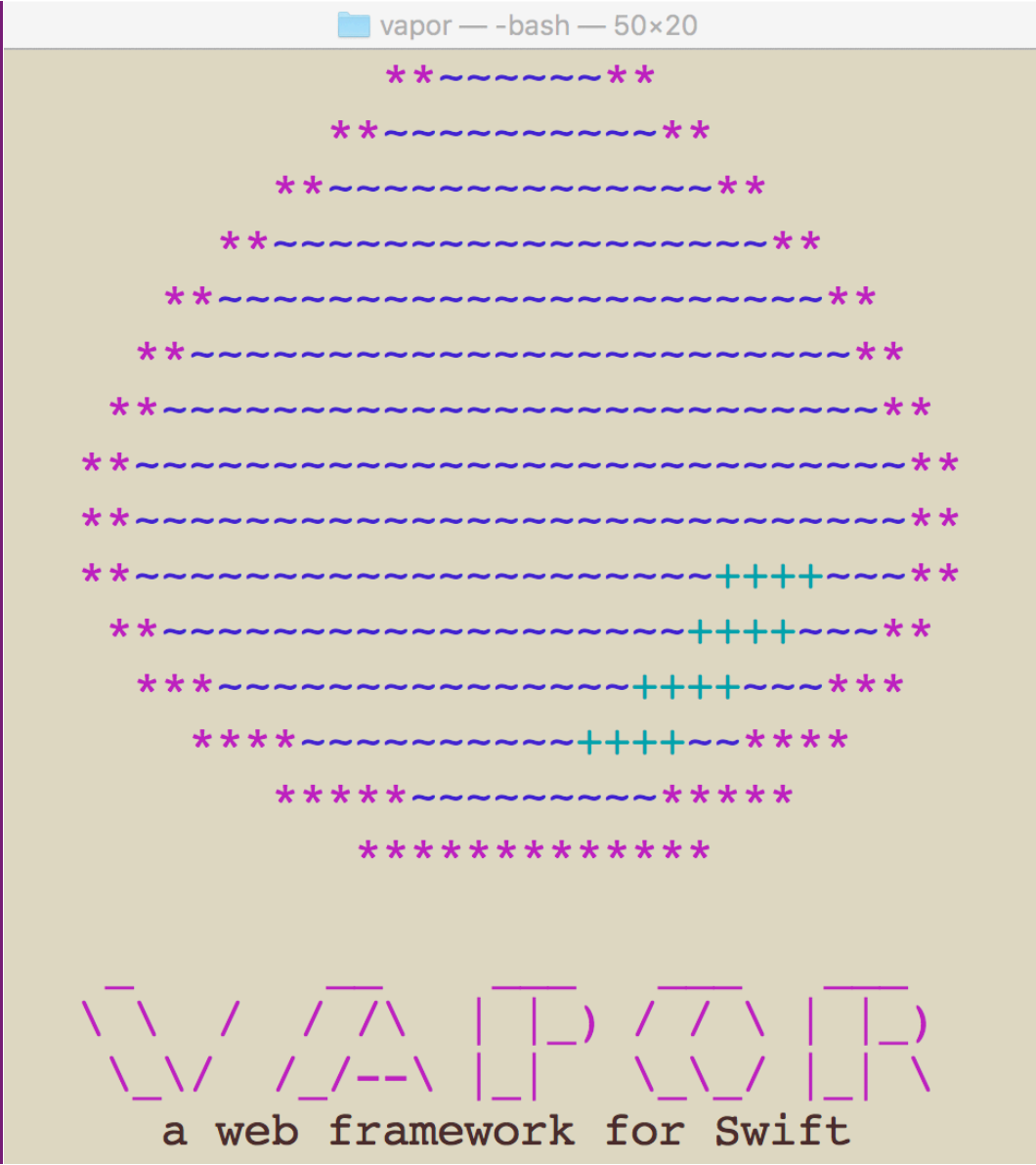

vapor — -bash — 80×20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
542 % vapor --help
Usage: vapor <new|build|update|run|fetch|clean|test|xcode|version|self|heroku>
Join our Slack if you have questions, need help,
or want to contribute: http://vapor.team
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
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>]
```

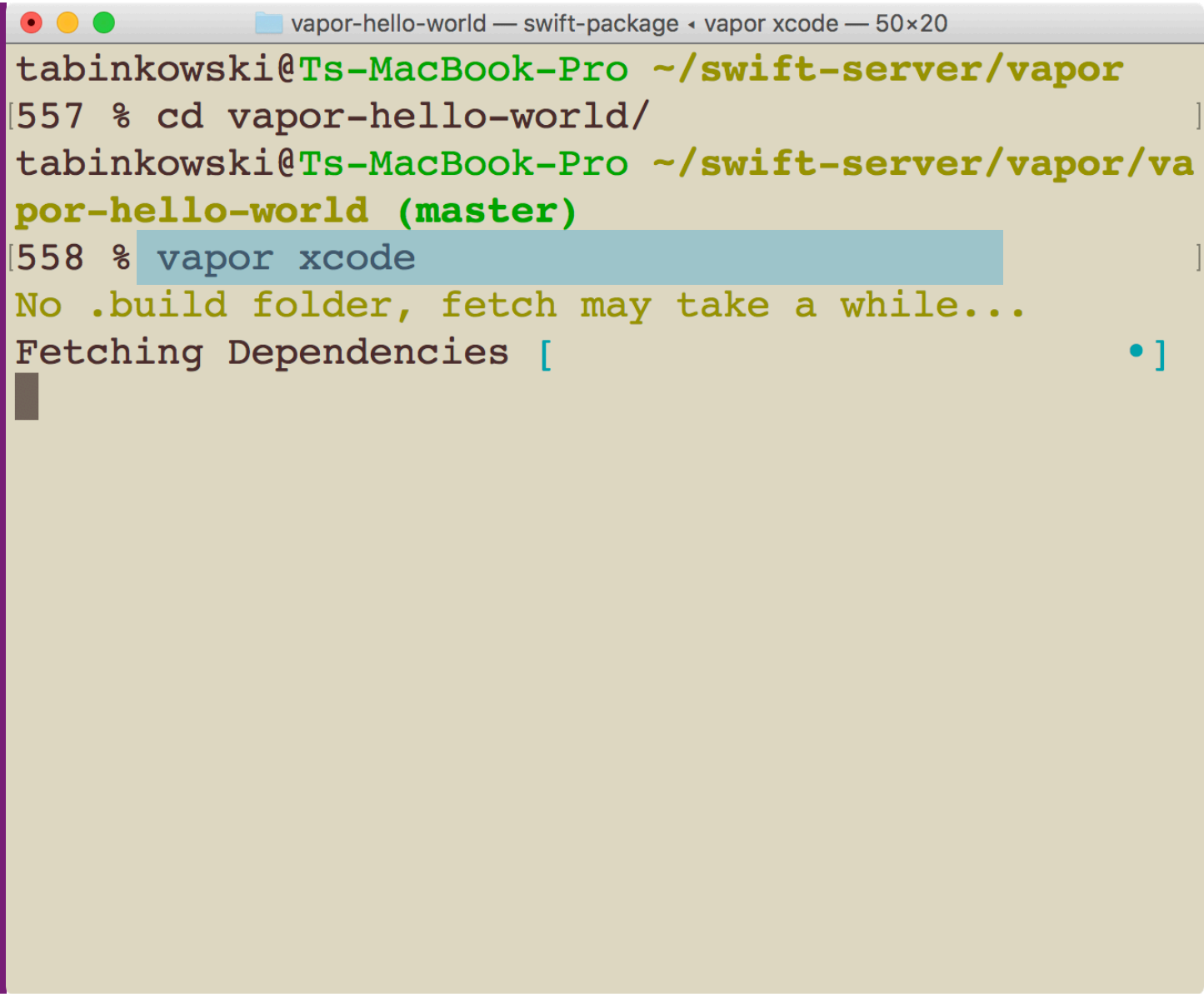
VAPOR

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



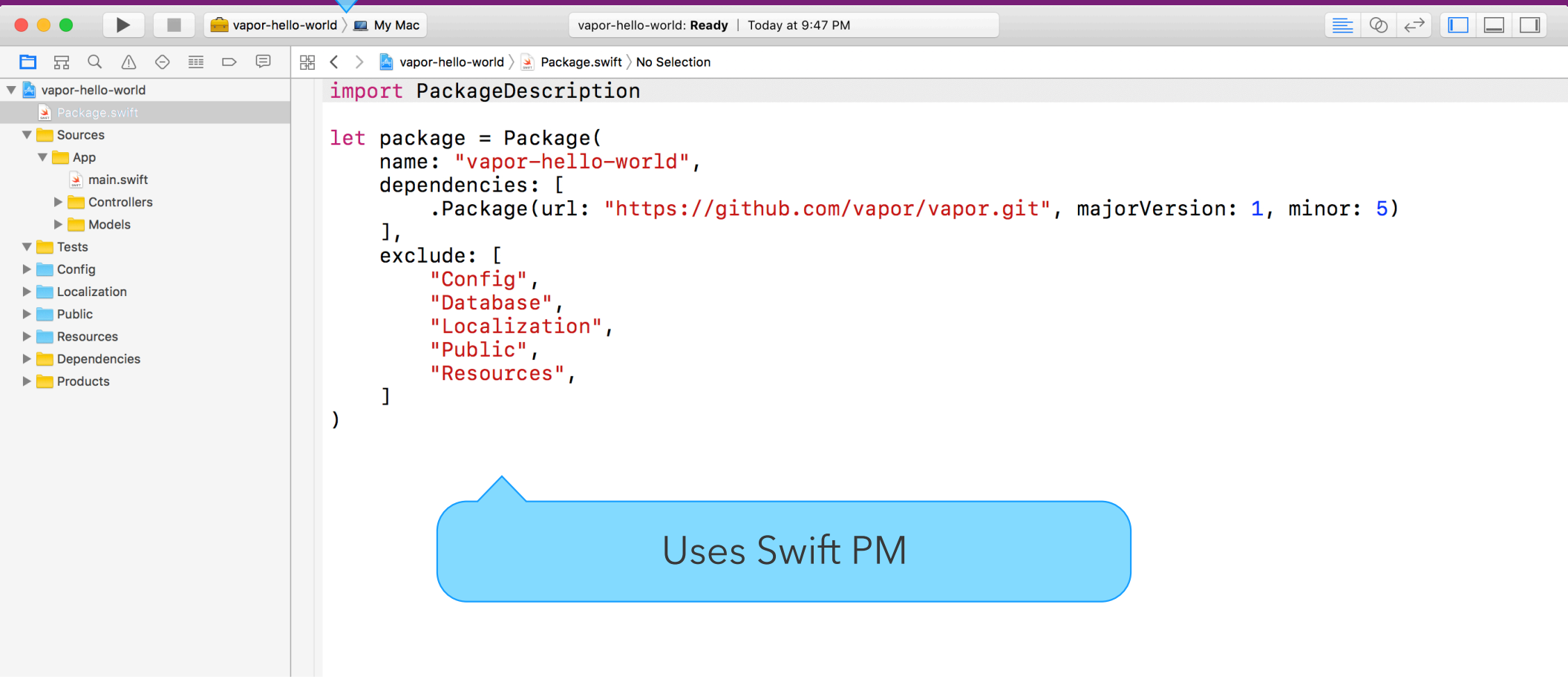
VAPOR

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



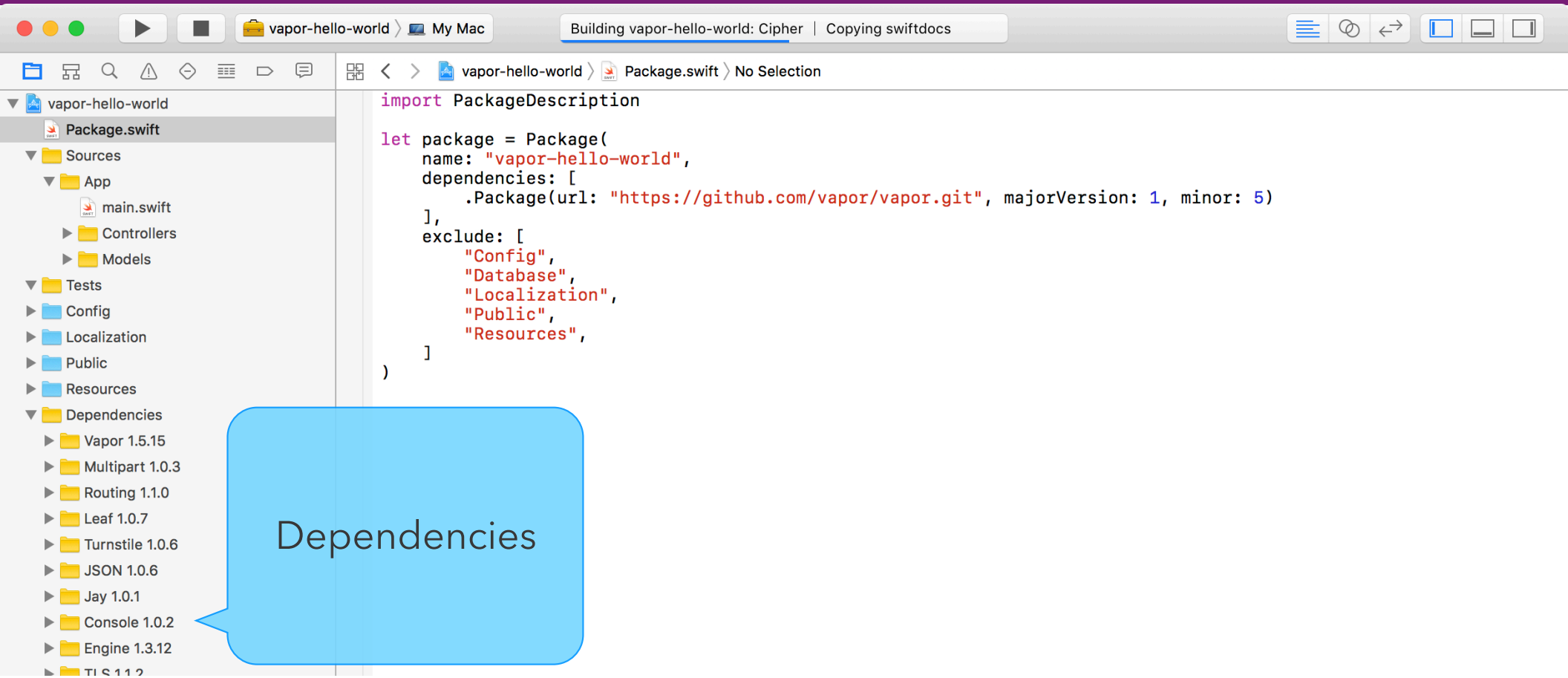
VAPOR

Builds a Framework and App

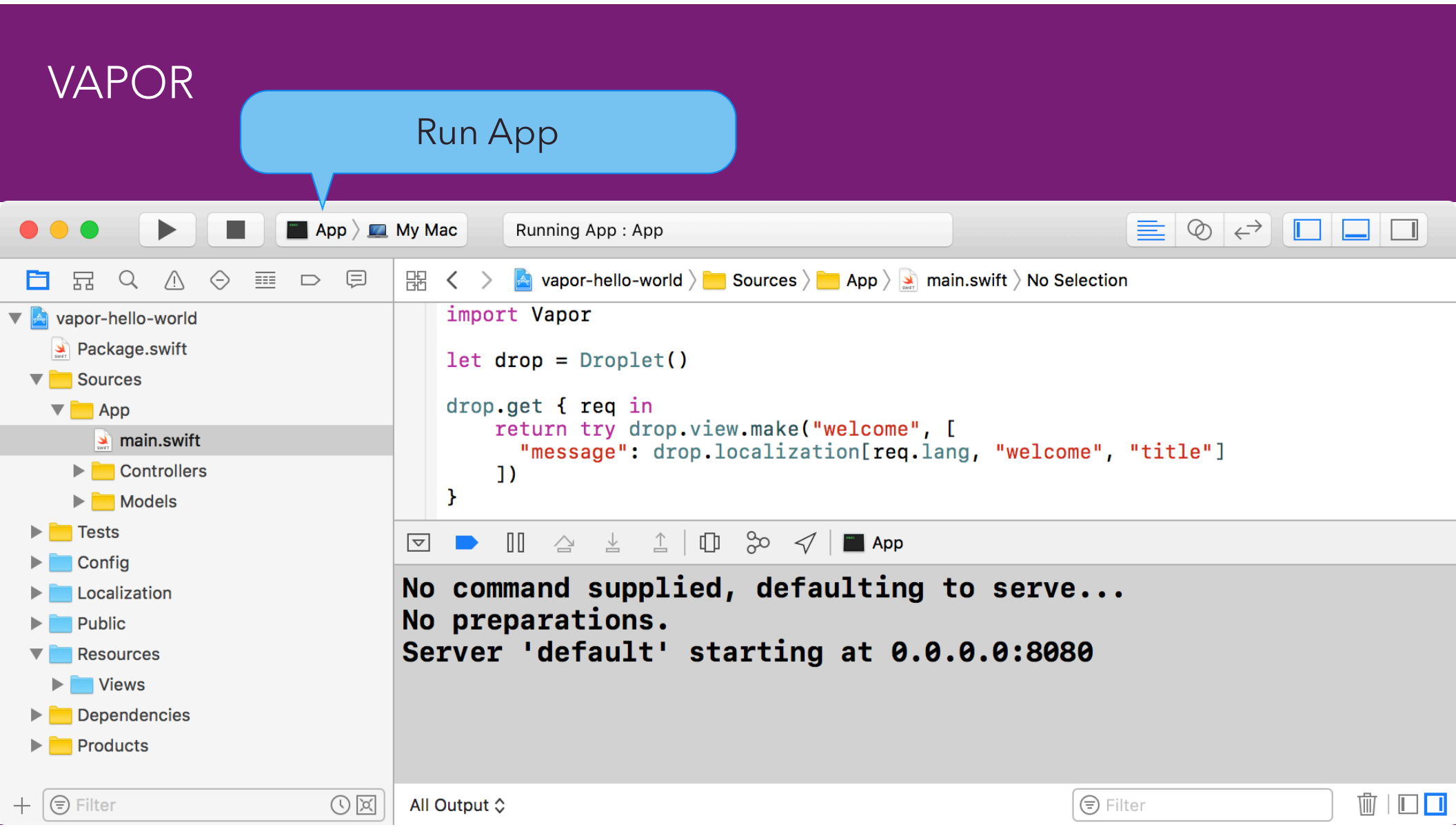


Uses Swift PM

VAPOR

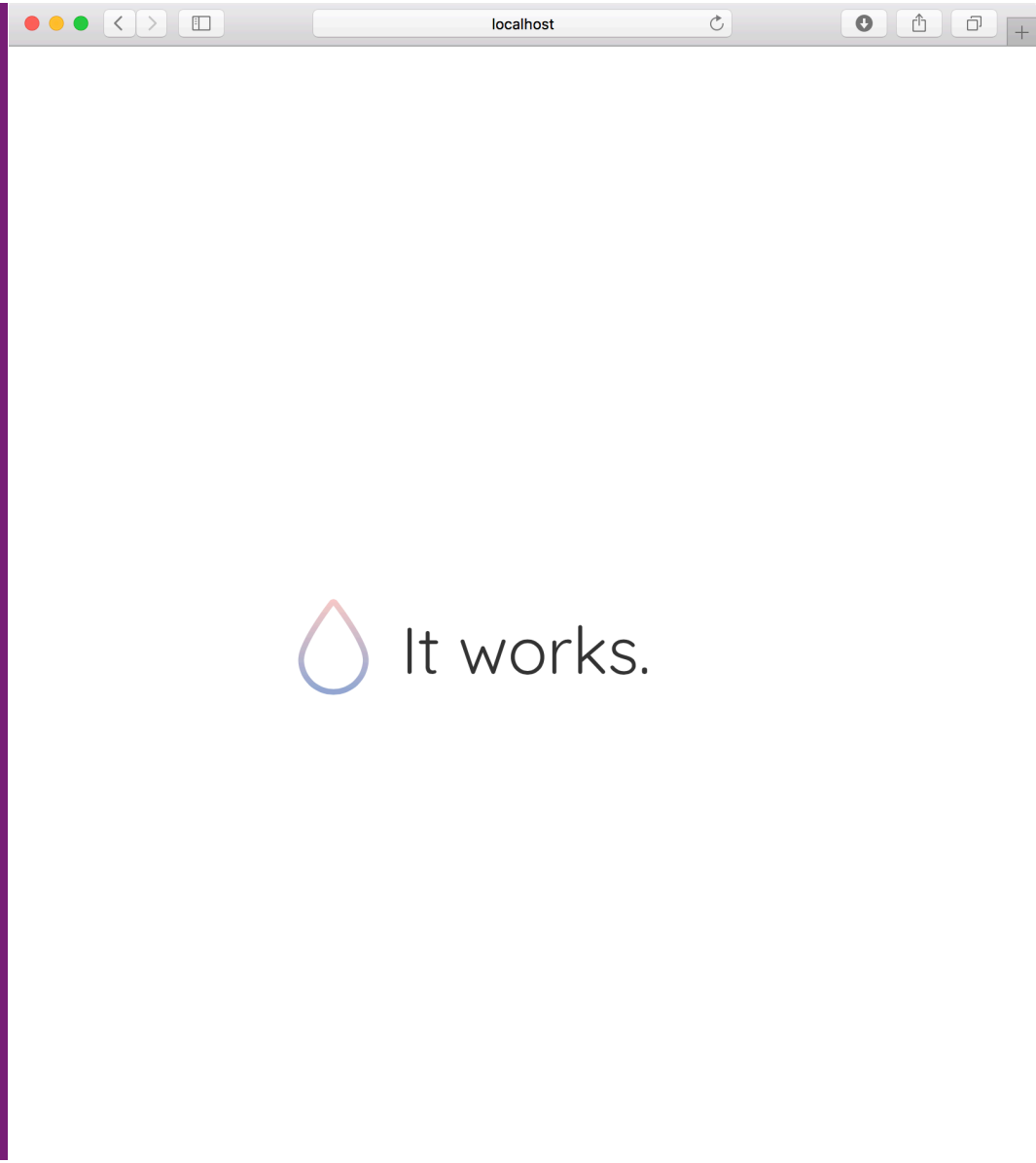

vapor-hello-world 〉 My Mac
Building vapor-hello-world: Cipher | Copying swiftdocs
vapor-hello-world 〉 Package.swift 〉 No Selection
vapor-hello-world
Package.swift
Sources
App
main.swift
Controllers
Models
Tests
Config
Localization
Public
Resources
Dependencies
Vapor 1.5.15
Multipart 1.0.3
Routing 1.1.0
Leaf 1.0.7
Turnstile 1.0.6
JSON 1.0.6
Jay 1.0.1
Console 1.0.2
Engine 1.3.12
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",
]
)
Dependencies

VAPOR

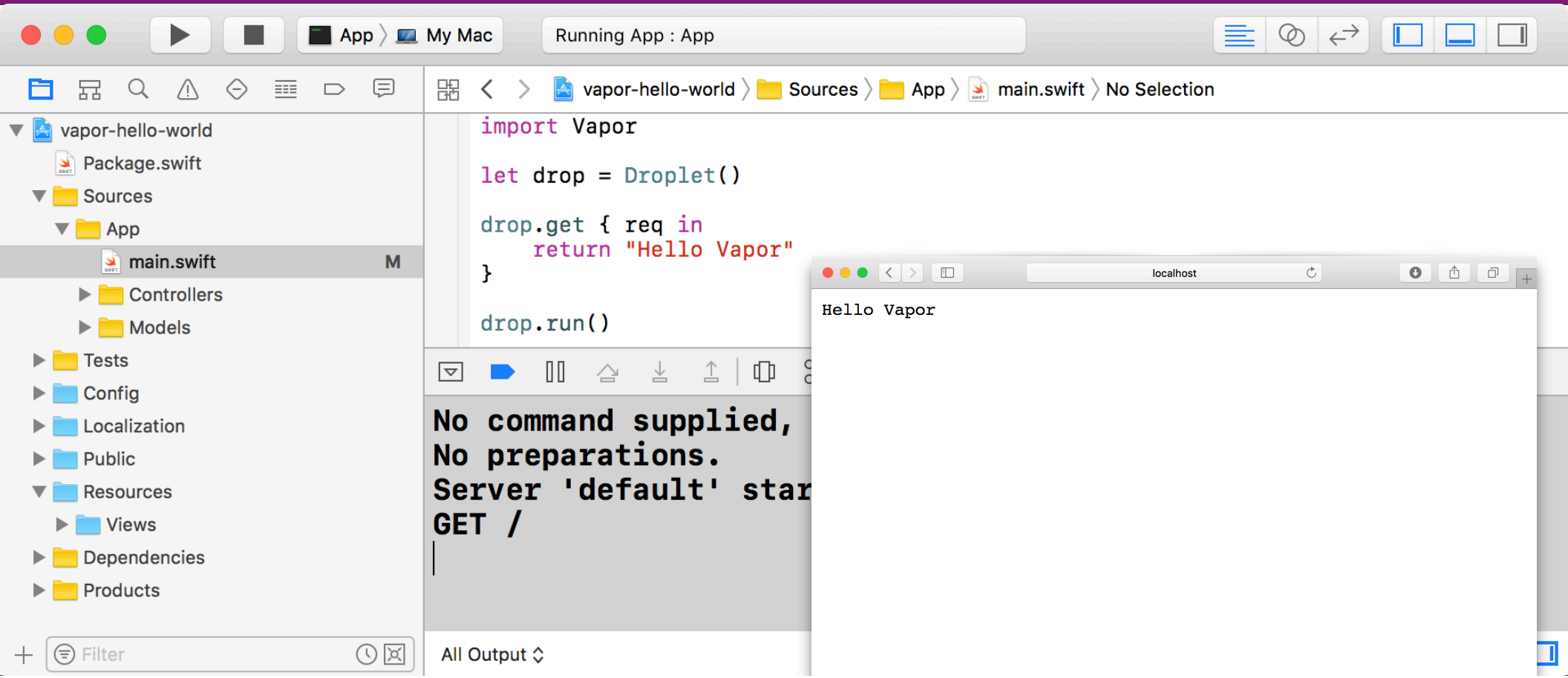

Run App
App
My Mac
Running App : App
vapor-hello-world
Sources
App
main.swift
No Selection
vapor-hello-world
Package.swift
Sources
App
main.swift
Controllers
Models
Tests
Config
Localization
Public
Resources
Views
Dependencies
Products
import Vapor
let drop = Droplet()
drop.get { req in
return try drop.view.make("welcome", [
"message": drop.localization[req.lang, "welcome", "title"]
])
}
App
No command supplied, defaulting to serve...
No preparations.
Server 'default' starting at 0.0.0.0:8080
Filter
All Output
Filter

VAPOR

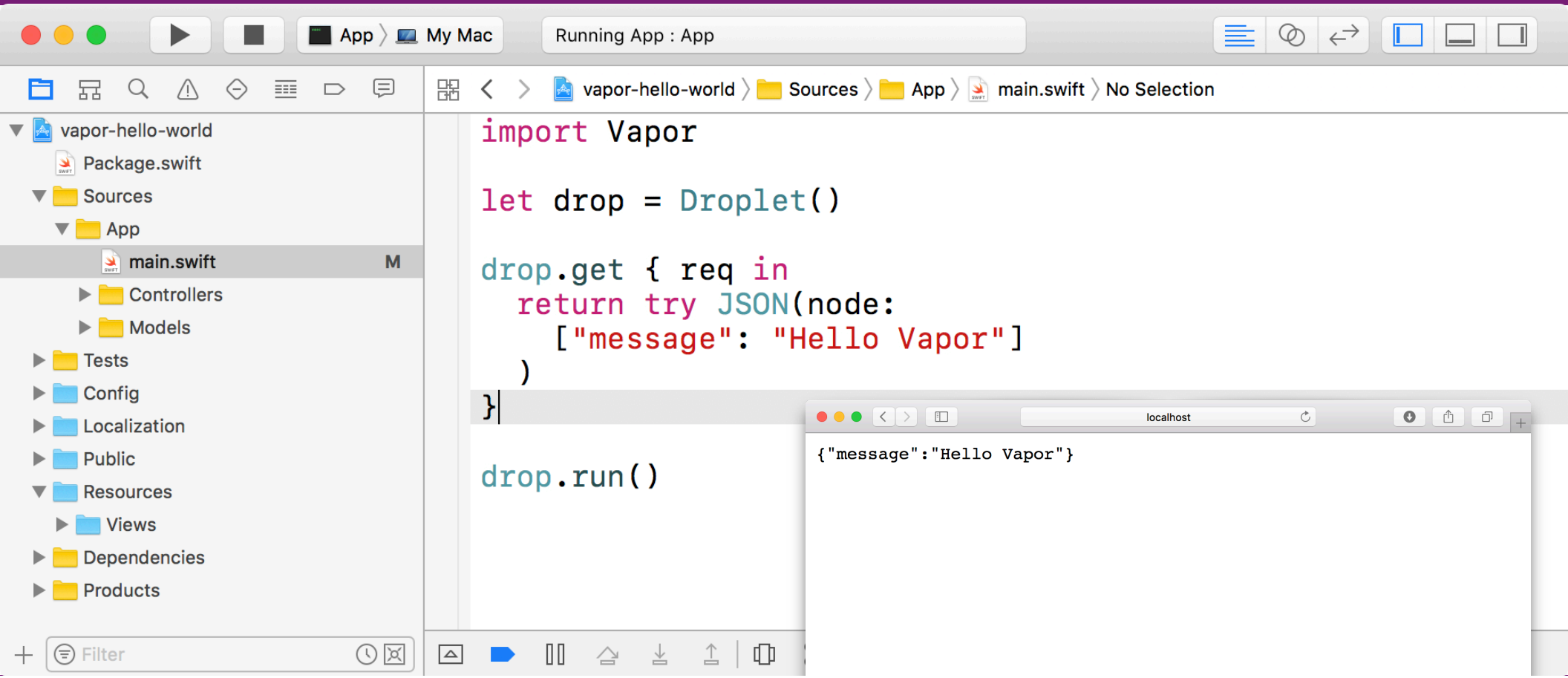
- Server runs on http://localhost:8080



VAPOR


App
My Mac
Running App : App
vapor-hello-world
Sources
App
main.swift
No Selection
vapor-hello-world
Package.swift
Sources
App
main.swift M
Controllers
Models
Tests
Config
Localization
Public
Resources
Views
Dependencies
Products
Filter
import Vapor
let drop = Droplet()
drop.get { req in
return "Hello Vapor"
}
drop.run()
No command supplied,
No preparations.
Server 'default' star
GET /
All Output
localhost
Hello Vapor

VAPOR

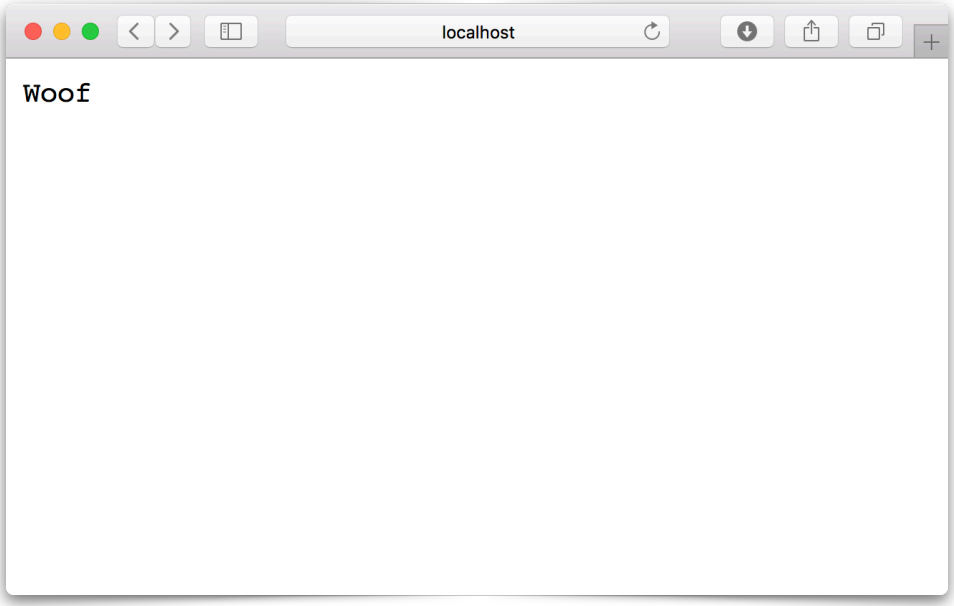

App
My Mac
Running App : App
vapor-hello-world
Sources
App
main.swift
No Selection
vapor-hello-world
Package.swift
Sources
App
main.swift M
Controllers
Models
Tests
Config
Localization
Public
Resources
Views
Dependencies
Products
Filter
import Vapor
let drop = Droplet()
drop.get { req in
return try JSON(node:
["message": "Hello Vapor"]
)
}
drop.run()
localhost
{"message":"Hello Vapor"}

VAPOR

vapor-hello-world 〉 Sources 〉 App 〉 main.swift 〉 No Selection

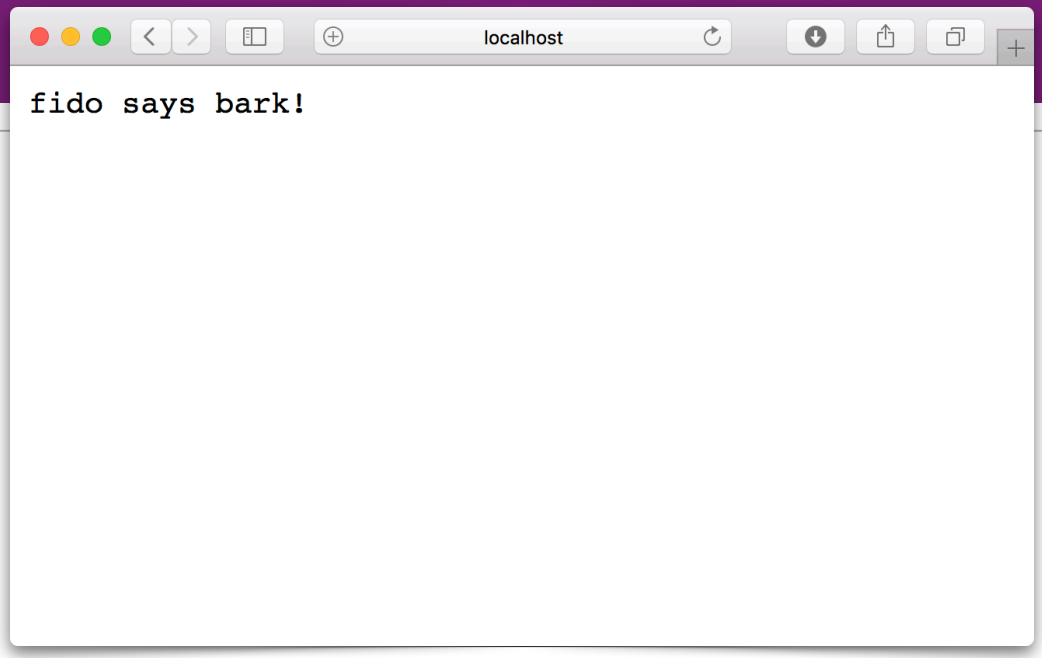
```swift
import Vapor

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()
```

Localhost:8080/dog



VAPOR

Localhost:8080/dog/speak/fido/

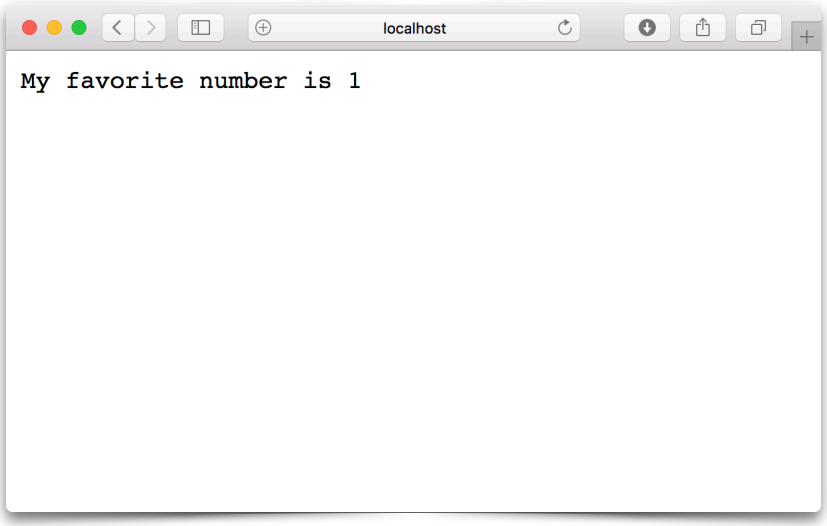


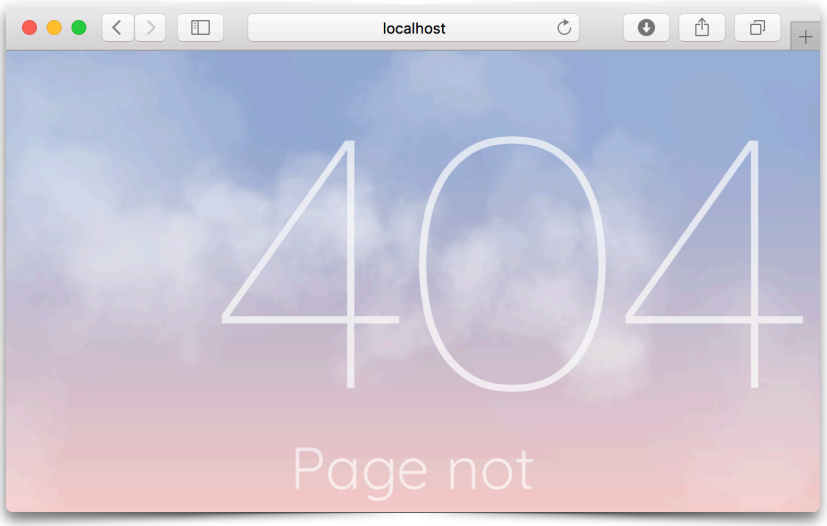

```
drop.get { req in
  return try JSON(node:
    ["message": "Hello Vapor"]
  )
}

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()
```

VAPOR

Localhost:8080/dog/speak/1/

Localhost:8080/dog/speak/fido/



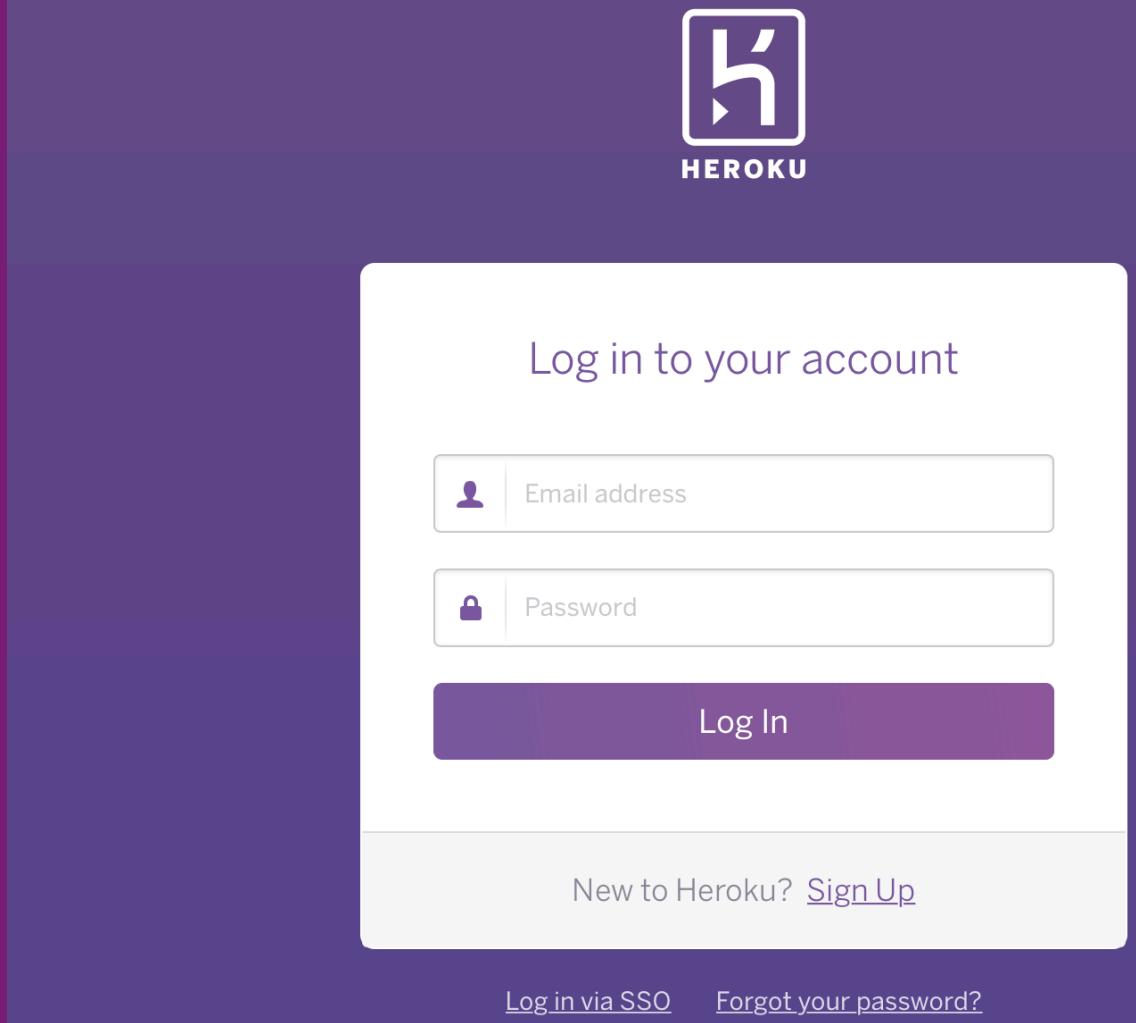



```
drop.get("dog","speak",Int.self) { req, number in
   return "My favorite number is \(number)"
}
```

DEPLOY VAPOR APP TO HEROKU

DEPLOY VAPOR APP TO HEROKU

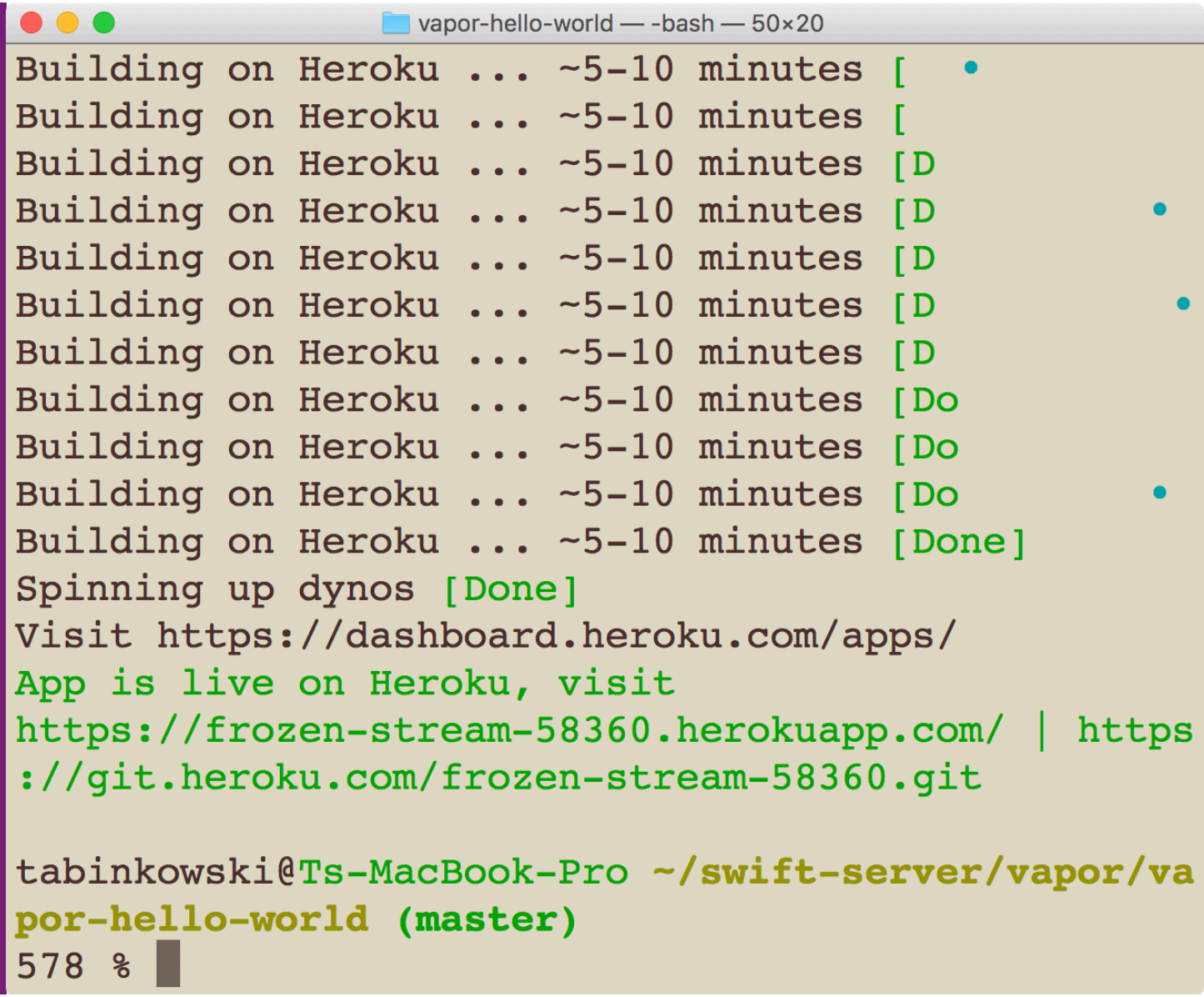
- Heroku is a platform as a service company
- Free 'hobby' tier to play around



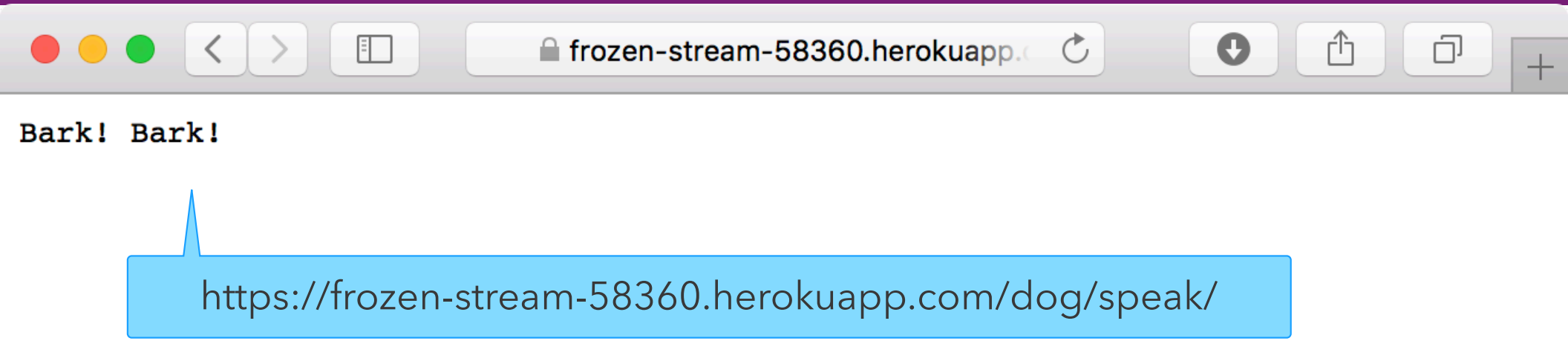
DEPLOY VAPOR APP TO HEROKU

```
brew install heroku
heroku login
```

DEPLOY VAPOR APP TO HEROKU



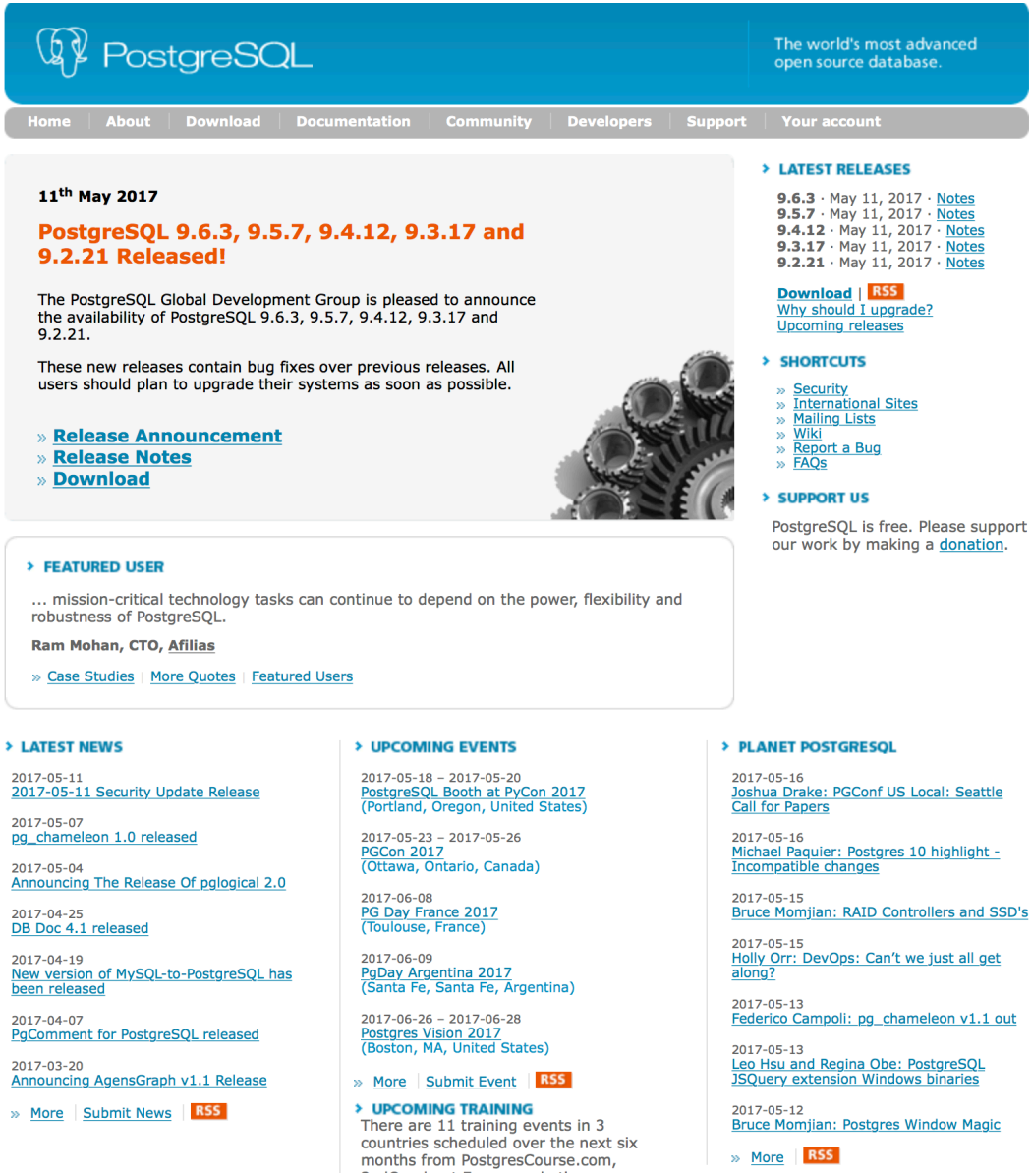
DEPLOY VAPOR APP TO HEROKU



VAPOR WITH POSTGRESQL

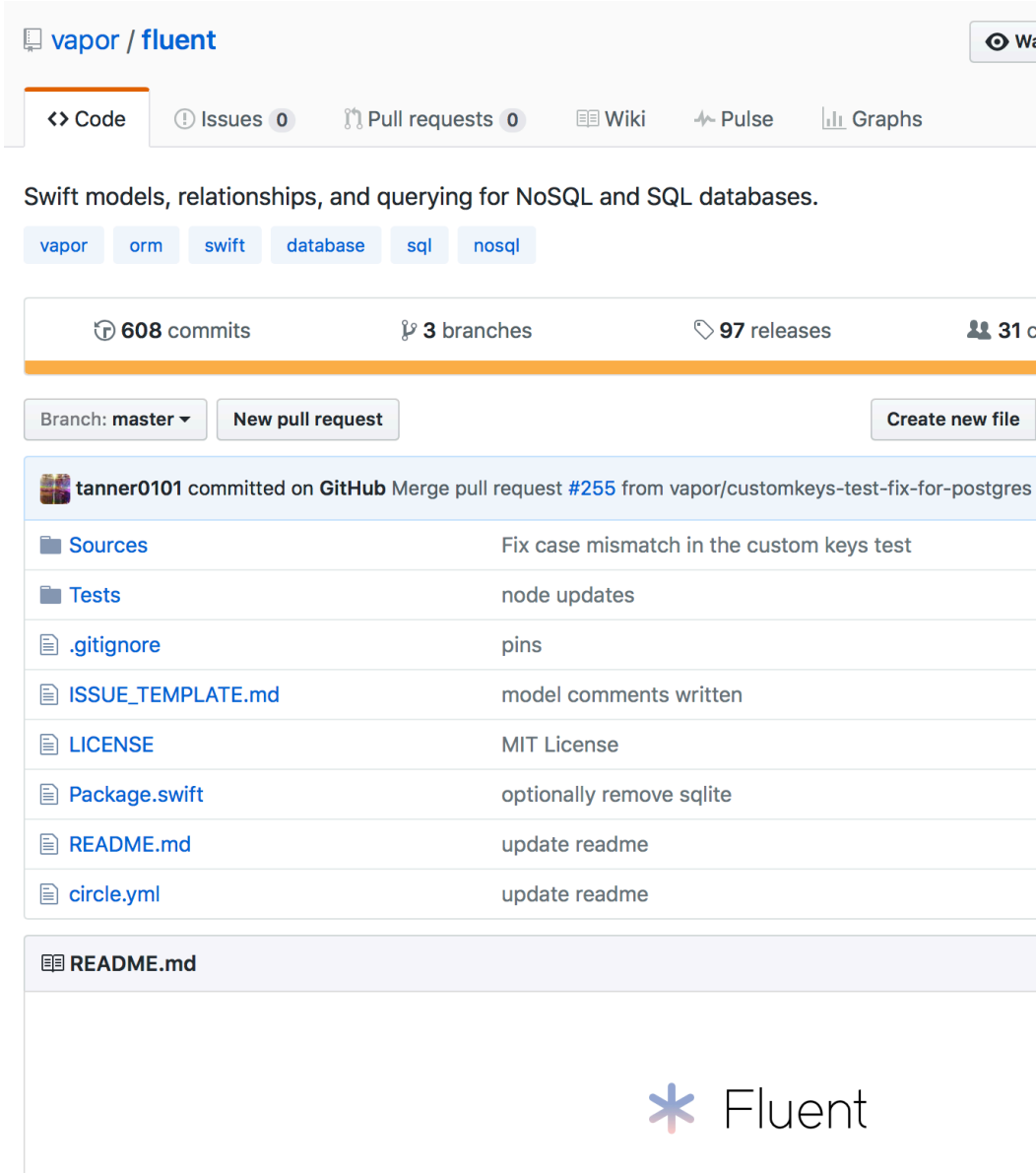
VAPOR WITH POSTGRESQL

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



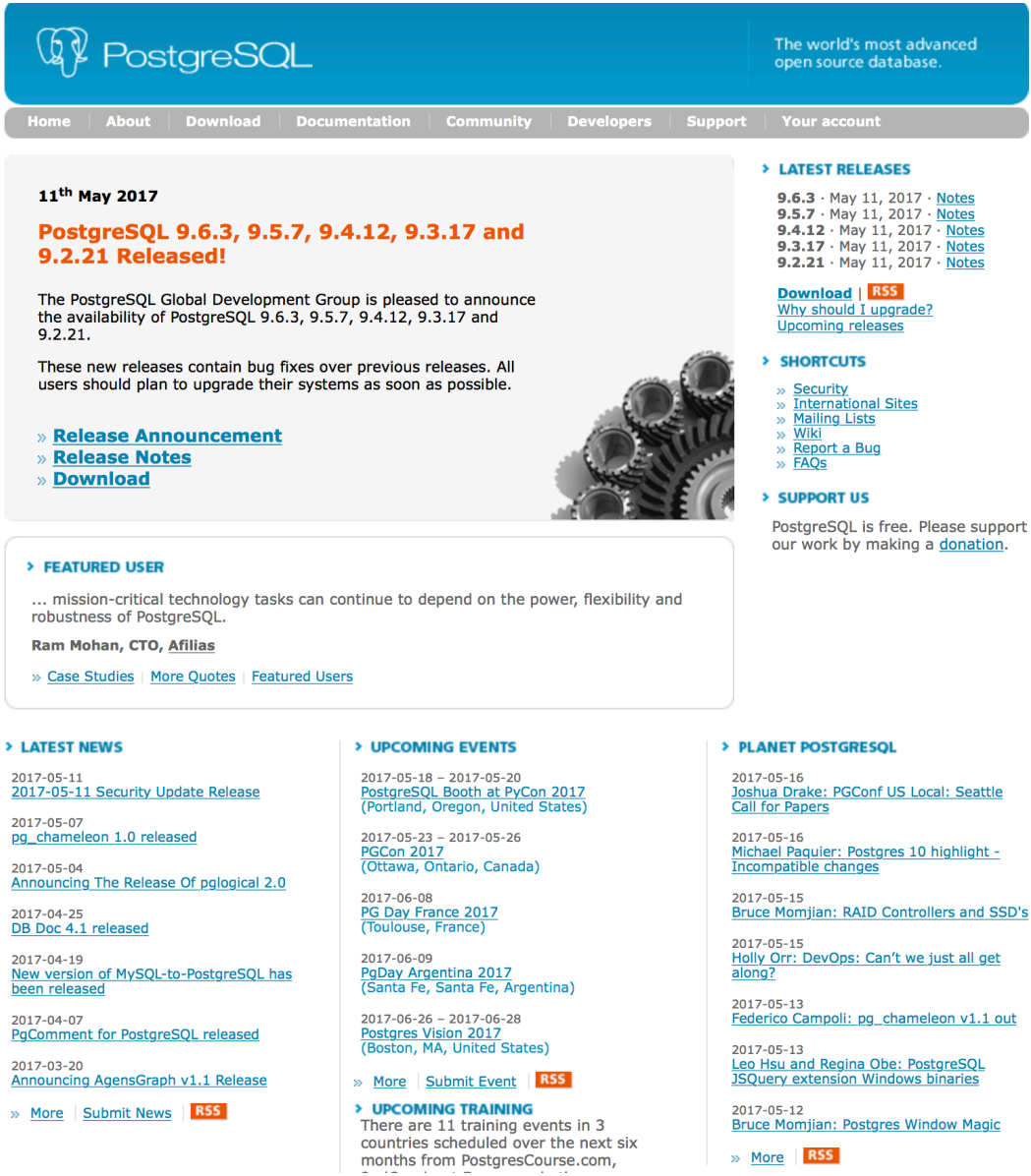
VAPOR WITH POSTGRESQL

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



VAPOR WITH POSTGRESQL

- Import package
- Configure droplet
- Add configuration file



VAPOR WITH POSTGRESQL

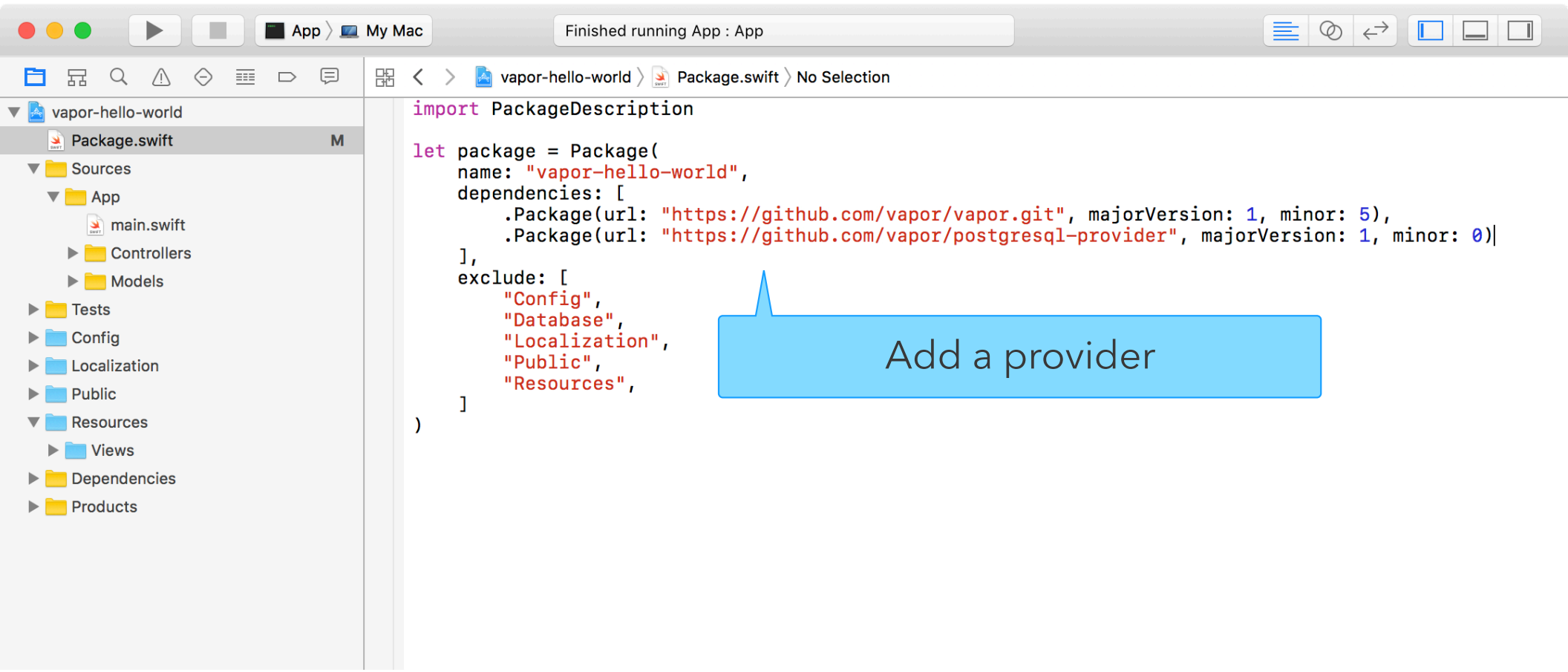
```
# Install postgres
brew install postgres

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

# Create a database
createdb dogs
```

VAPOR WITH POSTGRESQL

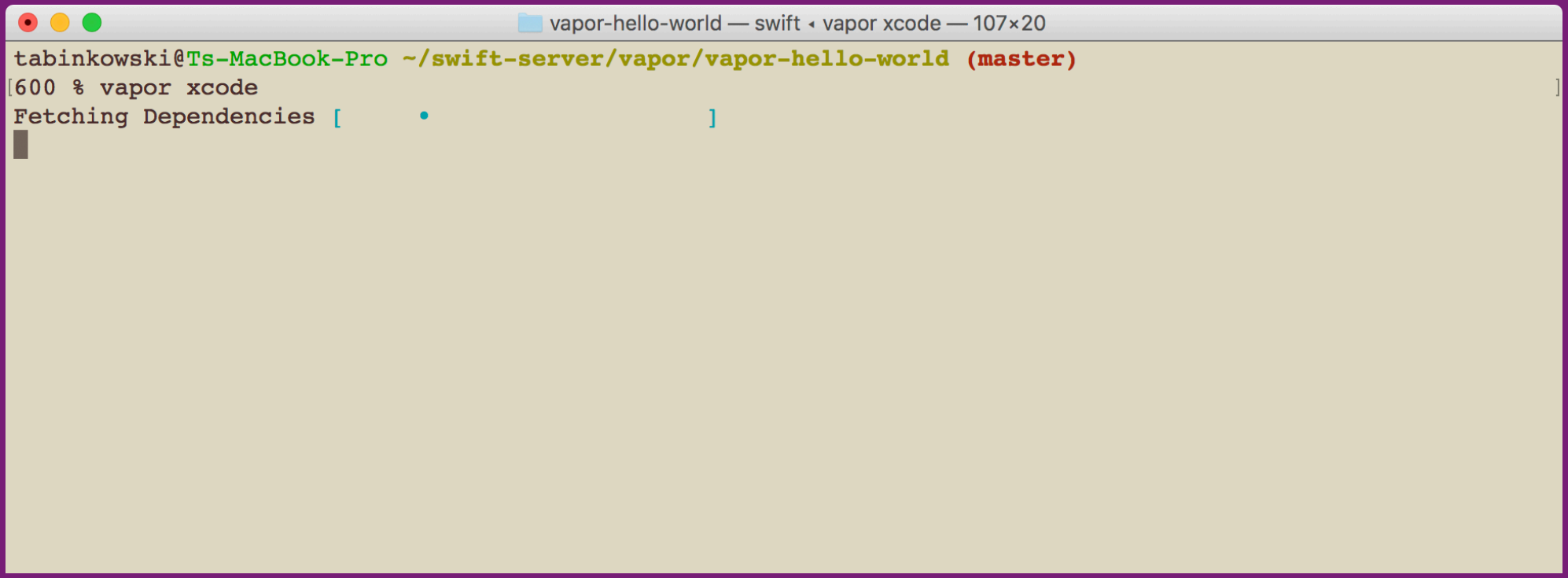
```
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[592 % psql
psql (9.6.3)
Type "help" for help.

[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



VAPOR WITH POSTGRESQL



- Update the project and rebuild your framework in Xcode

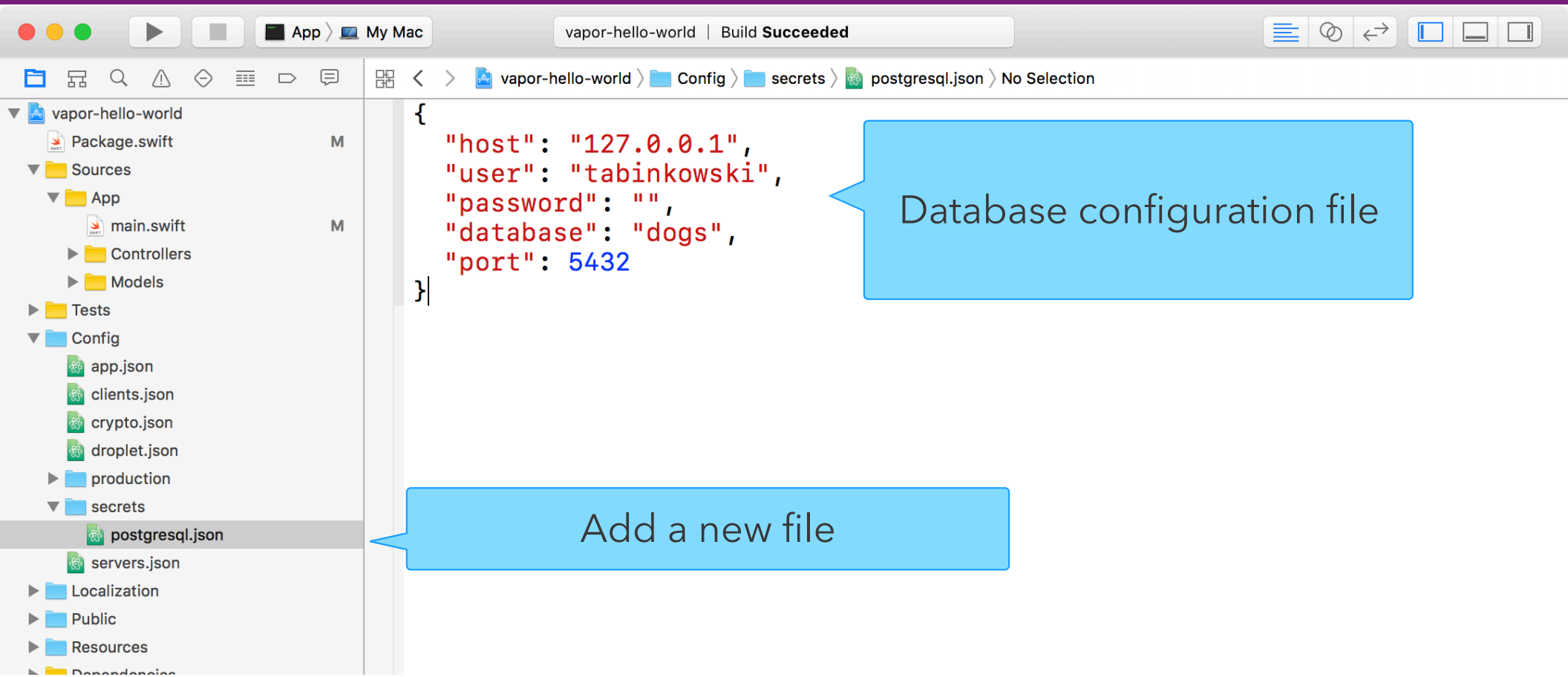
VAPOR WITH POSTGRESQL

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

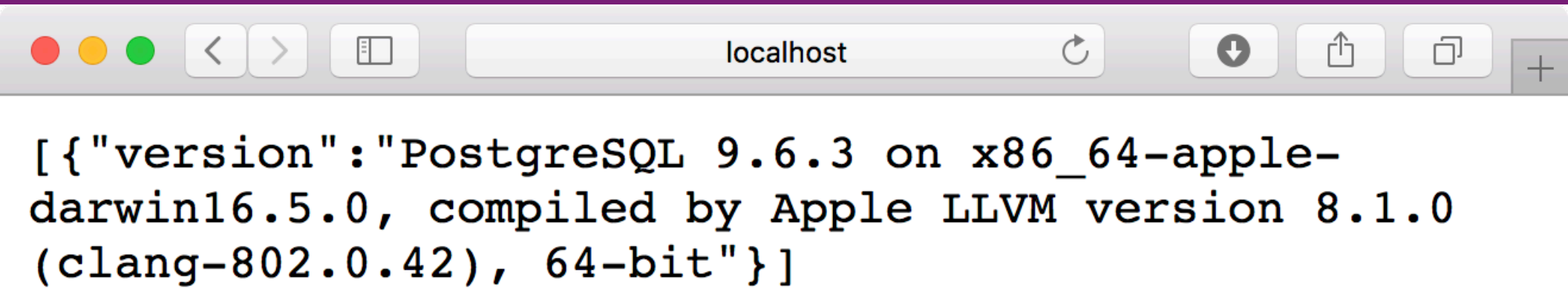


VAPOR WITH POSTGRESQL

```
// 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"
  }
}
```

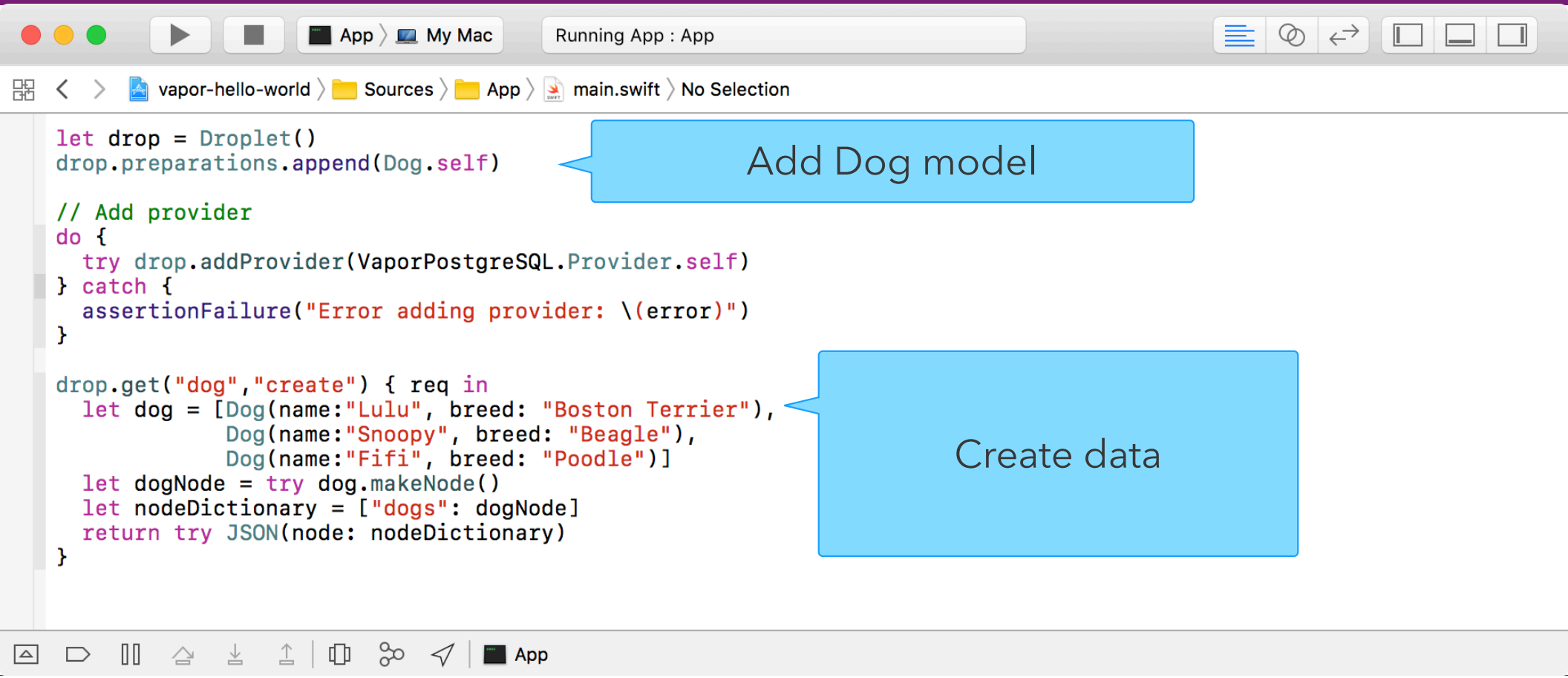
Print out database version

VAPOR WITH POSTGRESQL

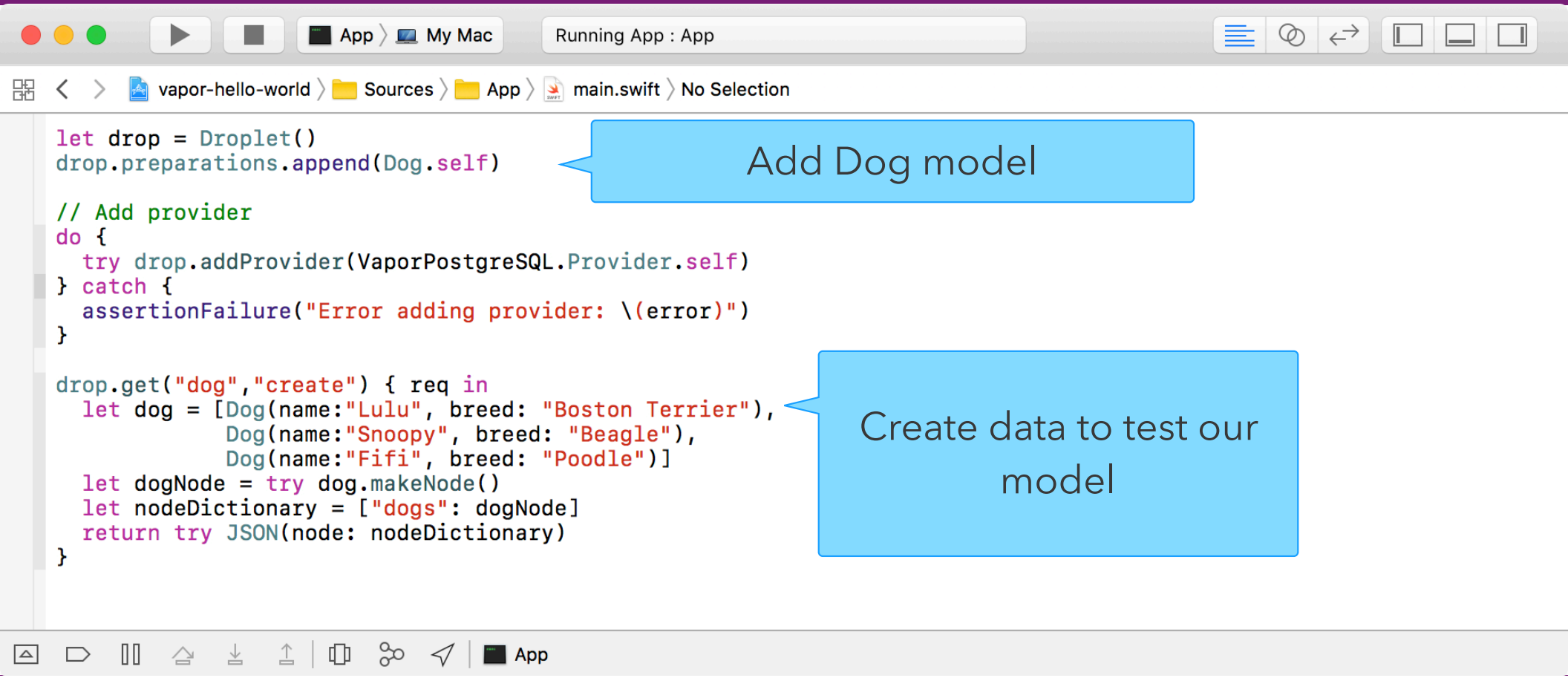


It's boring, but it works

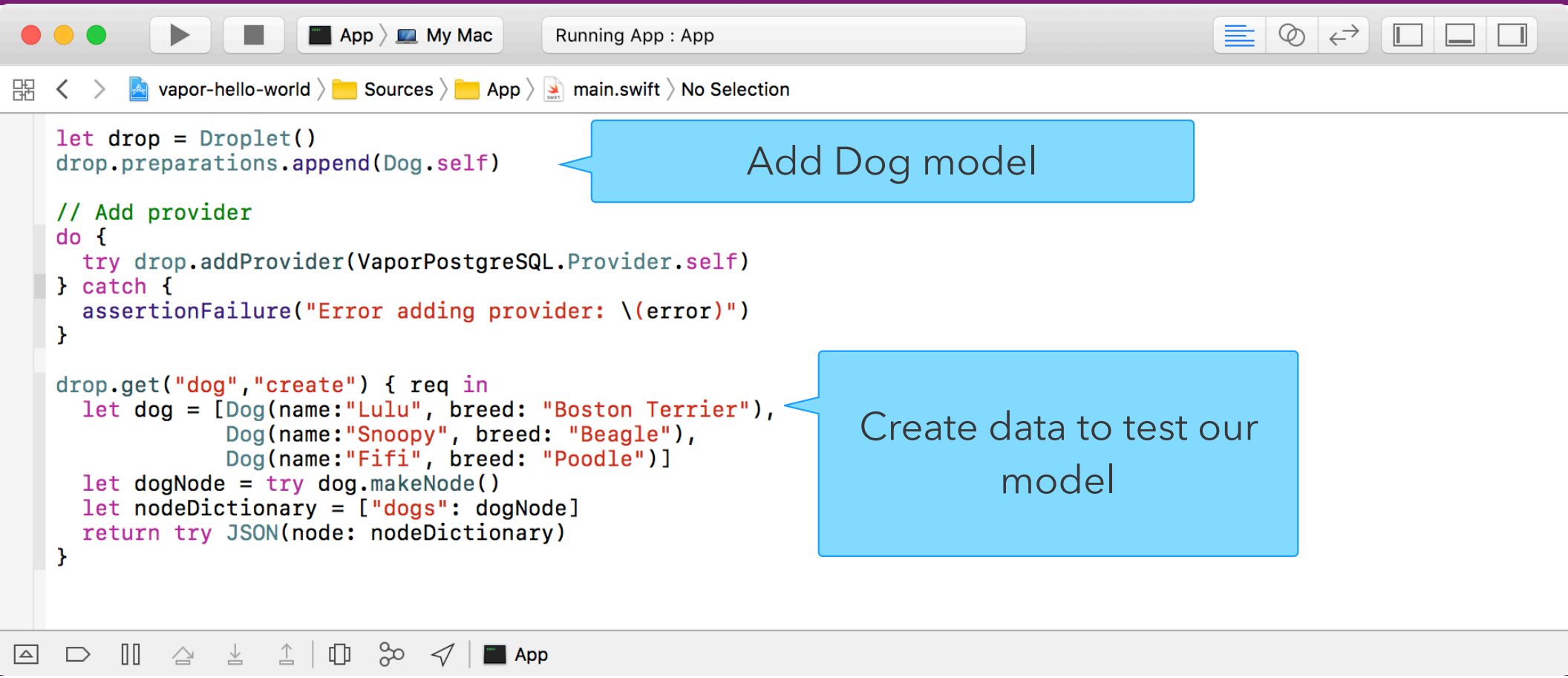
VAPOR WITH POSTGRESQL



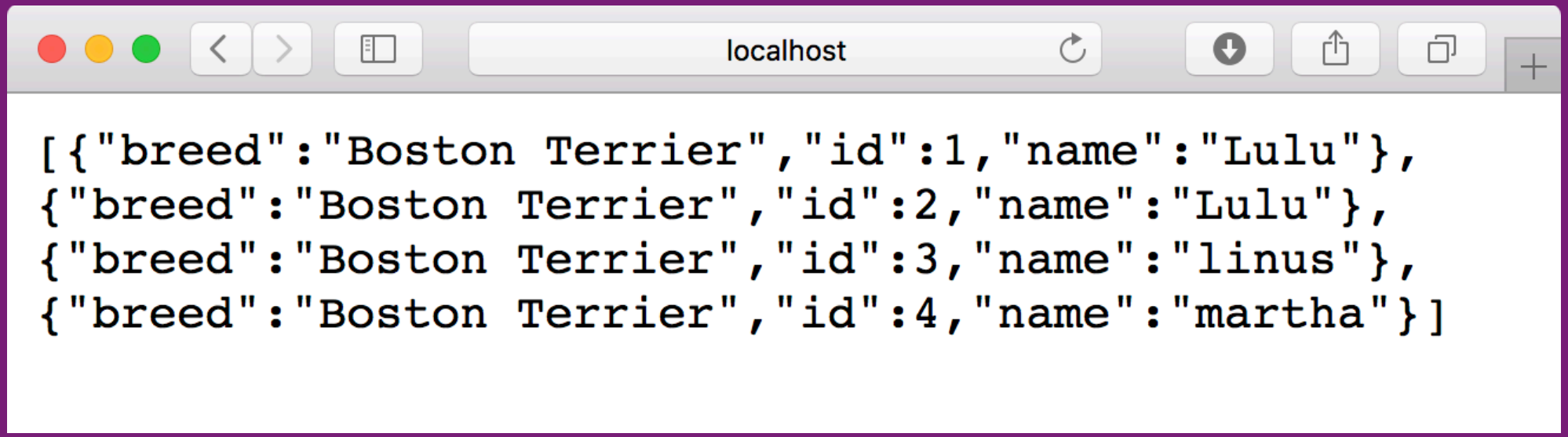
VAPOR WITH POSTGRESQL



VAPOR WITH POSTGRESQL



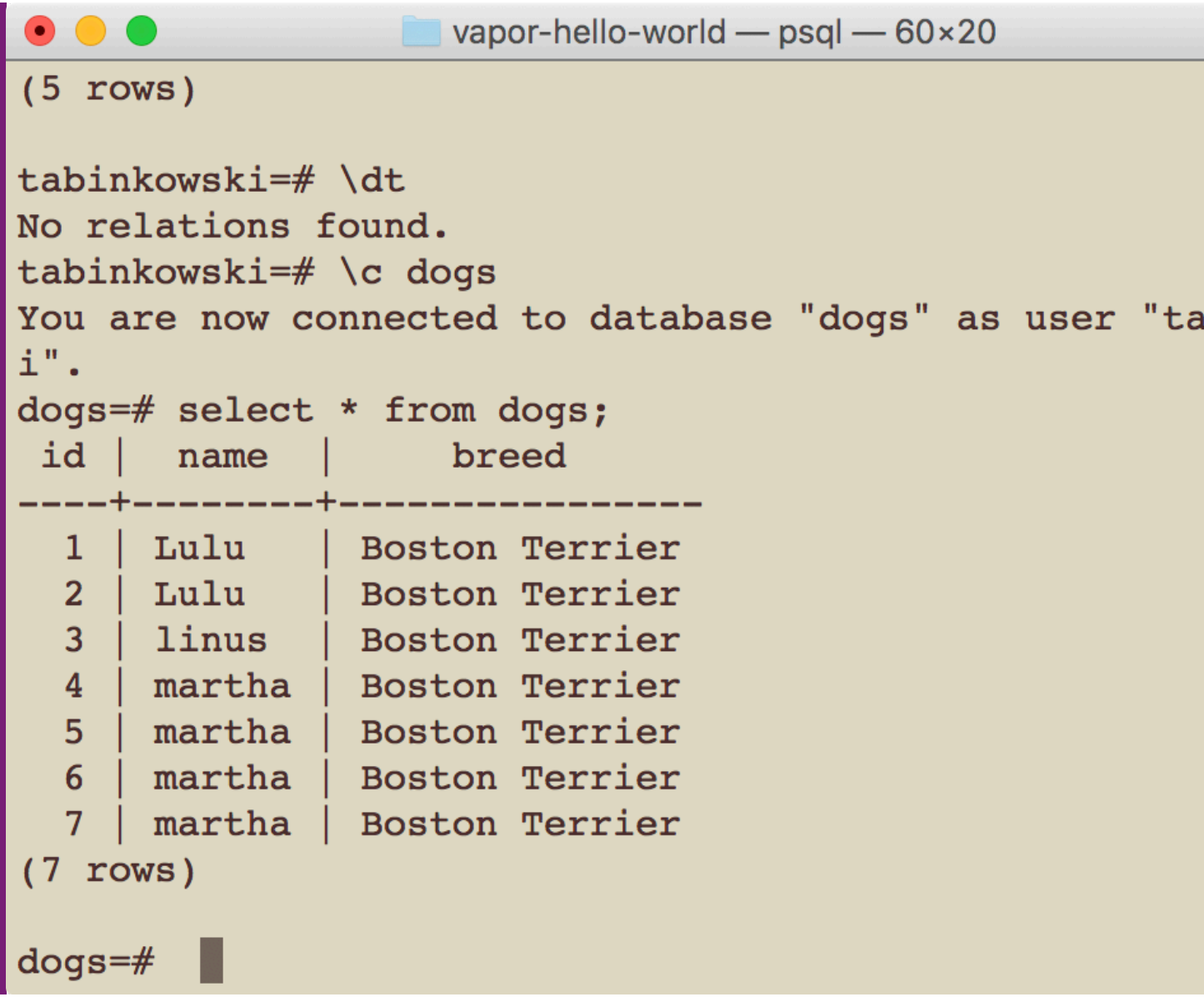
VAPOR WITH POSTGRESQL




```
// 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



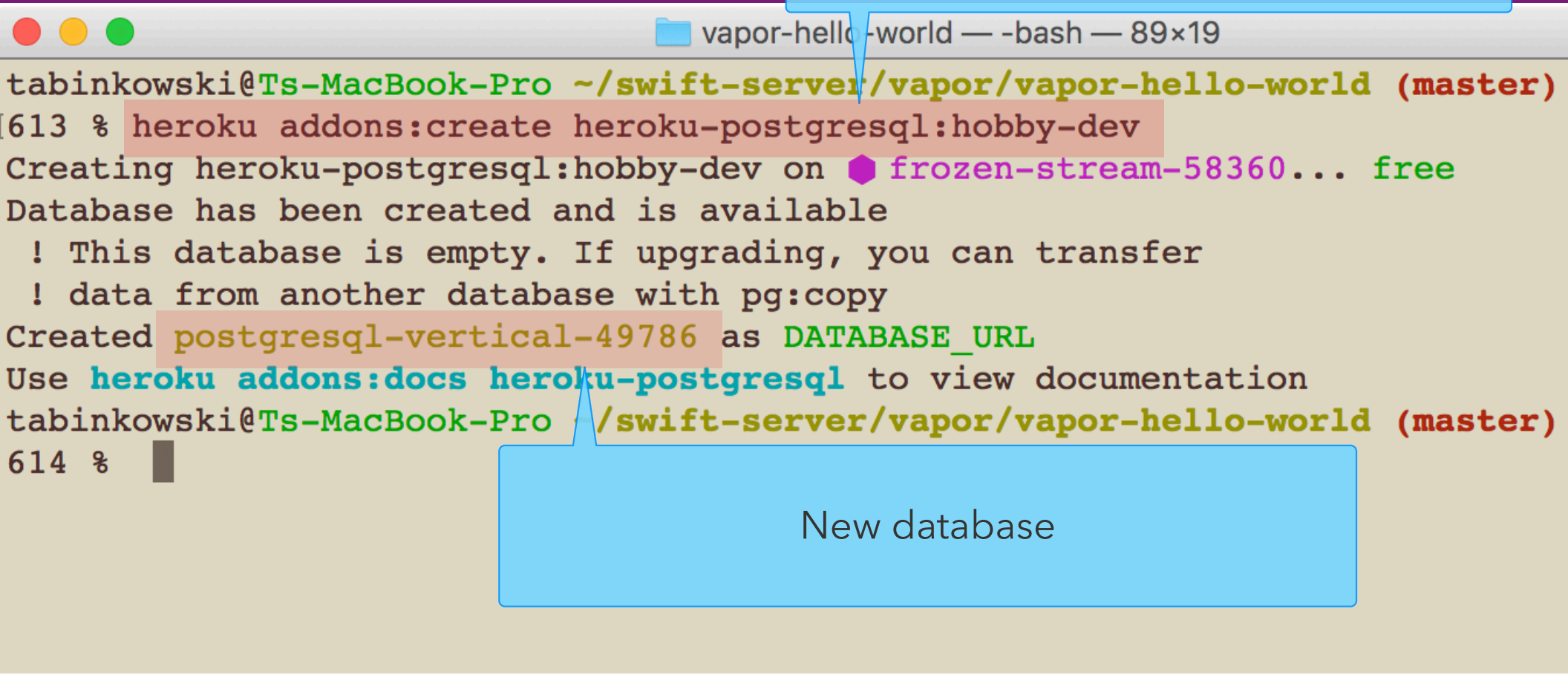
DEPLOY TO HEROKU WITH POSTGRESQL

DEPLOY TO HEROKU WITH POSTGRESQL

- Set up postgresql on heroku
  - Built-in, just need to add it
- Tell Postgresql where the database is locates
  - 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

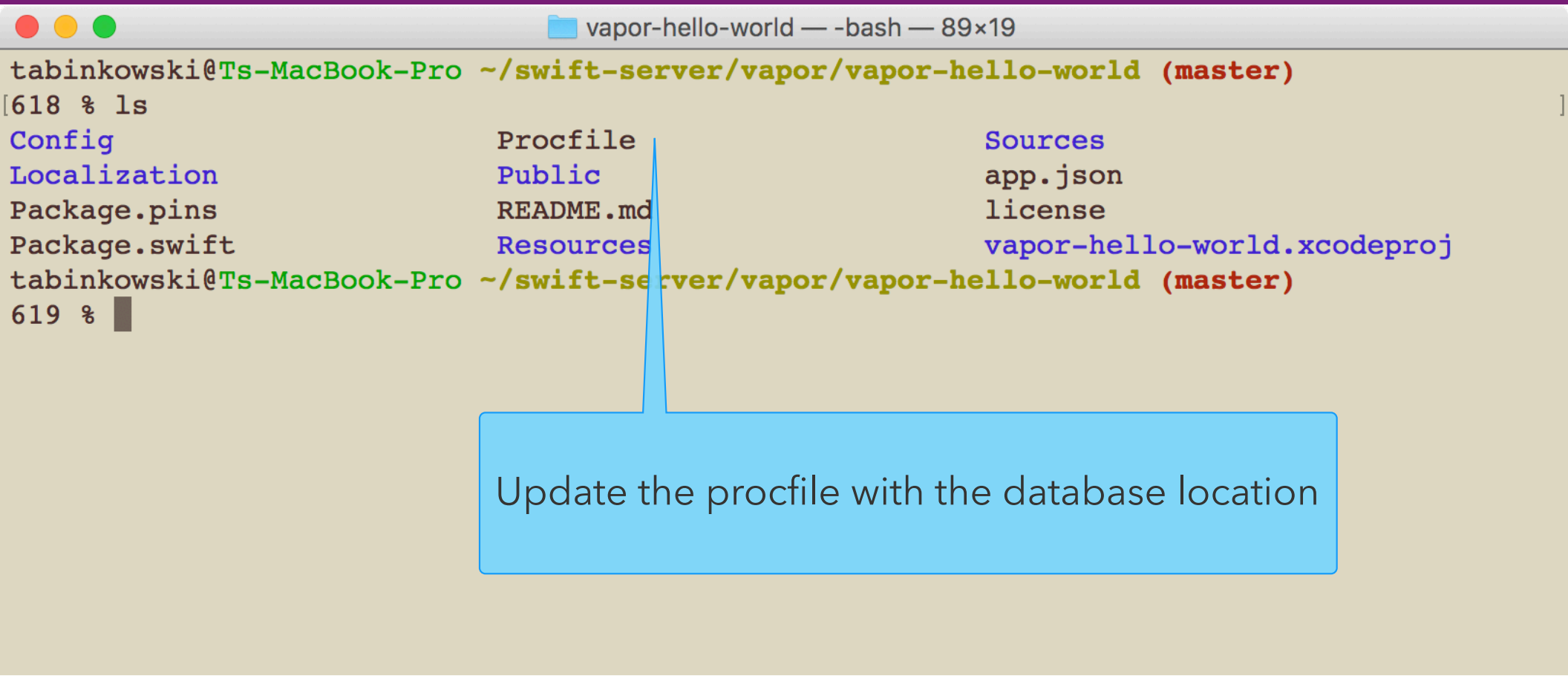


DEPLOY TO HEROKU WITH POSTGRESQL

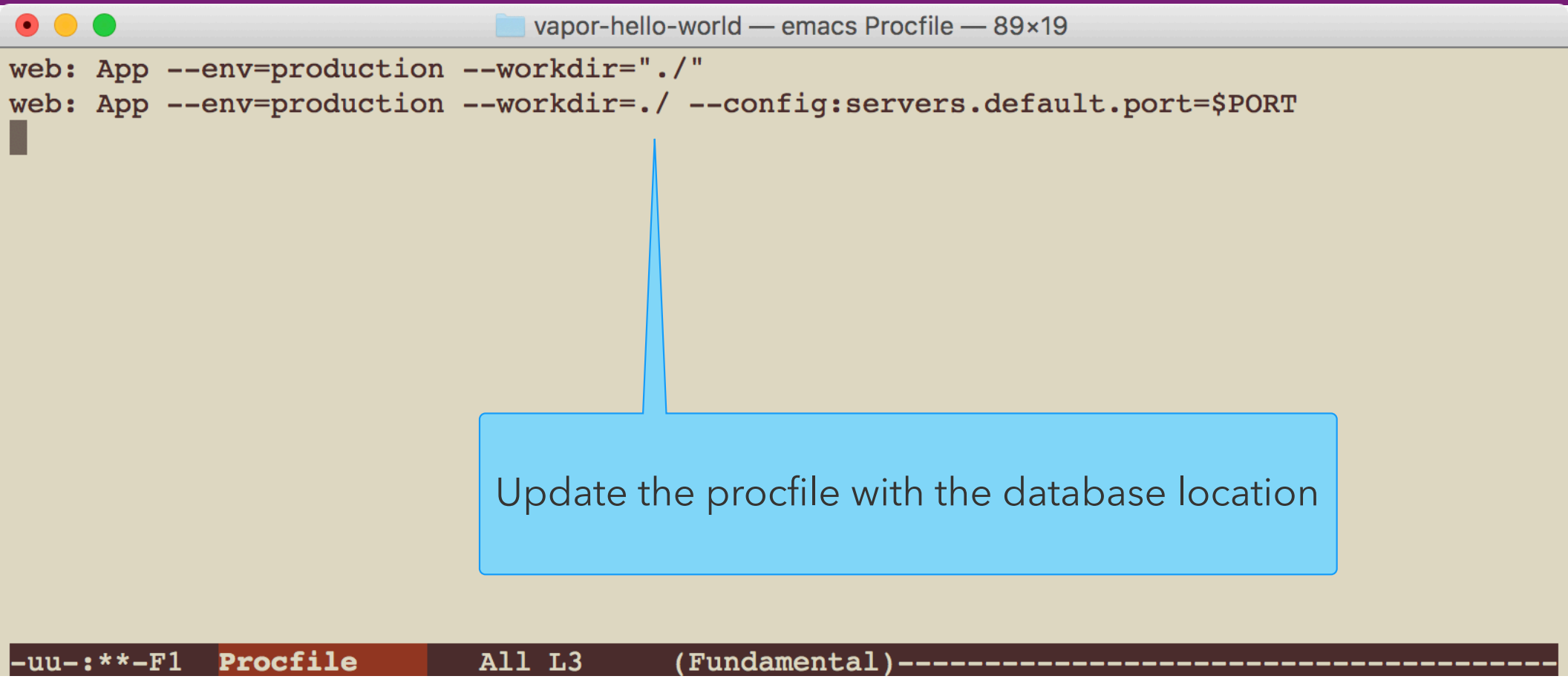
```
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[613 % heroku addons:create heroku-postgresql:hobby-dev
Creating heroku-postgresql:hobby-dev on ⬢ frozen-stream-58360... free
Database has been created and is available
 ! This database is empty. If upgrading, you can transfer
 ! data from another databa se with pg:copy
Created postgresql-vertical-49786 as DATABASE_URL
Use heroku addons:docs hero ku-postgresql to view documentation
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (ma ter)
[614 %  heroku config
=== frozen-stream-58360 Config Vars
DATABASE_URL: postgres://hzgtyzqowaztwa:4ad9847249b30c97f1835744ca06b1bae2b9012422c21d32d
4dad5e5088def04@ec2-23-23-227-188.compute-1.amazonaws.com:5432/d9hapt1pu1fgrf
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
615 %
```

Database location

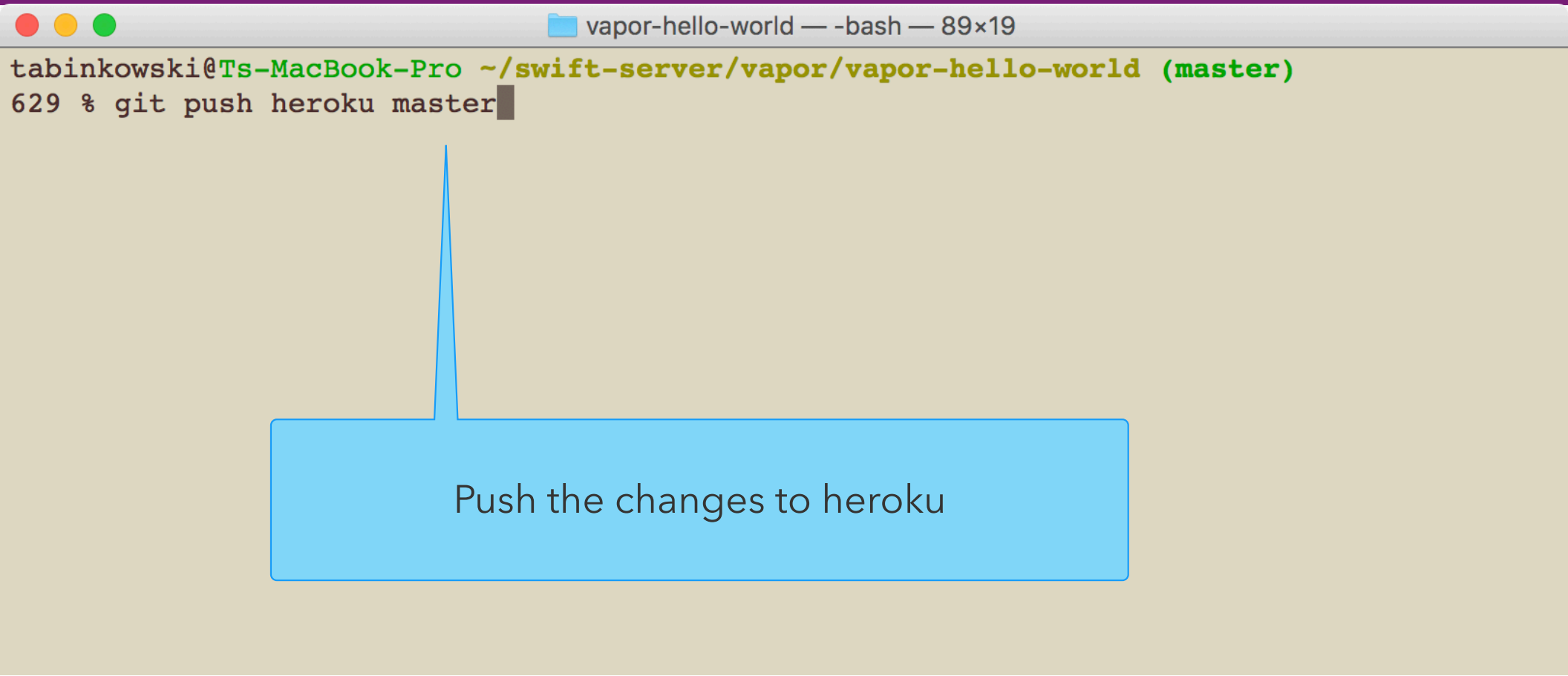
DEPLOY TO HEROKU WITH POSTGRESQL



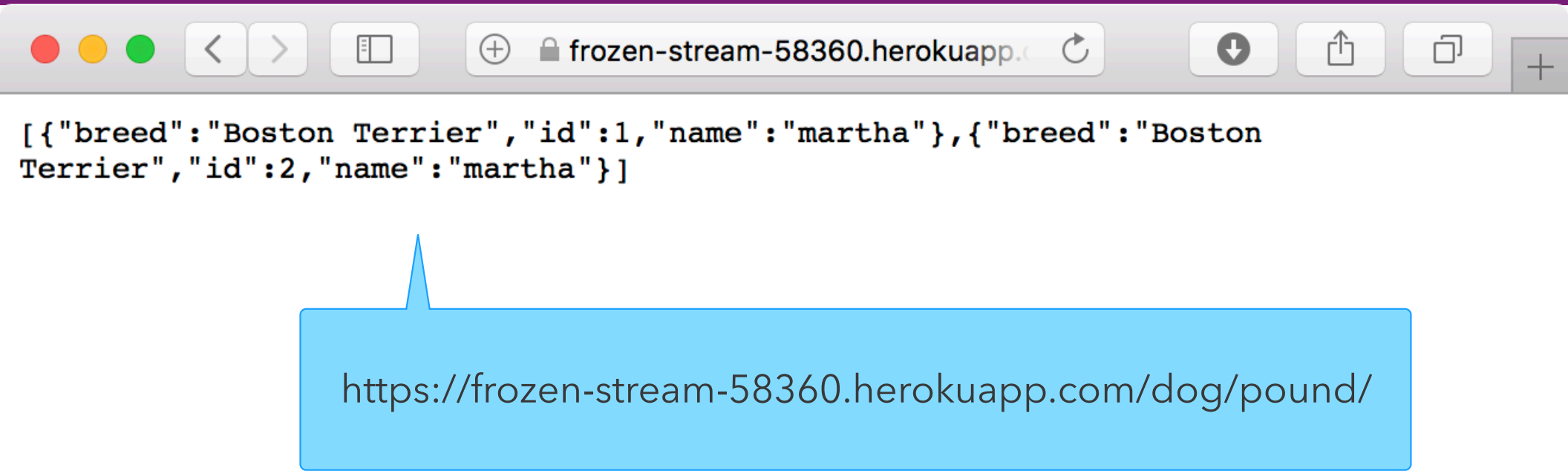
DEPLOY TO HEROKU WITH POSTGRESQL



DEPLOY TO HEROKU WITH POSTGRESQL



DEPLOY TO HEROKU WITH POSTGRESQL



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

```
//
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
}
```

Get the first entry

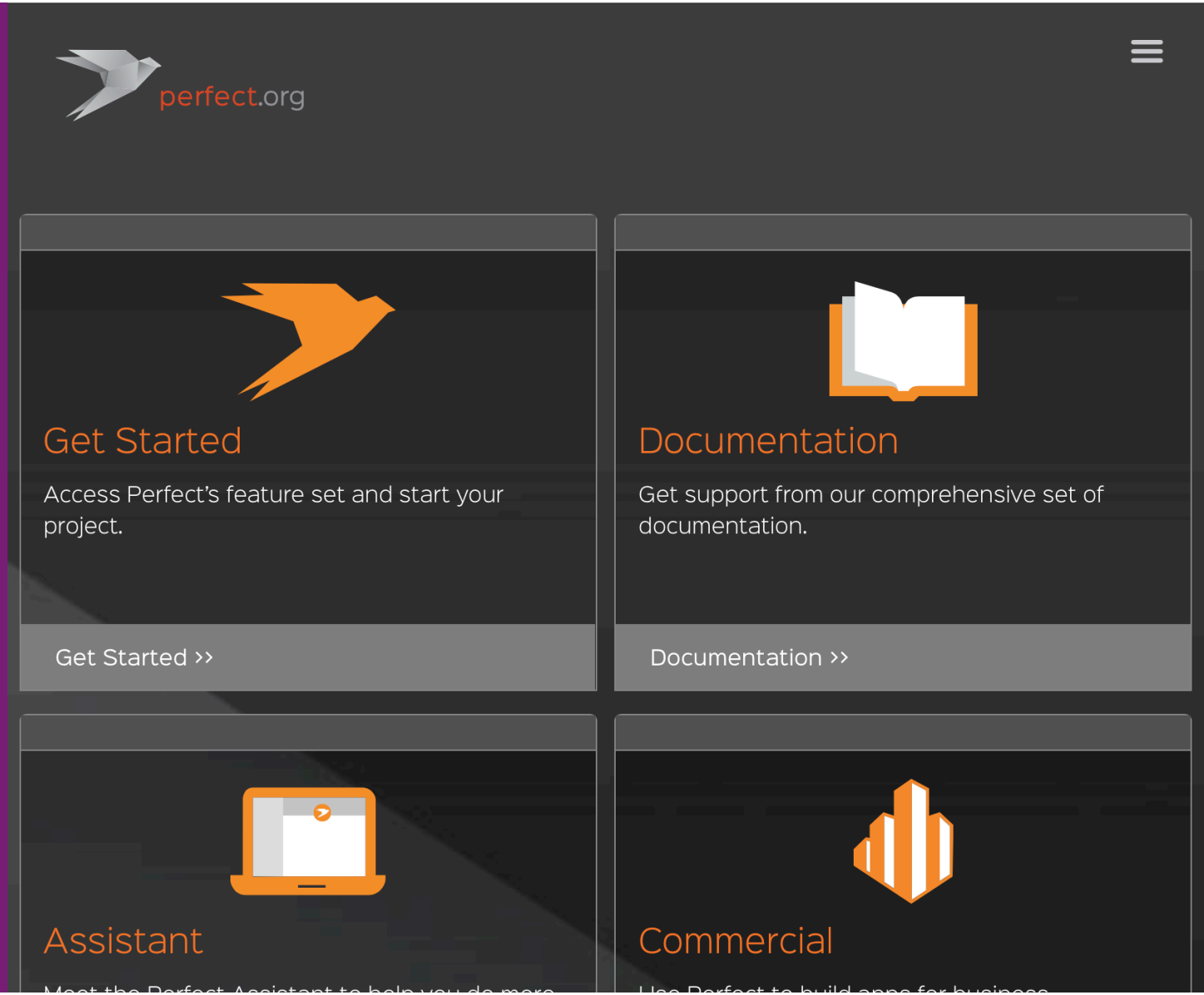
Change values

Save

PERFECT

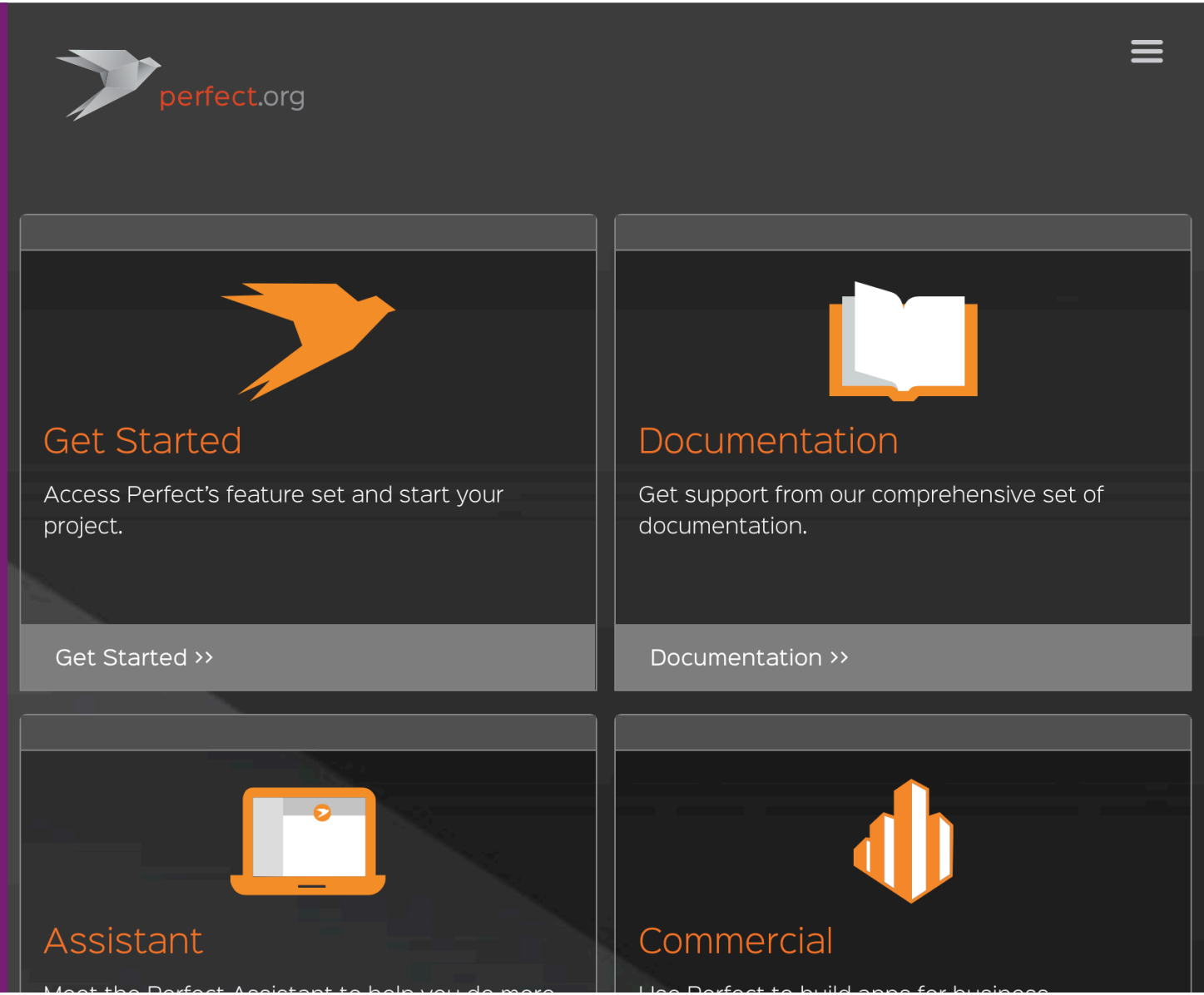
PERFECT

- Web framework and server for Swift server applications



PERFECT

- Requirements
  - Xcode 8 Mac
  - Linux toolchain

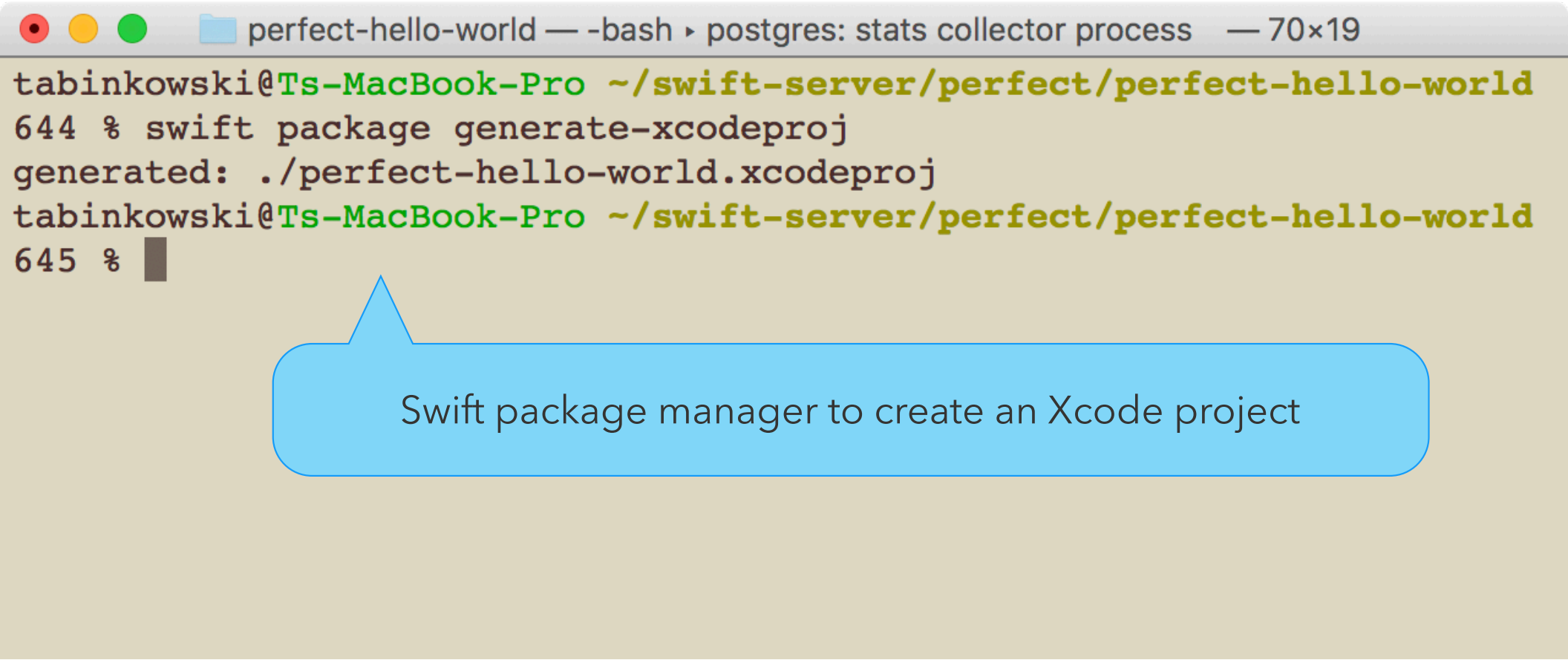


PERFECT

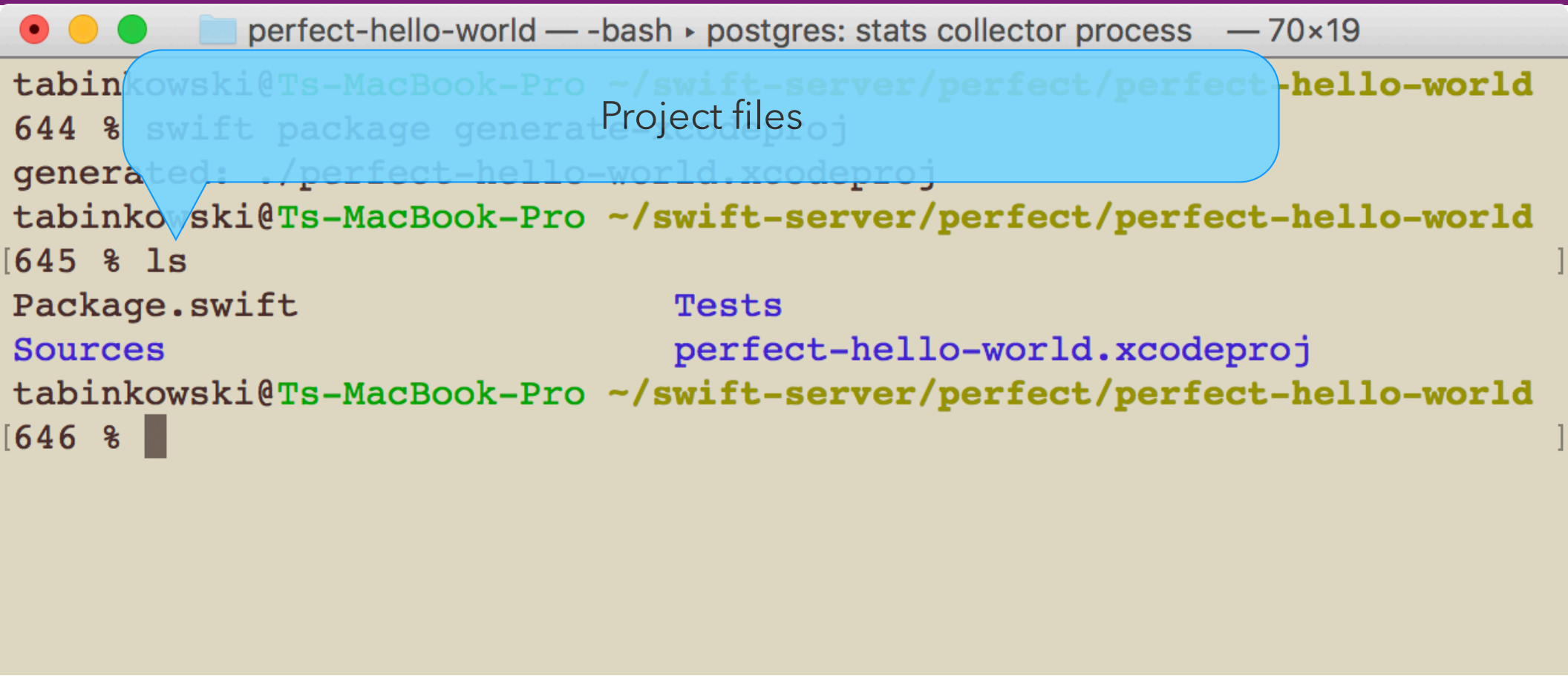
```
[637 % cd perfect/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[638 % ls
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[639 % mkdir perfect-hello-world
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[640 % cd perfect-hello-world/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[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 %
```

Swift package manager to create an executable package

PERFECT

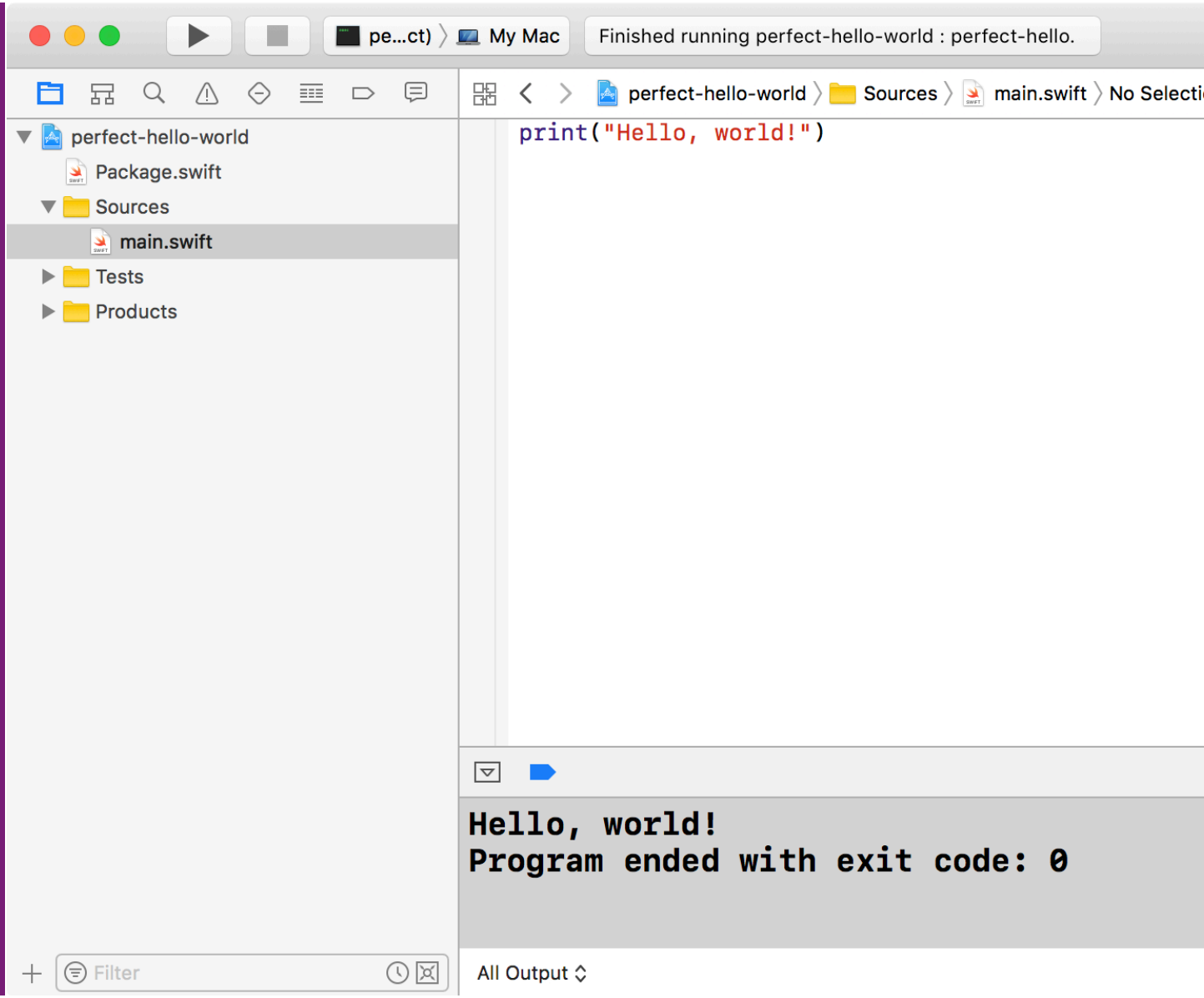

perfect-hello-world — -bash ▸ postgres: stats collector process — 70×19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
645 %
Swift package manager to create an Xcode project

PERFECT

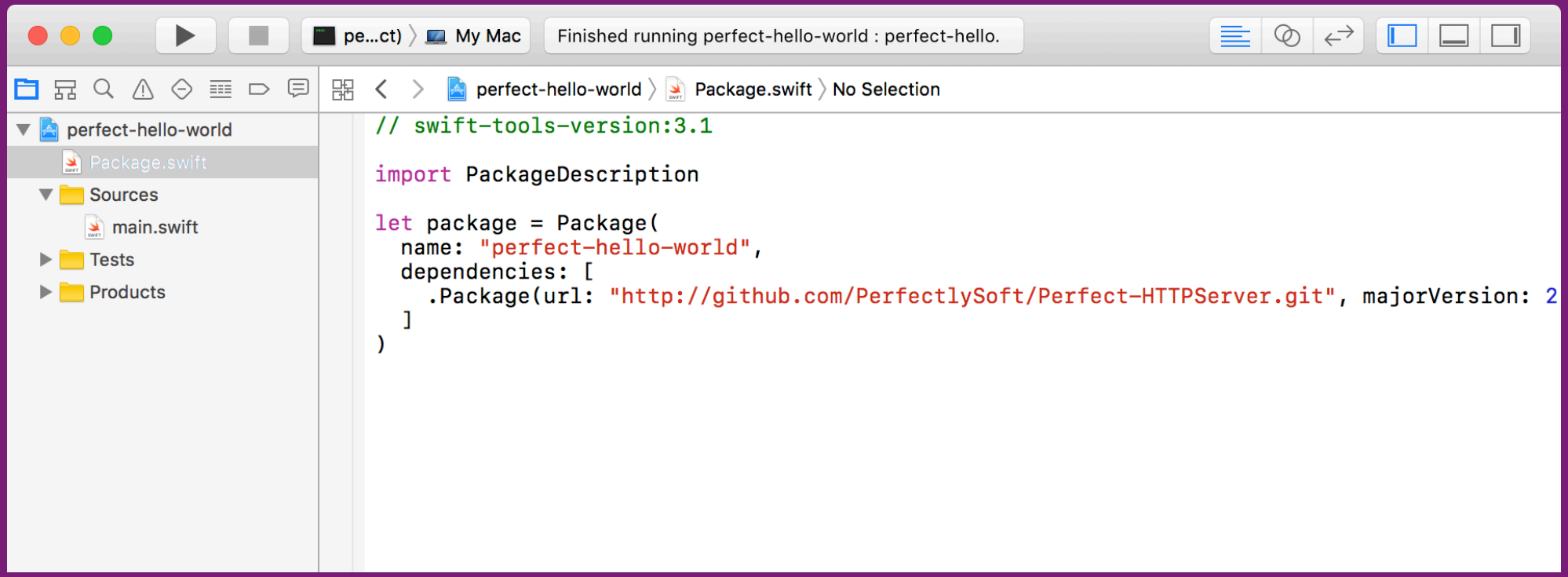

perfect-hello-world — -bash ▸ postgres: stats collector process — 70×19
Project files
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
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

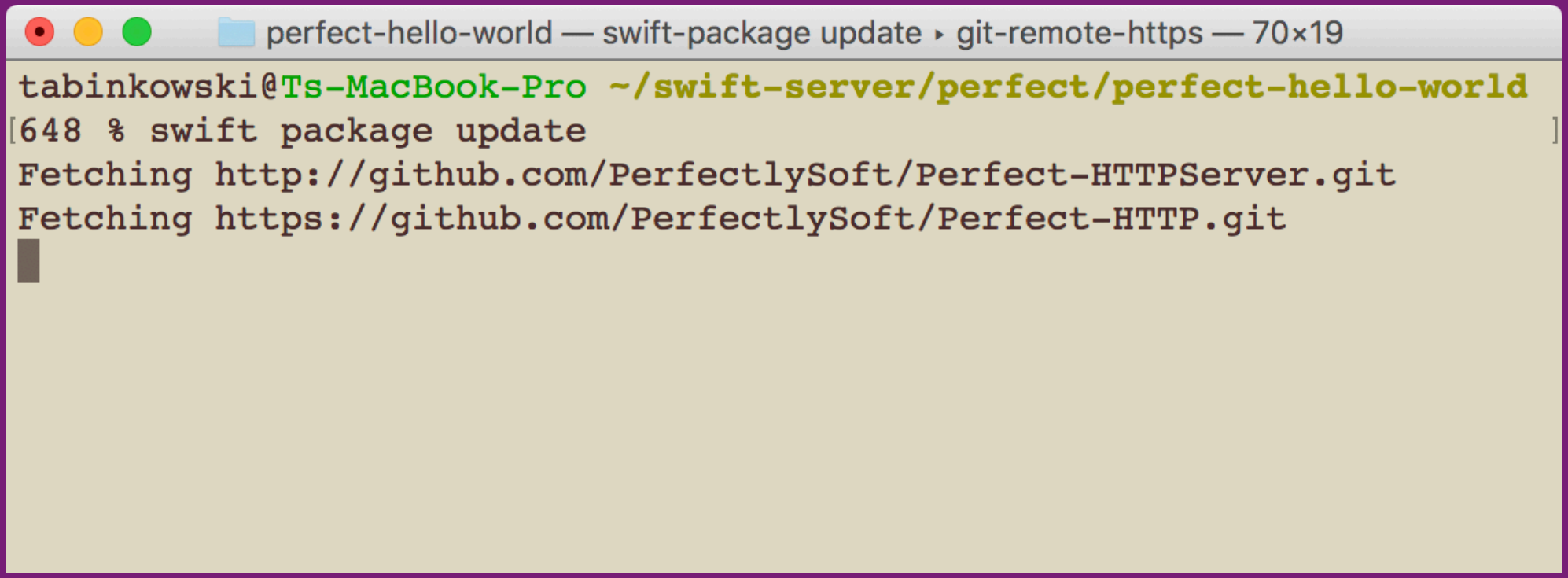


PERFECT



- Add Perfect as a project dependency

PERFECT

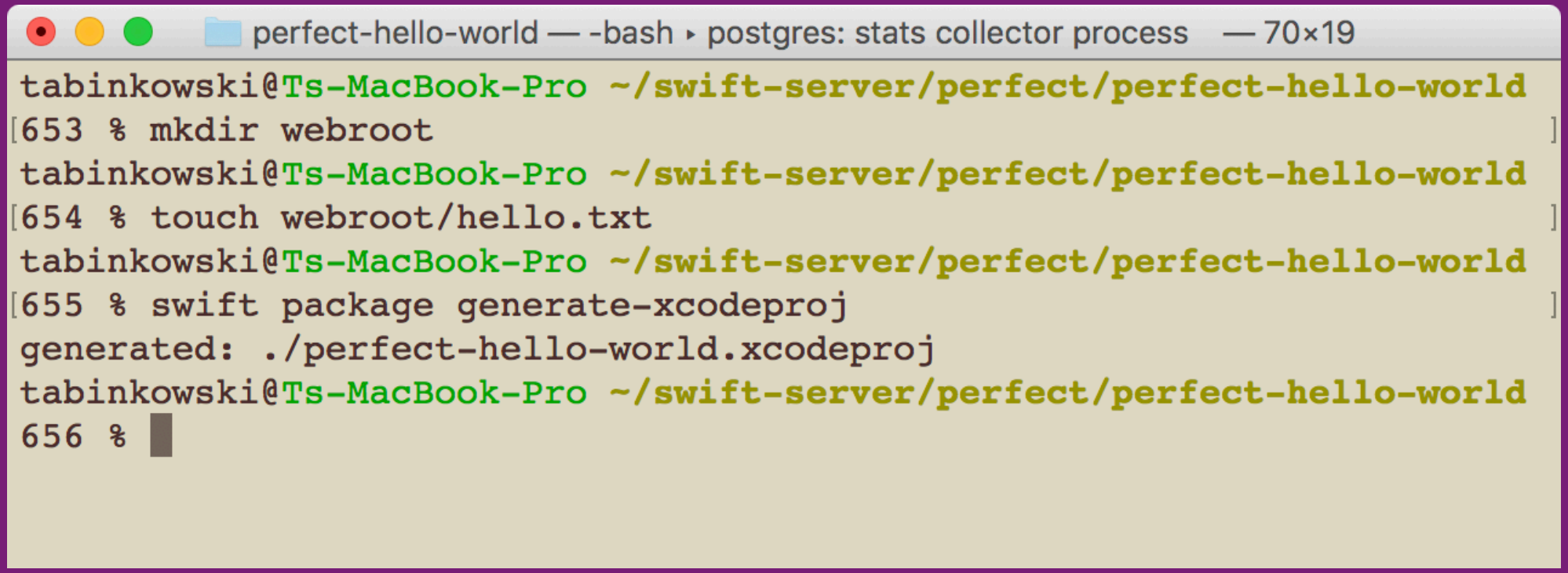


- Update the package manager

PERFECT

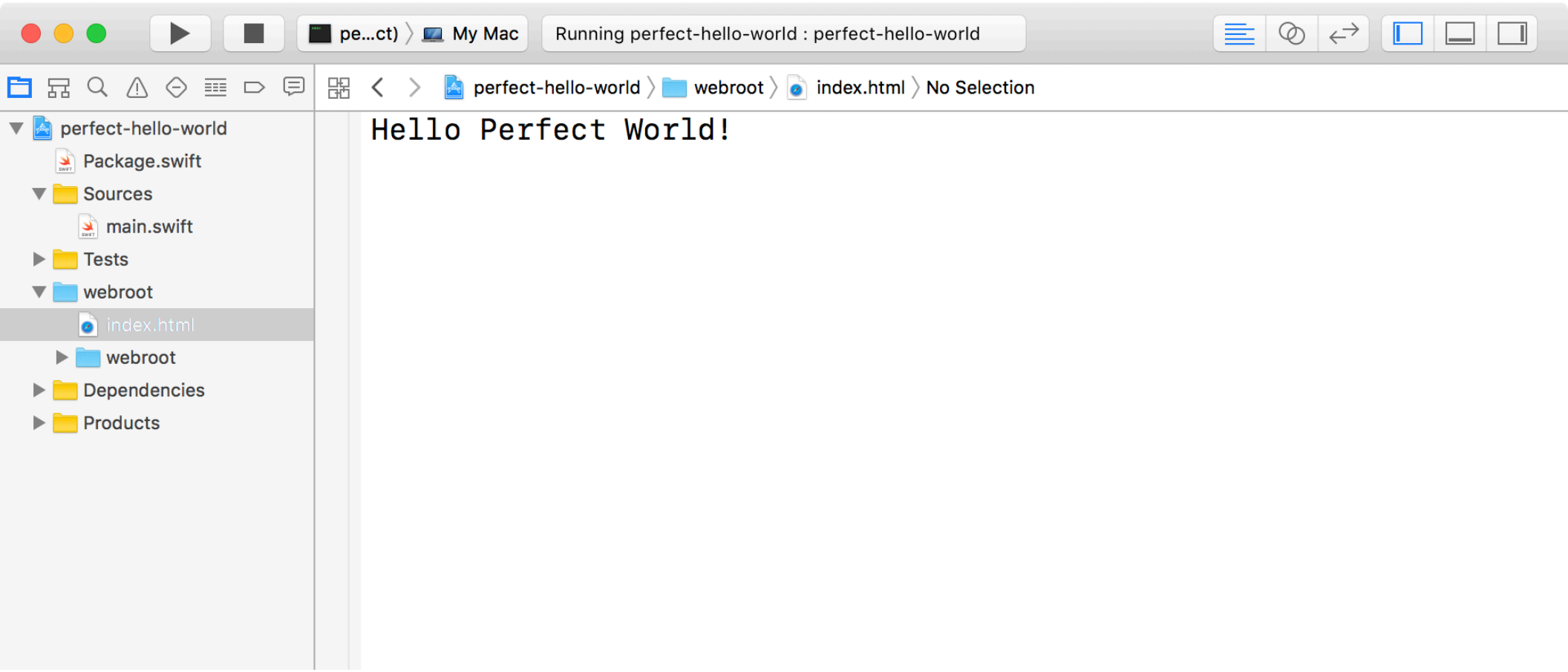
- Update the Xcode project

PERFECT



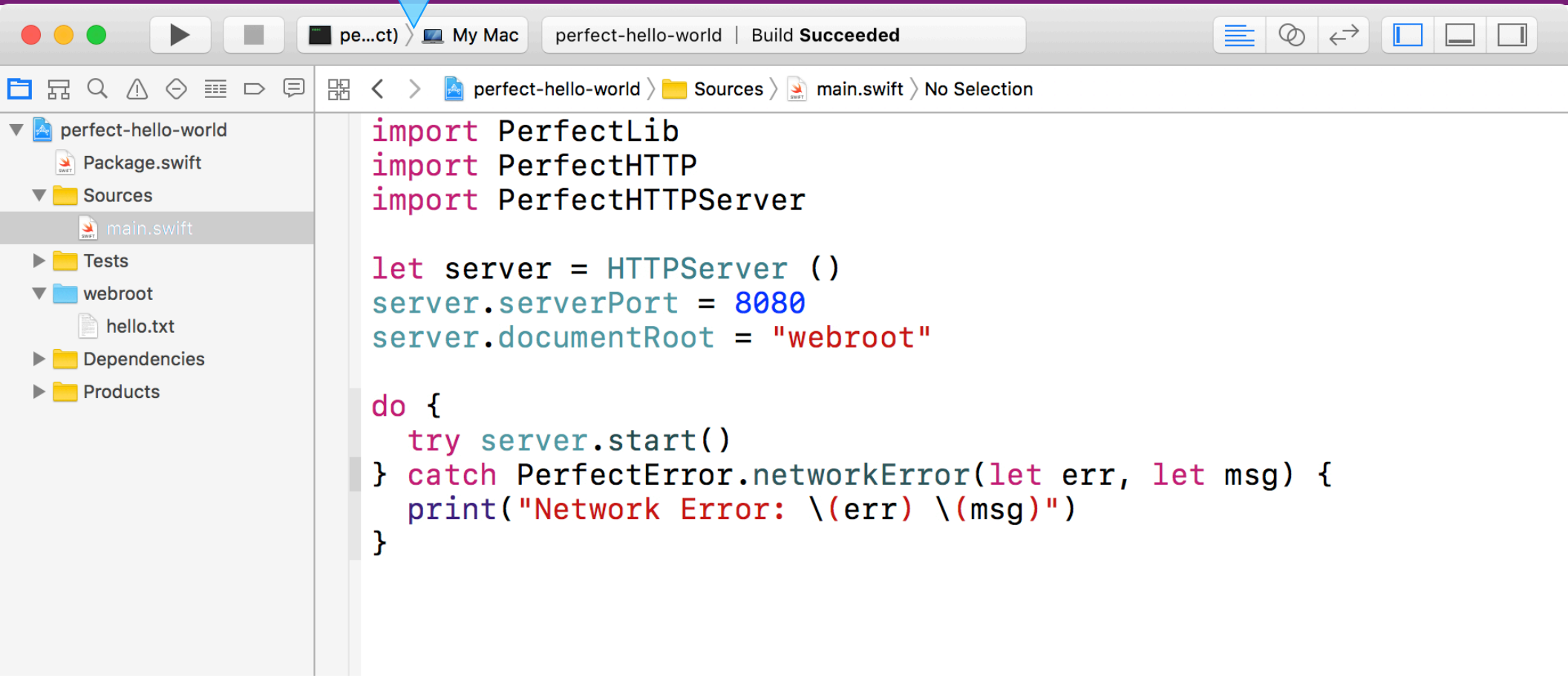
- Create a root directory for the web server

PERFECT

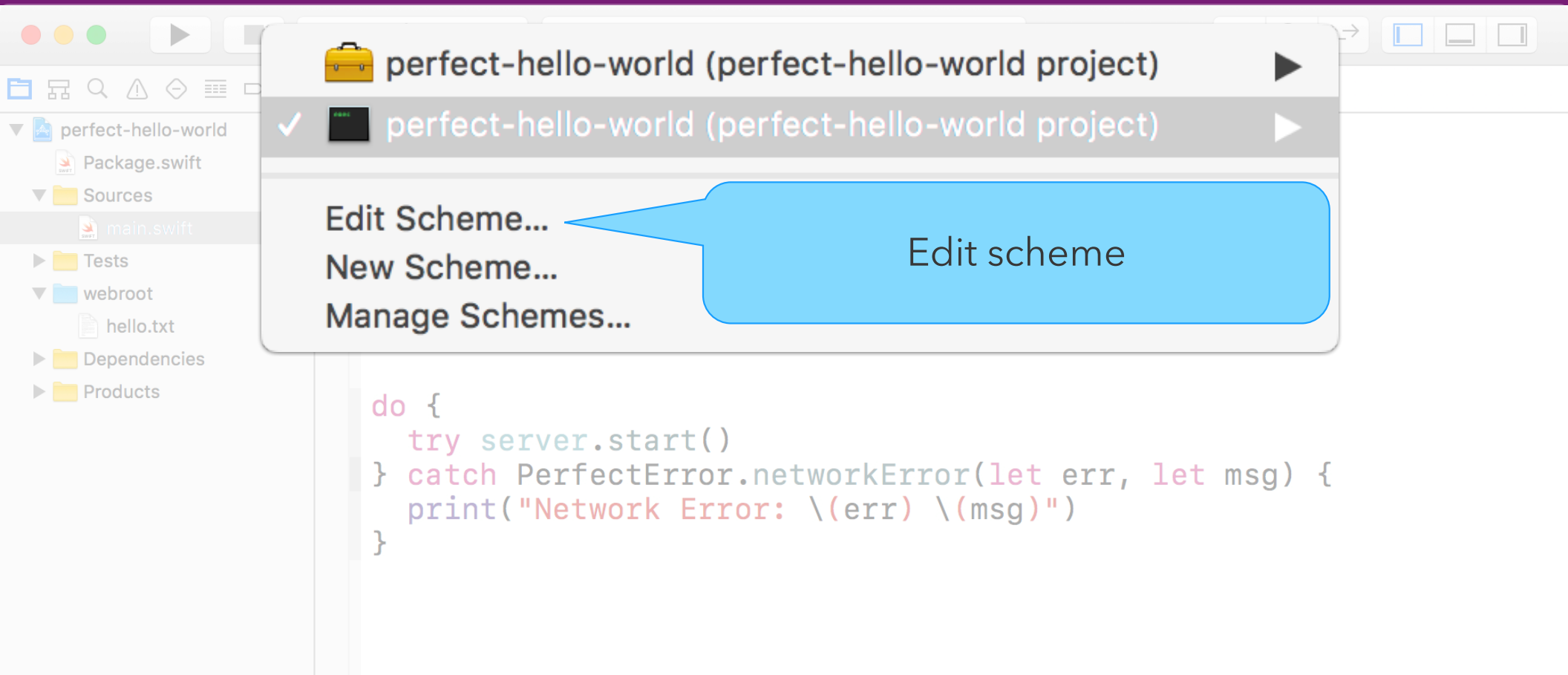

pe...ct) 〉 My Mac
Running perfect-hello-world : perfect-hello-world
perfect-hello-world 〉 webroot 〉 index.html 〉 No Selection
perfect-hello-world
Package.swift
Sources
main.swift
Tests
webroot
index.html
webroot
Dependencies
Products
Hello Perfect World!

PERFECT

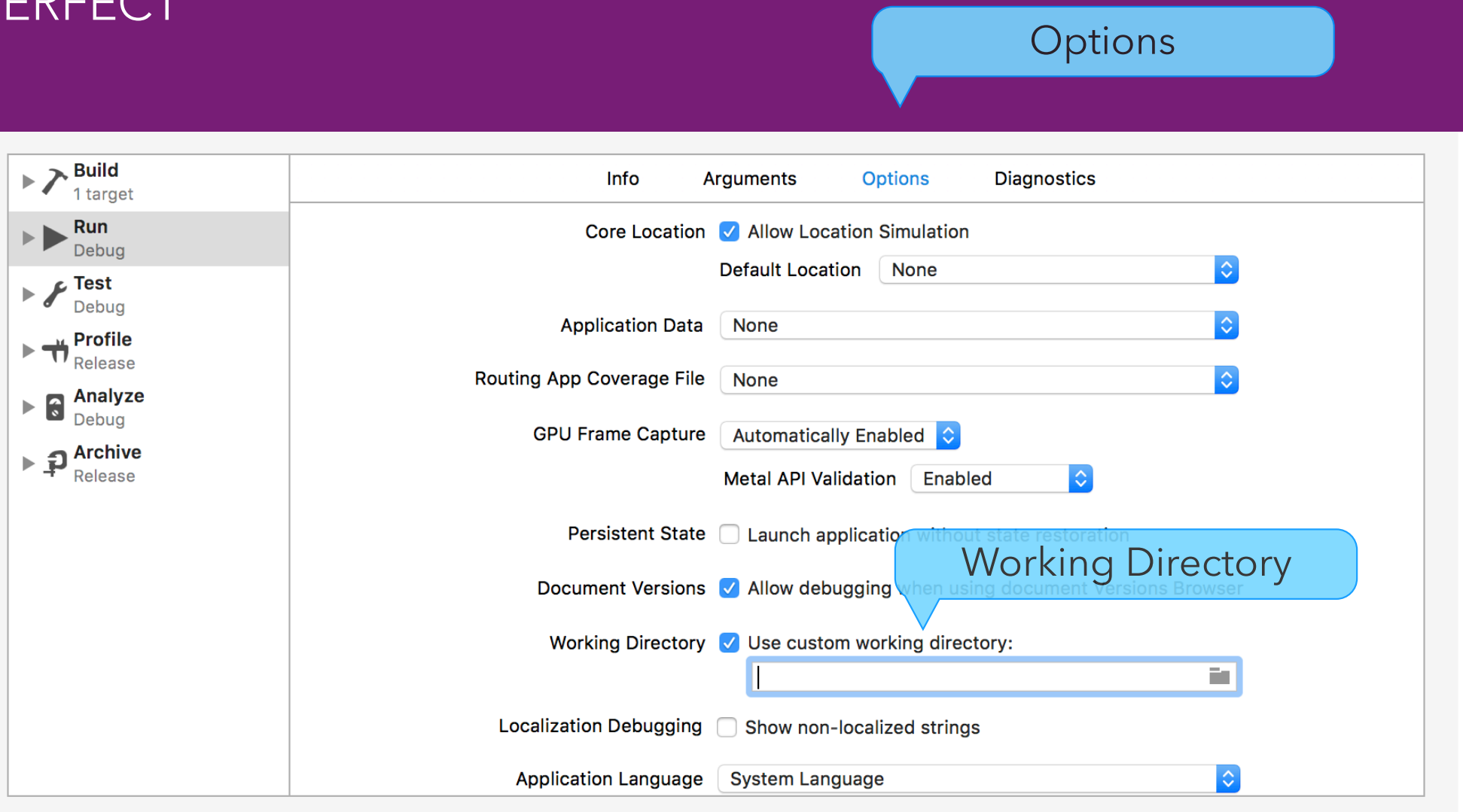
Create a server



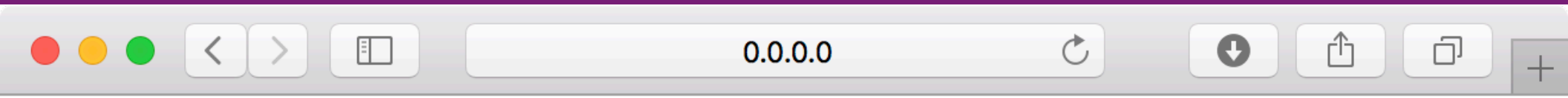
PERFECT


perfect-hello-world (perfect-hello-world project)
perfect-hello-world (perfect-hello-world project)
Edit Scheme...
New Scheme...
Manage Schemes...
Edit scheme
perfect-hello-world
Package.swift
Sources
main.swift
Tests
webroot
hello.txt
Dependencies
Products
do {
try server.start()
} catch PerfectError.networkError(let err, let msg) {
print("Network Error: \(err) \(msg)")
}

PERFECT

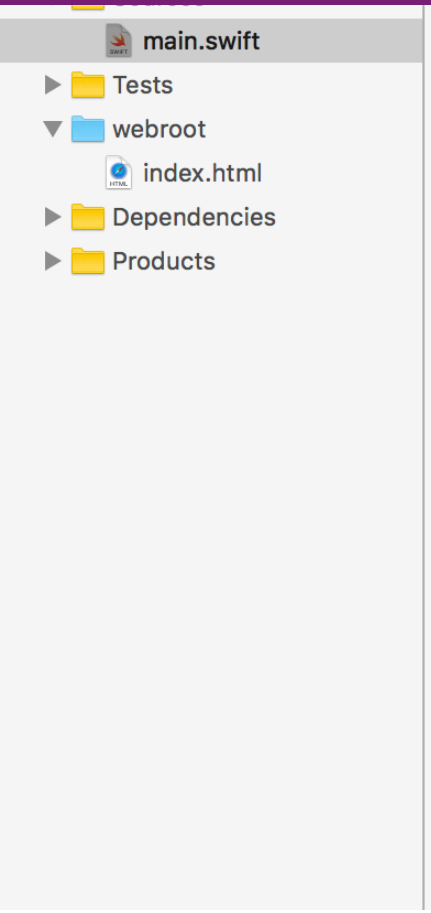

Options
Build
1 target
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Diagnostics
Core Location
Allow Location Simulation
Default Location
None
Application Data
None
Routing App Coverage File
None
GPU Frame Capture
Automatically Enabled
Metal API Validation
Enabled
Persistent State
Launch application without state restoration
Document Versions
Allow debugging when using document versions Browser
Working Directory
Working Directory
Use custom working directory:
Localization Debugging
Show non-localized strings
Application Language
System Language

PERFECT



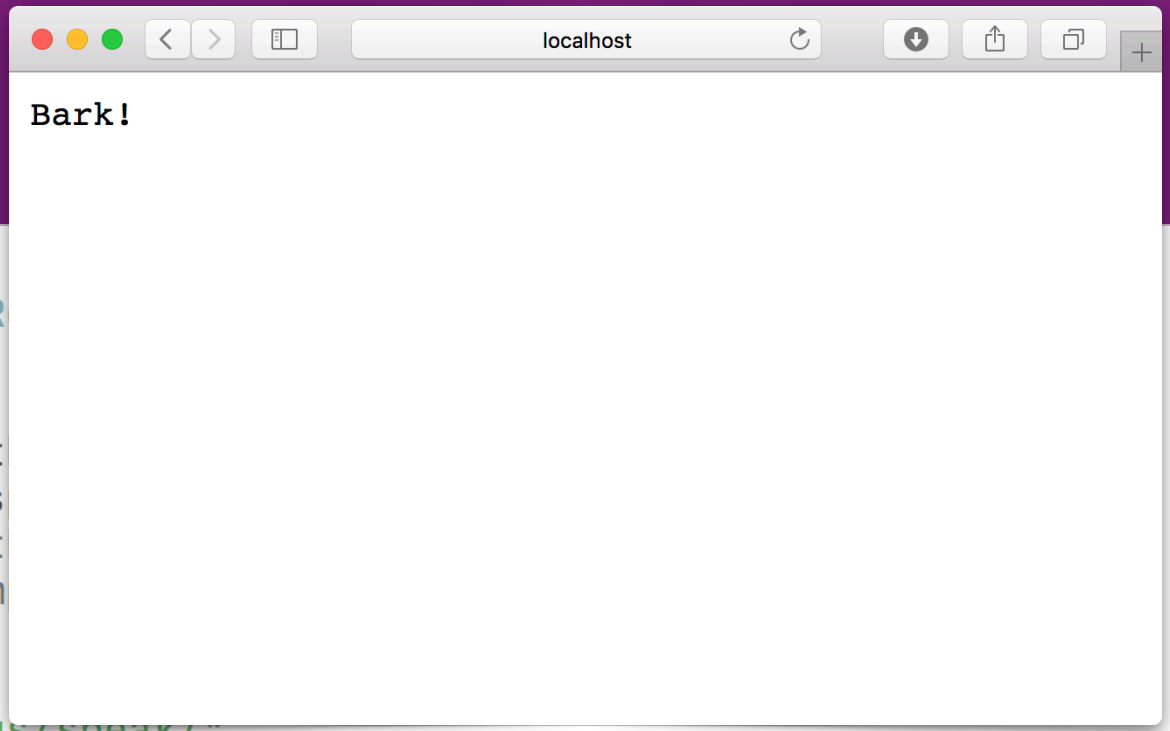
Hello Perfect World!

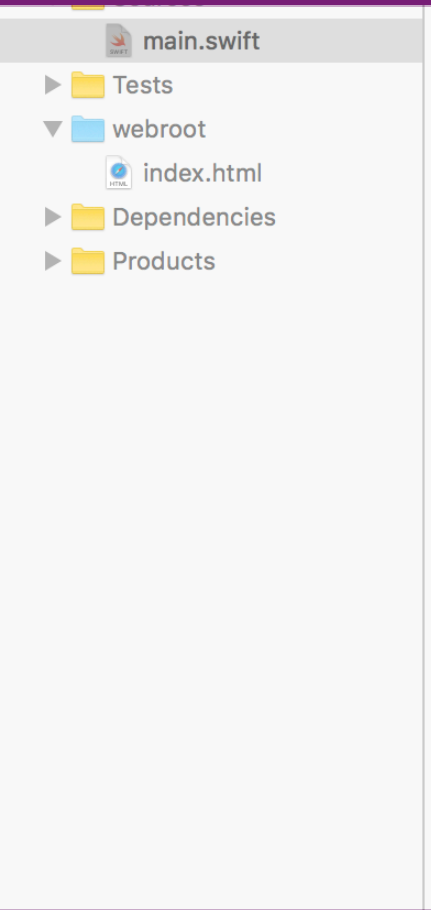
PERFECT




```
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



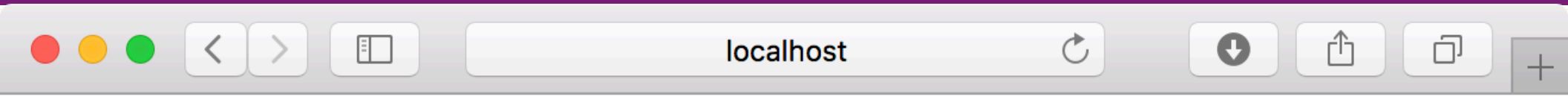



```
var routes = R

// Route "/"
routes.add(met
  request, res
  response.set
  response.com
})

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

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



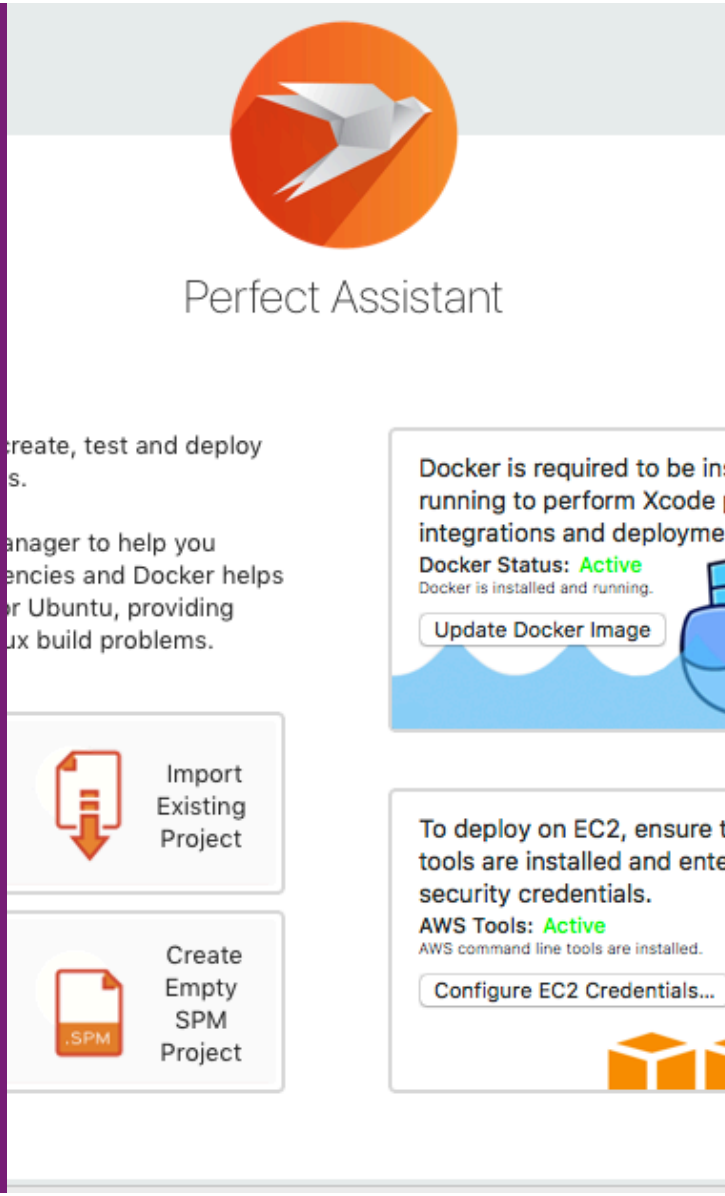

```
lassie says Bark!
```

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

PERFECT ASSISTANT

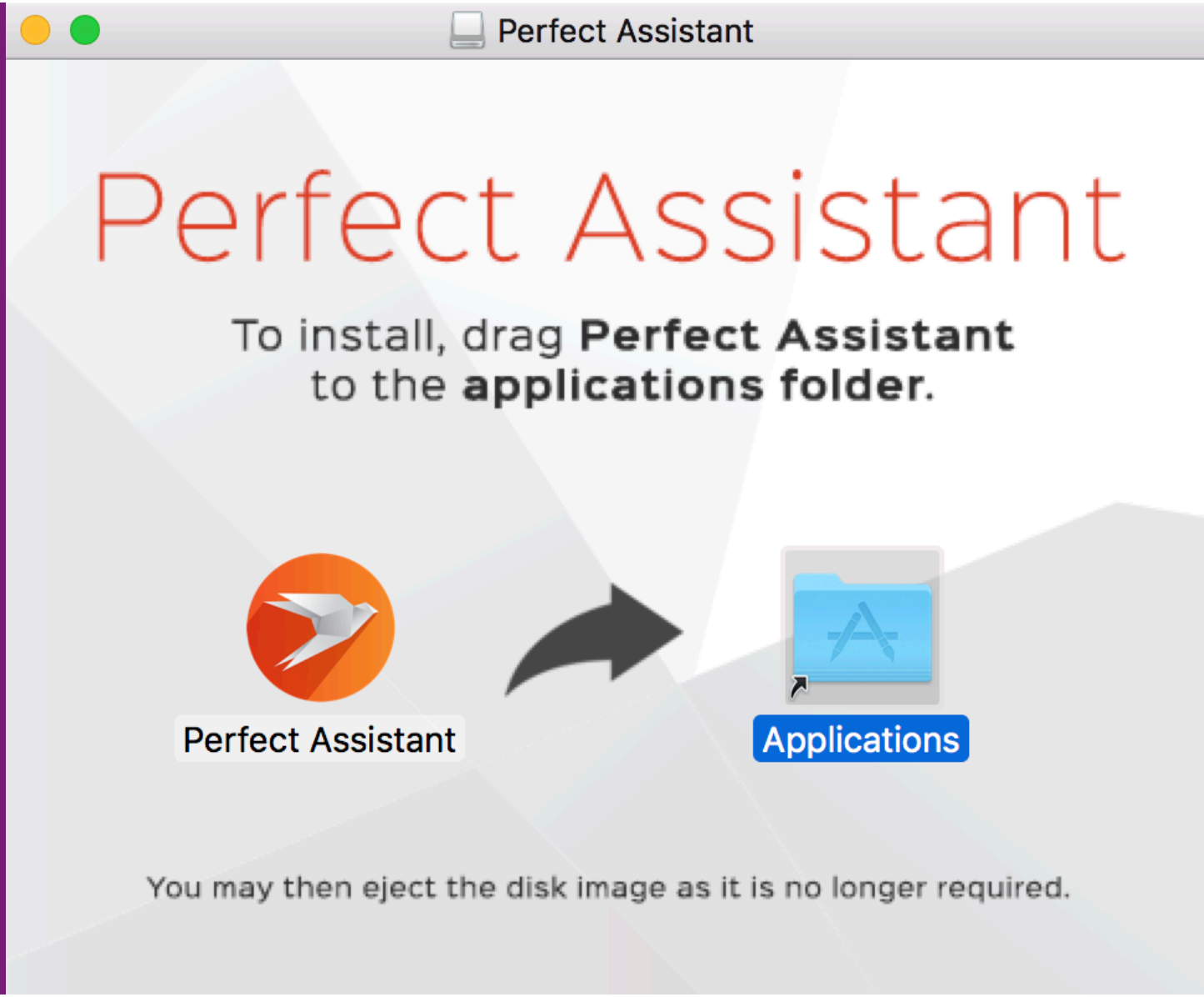
PERFECT ASSISTANT

- 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



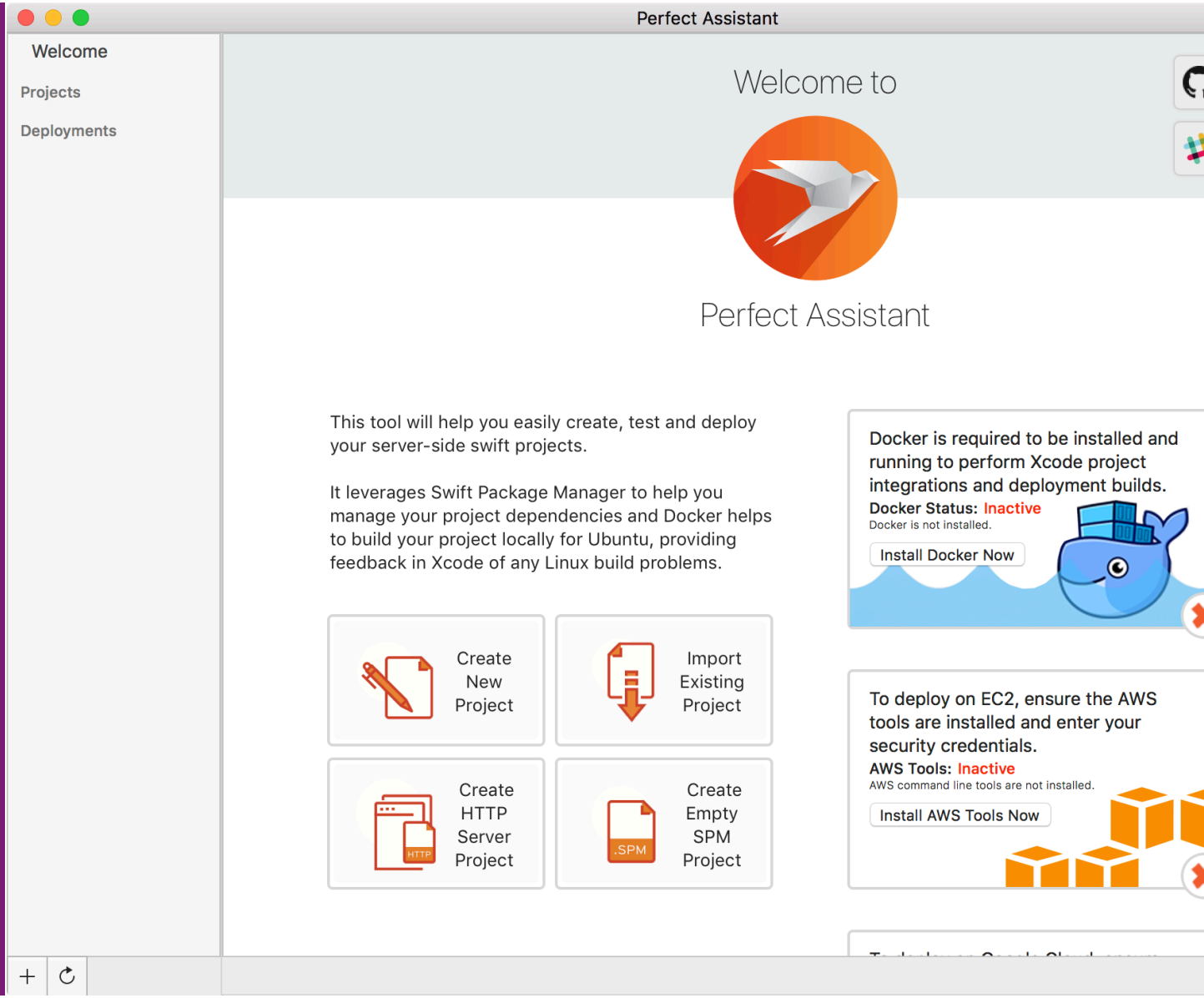
PERFECT ASSISTANT

- Download and install



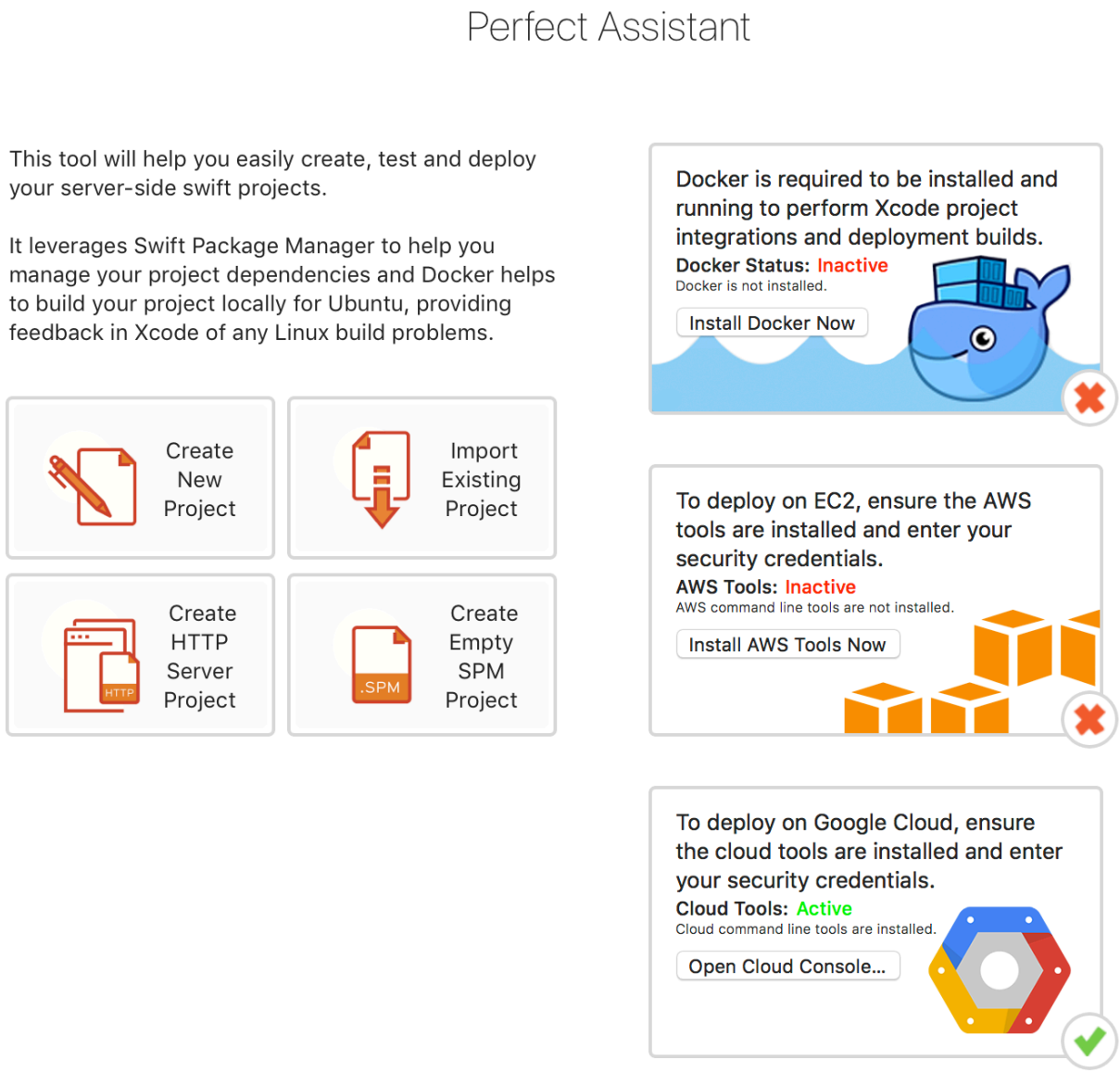
PERFECT ASSISTANT

- Download and install



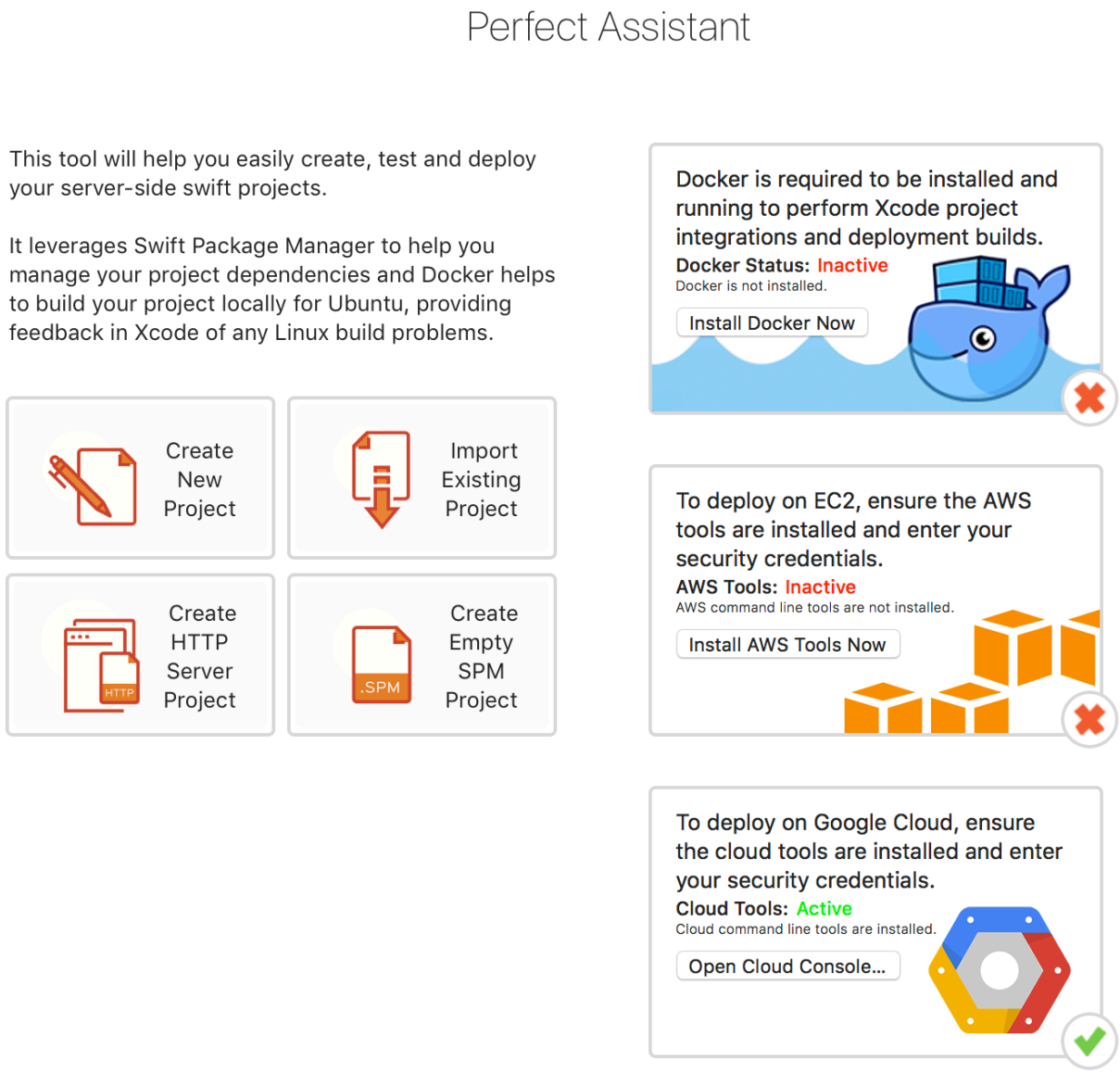
PERFECT ASSISTANT

- Deploy using Docker

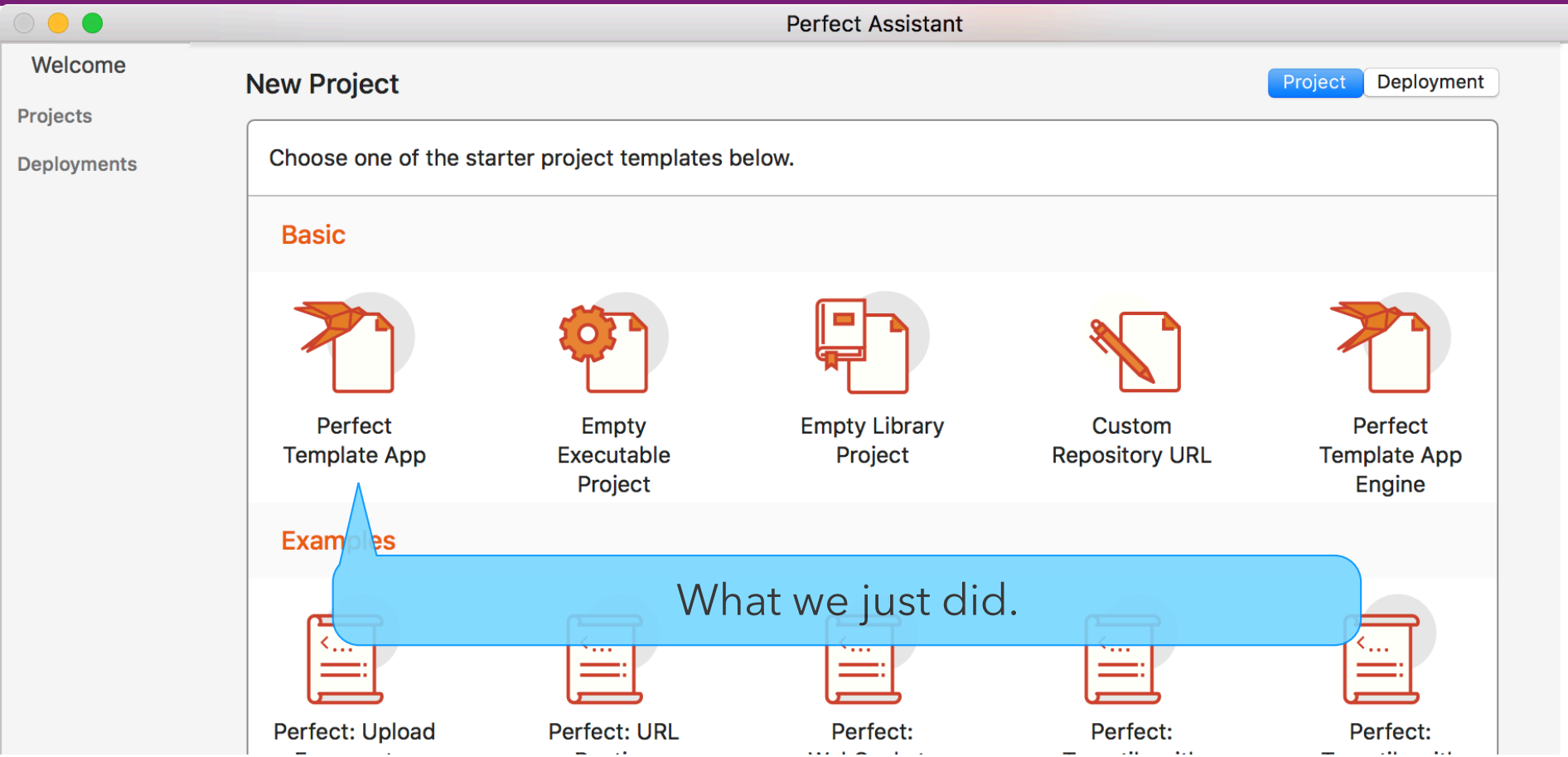


PERFECT ASSISTANT

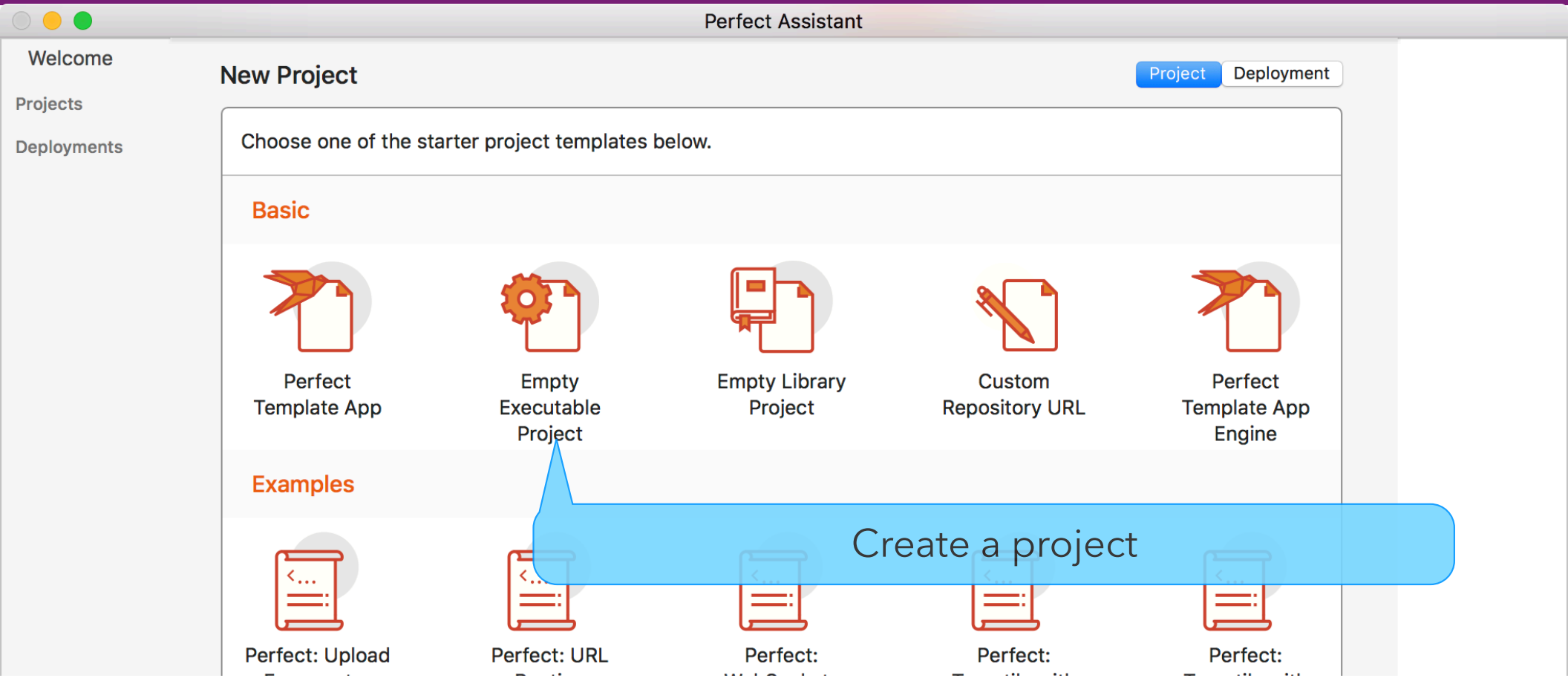
- Deploy using Docker



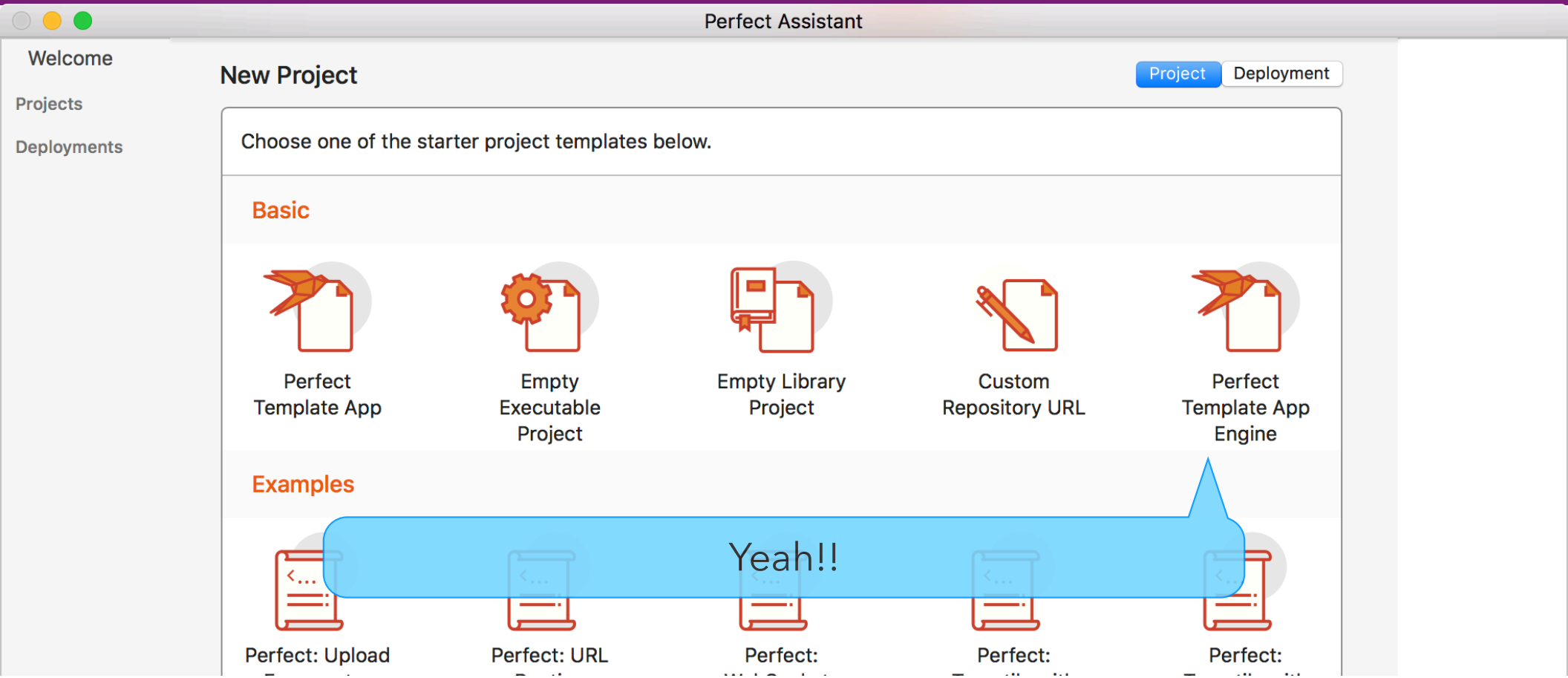
PERFECT ASSISTANT



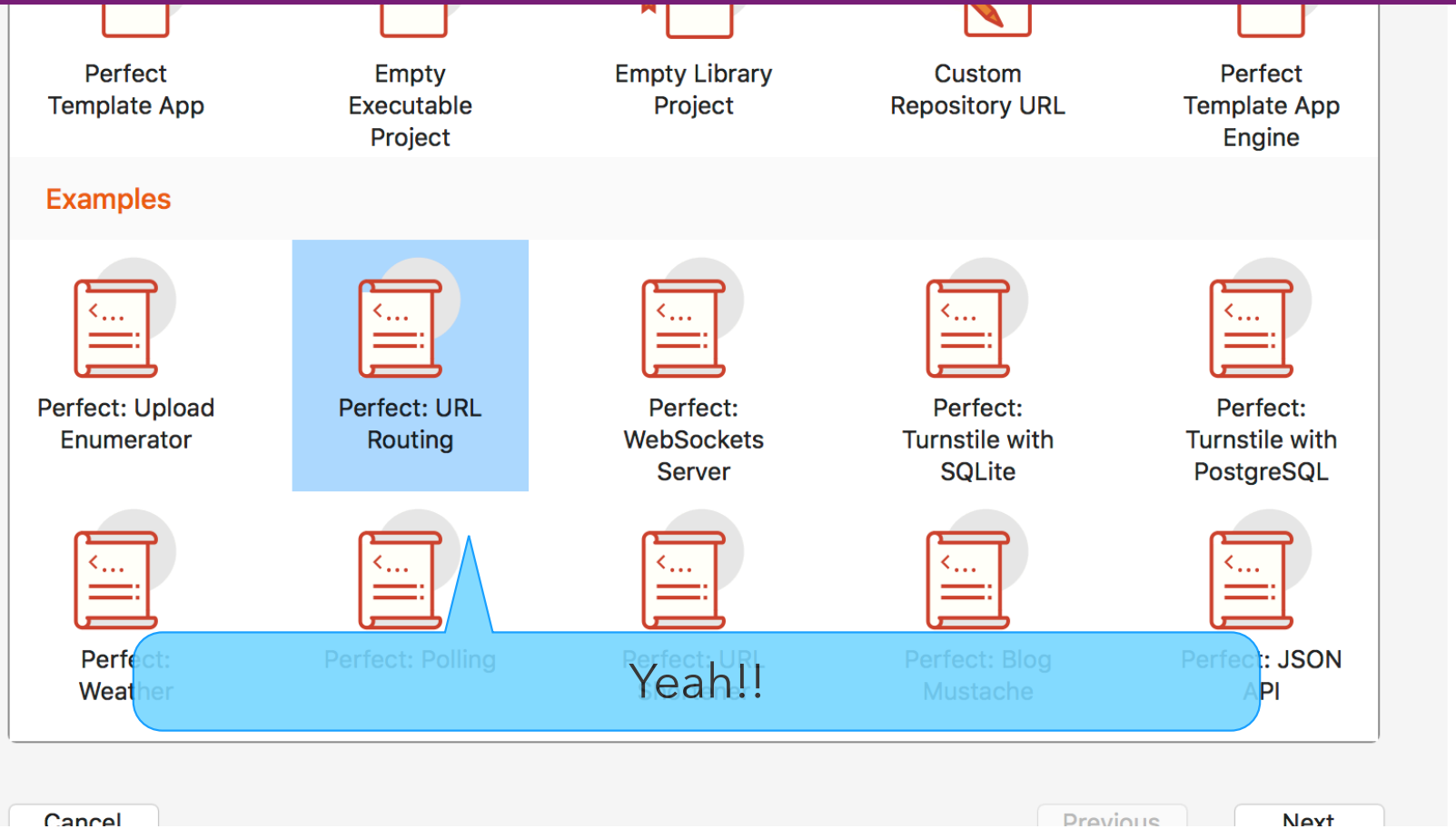
PERFECT ASSISTANT



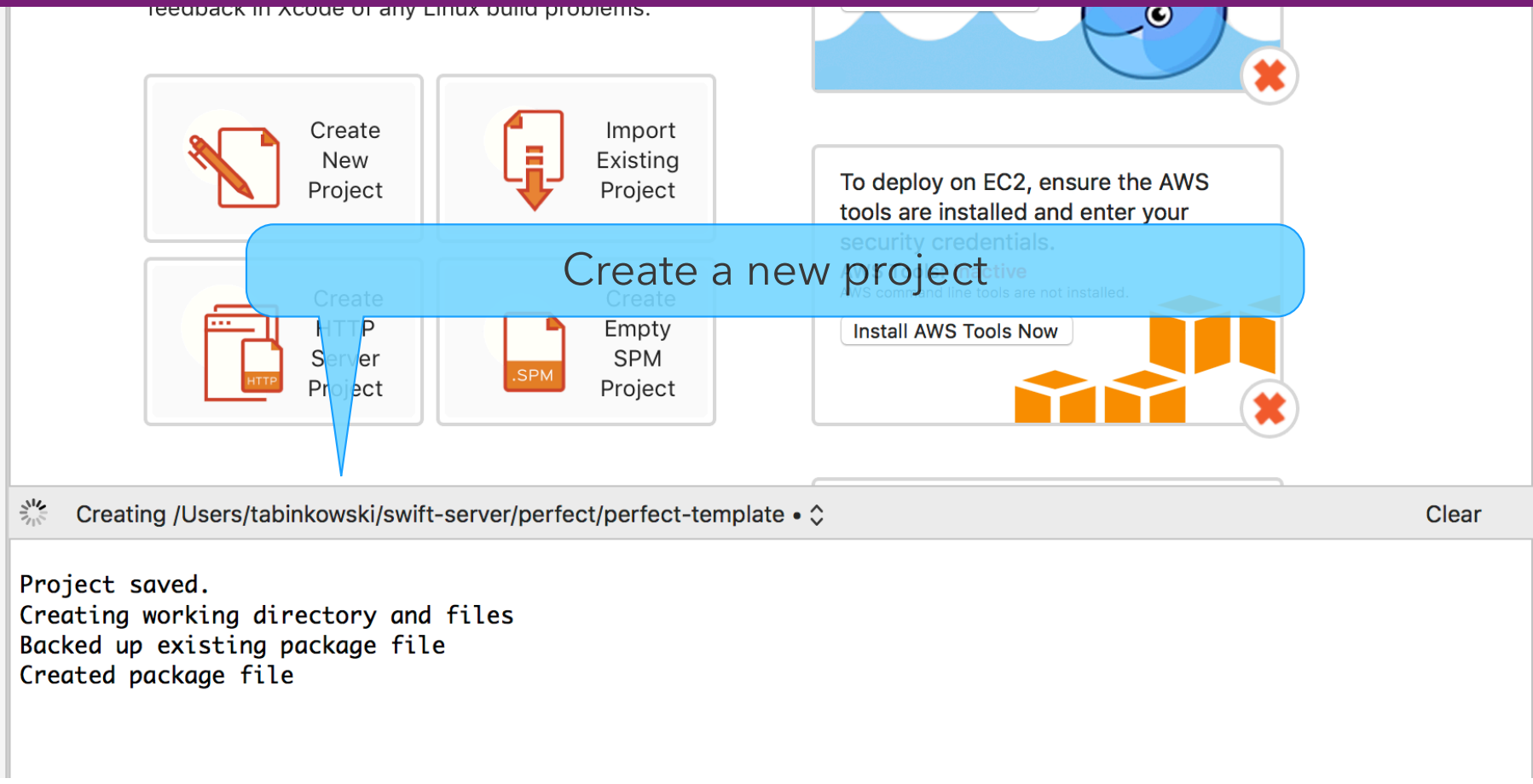
PERFECT ASSISTANT



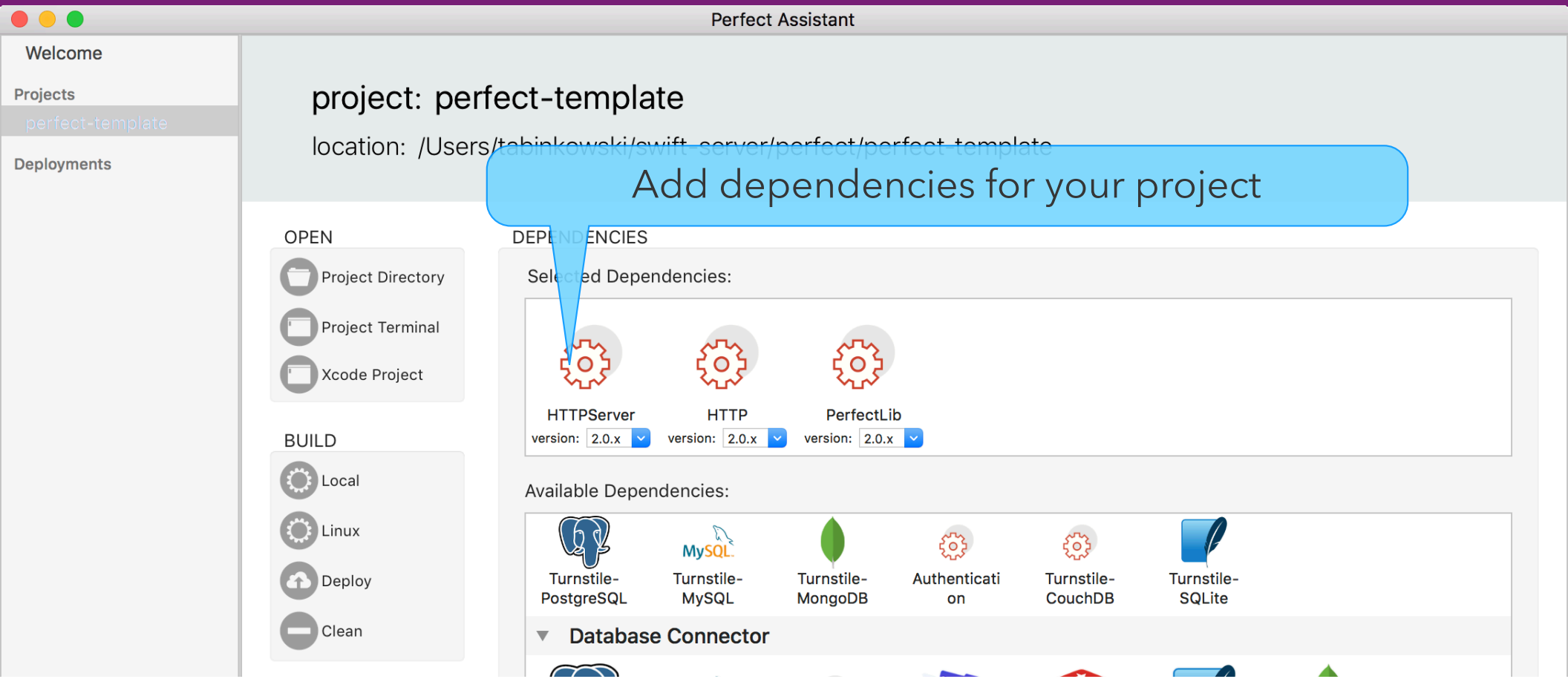
PERFECT ASSISTANT



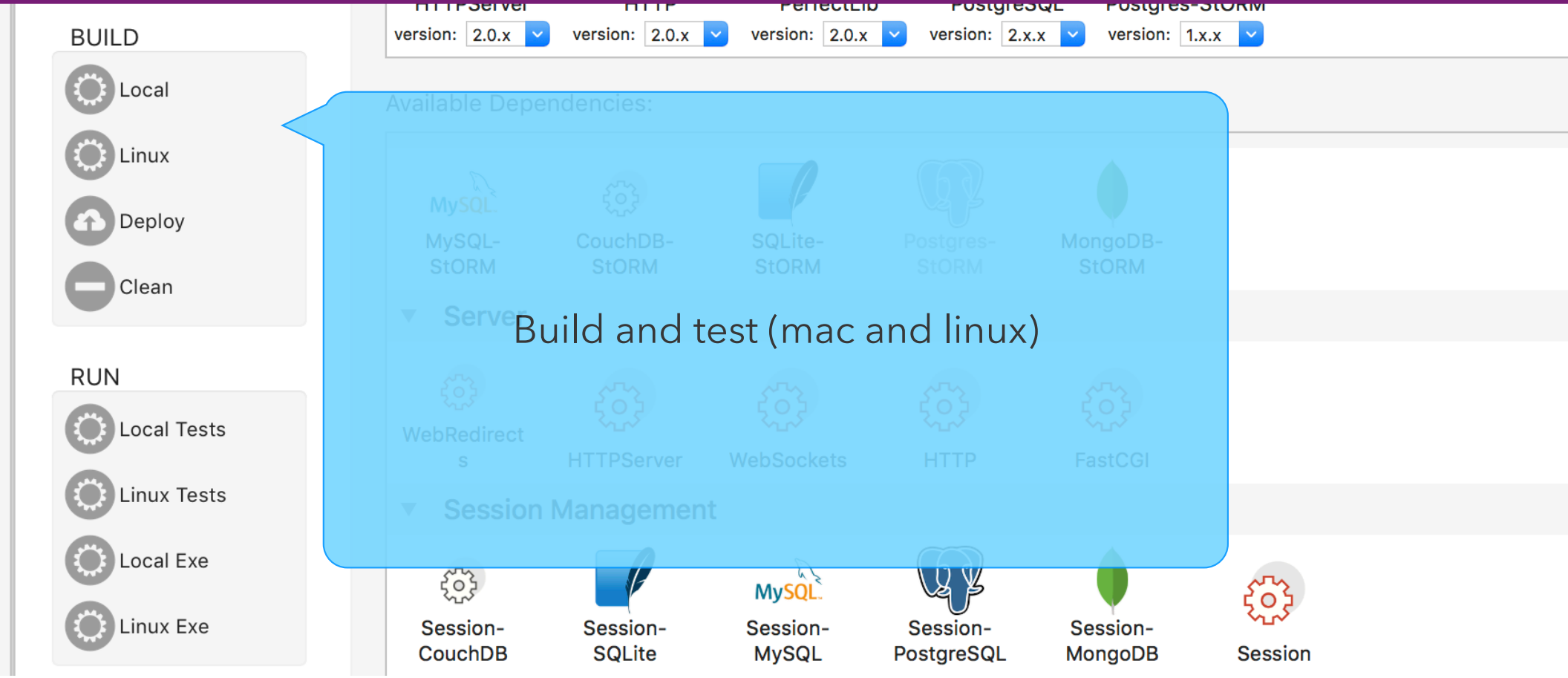
PERFECT ASSISTANT



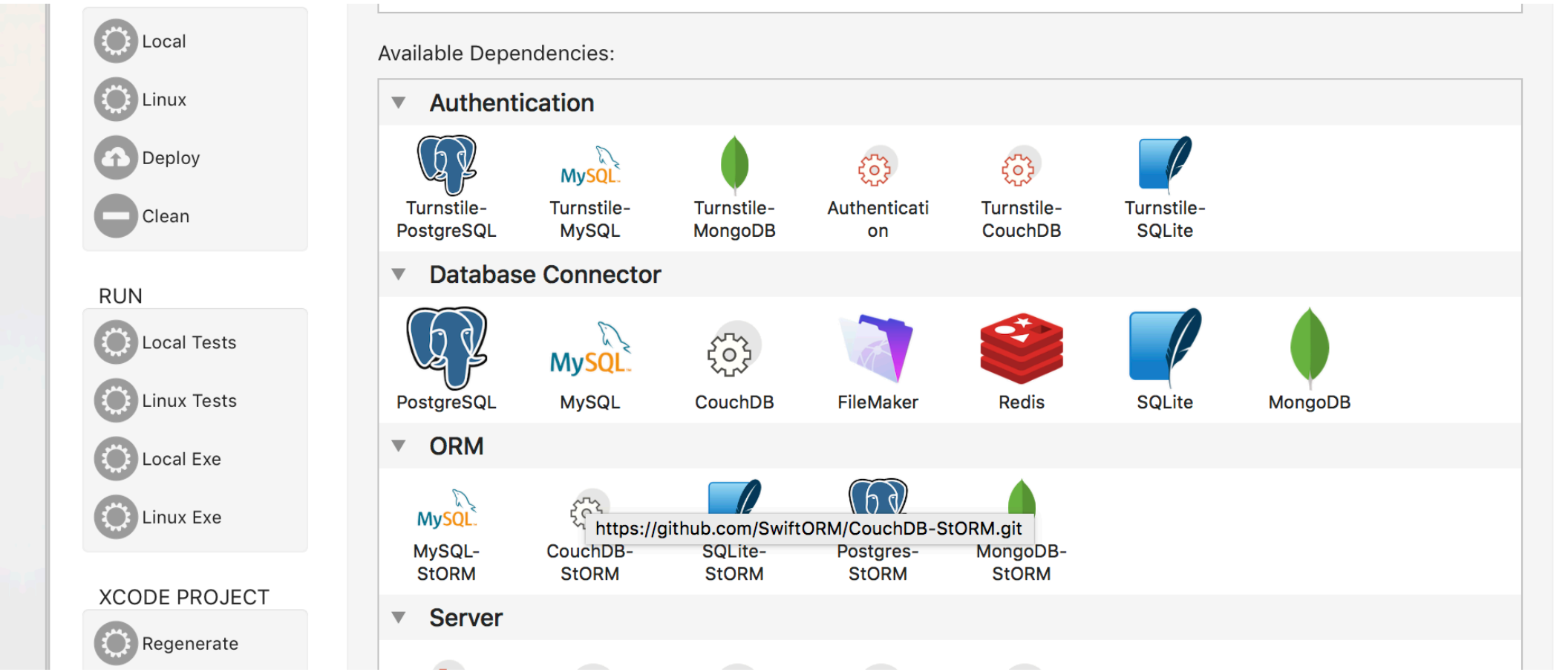
PERFECT ASSISTANT



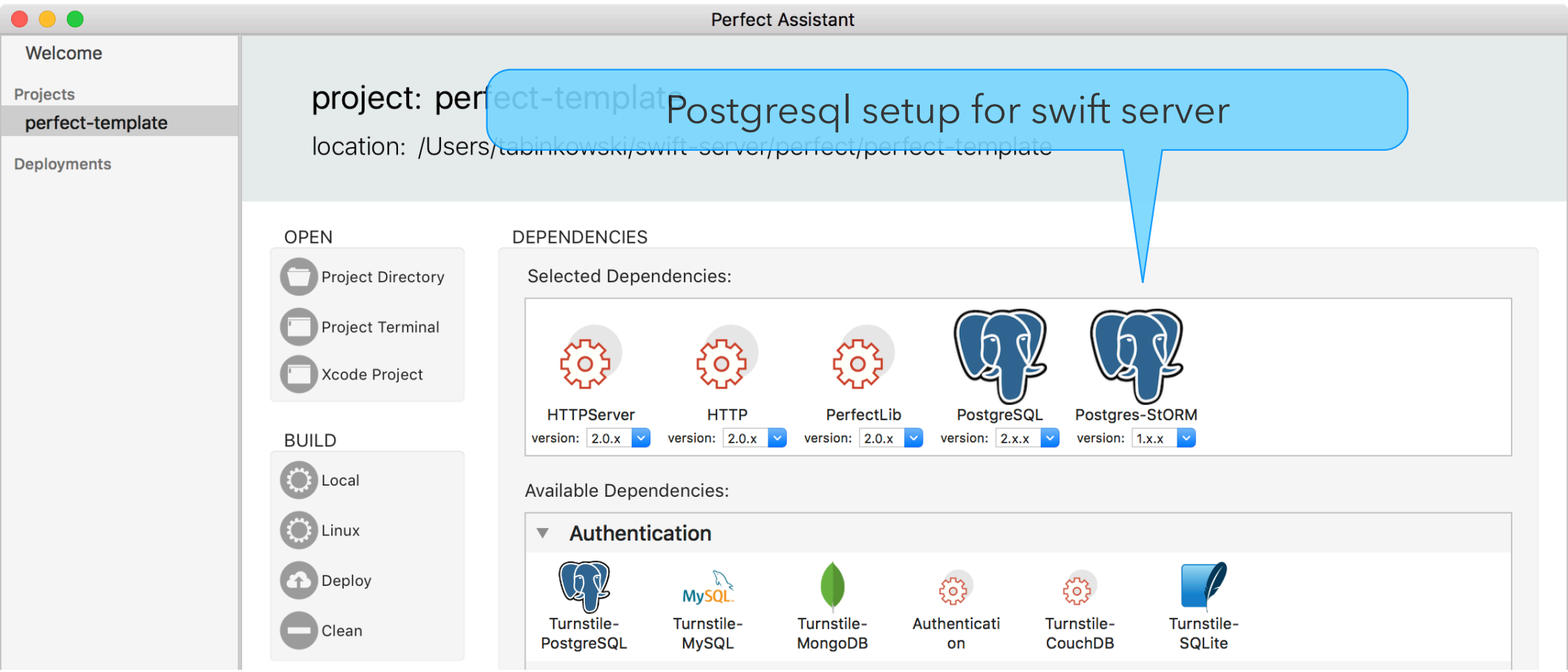
PERFECT ASSISTANT



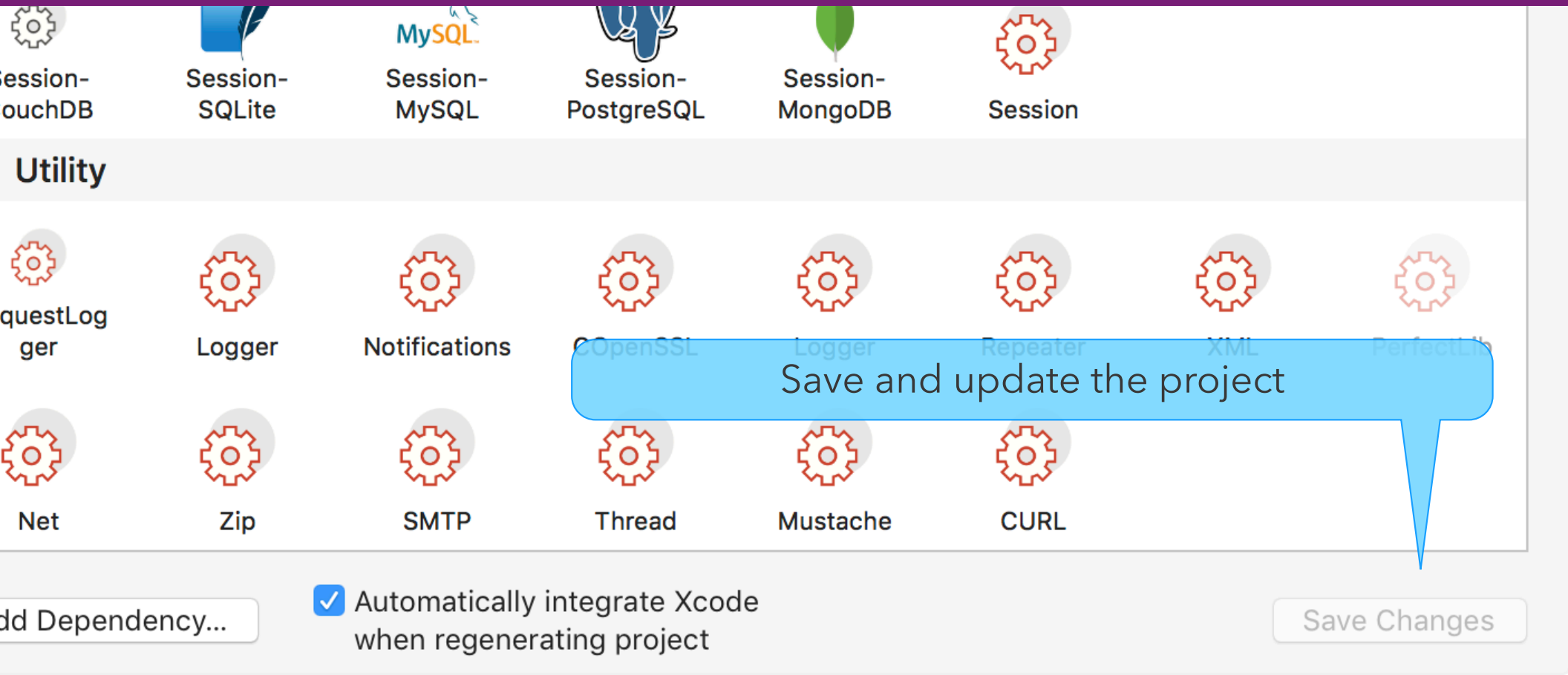
PERFECT ASSISTANT



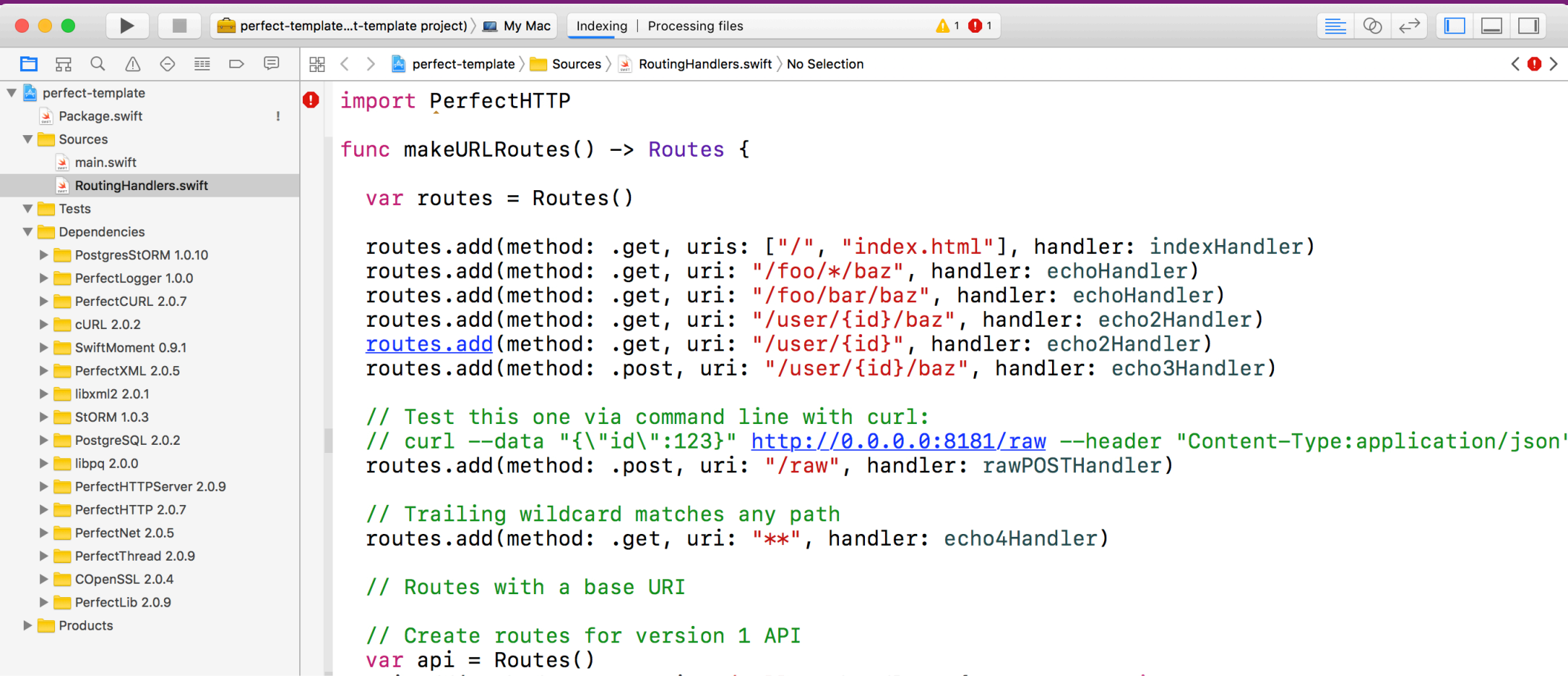
PERFECT ASSISTANT



PERFECT ASSISTANT



PERFECT ASSISTANT



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

	Perfect	Vapor	Kitura	Zewo	Node.js
Build Times					
1. --clean=dist	2nd Place	4th Place	1st Place	3rd Place	N/A
1. --clean	3rd Place	4th Place	2nd Place	1st Place	N/A
Memory Usage					
1. Starting Usage	1st Place	4th Place	3rd Place	2nd Place	5th Place
2. Peak Lower Load	1st Place	5th Place	4th Place	2nd Place	3rd Place
3. Peak Higher Load	1st Place	5th Place	4th Place	2nd Place	3rd Place
Thread Usage					
1. Starting Threads	1	1	4	1	7
2. Under Load	6	11	11	1	11
Blog Benchmarks					
1. Requests/Sec	1st Place	2nd Place	4th Place	5th Place	3rd Place
2. Latency	2nd Place	4th Place	1st Place	5th Place	3rd Place
JSON Benchmarks					
1. Requests/Sec	1st Place	3rd Place	2nd Place	5th Place	4th Place
2. Latency	1st Place	5th Place	2nd Place	4th Place	3rd Place

FUTURE OF SWIFT ON THE SERVER

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

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