THE UNIVERSITY OF CHICAGO

MPCS 51033 • AUTUMN 2019 • SESSION 8

BACKENDS FOR MOBILE APPLICATIONS

FINAL PROJECTS

CLASS NEWS

CLASS NEWS

- Office hours Thursday from 10-11:30 in Young 308

CLASS NEWS

- Week 7 - CloudKit
  - Assignment 5 assigned
- Week 8 - CloudKit Server Extensions; Swift on the Server
  - Case Study assigned
  - Finish server component of Assignment 5
- Week 9 - Case Studies in Class (Assignment 5 Due)
- Week 10 - Cloud Roundup (Realm, Serverless, etc.)

CLASS NEWS

- Case Studies
  - Find an area of interest in mobile/cloud computing and present to class
  - Ideas:
    - New service (serverless, azure, aws lambda)
    - Technique (sharding :), mysql with app engine)
    - Case study (Snapchat on App Engine, what does XX company use)
    - Deeper dive on topic covered (eg. Google Vision API, etc.)
  - 30 minutes

CLASS NEWS

- Week 7 - CloudKit
  - Assignment 5 assigned
- Week 8 - CloudKit Server Extensions; Swift on the Server
  - Case Study assigned
  - Finish server component of Assignment 5
- Week 9 - Case Studies in Class (Assignment 5 Due)
- Week 10 - Cloud Roundup (Realm, Serverless, etc.)

MAKE-UP CLASS MATERIALS AFTER PRESENTATIONS

CLASS NEWS

- Final Projects
  - Opportunity for you to apply what you learned in class to a project you care about
  - Can use old iOS project or extend assignment
  - Talk me about your ideas early

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

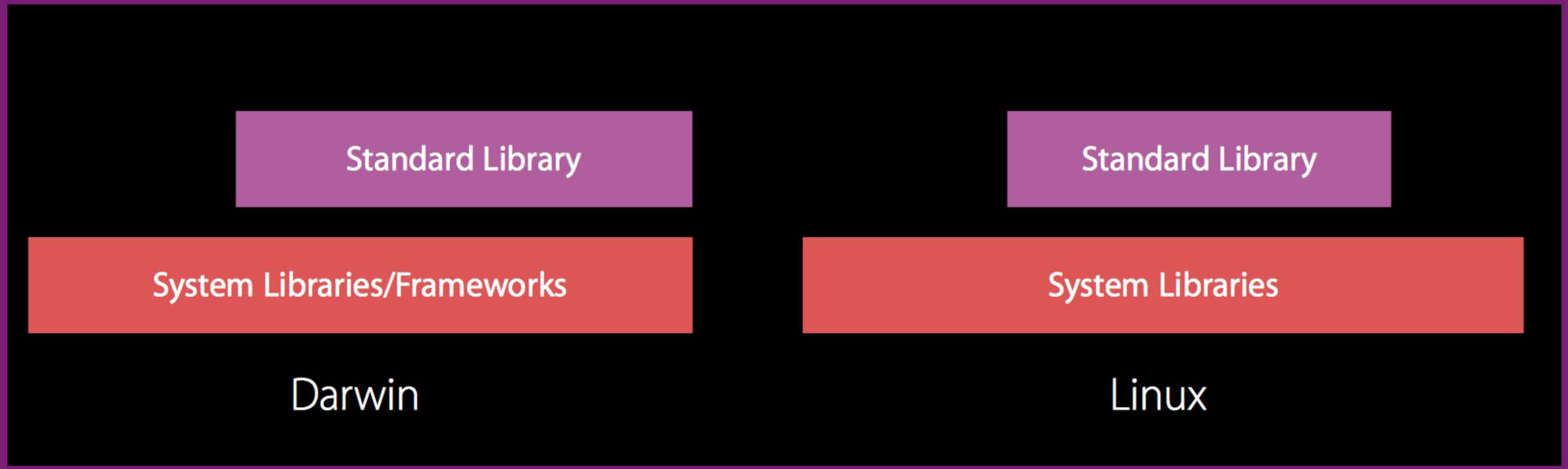
Swift.org is an entirely new site dedicated to open source Swift. This site hosts resources for the community of developers that want to help evolve Swift, contribute fixes, and most importantly, interact with each other. Swift.org hosts:

- 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

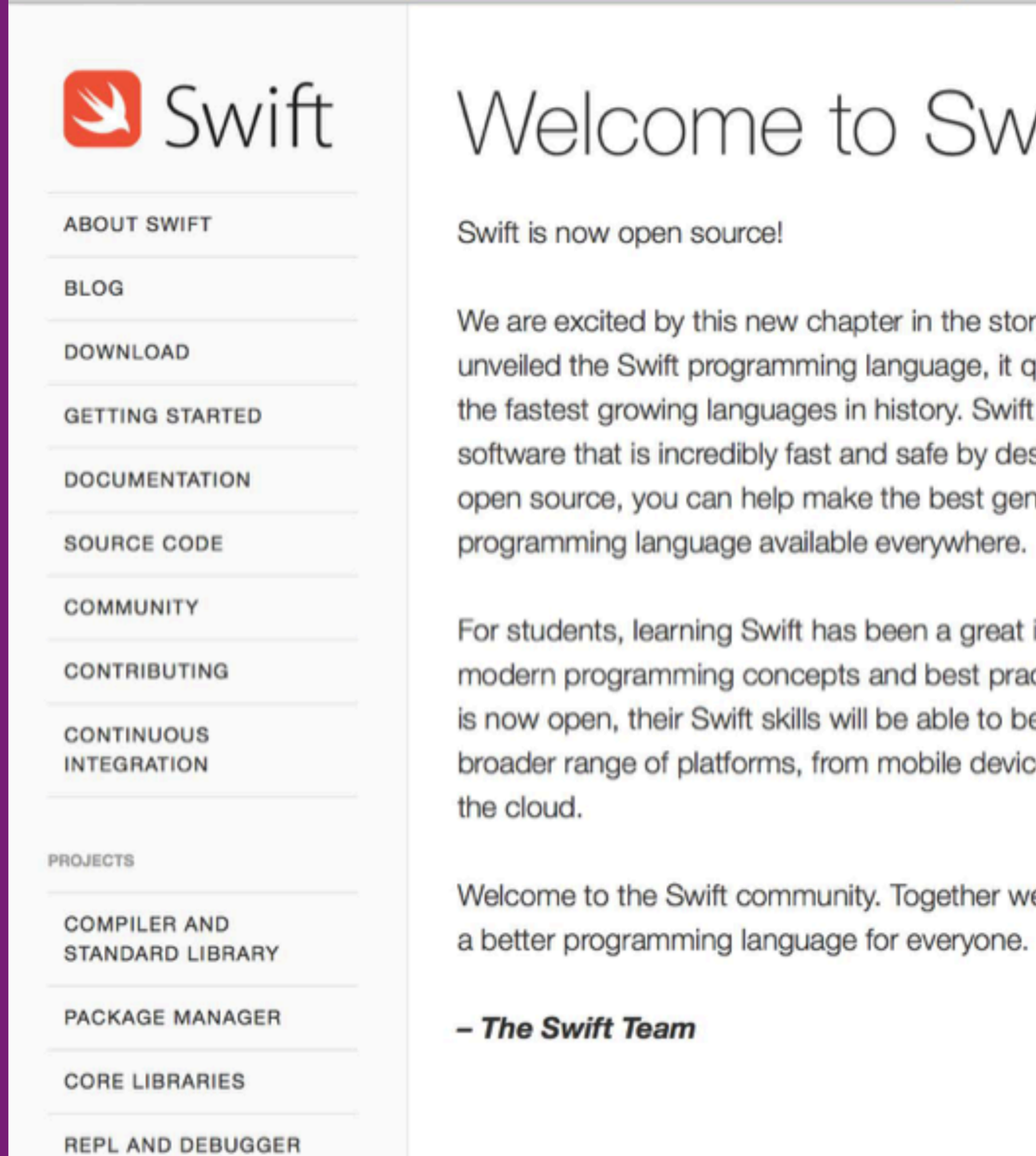
OPEN SOURCE SWIFT



- Something is missing...

OPEN SOURCE SWIFT

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



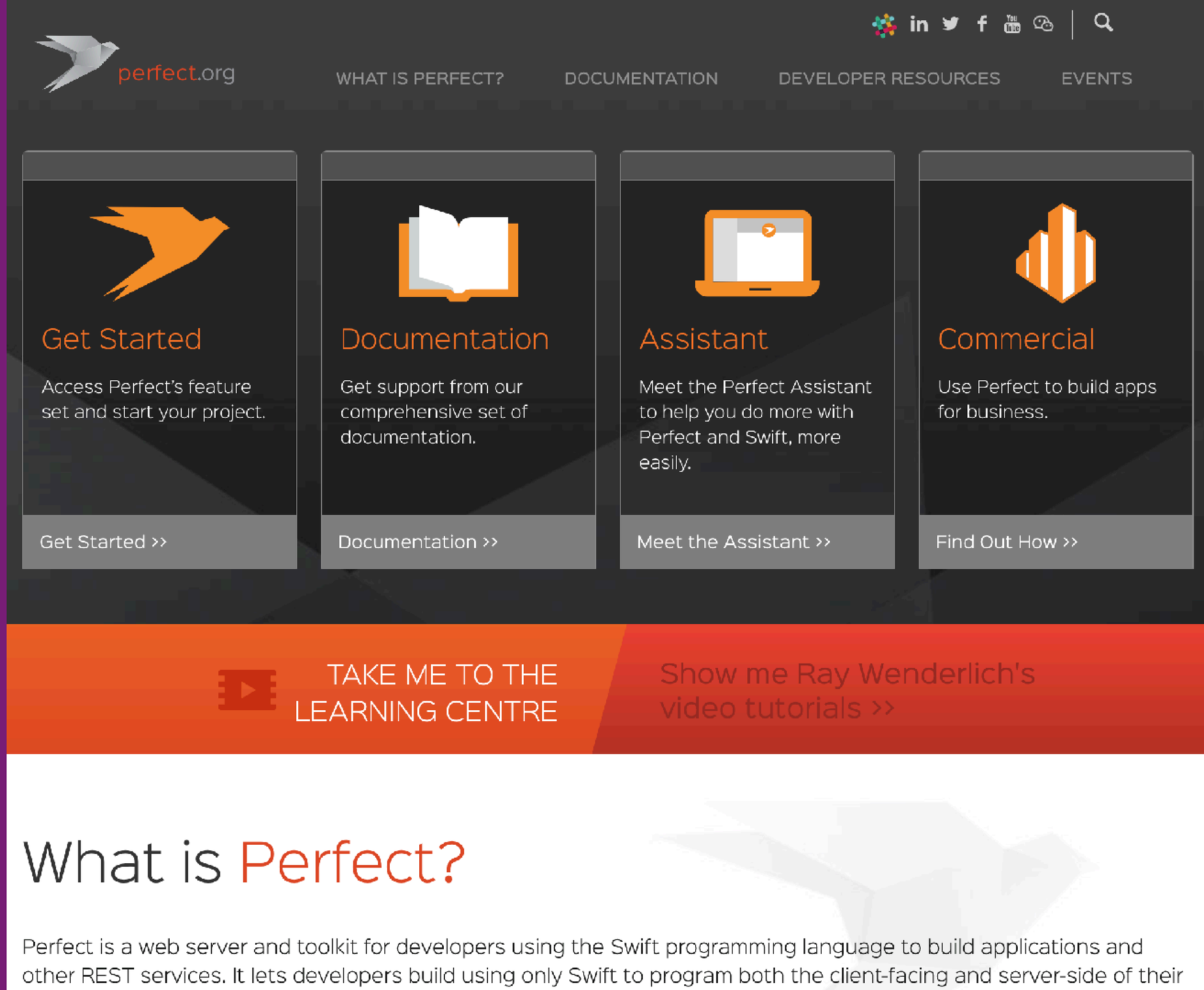
STATE OF SWIFT ON THE SERVER

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



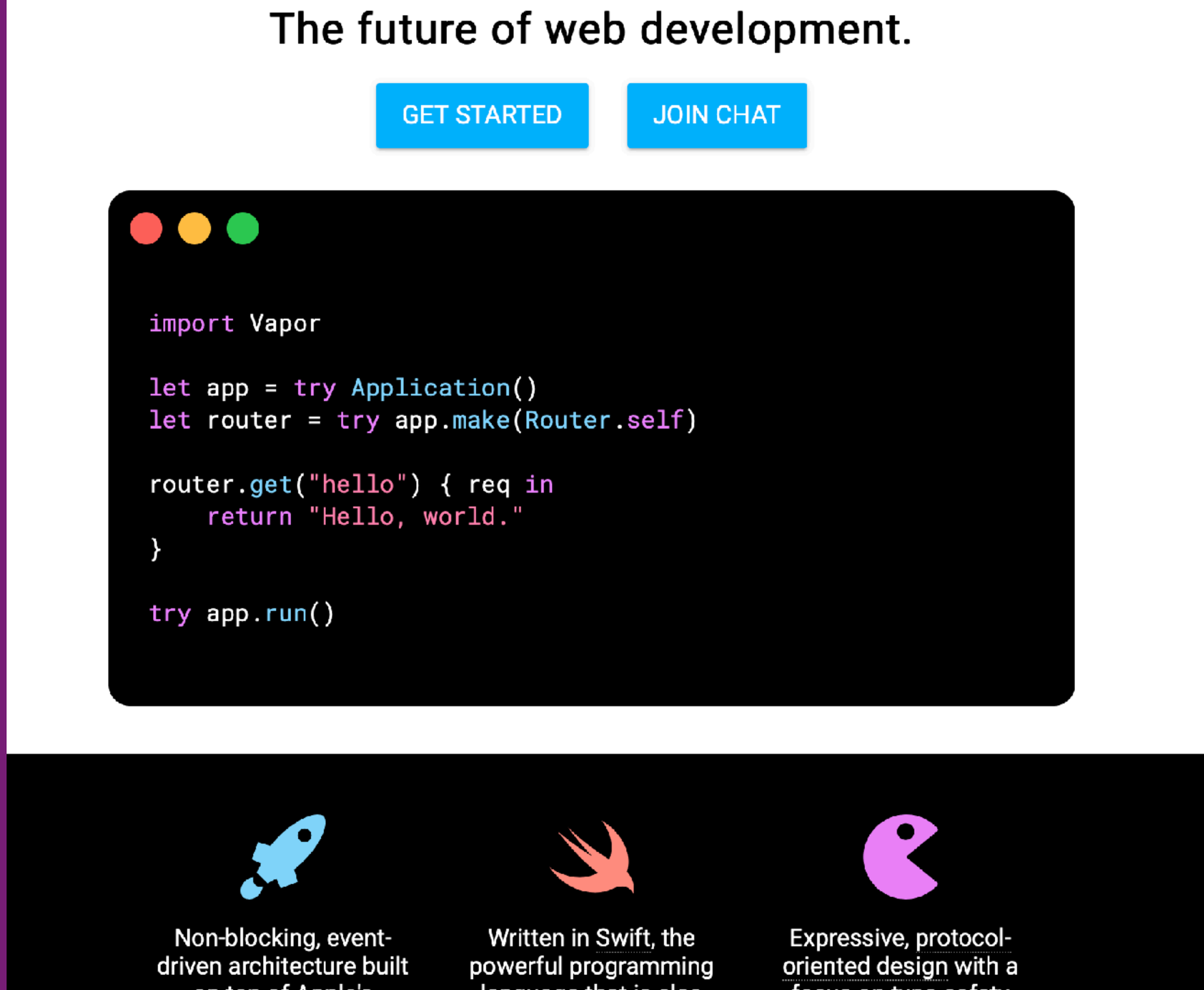
STATE OF SWIFT ON THE SERVER

- 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()
    }
}
```

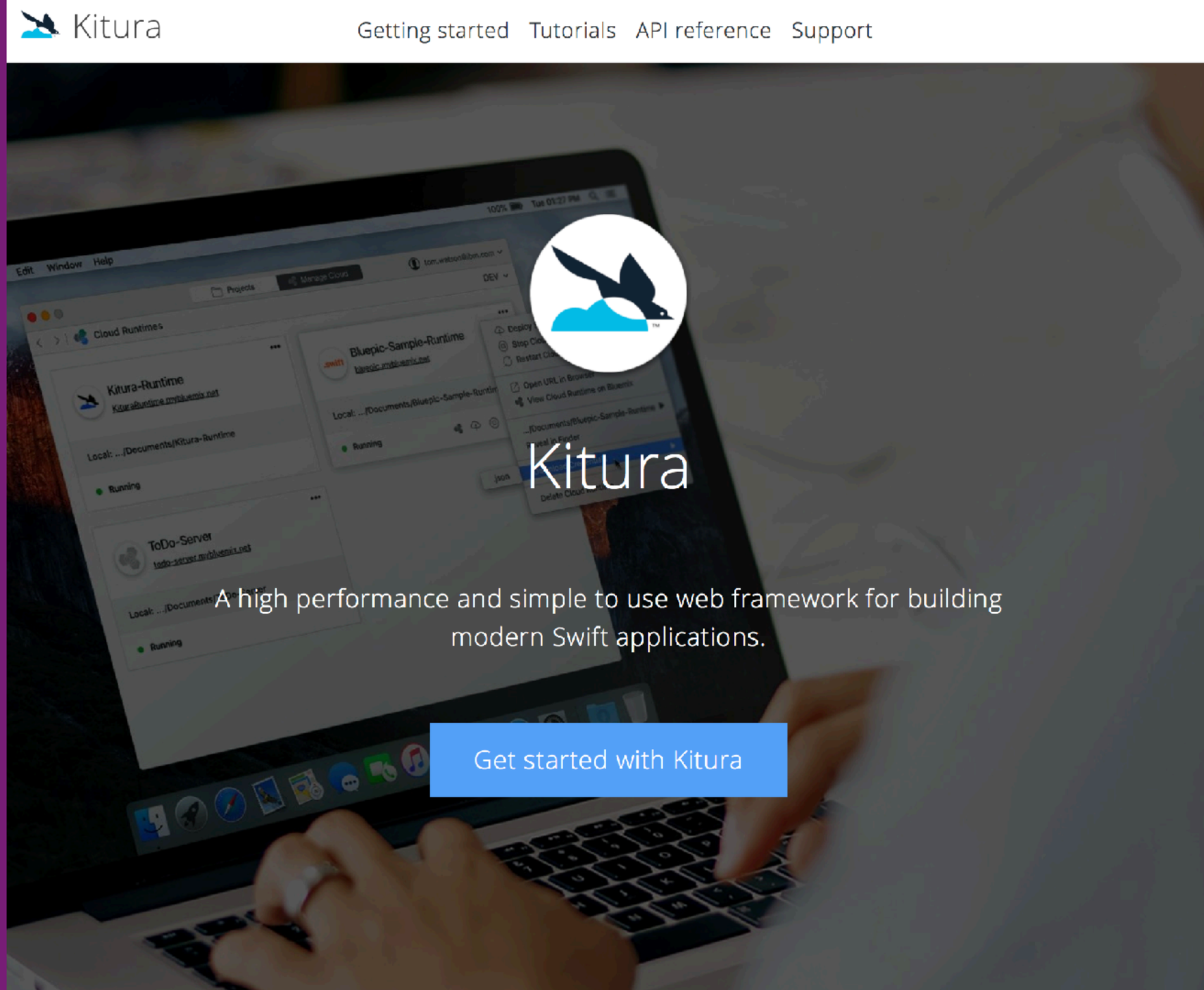
STATE OF SWIFT ON THE SERVER

- Vapor
  - 17,479 ⭐
  - Laravel (php) inspired



STATE OF SWIFT ON THE SERVER

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



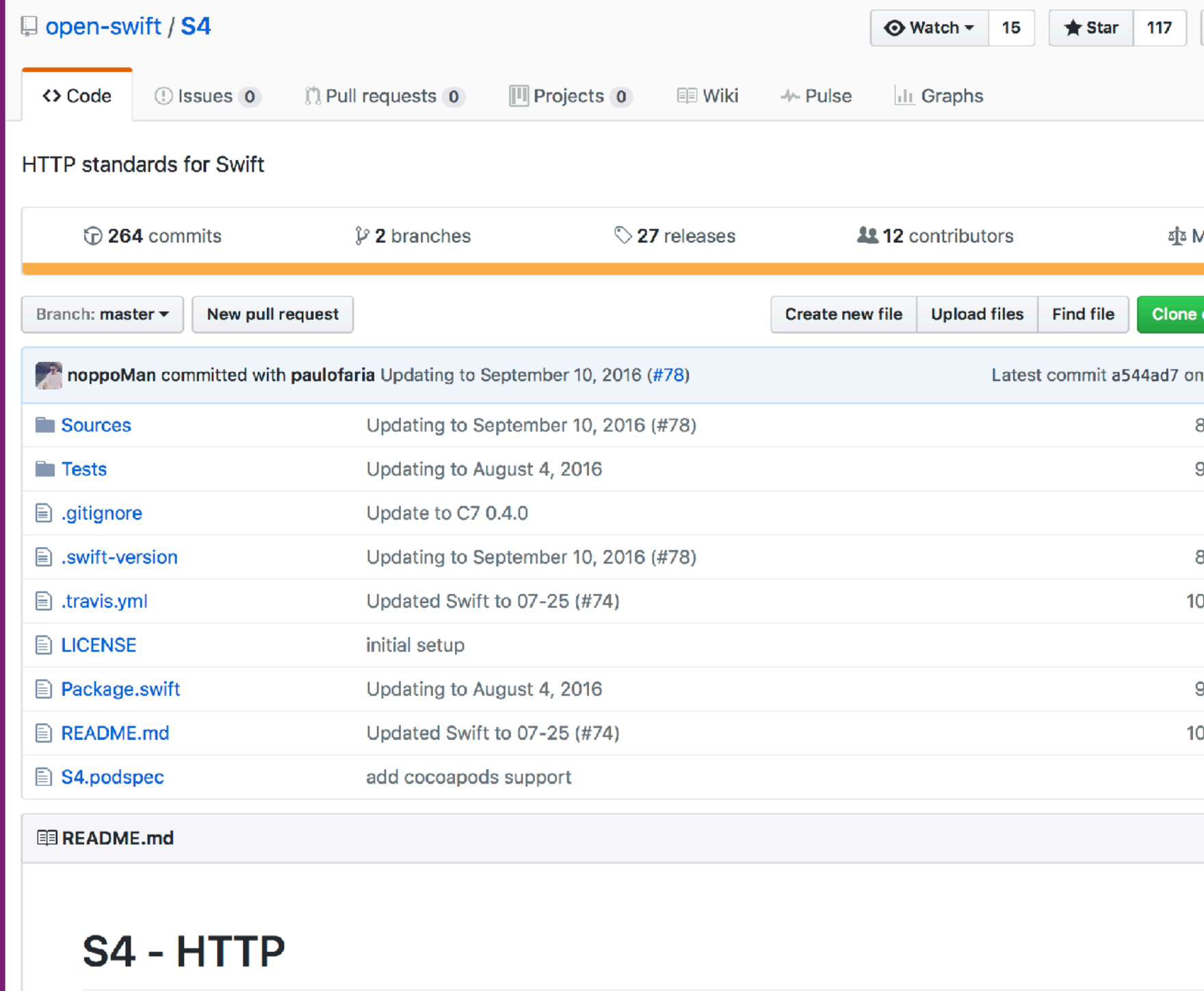
STATE OF SWIFT ON THE SERVER

- 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



STATE OF SWIFT ON THE SERVER

- SwiftNIO
- Event-driven network application framework for high performance protocol servers & clients, non-blocking

apple / swift-nio

Watch 162 | Star 5.4k | Fork

Code | Issues 73 | Pull requests 23 | Projects 0 | Security | Insights

Event-driven network application framework for high performance protocol servers & clients, non-blocking.
https://apple.github.io/swift-nio/

swift | asynchronous-io | networking | event-driven | high-performance | non-blocking-io | non-blocking | swiftnio | swift5 | swift-server

1,430 commits | 16 branches | 0 packages | 62 releases | 89 contributors | Apache-2.0

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

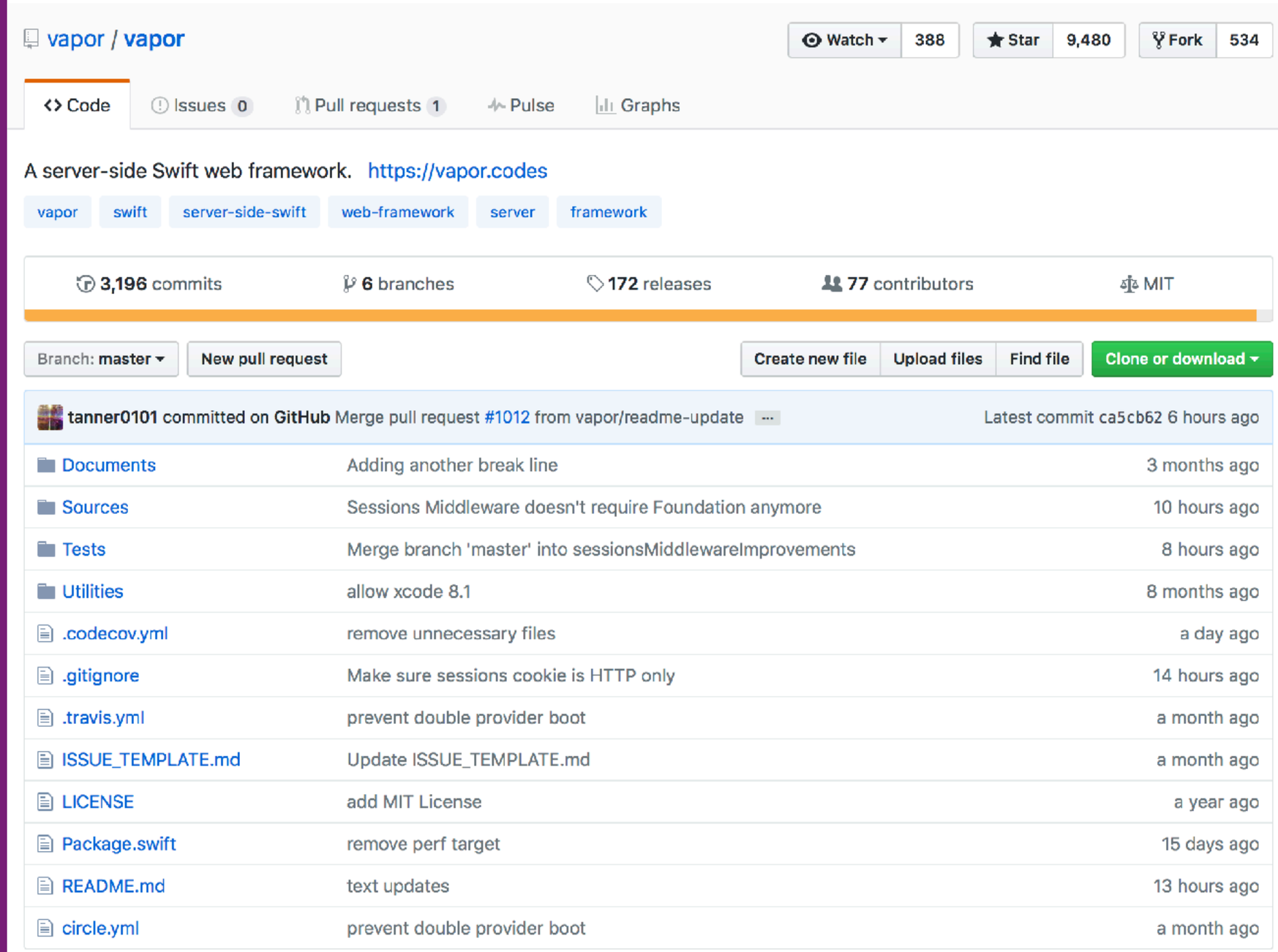
przala and weissi Added parentheses in TimeAmount constructors to generate optimal code... | Latest commit 86f41d8 yester

.github	Move GitHub files to .github folder (#71)	2 years
IntegrationTests	Fix running run-nio-alloc-counter-tests.sh for a single test (#1245)	5 days
Sources	Added parentheses in TimeAmount constructors to gen	ester
Tests	HTTP1TestServer: close accepted channel on `stop` (#	days
dev	make-single-file-spm: support lines ending in `n` (#125	ester
docker	EventLoopFuture: save one allocation per future (#1224	days
docs	Add an optimization tips document. (#1024)	nths
scripts	perf tests: switch printed metric to mean (#1244)	days
.gitignore	assert EventLoopGroup::syncShutdownGracefully is not	nths
.mailmap	update contributors (#1225)	days
CODE_OF_CONDUCT.md	update conduct email group (#1120)	nths
CONTRIBUTING.md	Make clear generate_linux_tests must be run (#113)	2 years

VAPOR

VAPOR

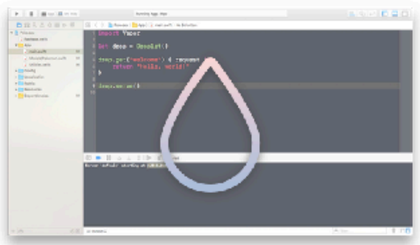
- Web framework and server for Swift
- Works on macOS and Ubuntu



VAPOR

- Great documentation and resources

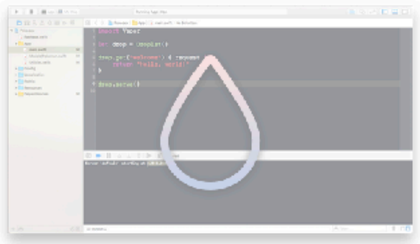
vapor.university



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

Install on macOS

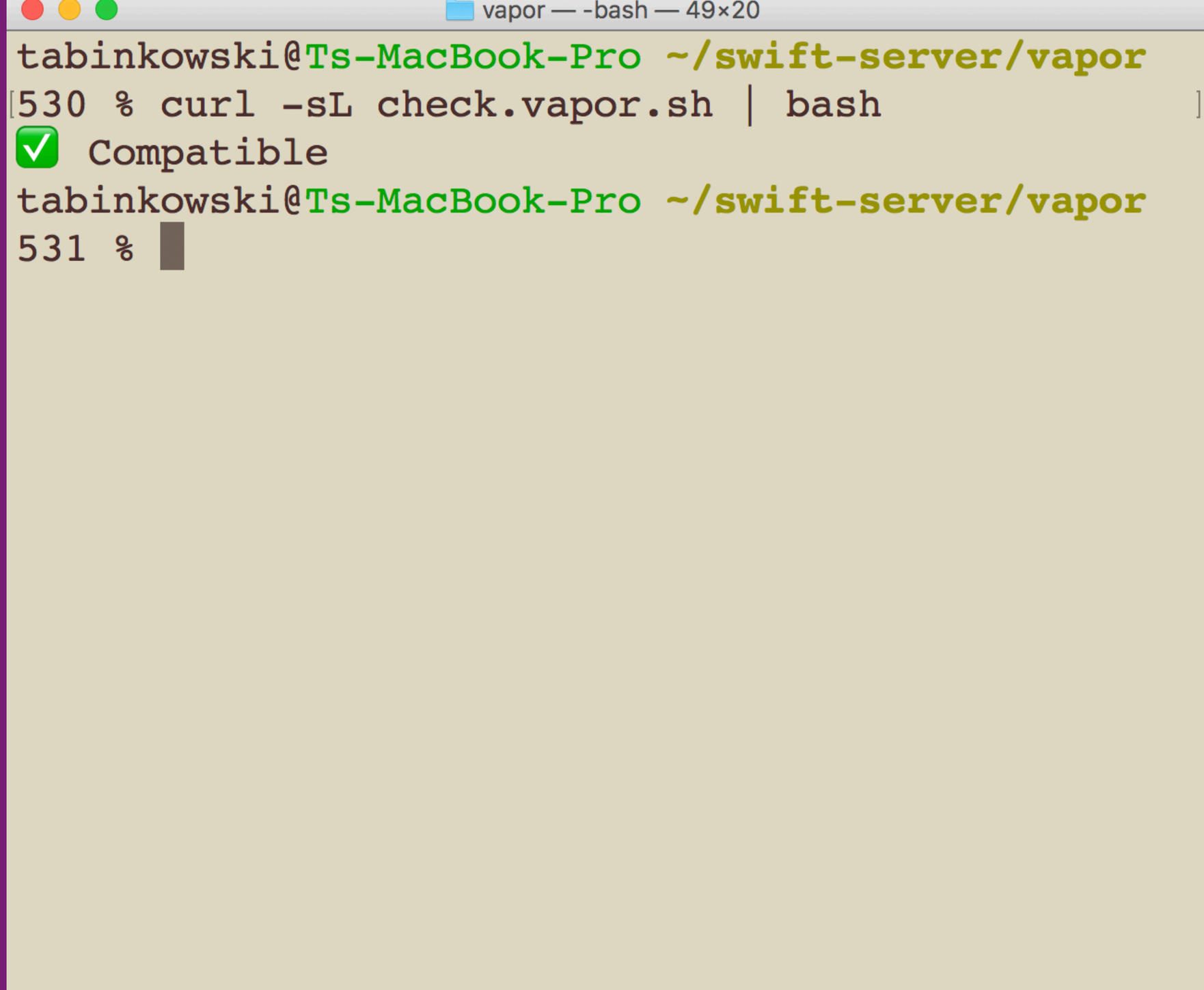
To use Vapor on macOS, you just need to have Xcode 9.3 or greater installed.

Install Xcode

Install Xcode 9.3 or greater from the Mac App Store.

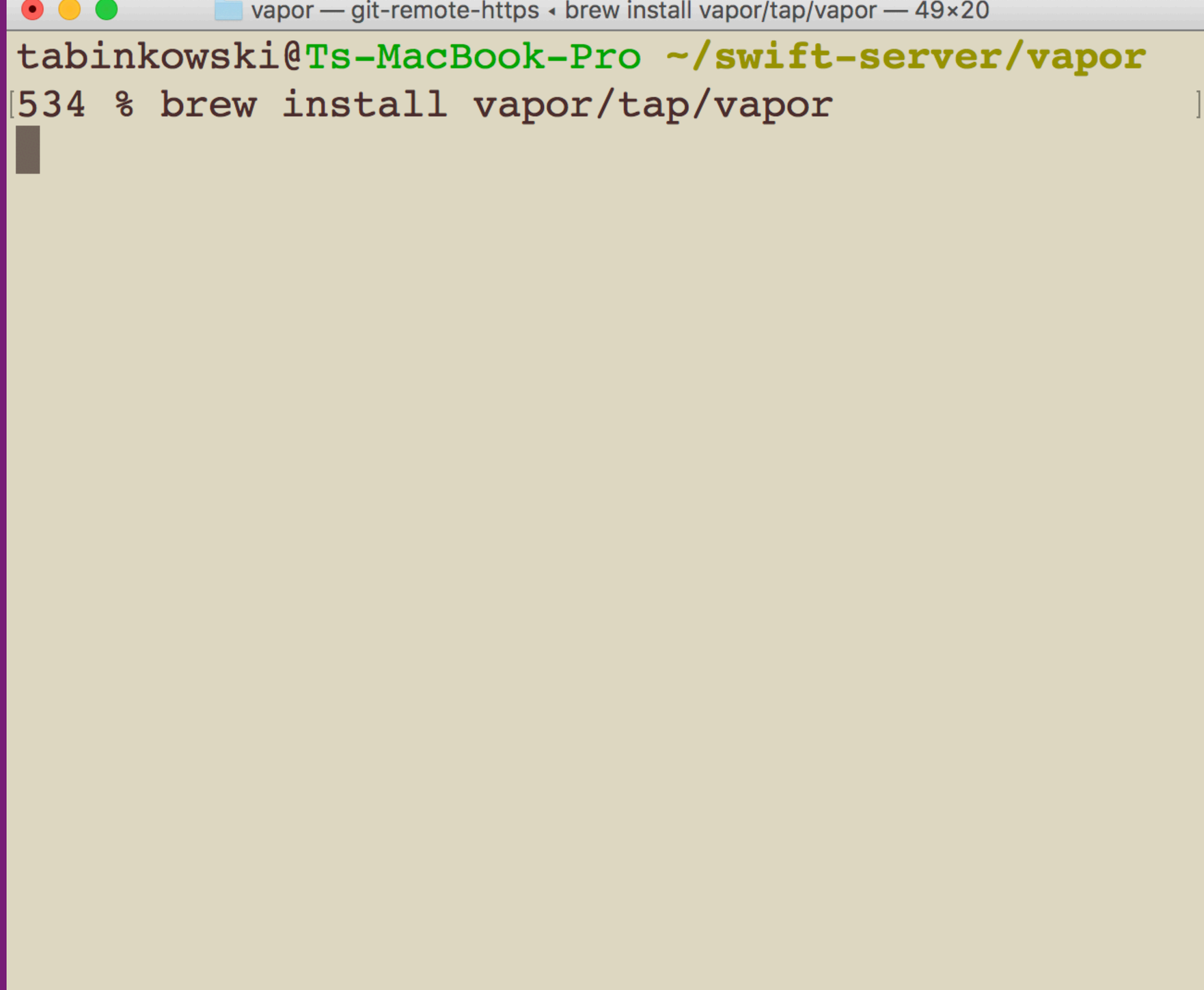
VAPOR

- Install Xcode
- Check version for vapor



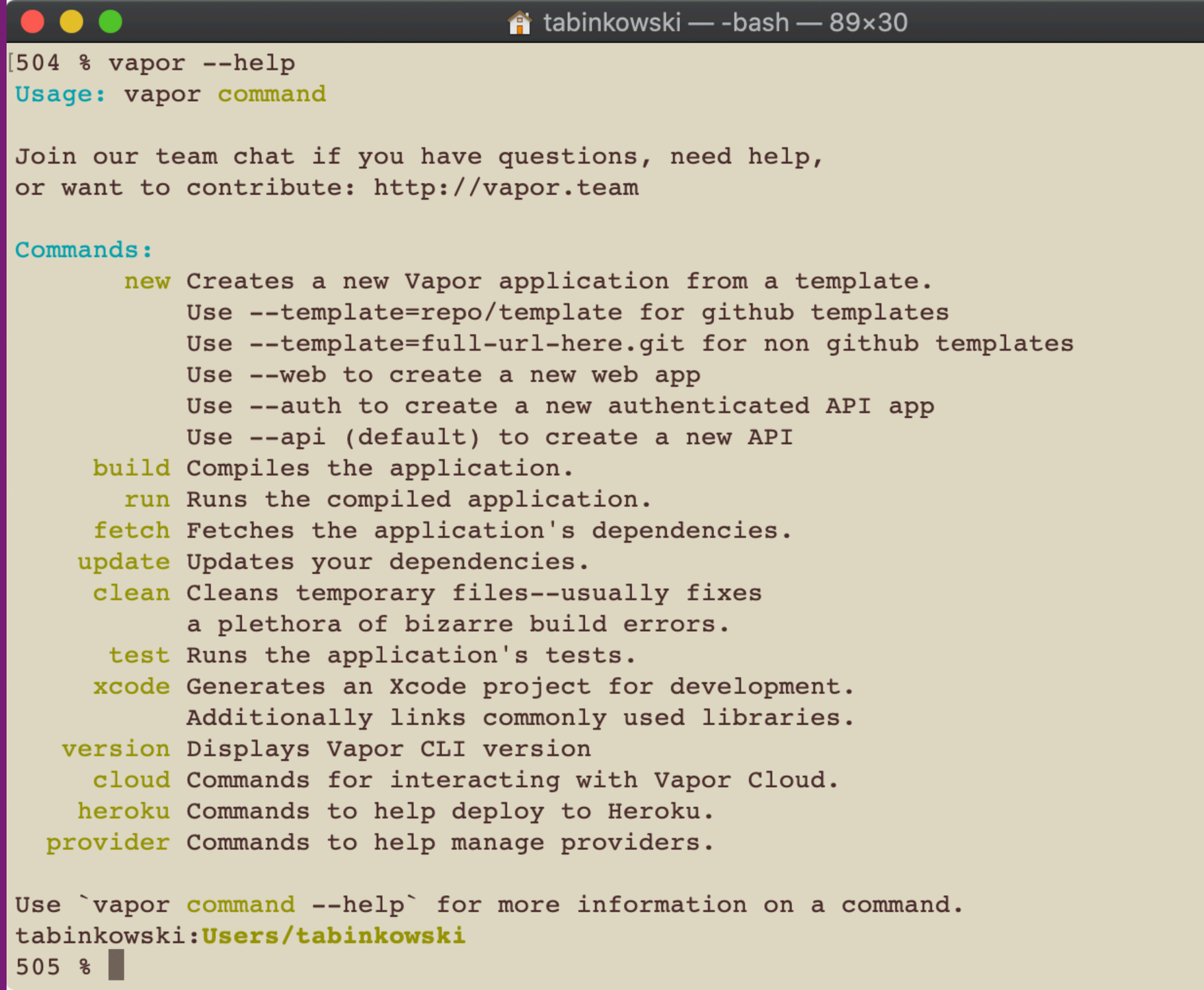
VAPOR

- 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 🍺

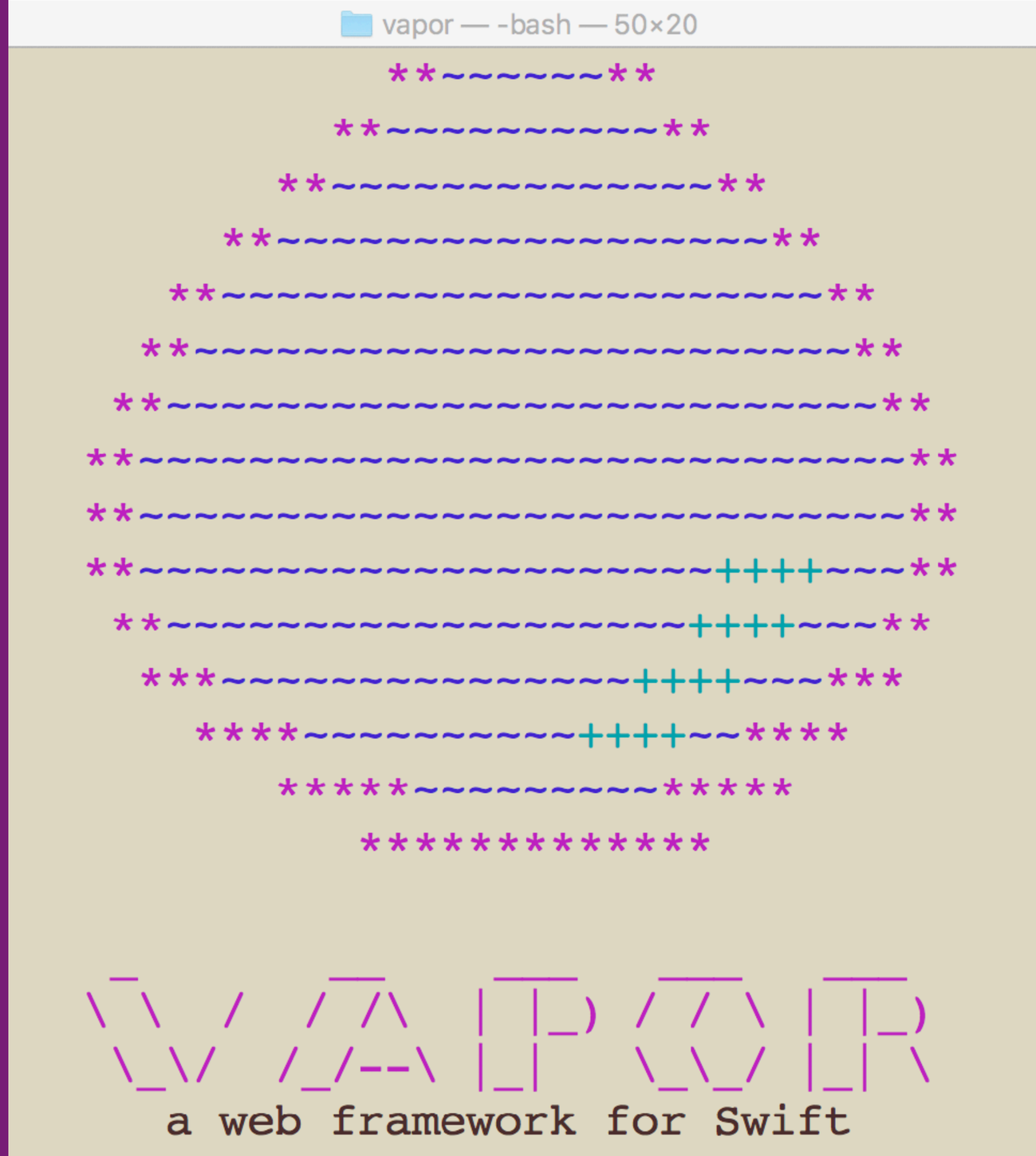


VAPOR

```
# 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>]
```

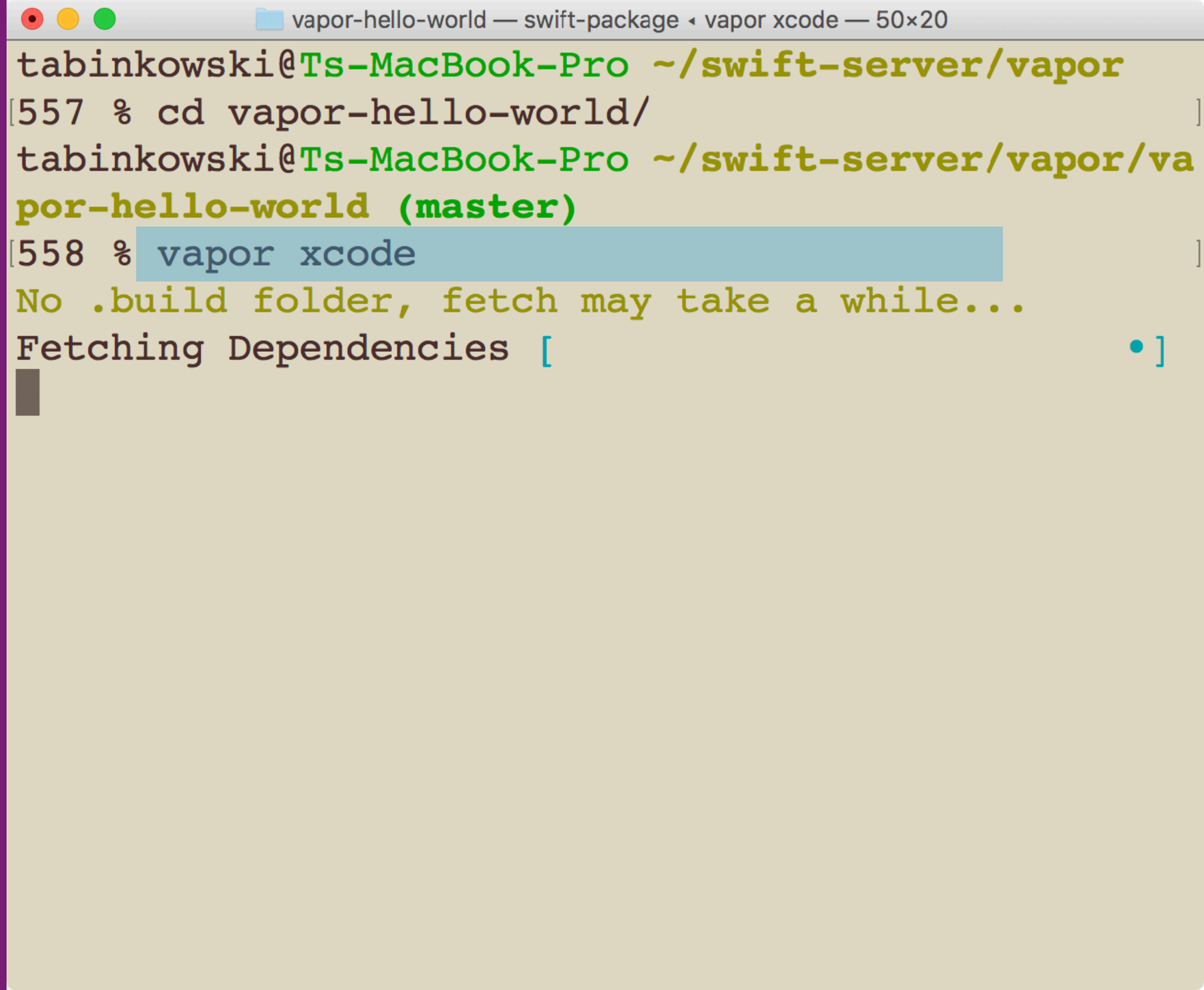
VAPOR

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



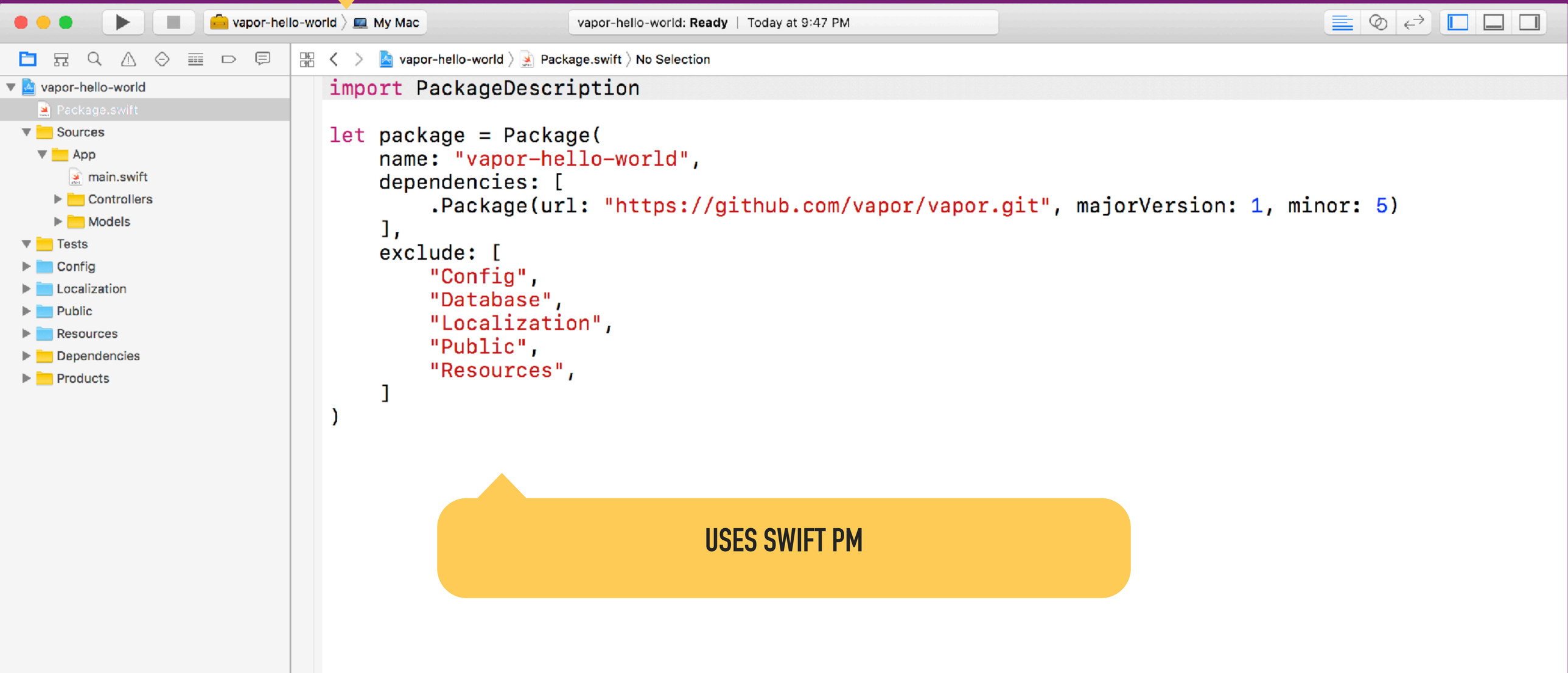
VAPOR

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



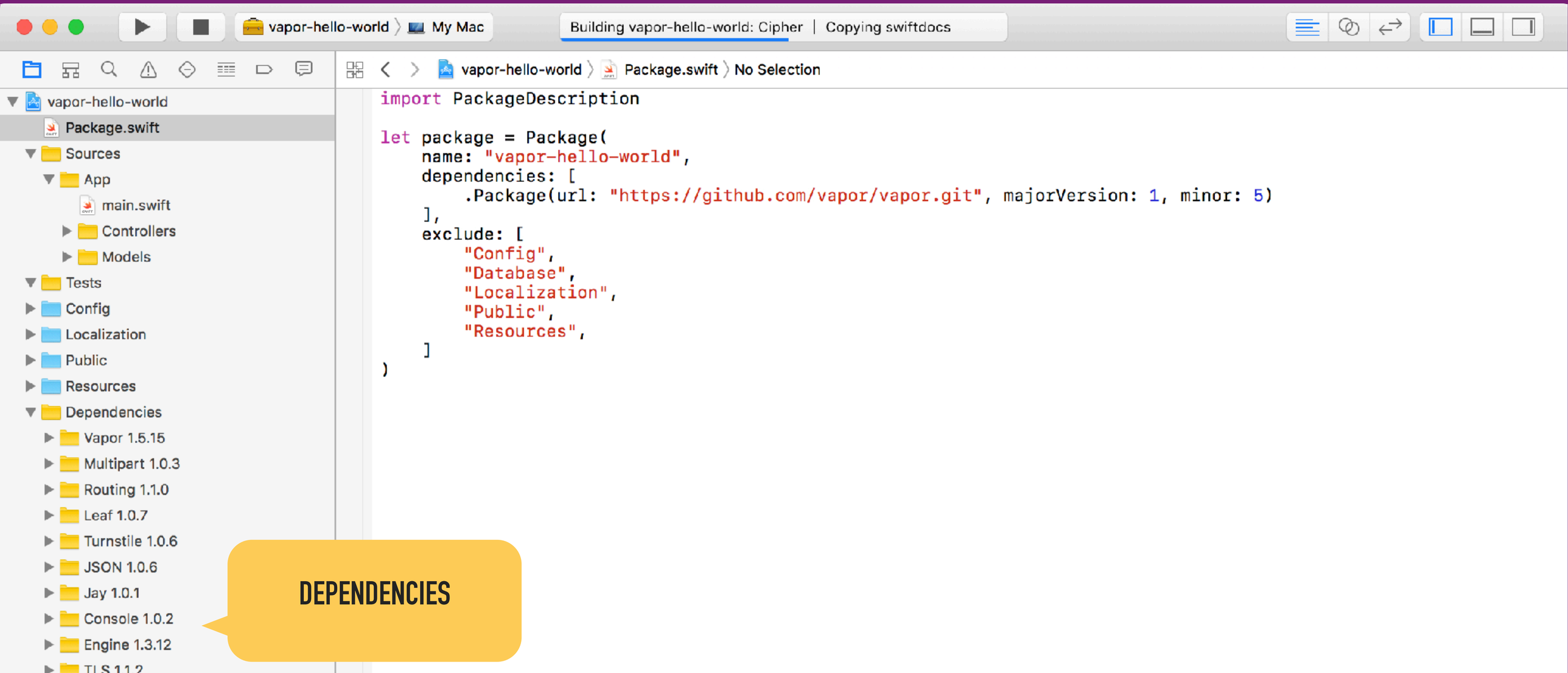
VAPOR

BUILDS A FRAMEWORK AND APP



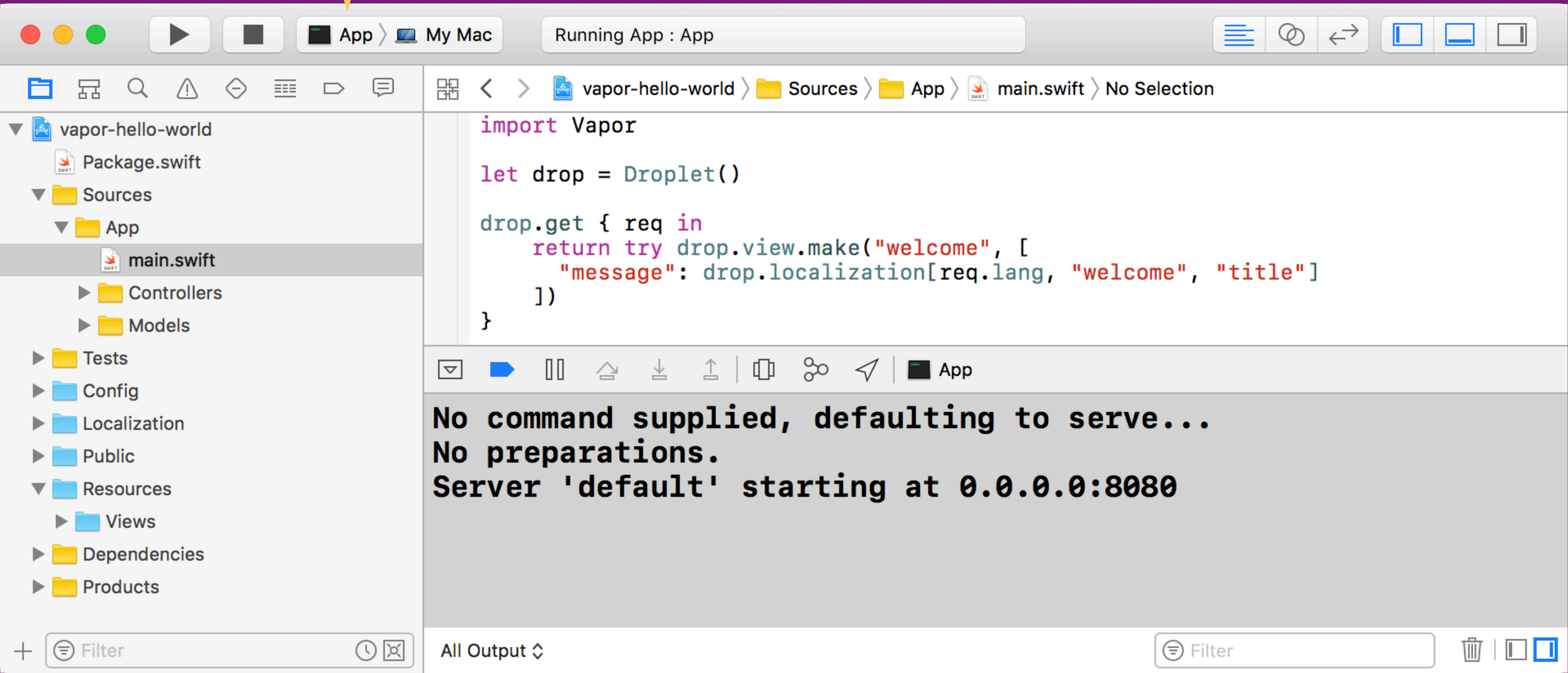
USES SWIFT PM

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

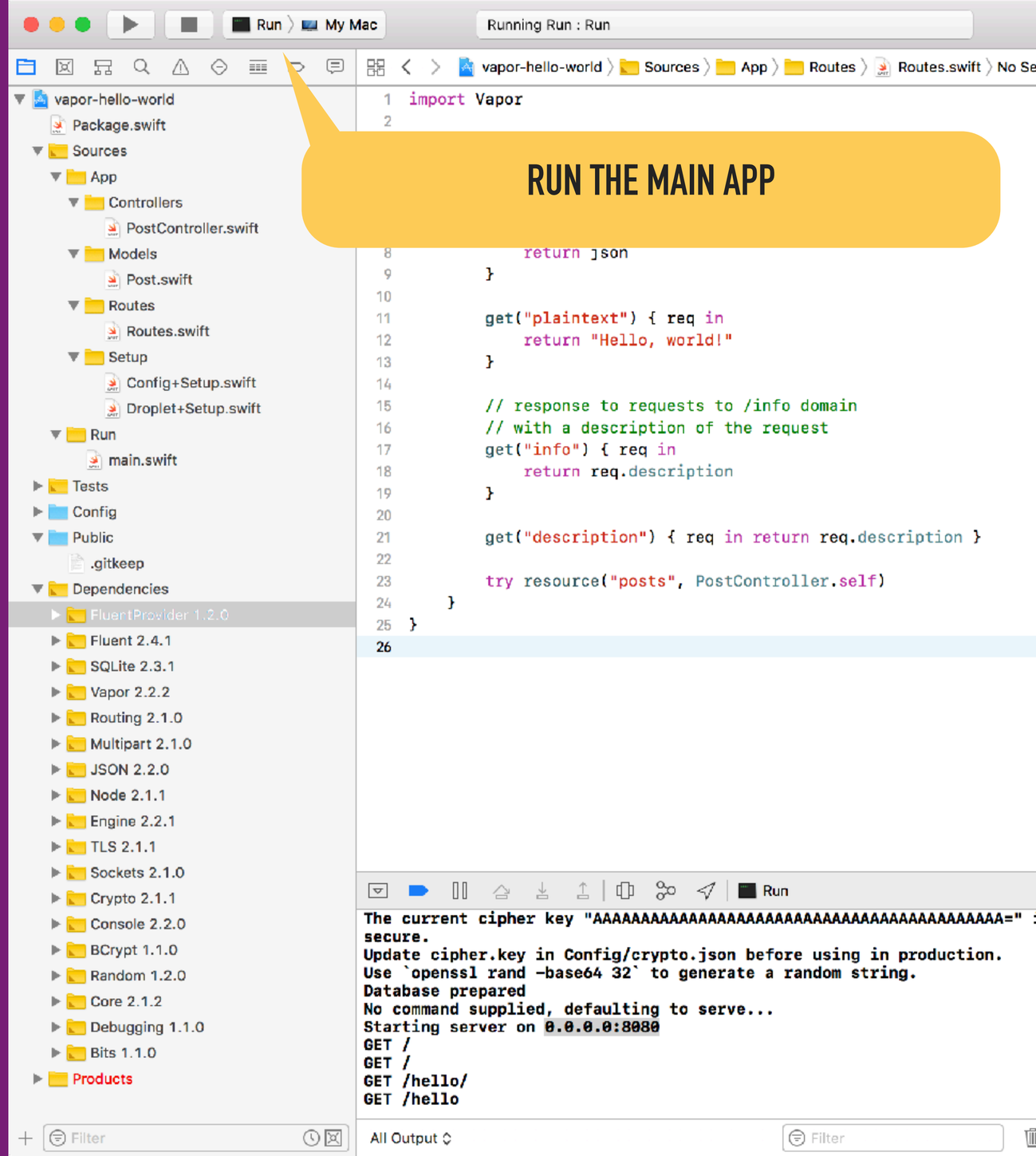
VAPOR

RUN APP



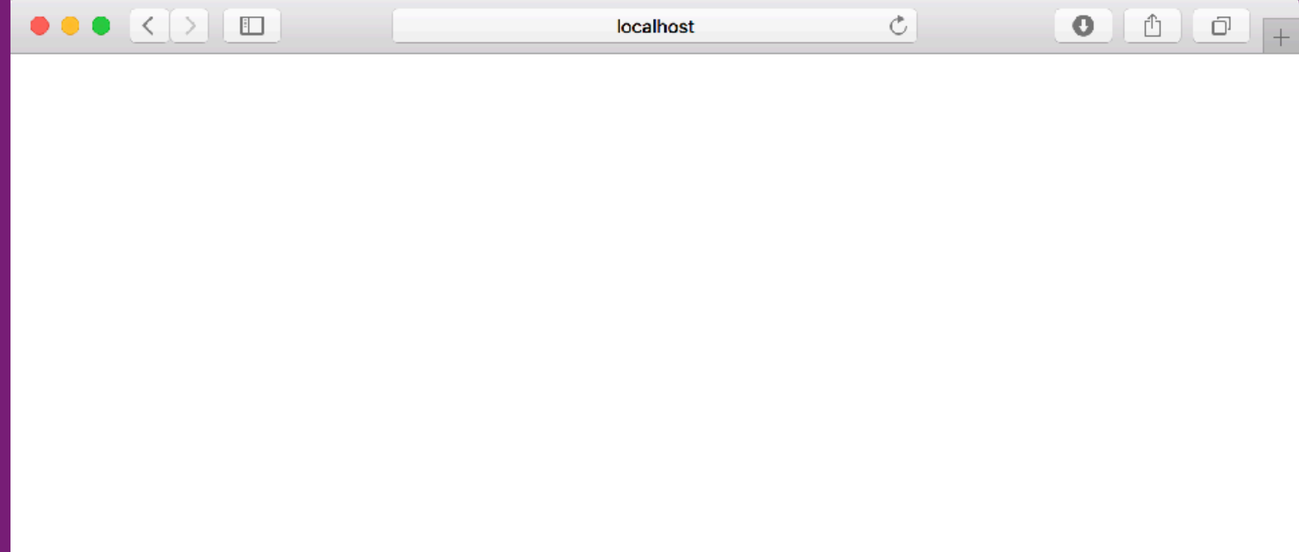
VAPOR

- Builds all the package dependencies and links them



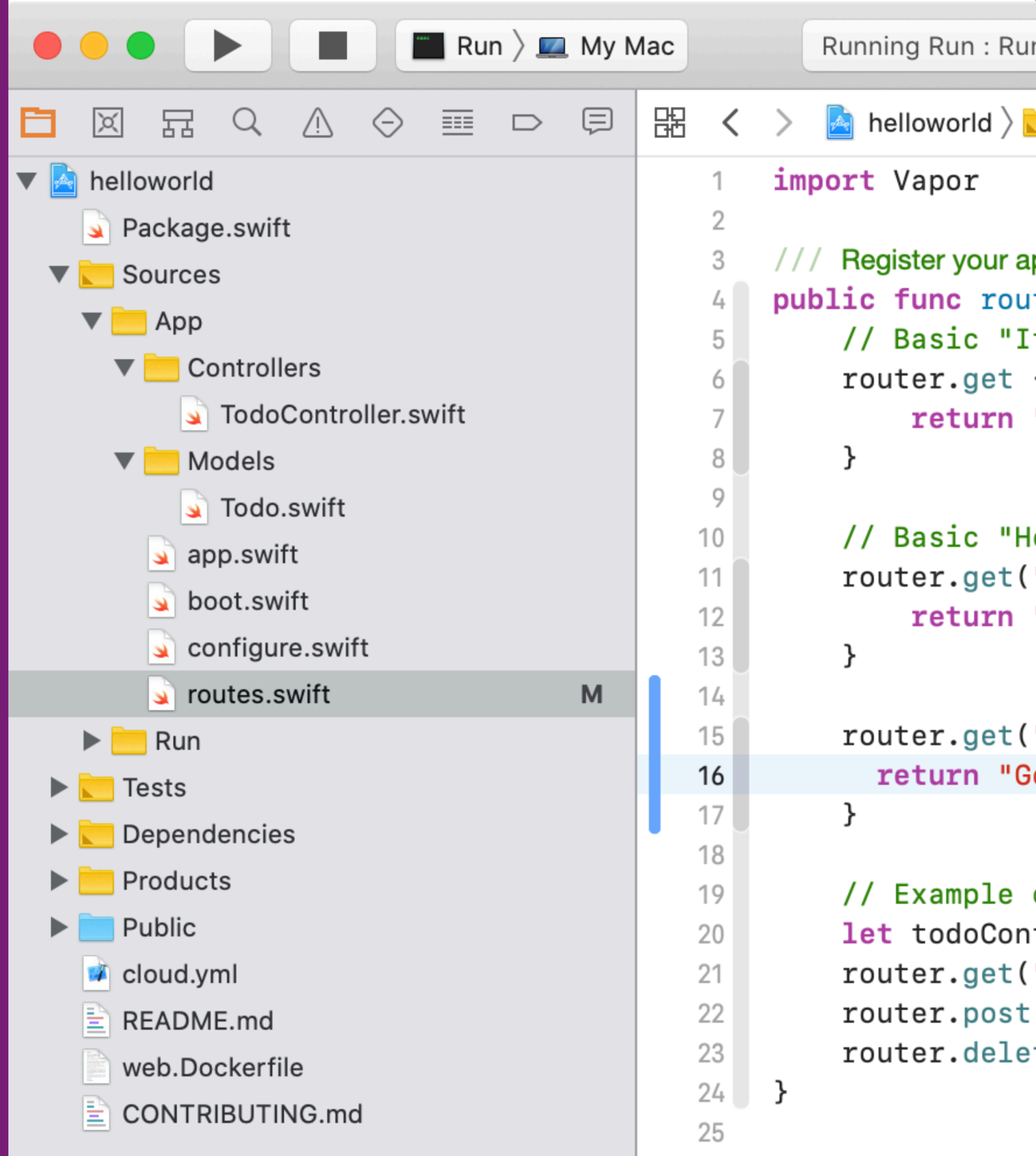
VAPOR

- Server runs on http://localhost:8080

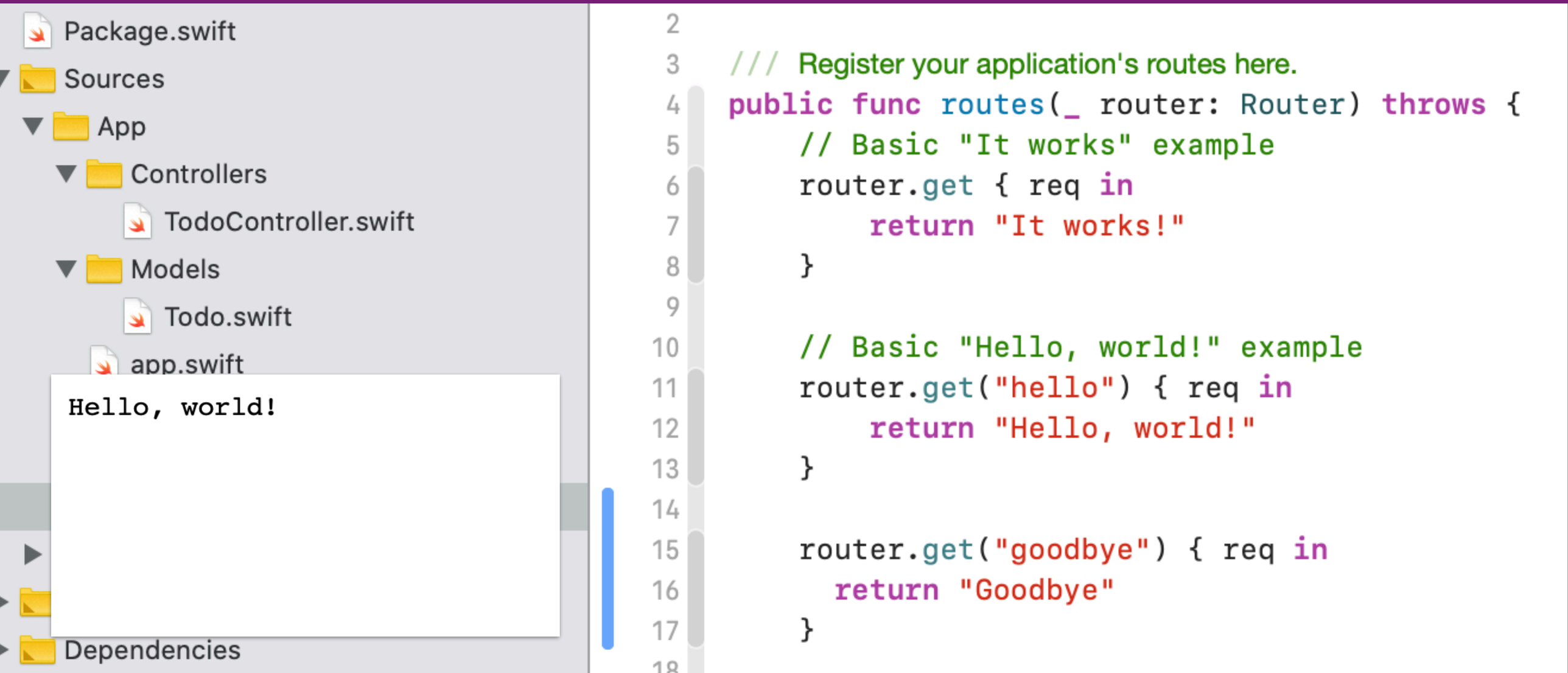


VAPOR

- File structure is designed to make testing easier 🤷‍♀️



VAPOR

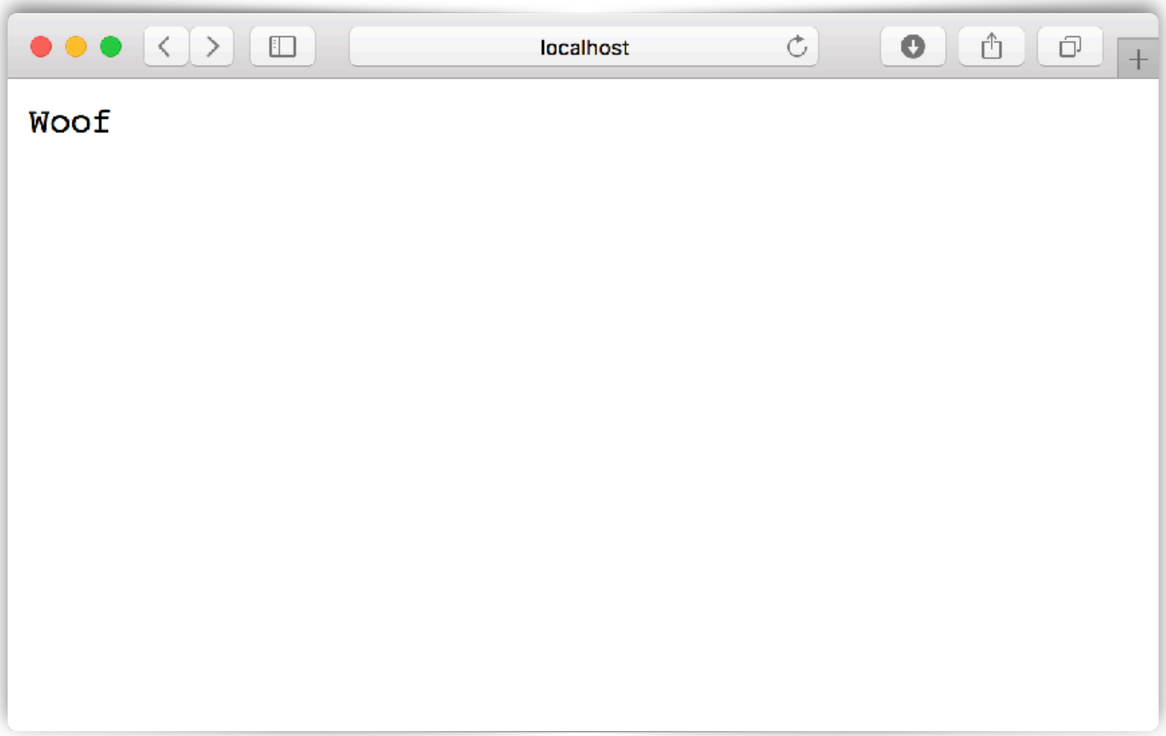

Package.swift
Sources
App
Controllers
TodoController.swift
Models
Todo.swift
app.swift
Hello, world!
Dependencies
2
3 /// Register your application's routes here.
4 public func routes(_ router: Router) throws {
5 // Basic "It works" example
6 router.get { req in
7 return "It works!"
8 }
9
10 // Basic "Hello, world!" example
11 router.get("hello") { req in
12 return "Hello, world!"
13 }
14
15 router.get("goodbye") { req in
16 return "Goodbye"
17 }

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()
```

LOCALHOST:8080/DOG



VAPOR

LOCALHOST:8080/GOODBYE

```
router.get("goodbye") { req in
   return "Goodbye"
}
```

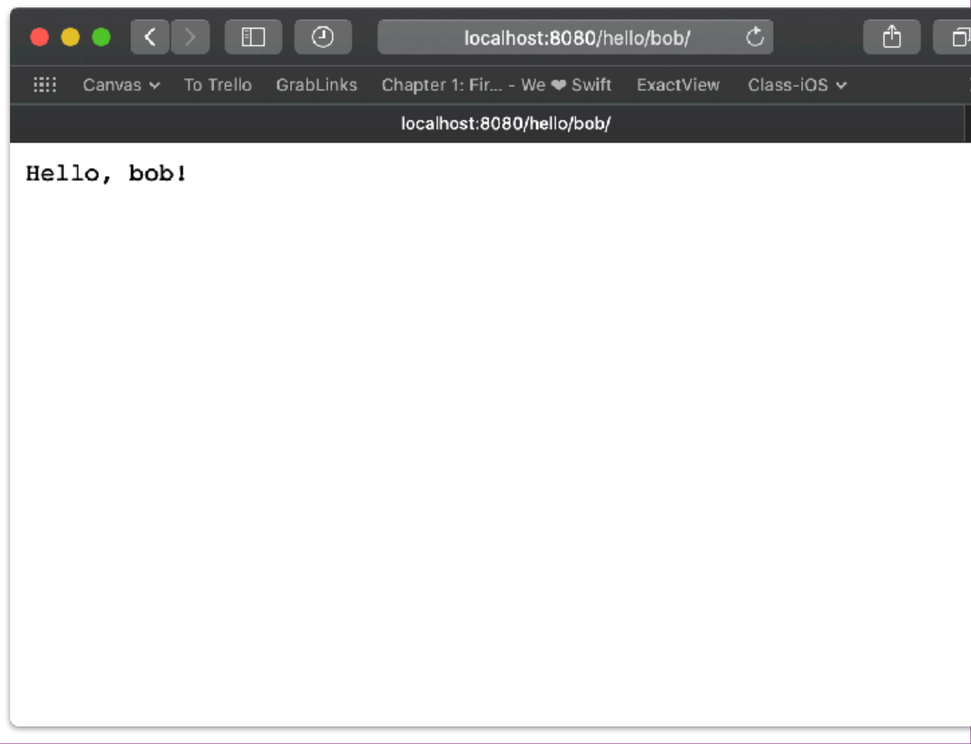
LOCALHOST:8080/USER/LOLA

```
router.get("user","lola") { req in
return "Goodbye"
```

VAPOR

LOCALHOST:8080/HELLO/BOB/

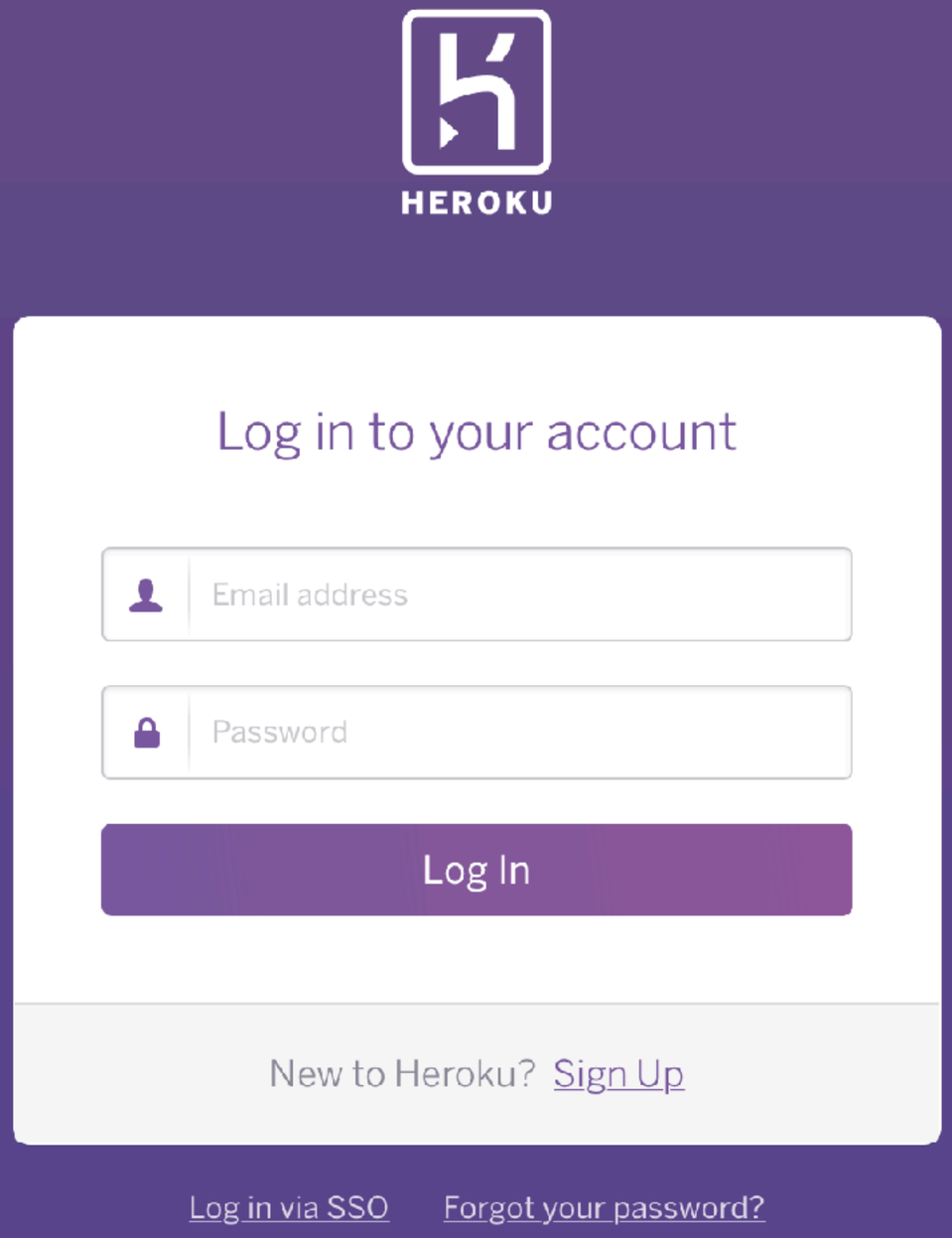
```
router.get("hello", String.parameter) { req -> String in
  //2
  let name = try req.parameters.next(String.self)
  // 3
  return "Hello, \(name)!"
}
```



DEPLOY VAPOR APP TO HEROKU

DEPLOY VAPOR APP TO HEROKU

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



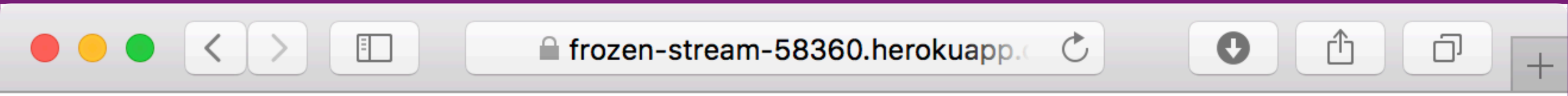
DEPLOY VAPOR APP TO HEROKU

```
# Install tools
brew install heroku


# Set credentials
heroku login


# Upload
vapor heroku init
```

DEPLOY VAPOR APP TO HEROKU

```
Building on Heroku ... ~5-10 minutes [D
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
```

DEPLOY VAPOR APP TO HEROKU



Bark! Bark!

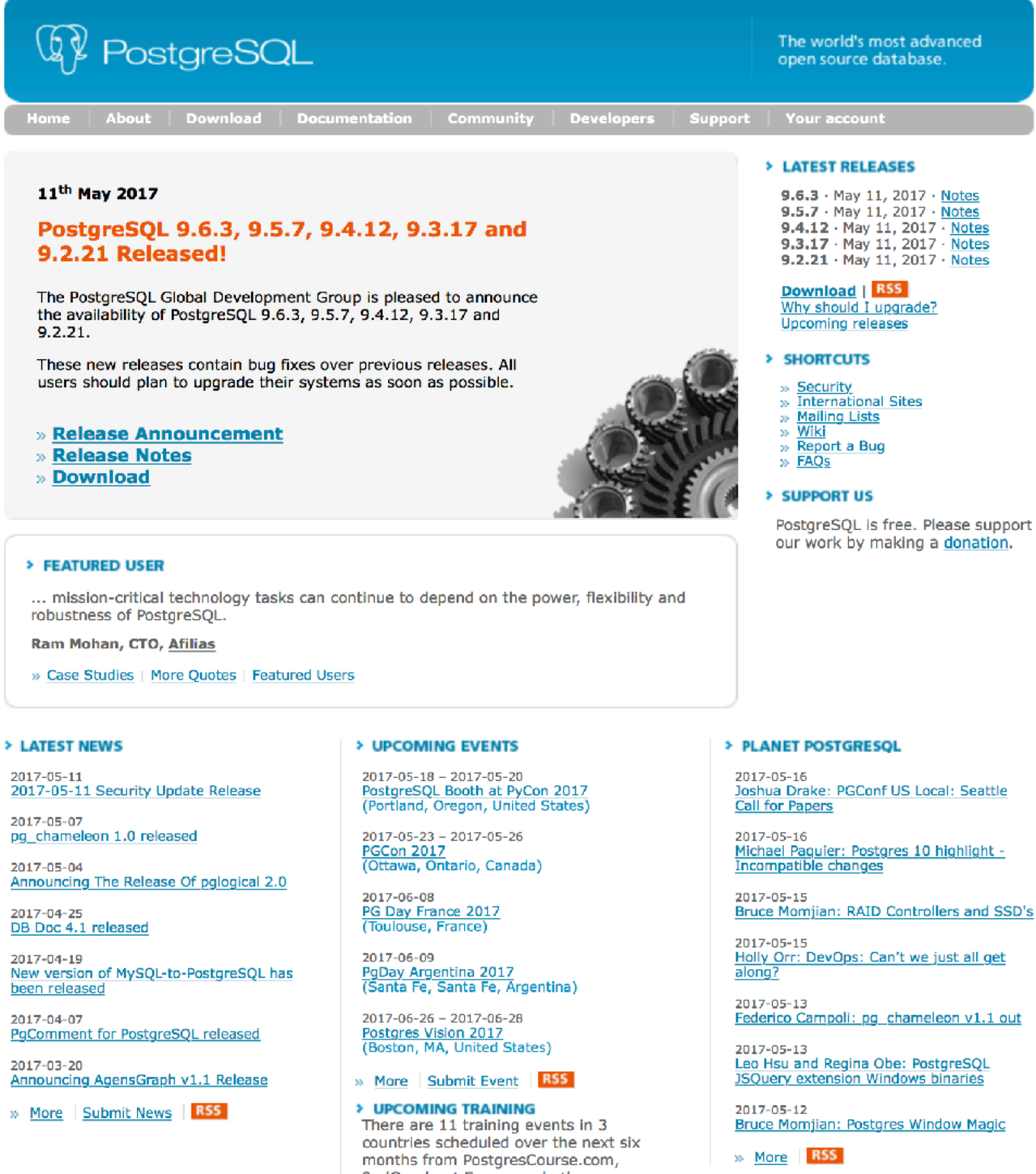
https://frozen-stream-58360.herokuapp.com/dog/speak/

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

VAPOR WITH POSTGRESQL

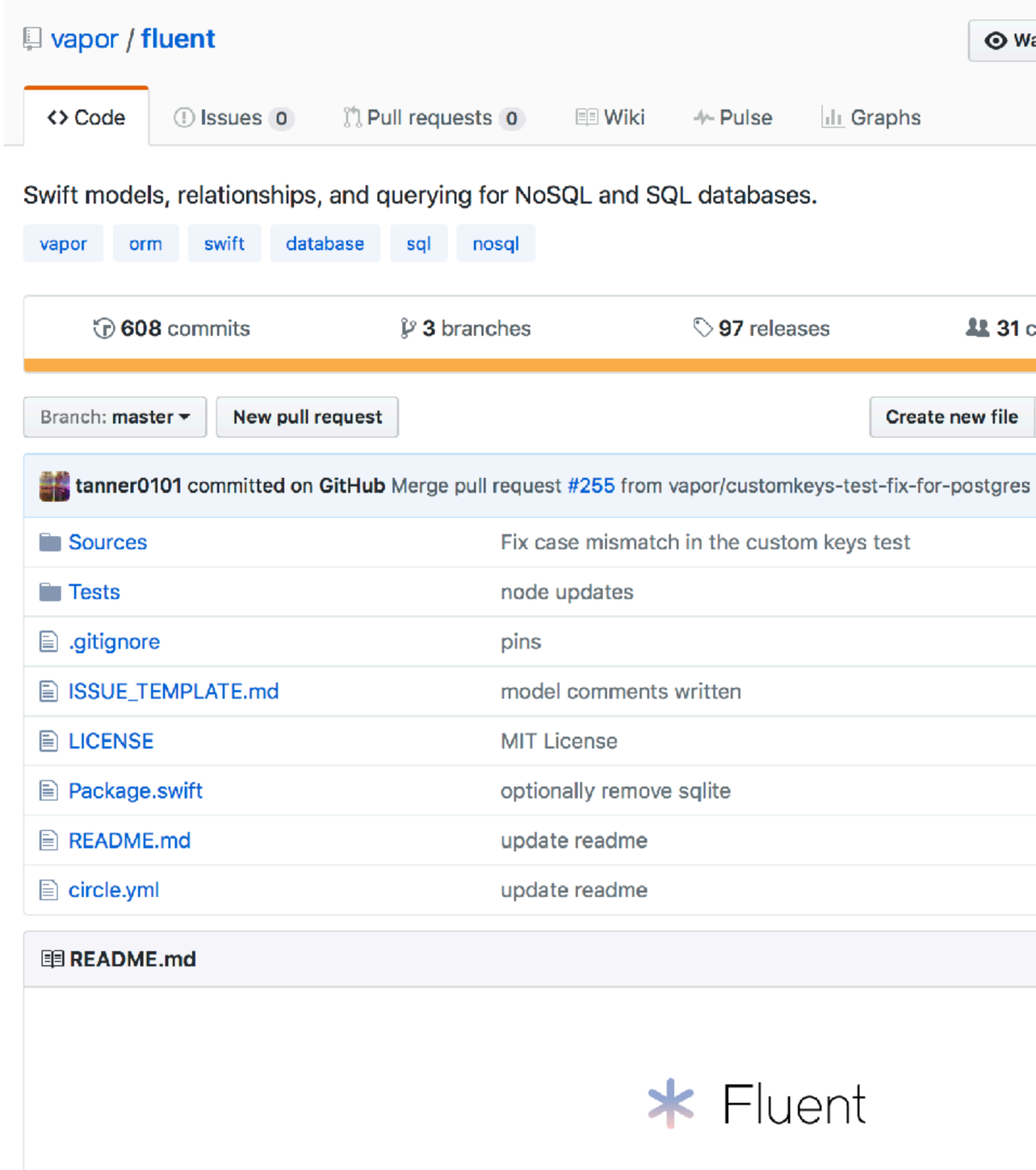
VAPOR WITH POSTGRESQL

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



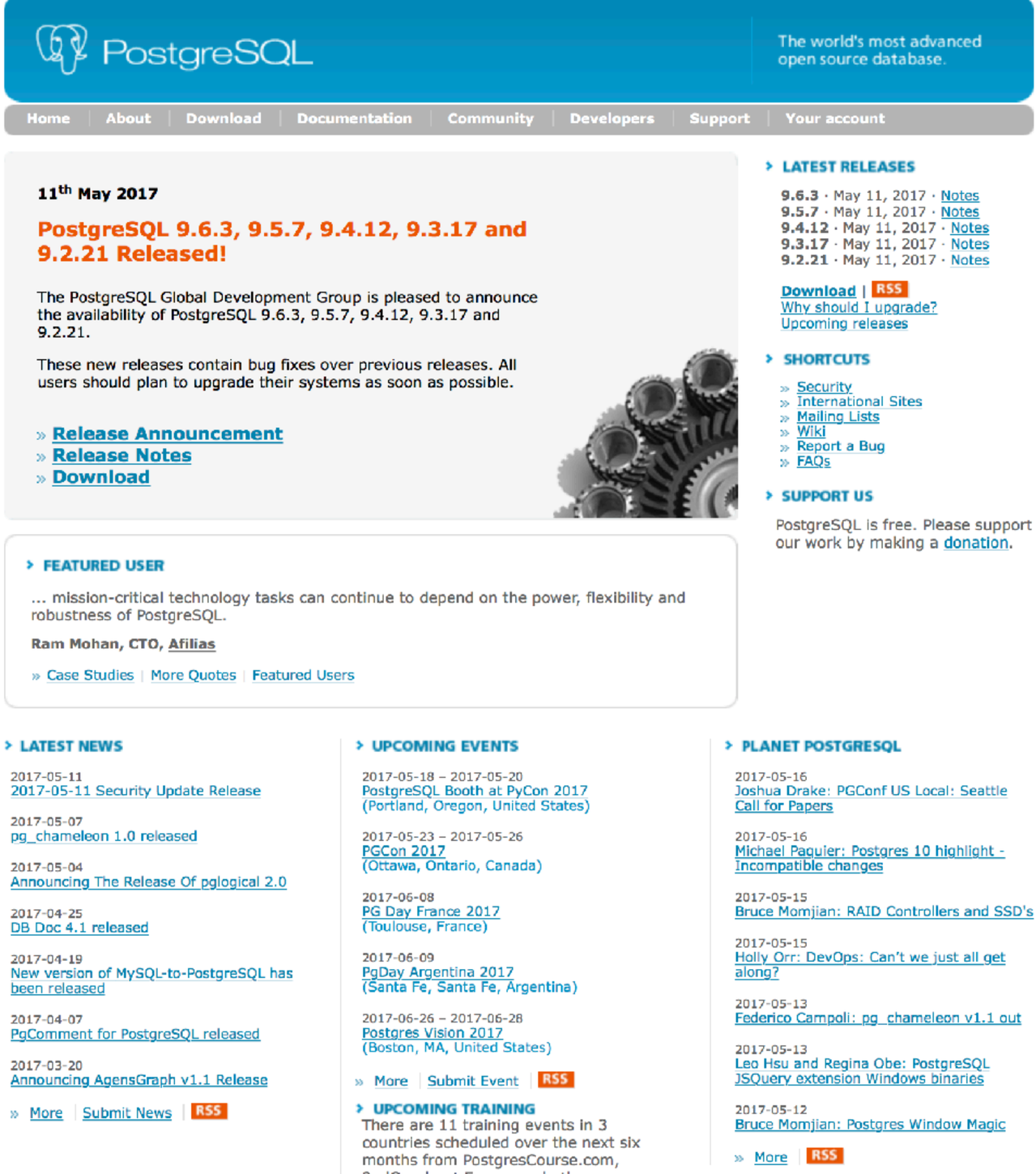
VAPOR WITH POSTGRESQL

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



VAPOR WITH POSTGRESQL

- Import package
- Configure droplet
- Add configuration file



VAPOR WITH POSTGRESQL

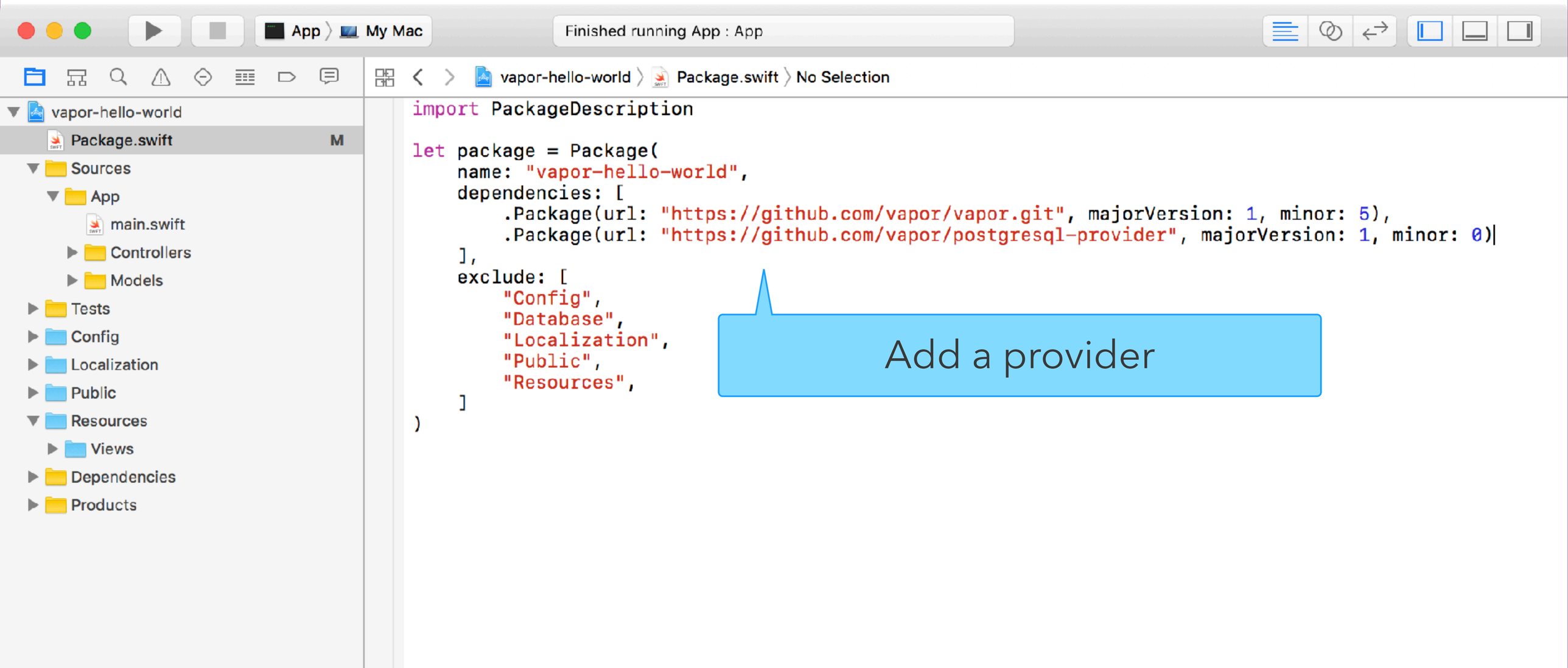
```
# Install postgres
brew install postgres

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

# Create a database
createdb dogs
```

VAPOR WITH POSTGRESQL

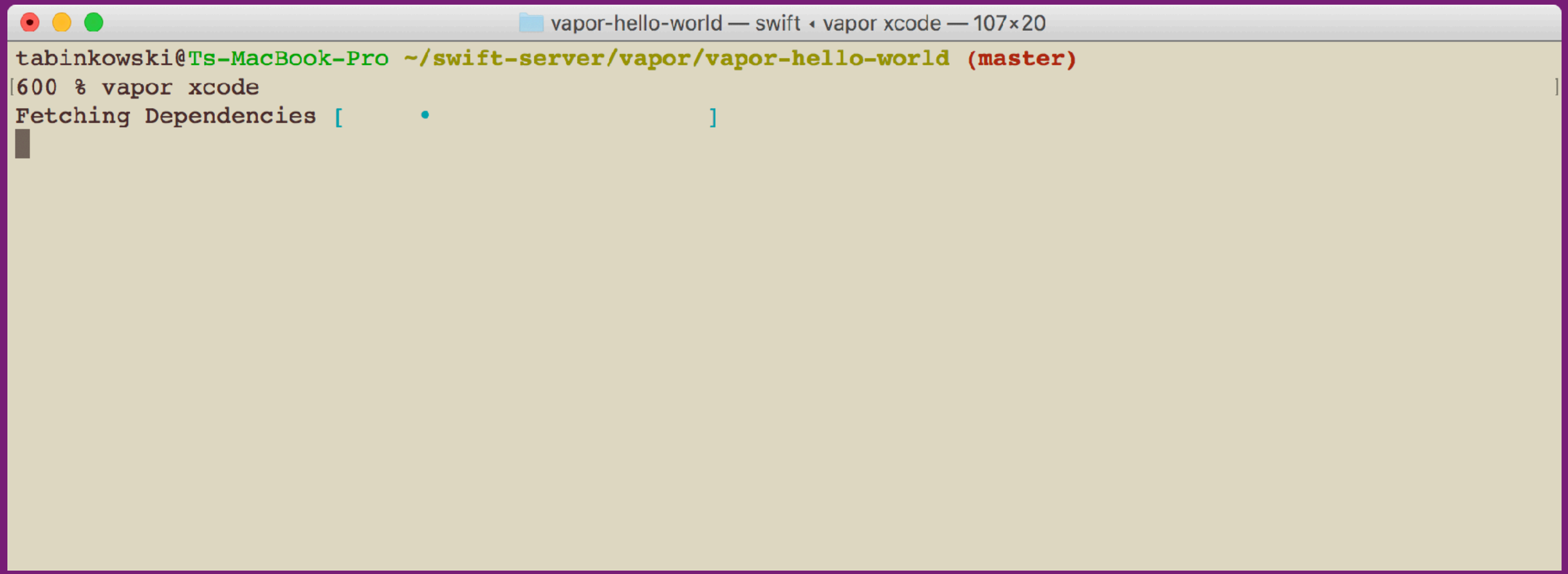
vapor-hello-world — psql — 107×20

```
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 WITH POSTGRESQL



VAPOR WITH POSTGRESQL



- Update the project and rebuild your framework in Xcode

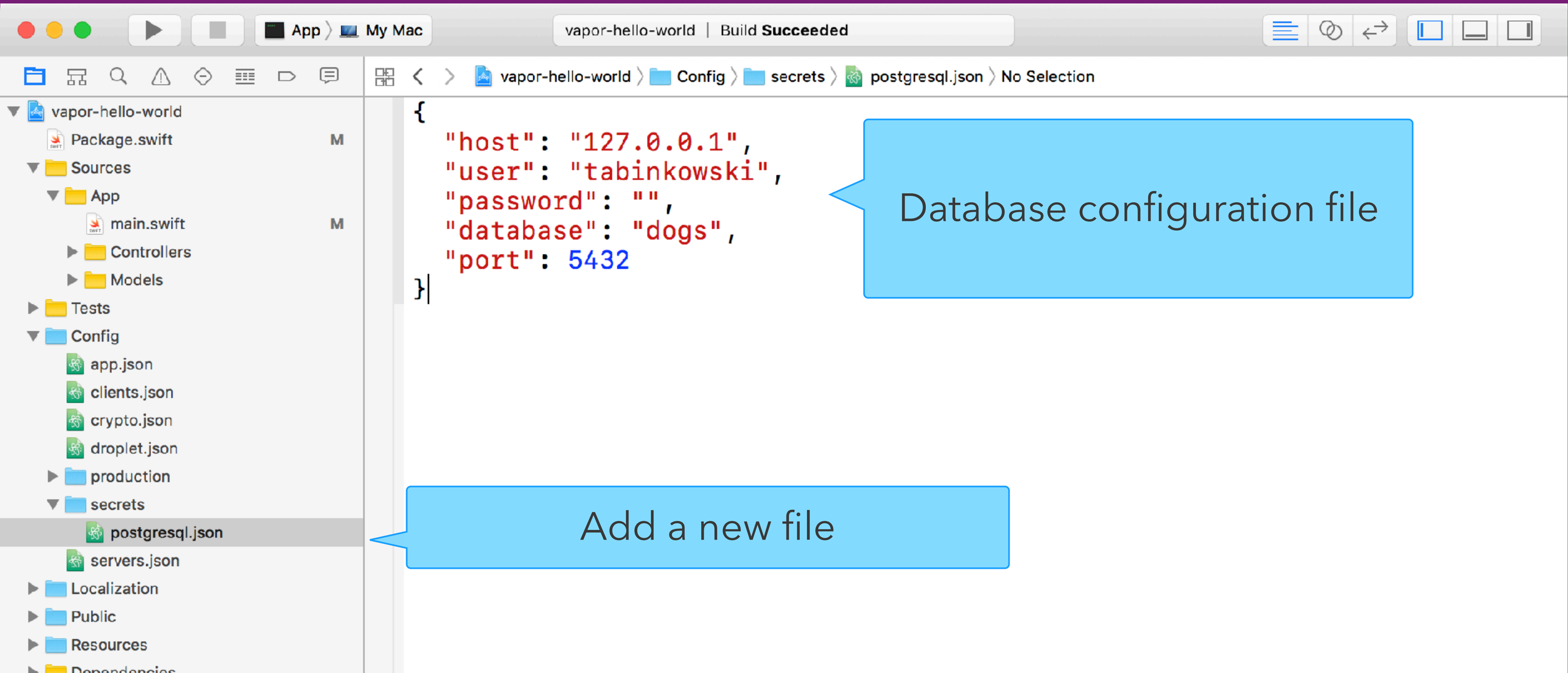
VAPOR WITH POSTGRESQL

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

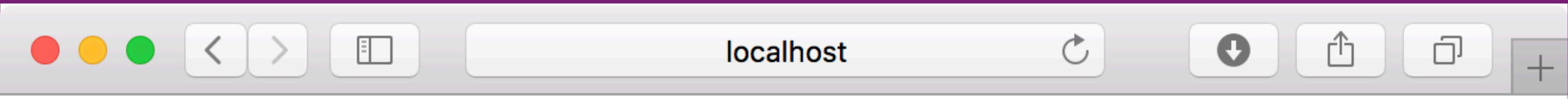


VAPOR WITH POSTGRESQL

```
// 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




```
[{"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

App 〉 My Mac | Running App : App

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

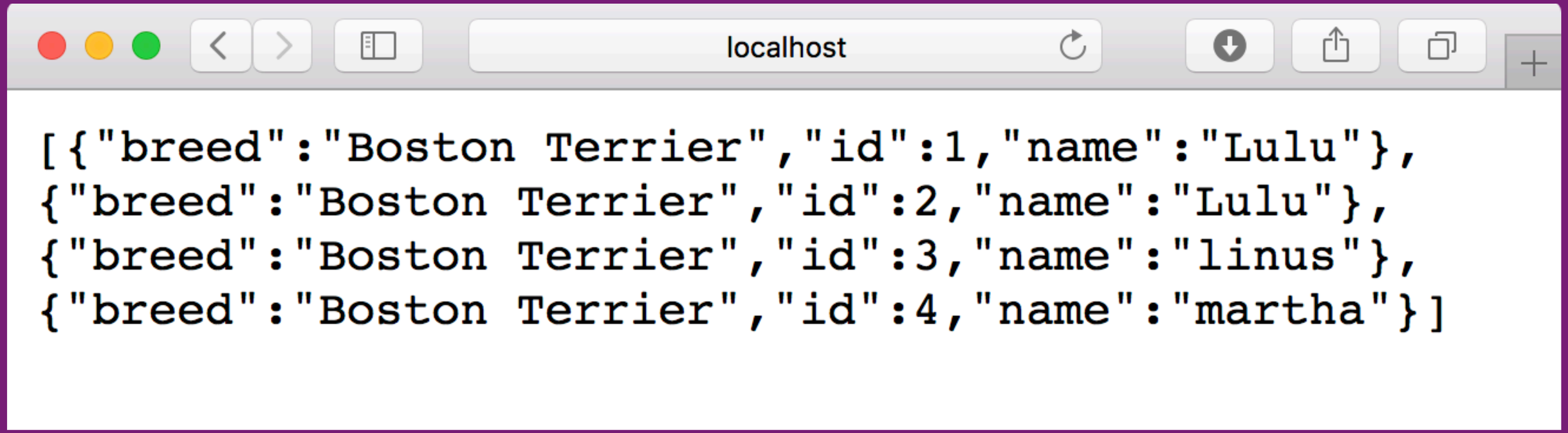
```
let drop = Droplet()
drop.preparations.append(Dog.self)

// 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)
}
```

ADD DOG MODEL

CREATE DATA

App

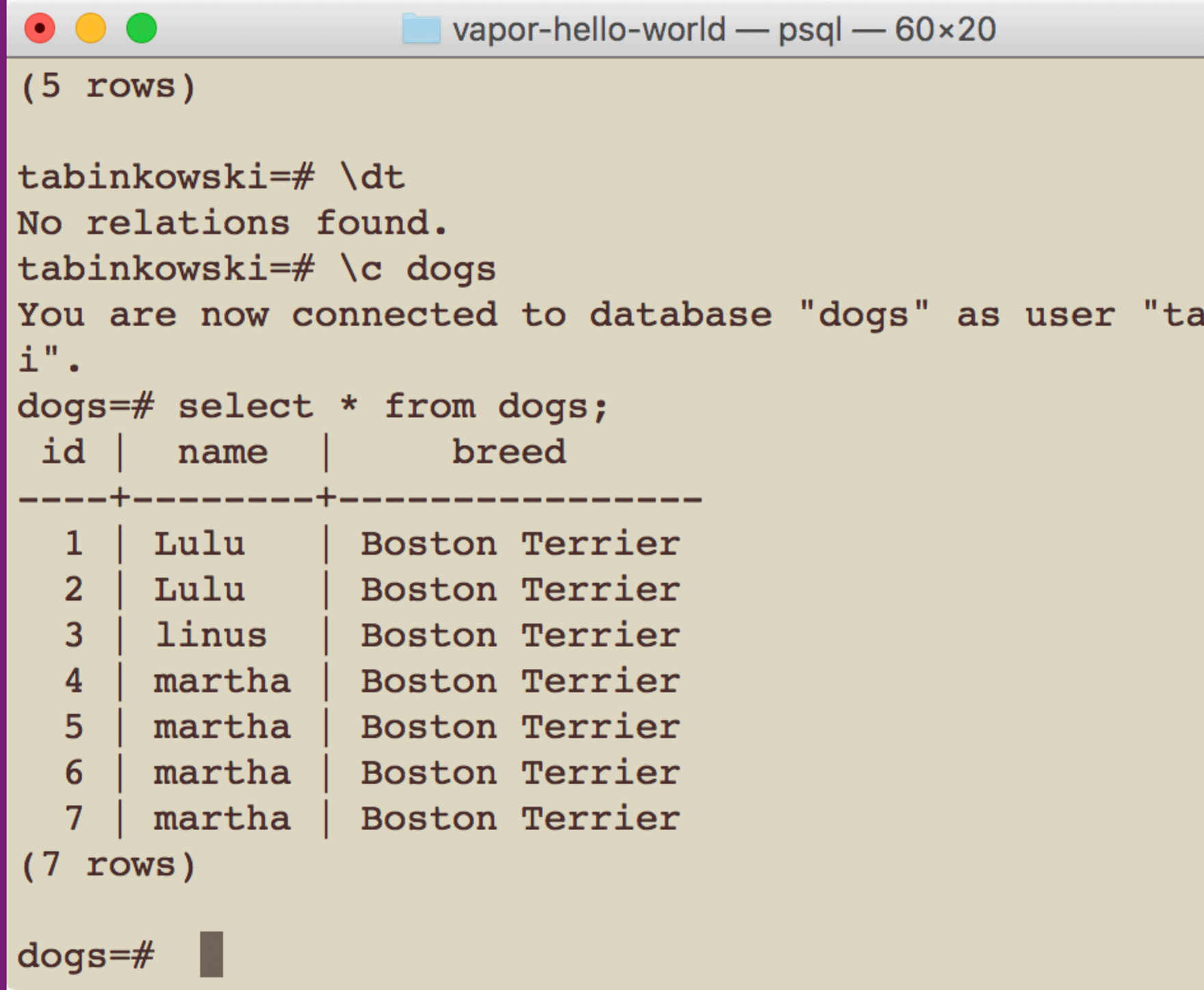
VAPOR WITH POSTGRESQL




```
// 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



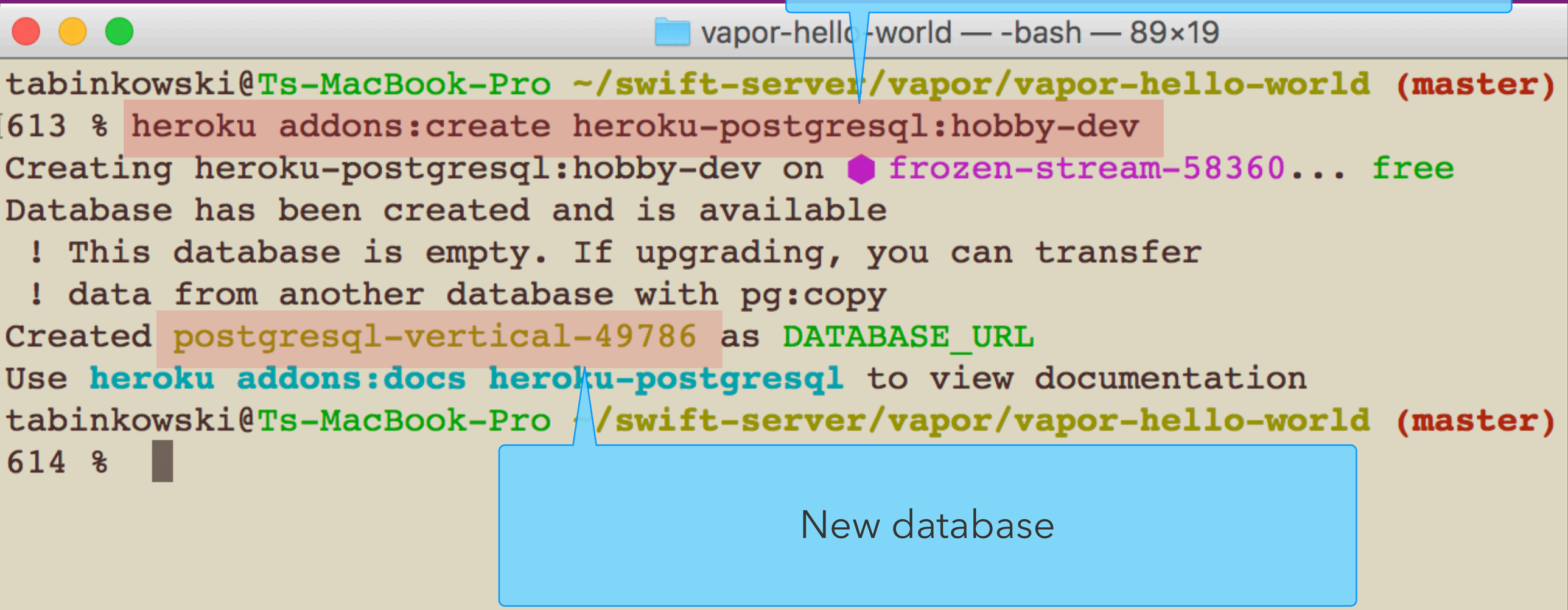
DEPLOY TO HEROKU WITH POSTGRESQL

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

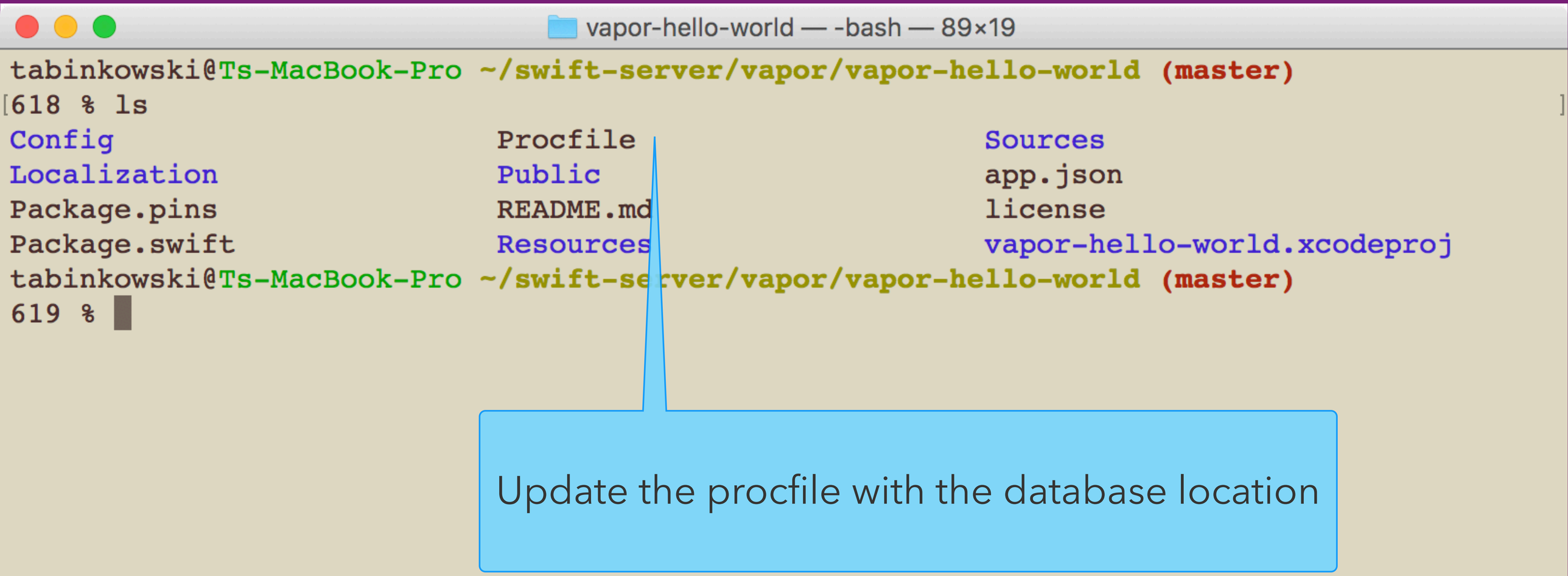


DEPLOY TO HEROKU WITH POSTGRESQL

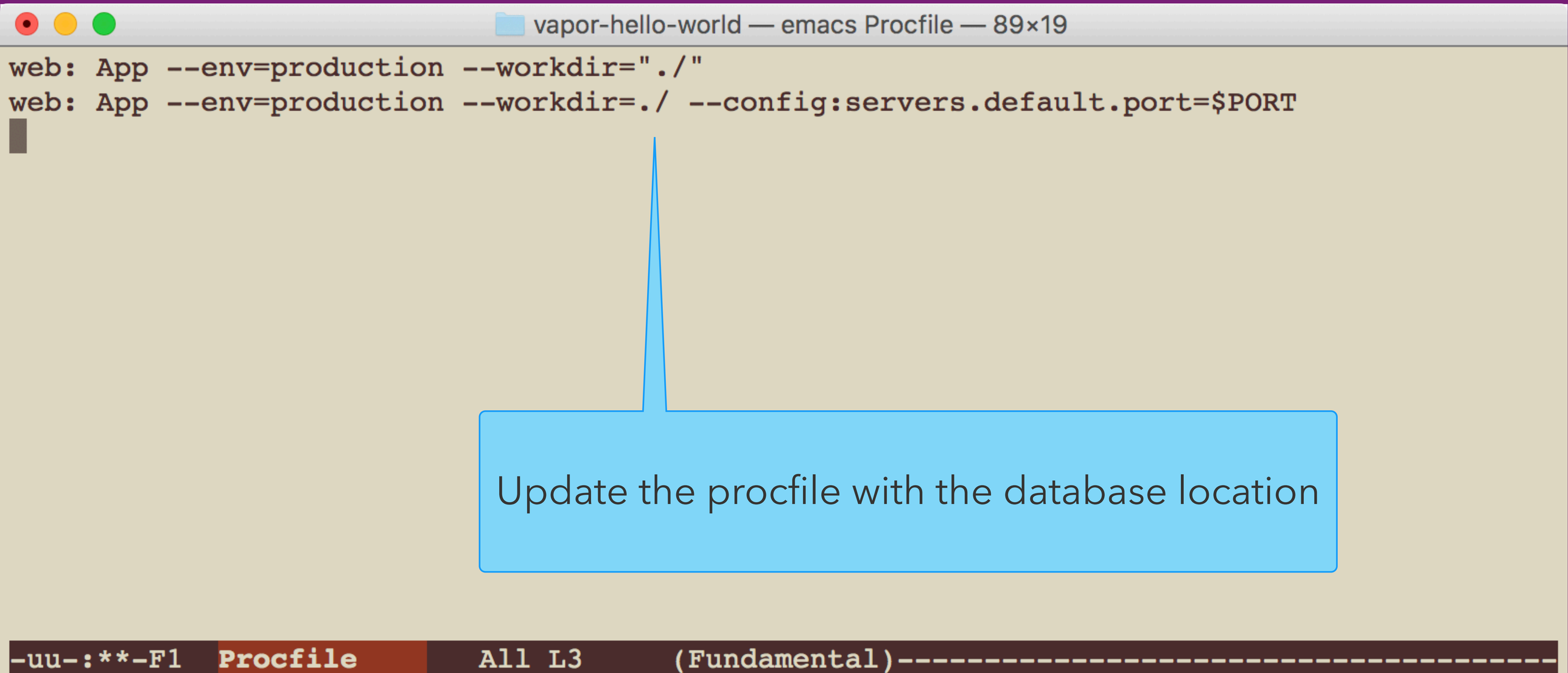
```
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[613 % heroku addons:create heroku-postgresql:hobby-dev
Creating heroku-postgresql:hobby-dev on ⬢ frozen-stream-58360... free
Database has been created and is available
 ! This database is empty. If upgrading, you can transfer
 ! data from another 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/d9hapt1pu1fgrf
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
615 %
```

Database location

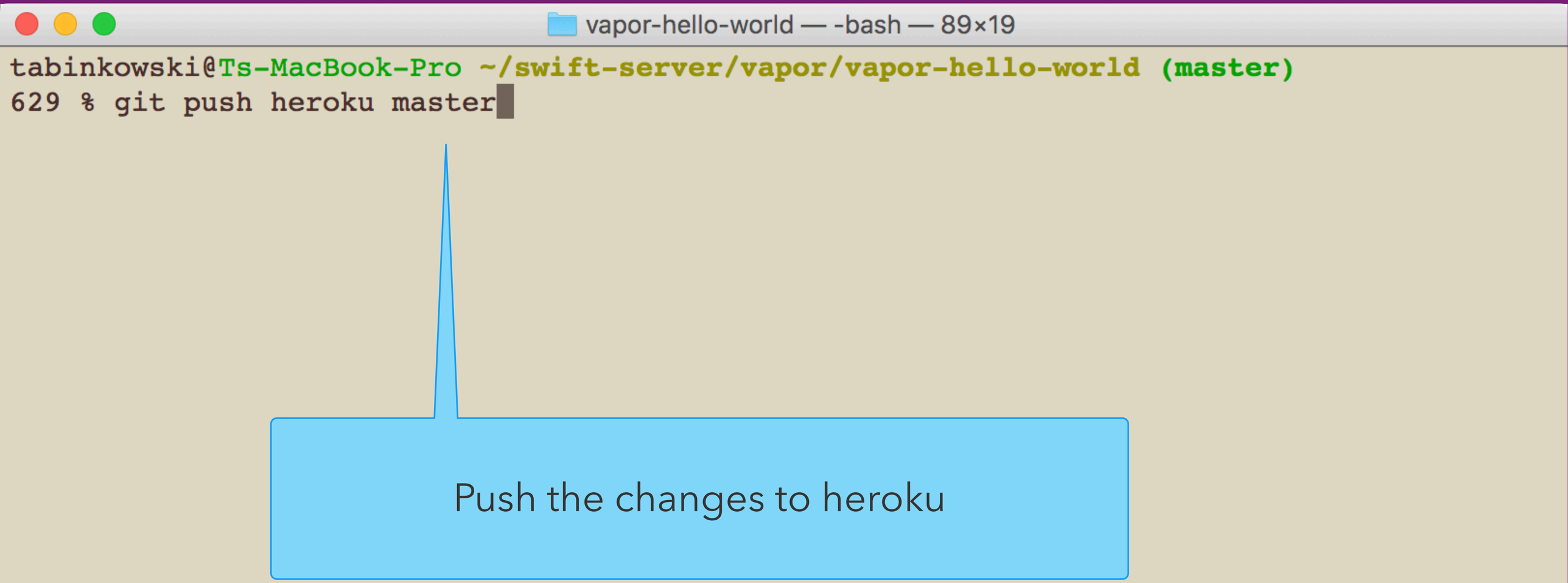
DEPLOY TO HEROKU WITH POSTGRESQL



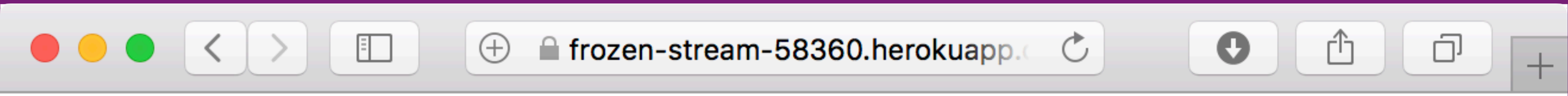
DEPLOY TO HEROKU WITH POSTGRESQL



DEPLOY TO HEROKU WITH POSTGRESQL



DEPLOY TO HEROKU WITH POSTGRESQL




```
[{"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

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

Filter

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

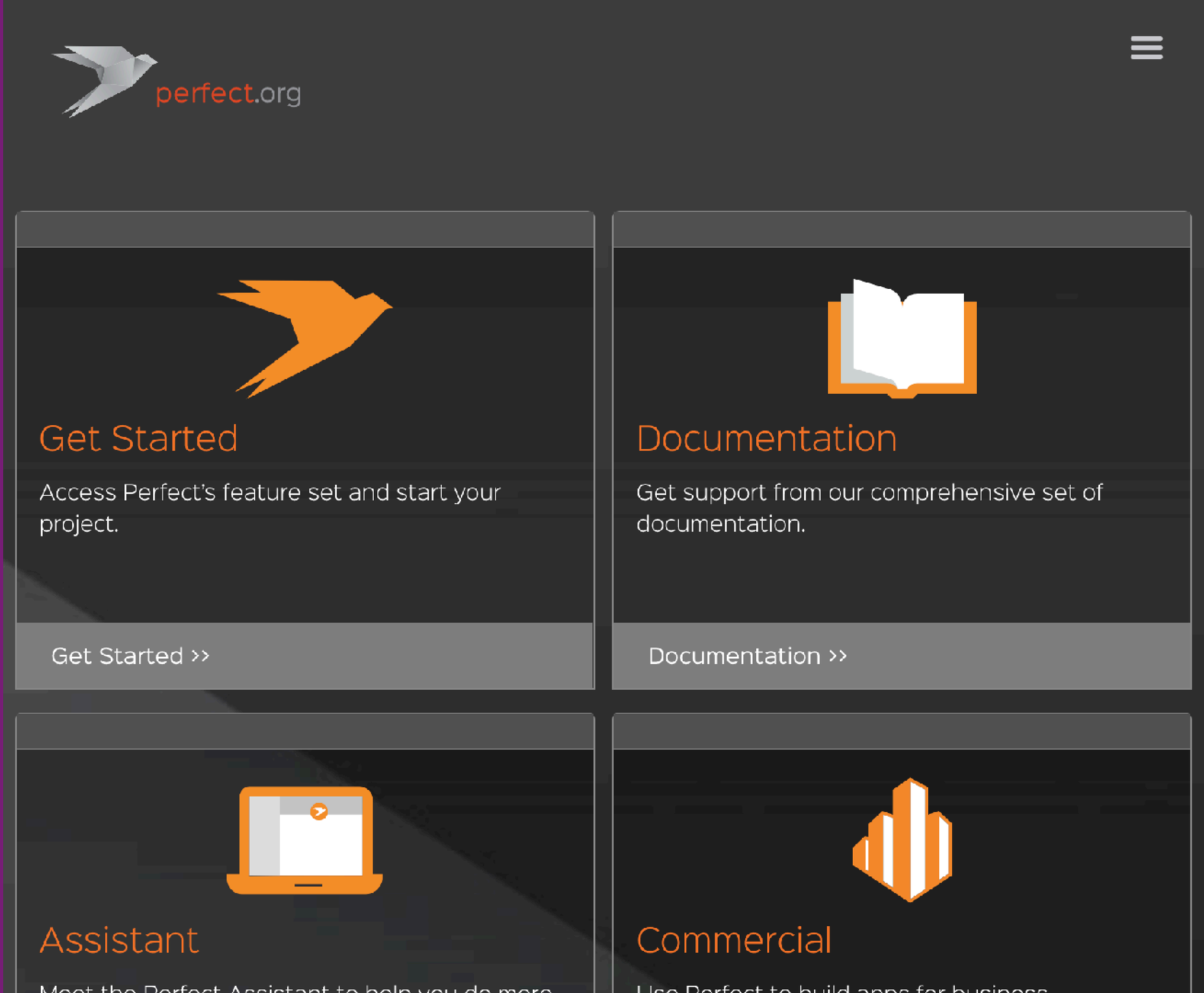
Save

BREAK TIME 

PERFECT

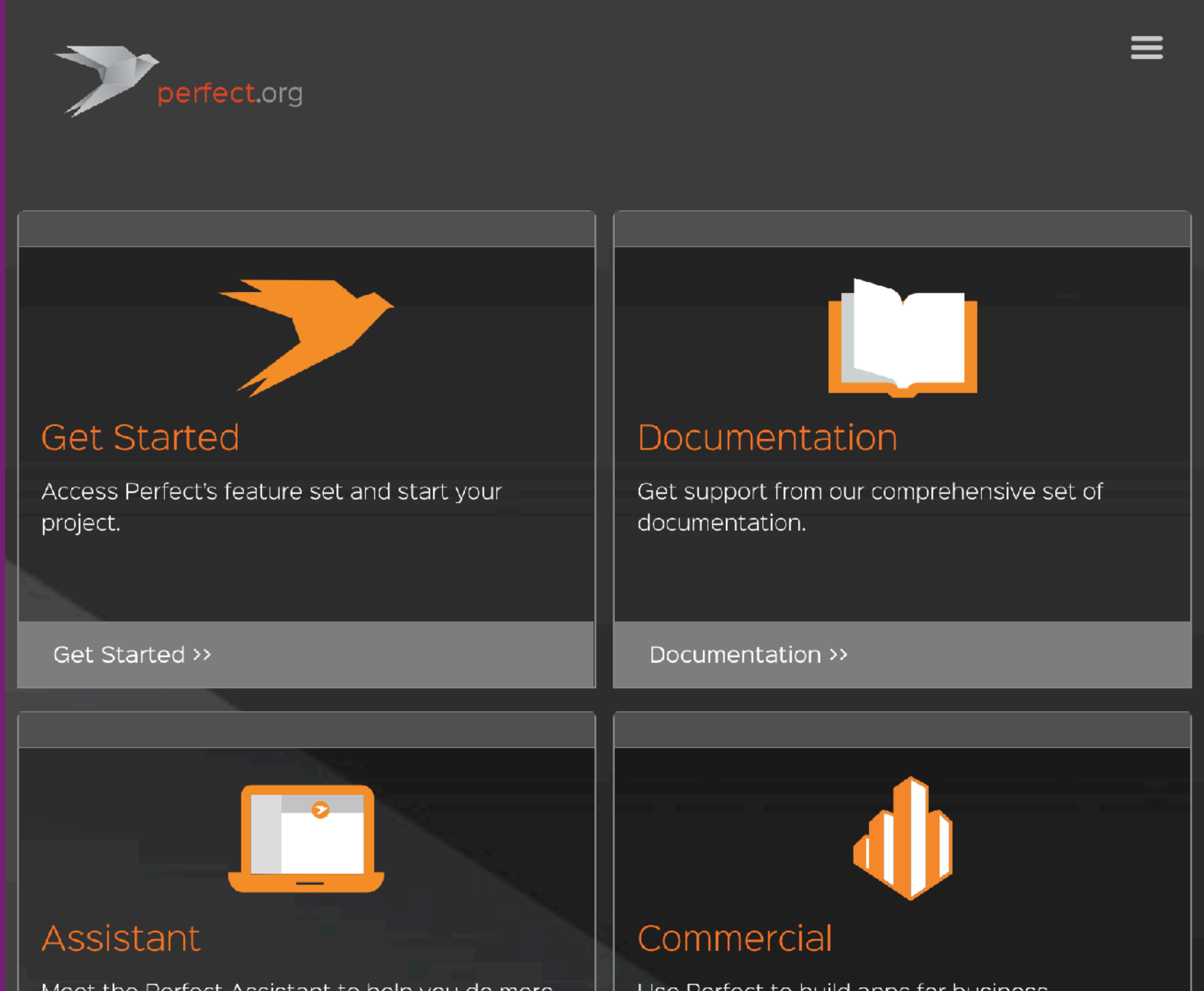
PERFECT

- Web framework and server for Swift server applications



PERFECT

- Requirements
  - Xcode 8 Mac
  - Linux toolchain

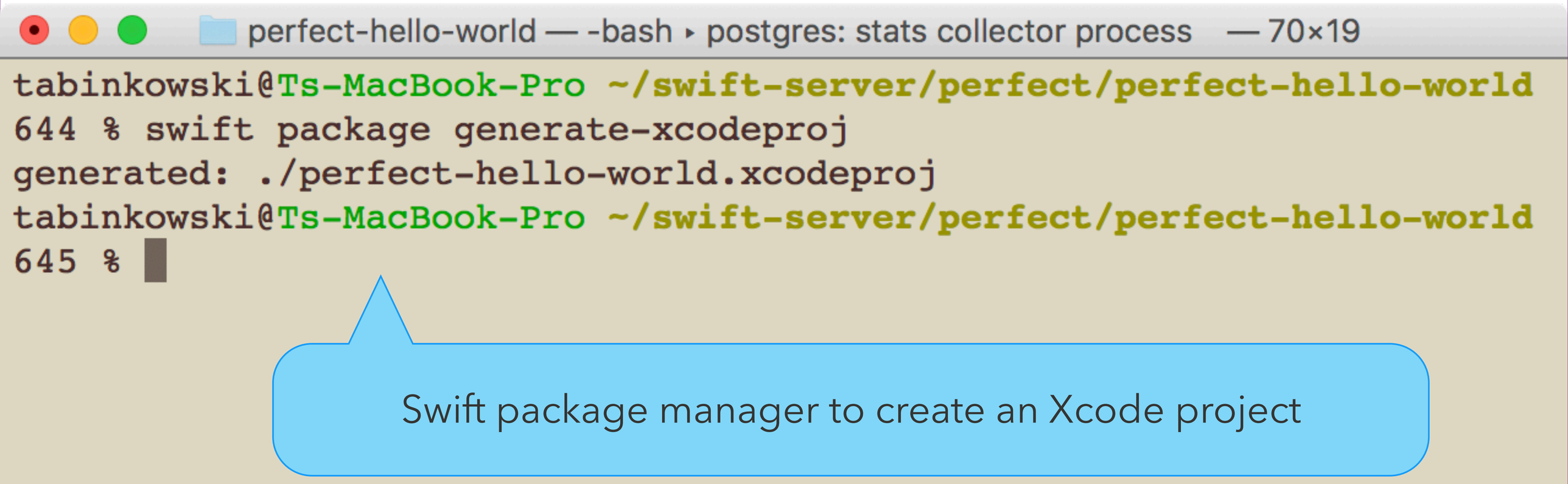


PERFECT

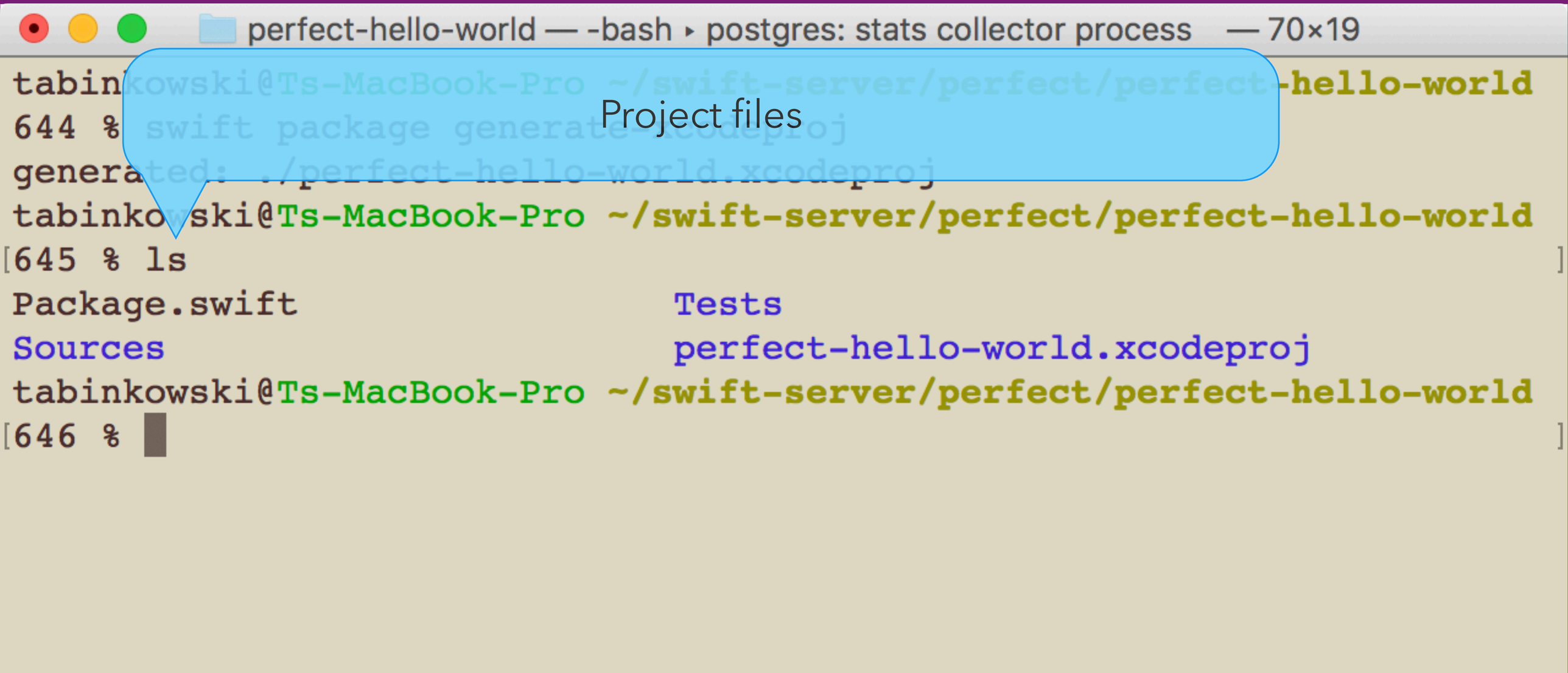
```
[637 % cd perfect/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[638 % ls
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[639 % mkdir perfect-hello-world
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect
[640 % cd perfect-hello-world/
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world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

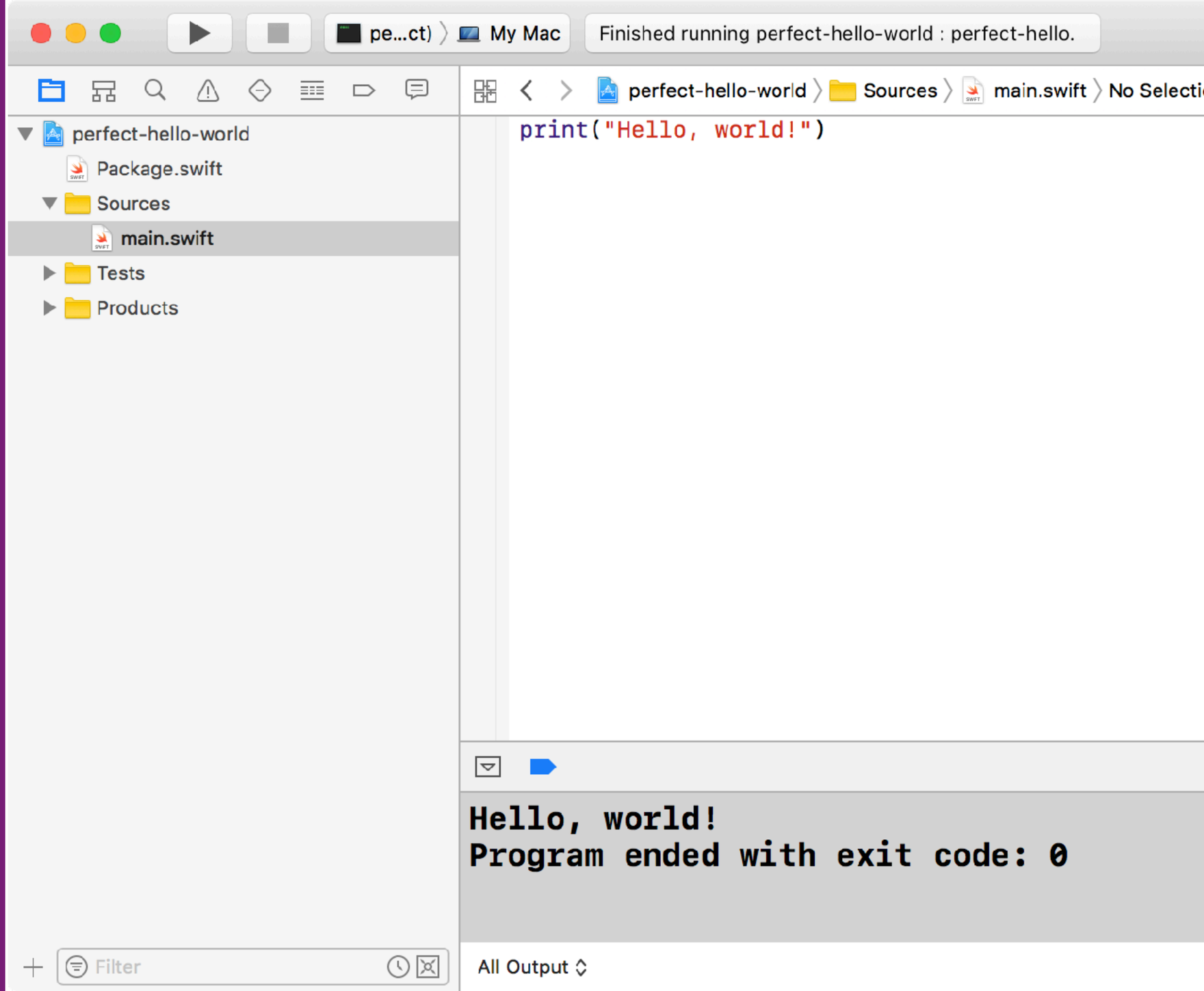

perfect-hello-world — -bash ▸ postgres: stats collector process — 70×19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
645 %
Swift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

PERFECT

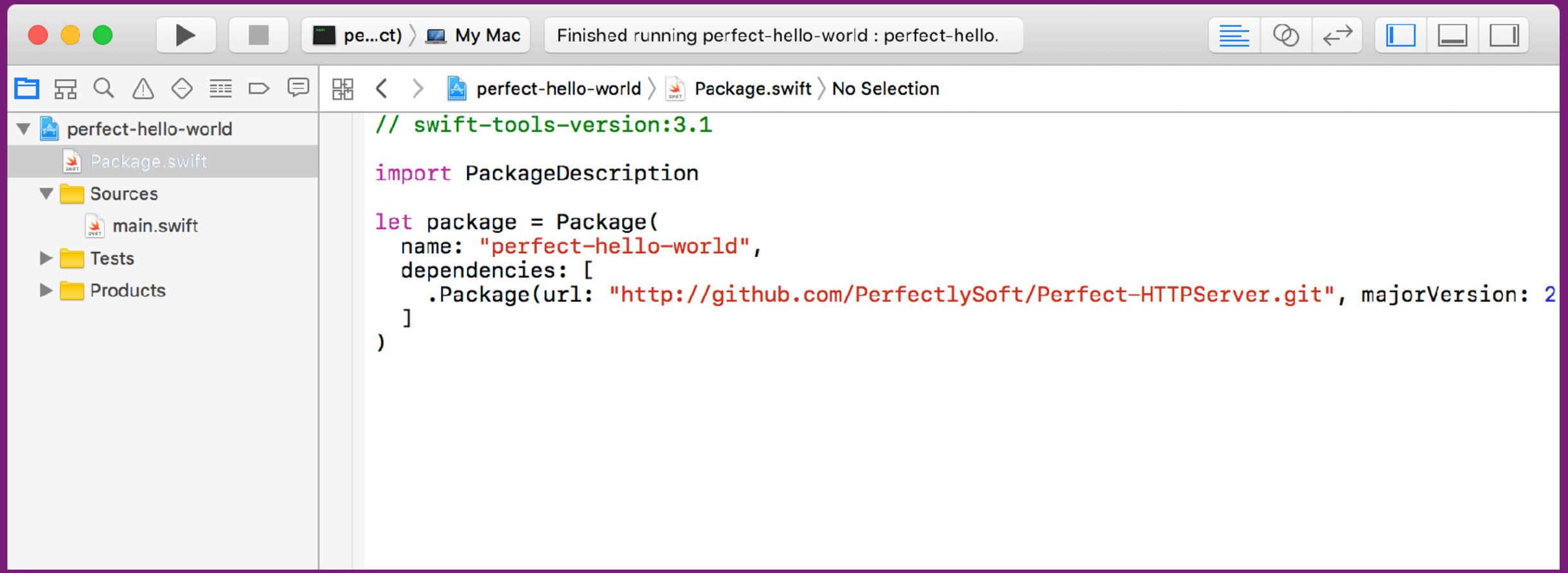

perfect-hello-world — -bash ▸ postgres: stats collector process — 70×19
Project files
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world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

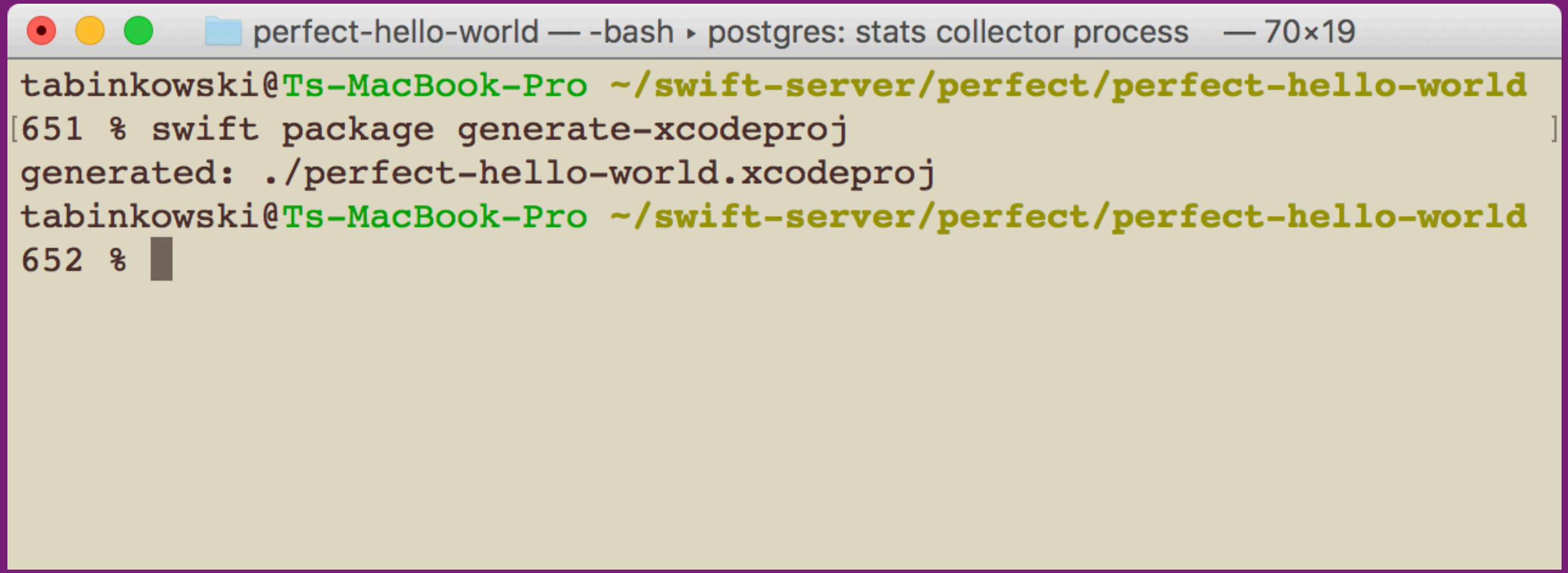


- Add Perfect as a project dependency

PERFECT

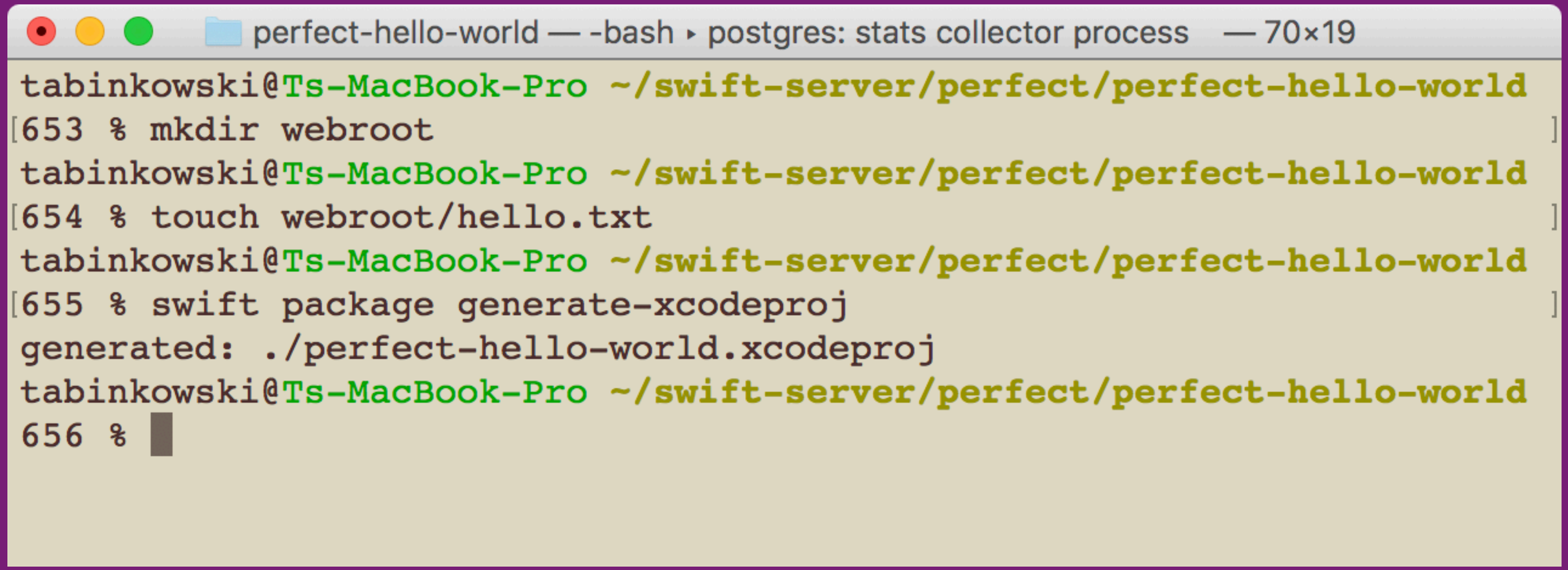
- Update the package manager

PERFECT



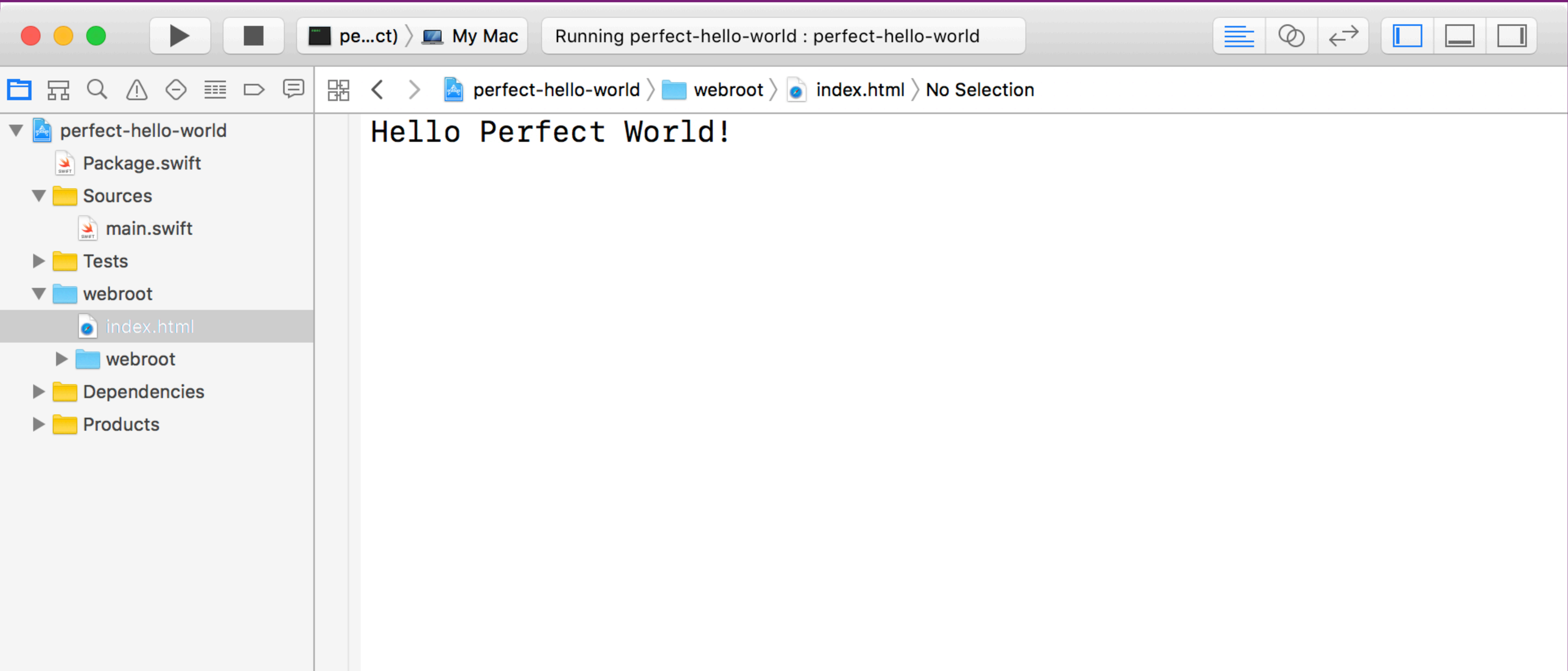
- Update the Xcode project

PERFECT



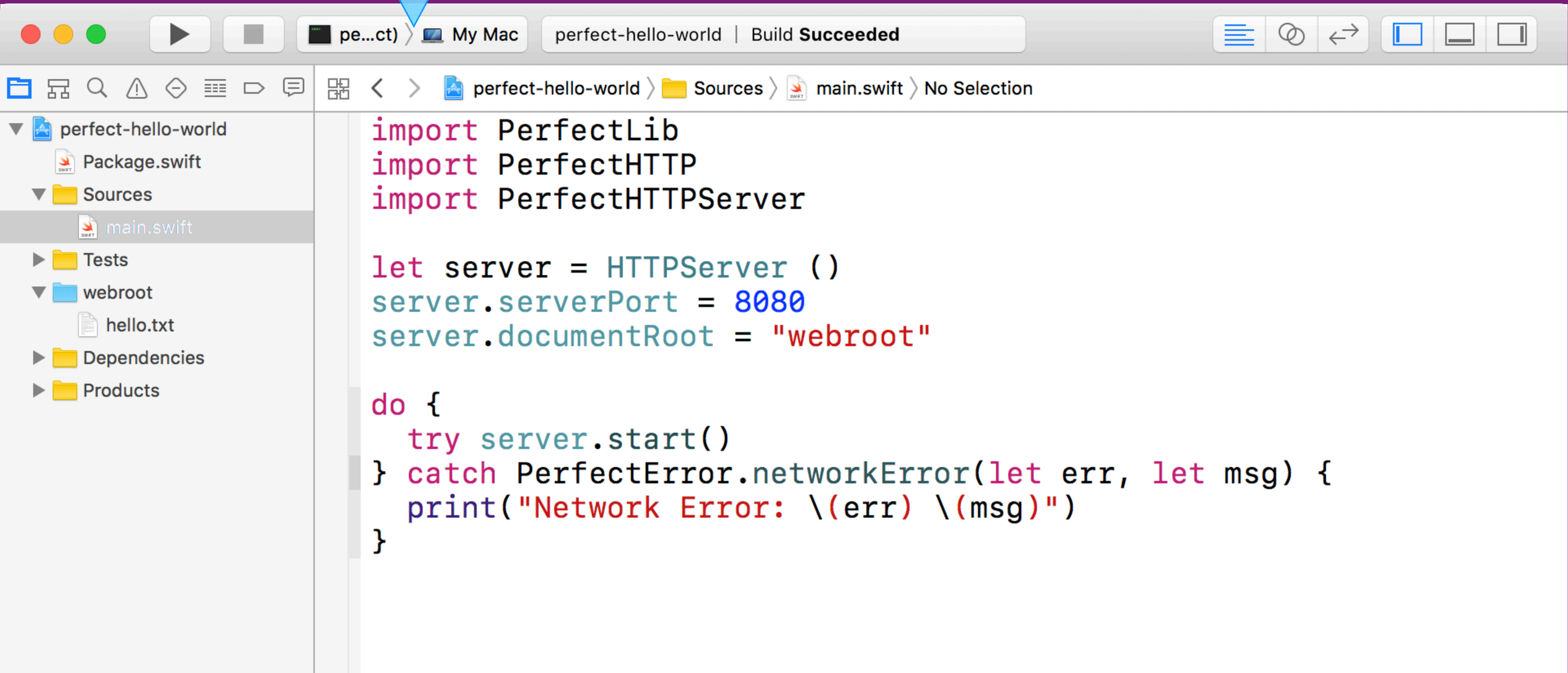
- Create a root directory for the web server

PERFECT

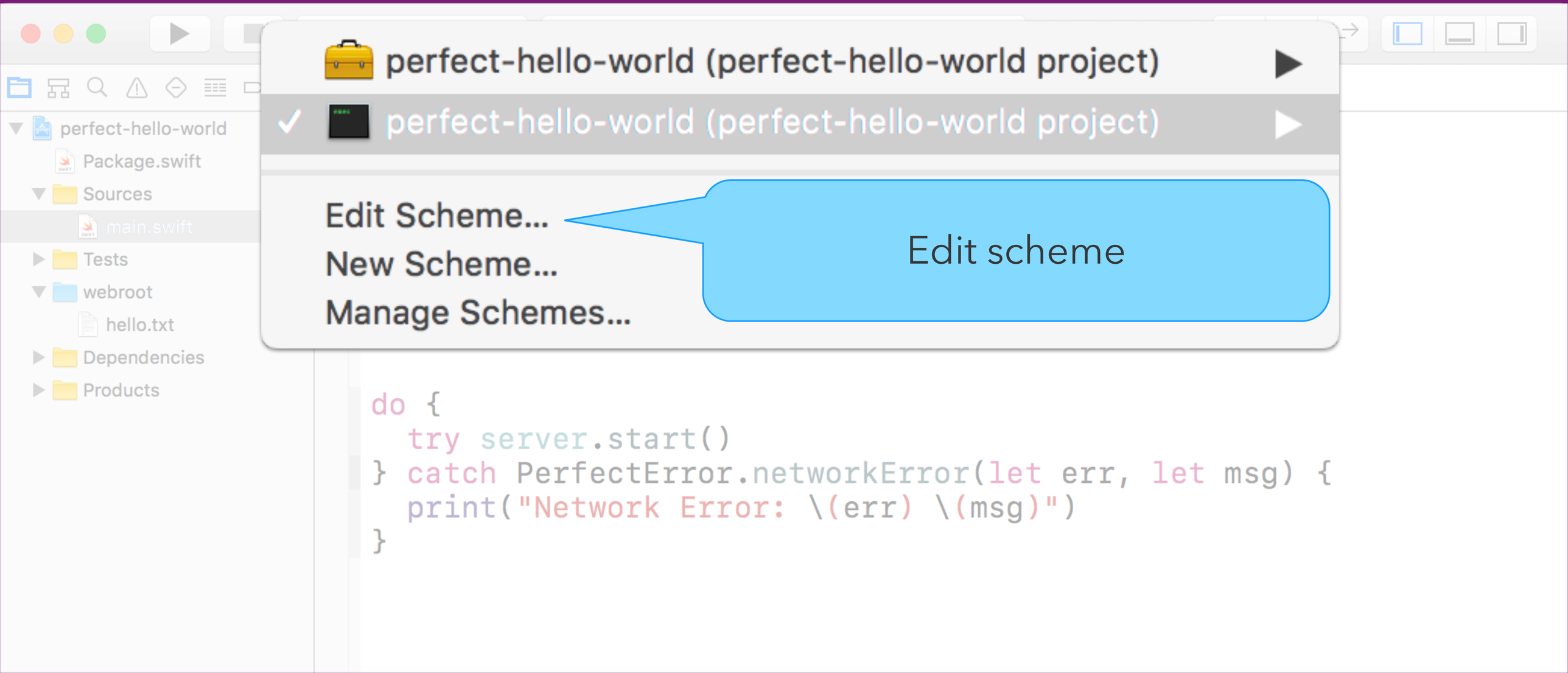

pe...ct) 〉 My Mac
Running perfect-hello-world : perfect-hello-world
perfect-hello-world 〉 webroot 〉 index.html 〉 No Selection
perfect-hello-world
Package.swift
Sources
main.swift
Tests
webroot
index.html
webroot
Dependencies
Products
Hello Perfect World!

PERFECT

Create a server

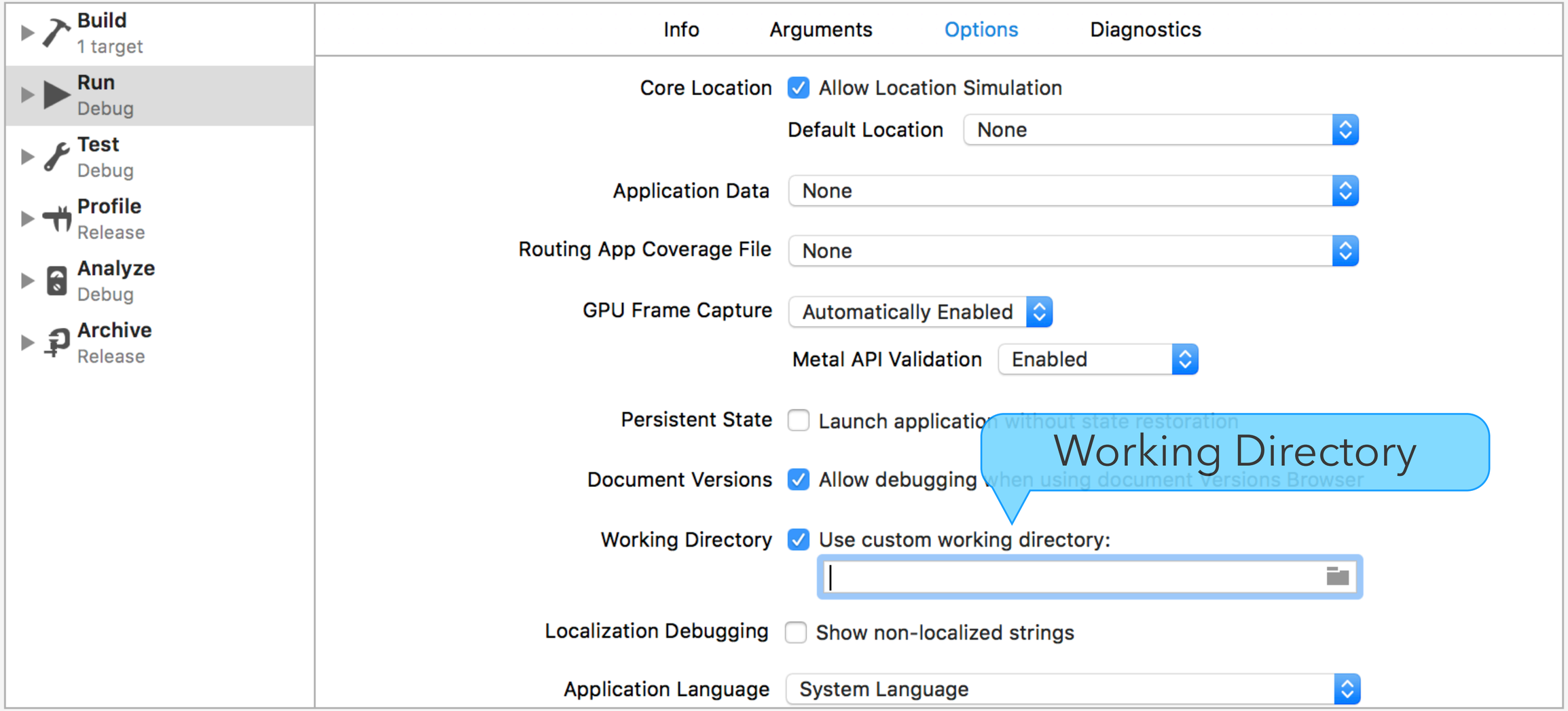


PERFECT

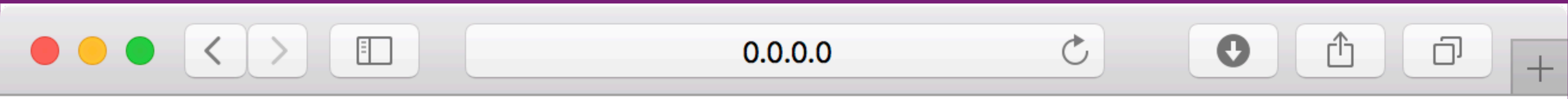

perfect-hello-world (perfect-hello-world project)
perfect-hello-world (perfect-hello-world project)
Edit Scheme...
New Scheme...
Manage Schemes...
Edit scheme
perfect-hello-world
Package.swift
Sources
main.swift
Tests
webroot
hello.txt
Dependencies
Products
do {
try server.start()
} catch PerfectError.networkError(let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

PERFECT

Options

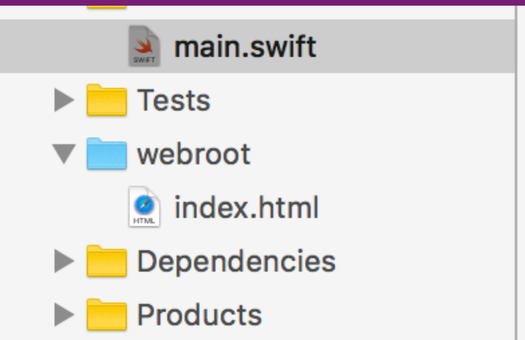


PERFECT



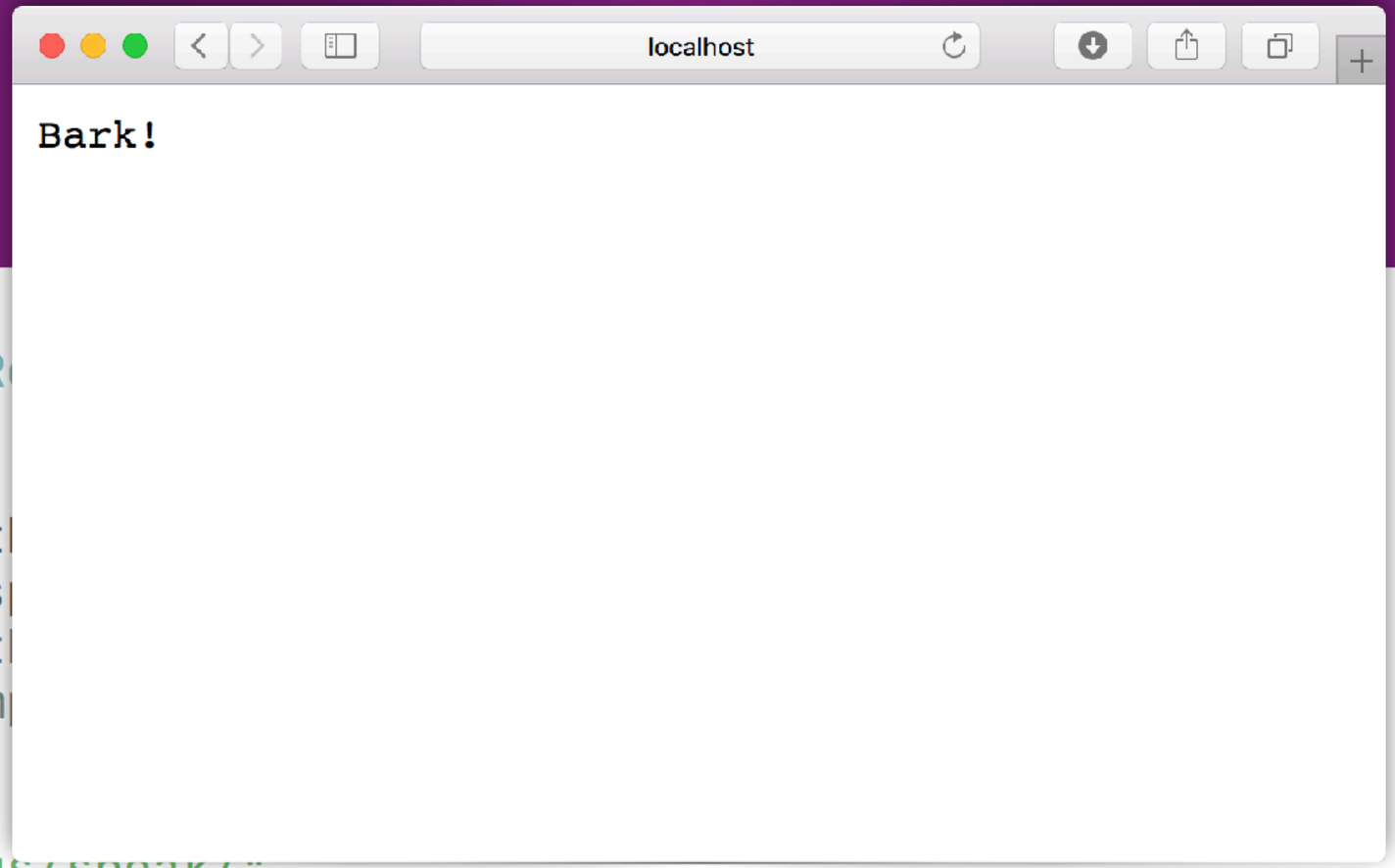
Hello Perfect World!

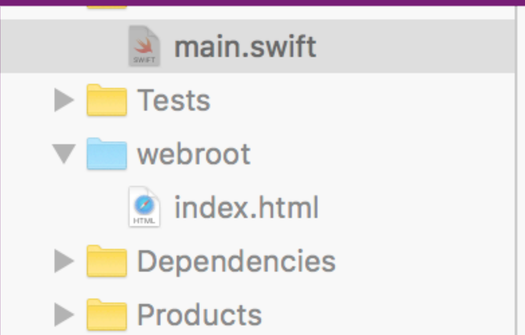
PERFECT




```
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



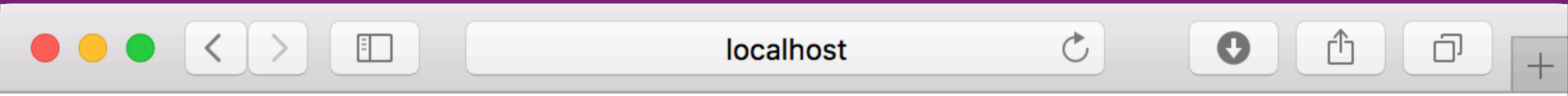



```
var routes = R

// 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



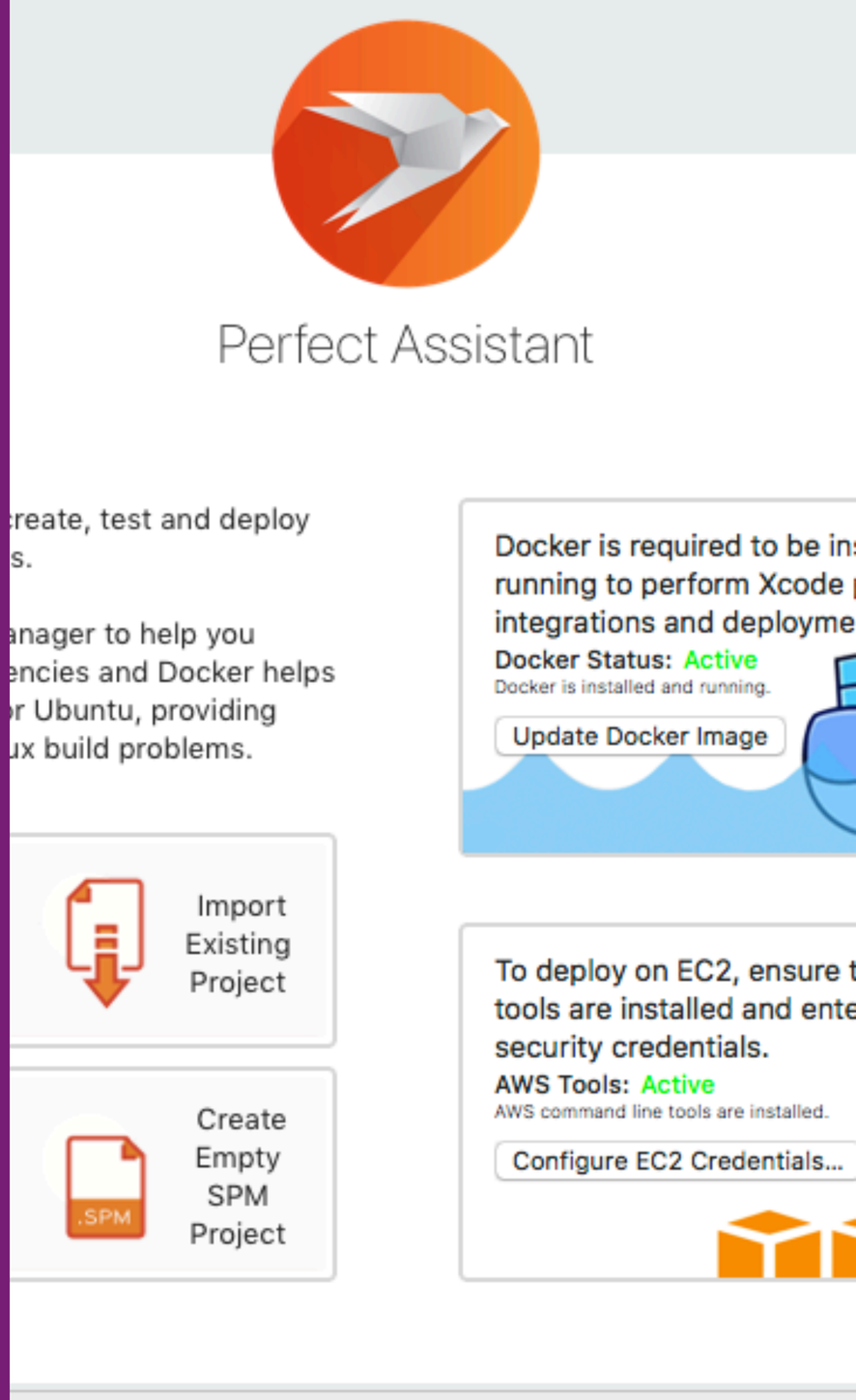

```
lassie says Bark!
```

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

PERFECT ASSISTANT

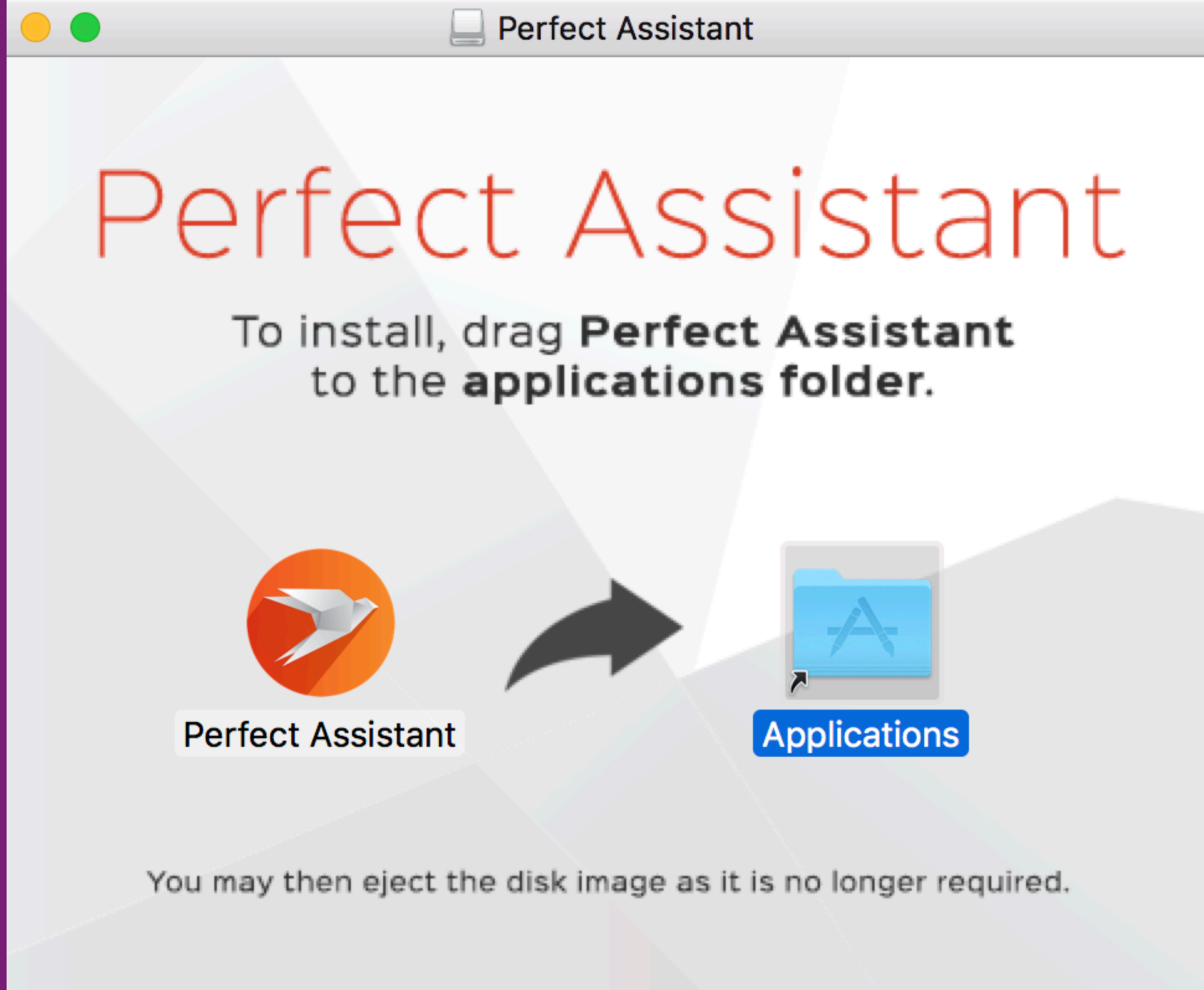
PERFECT ASSISTANT

- 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



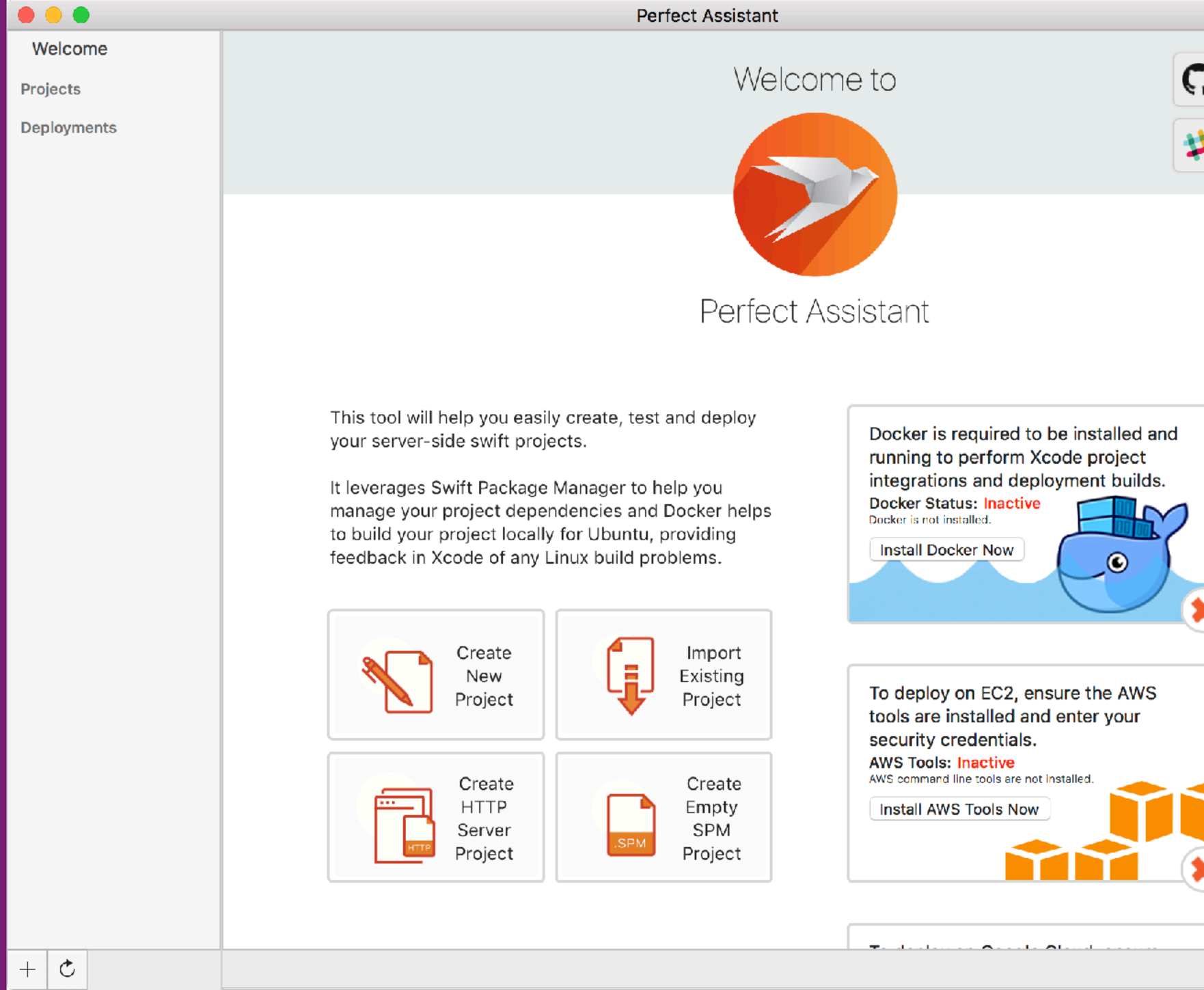
PERFECT ASSISTANT

- Download and install



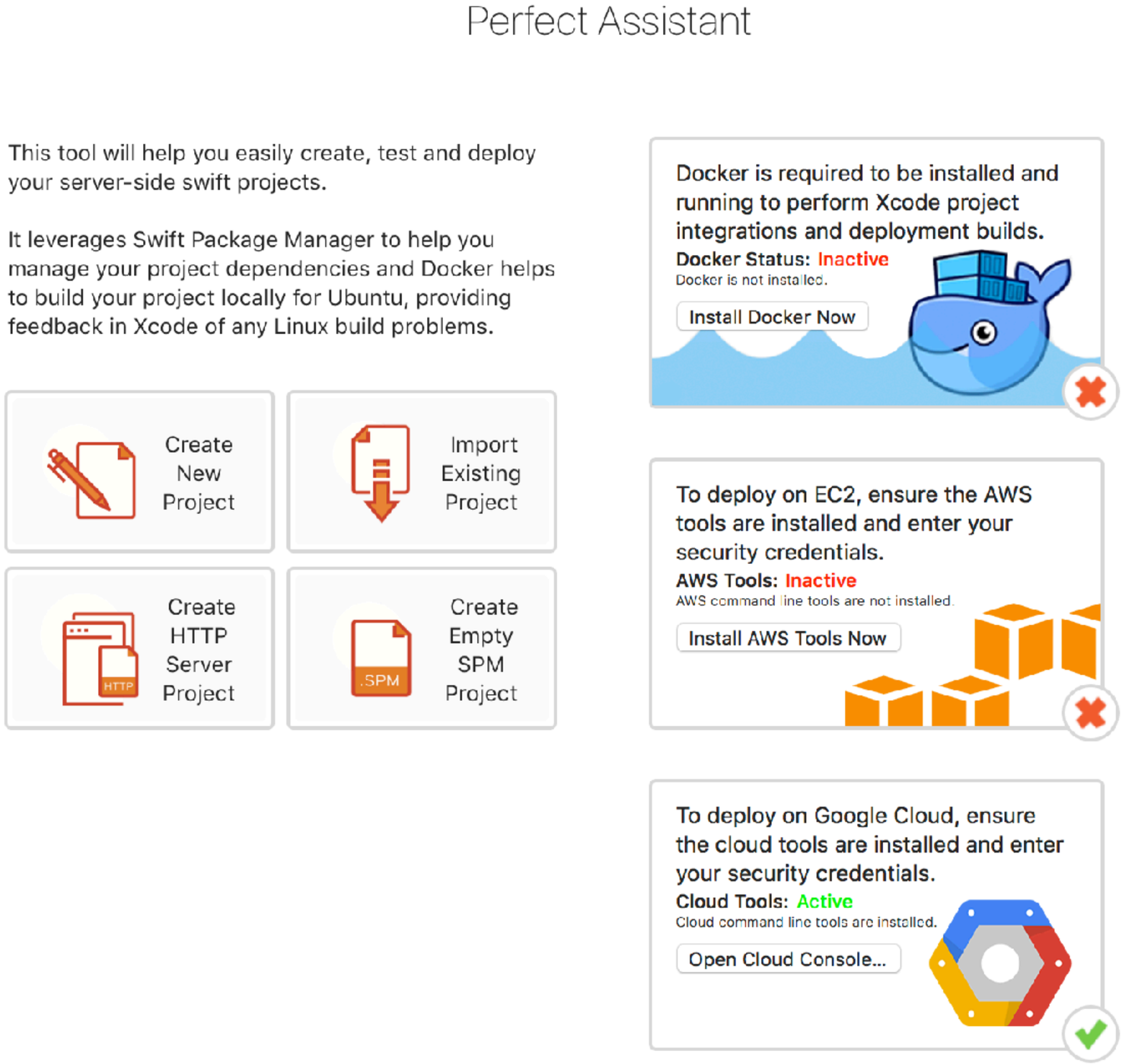
PERFECT ASSISTANT

- Download and install

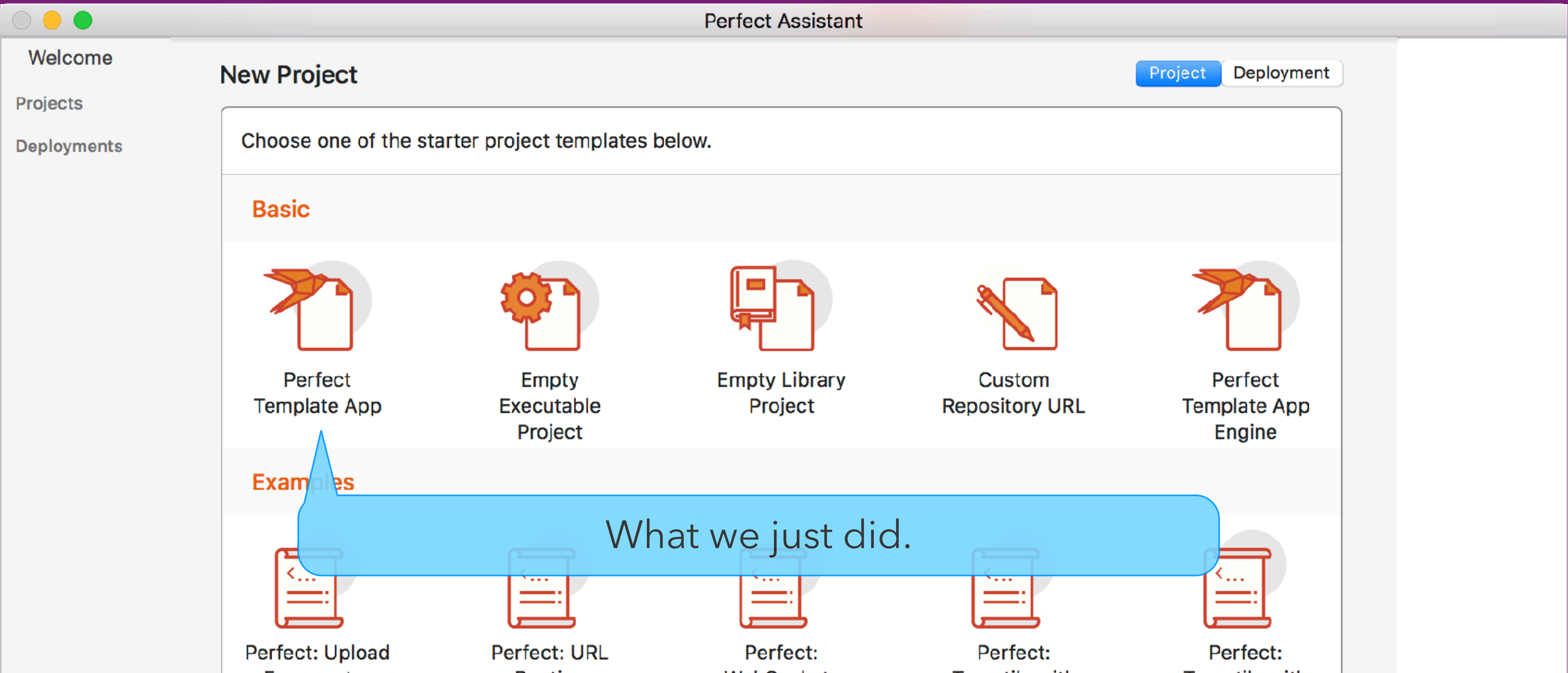


PERFECT ASSISTANT

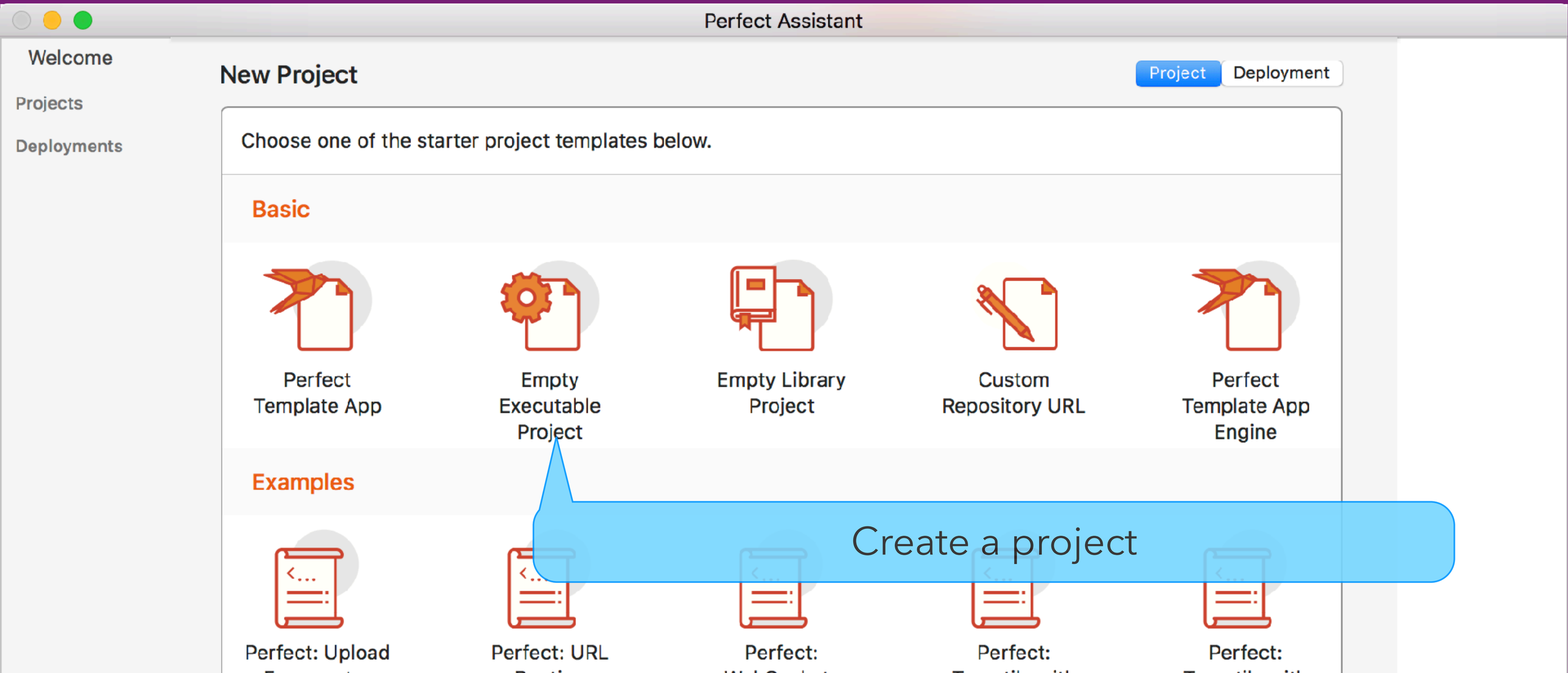
- Deploy using Docker



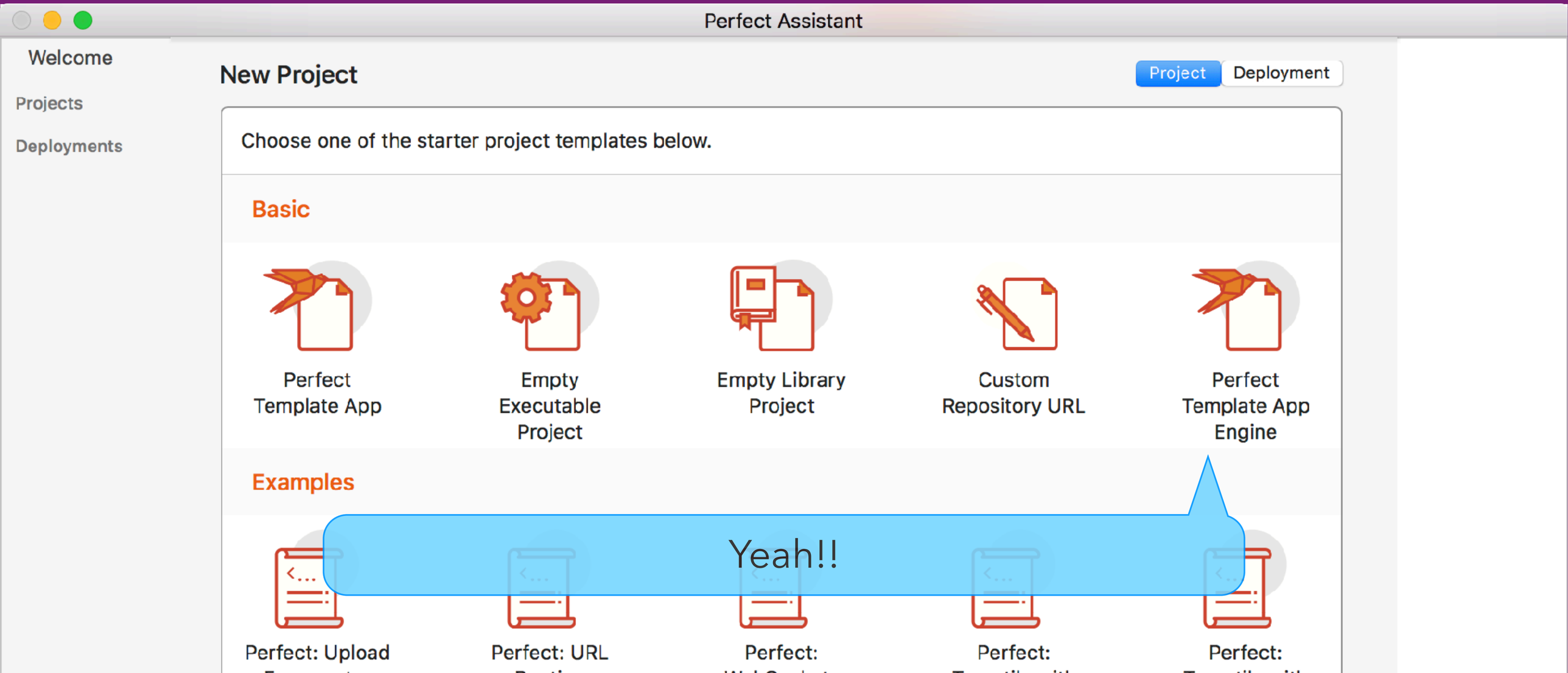
PERFECT ASSISTANT



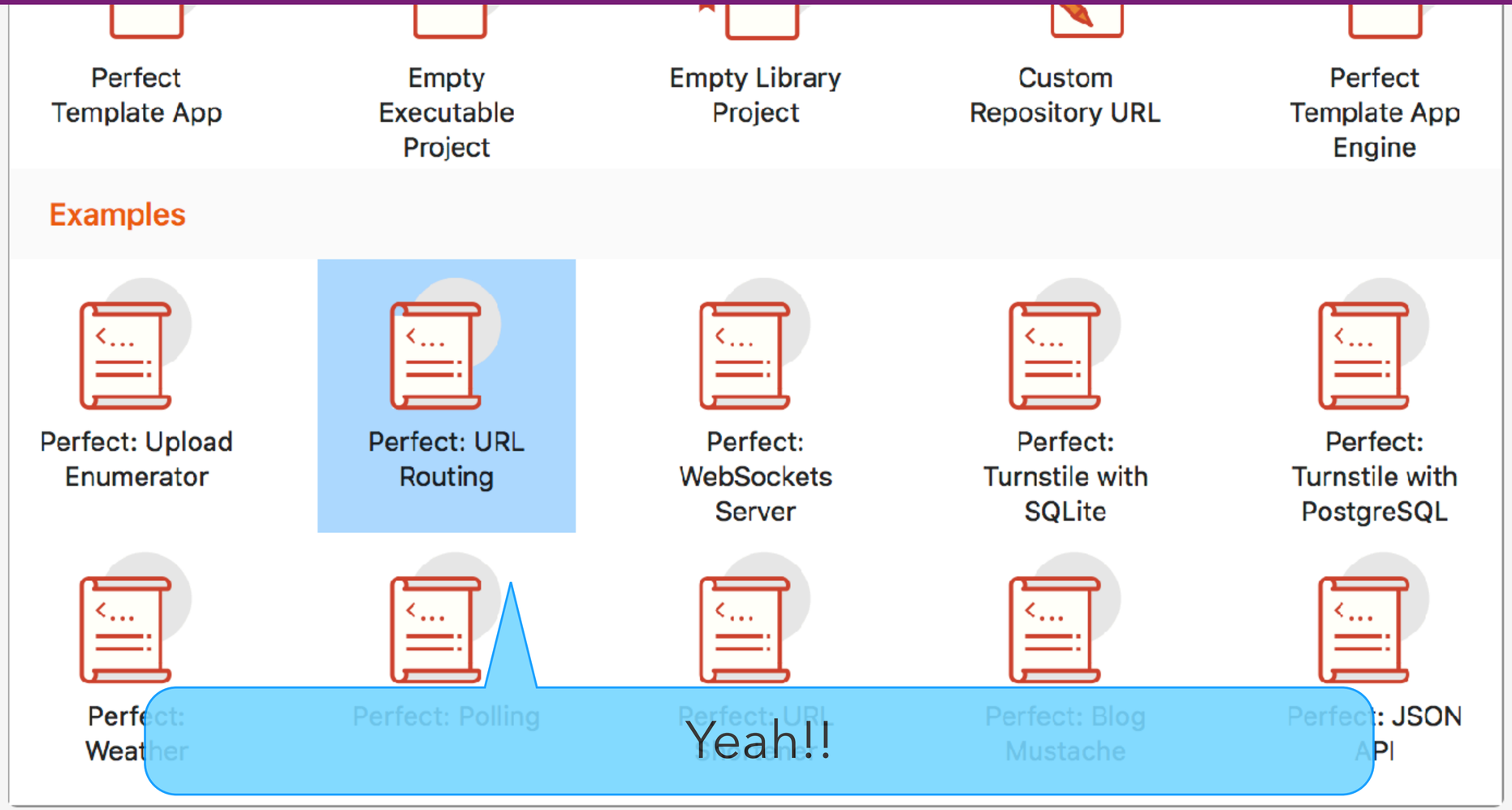
PERFECT ASSISTANT



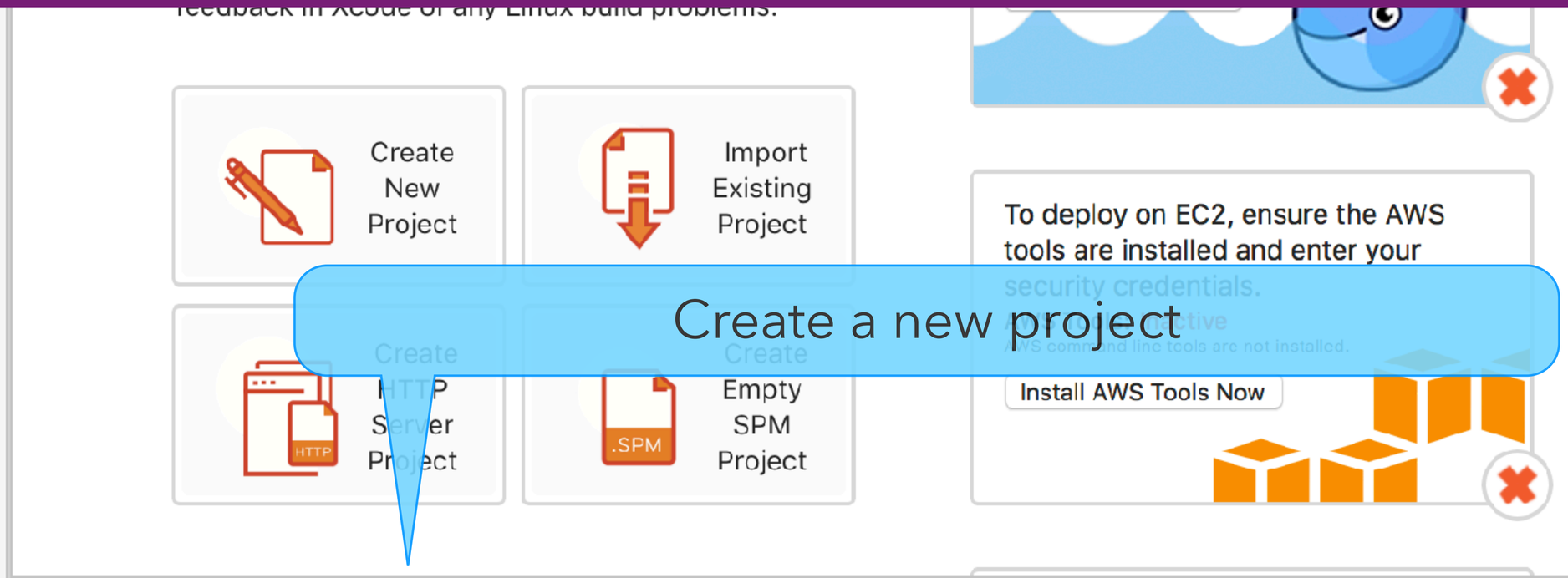
PERFECT ASSISTANT



PERFECT ASSISTANT



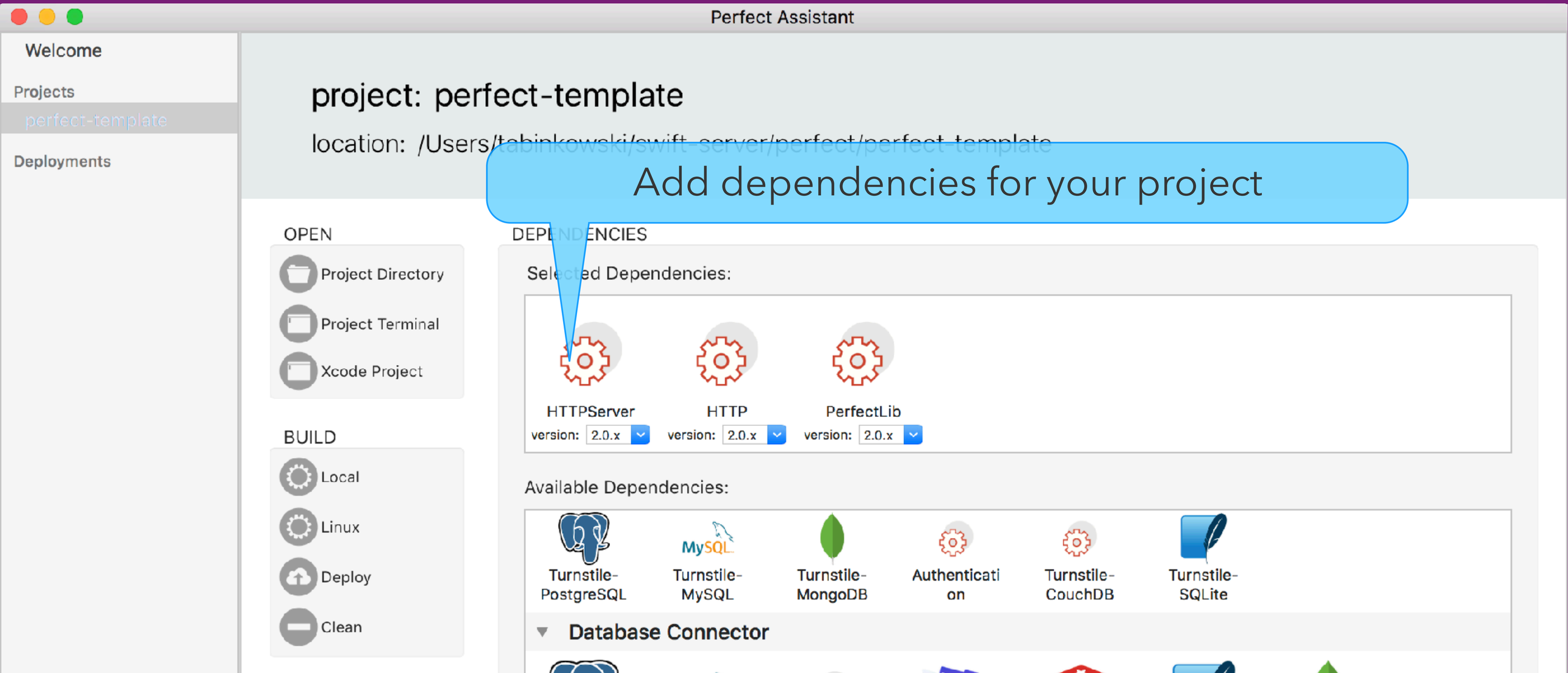
PERFECT ASSISTANT



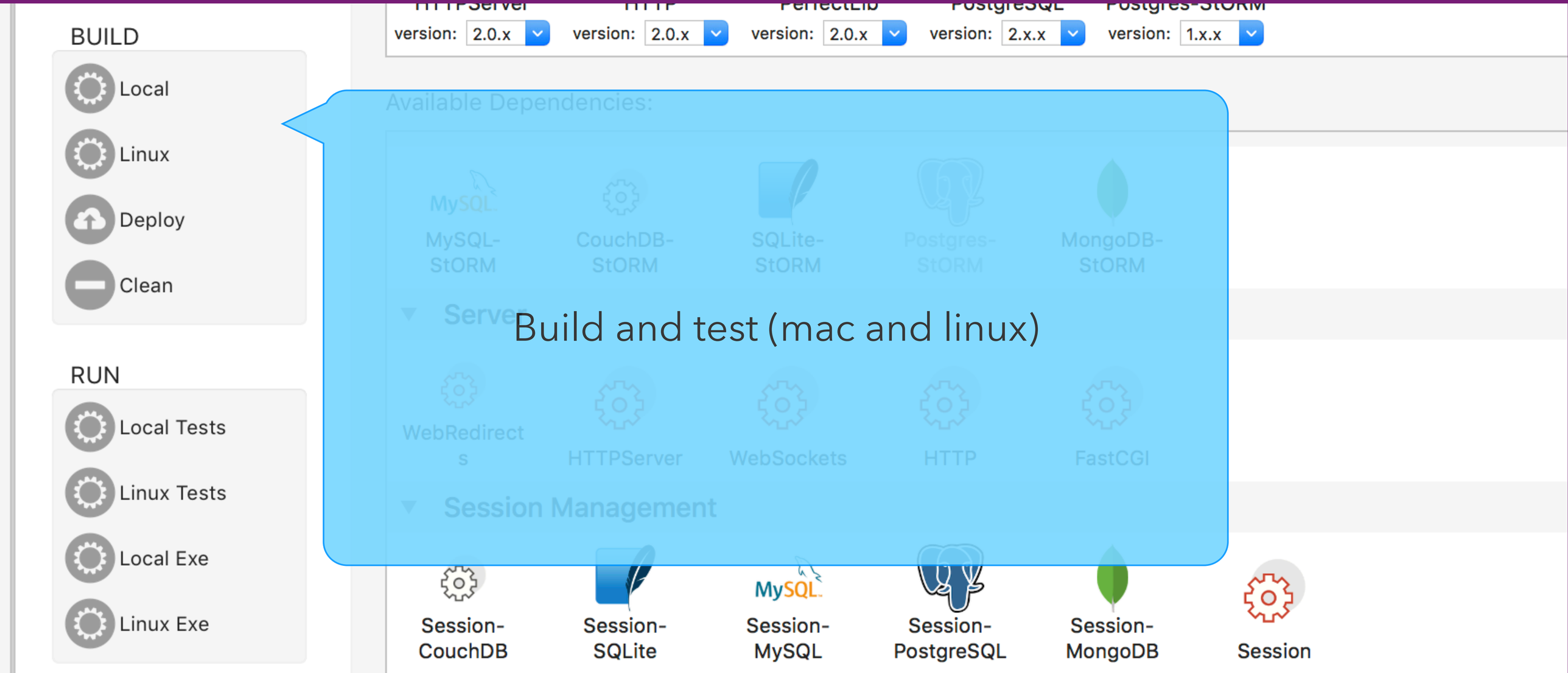
Creating /Users/tabinkowski/swift-server/perfect/perfect-template • Clear

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

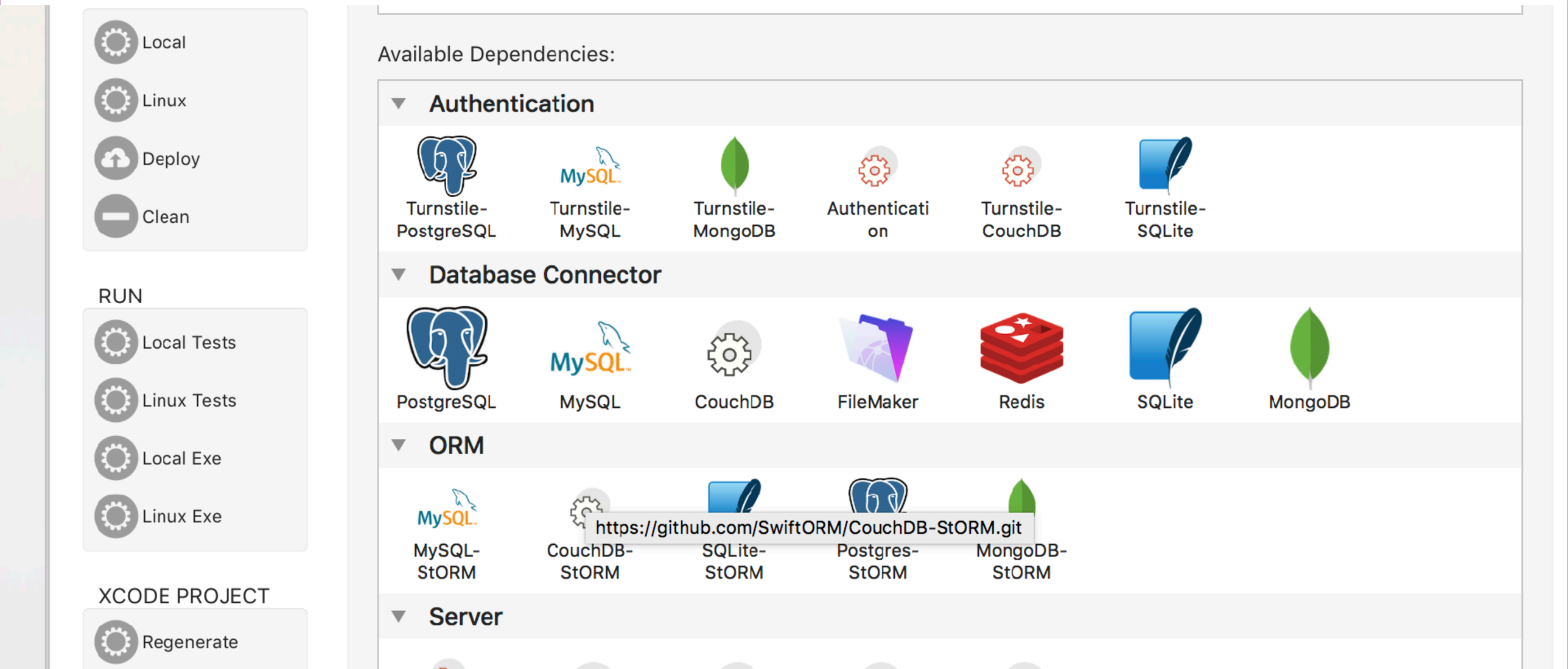
PERFECT ASSISTANT



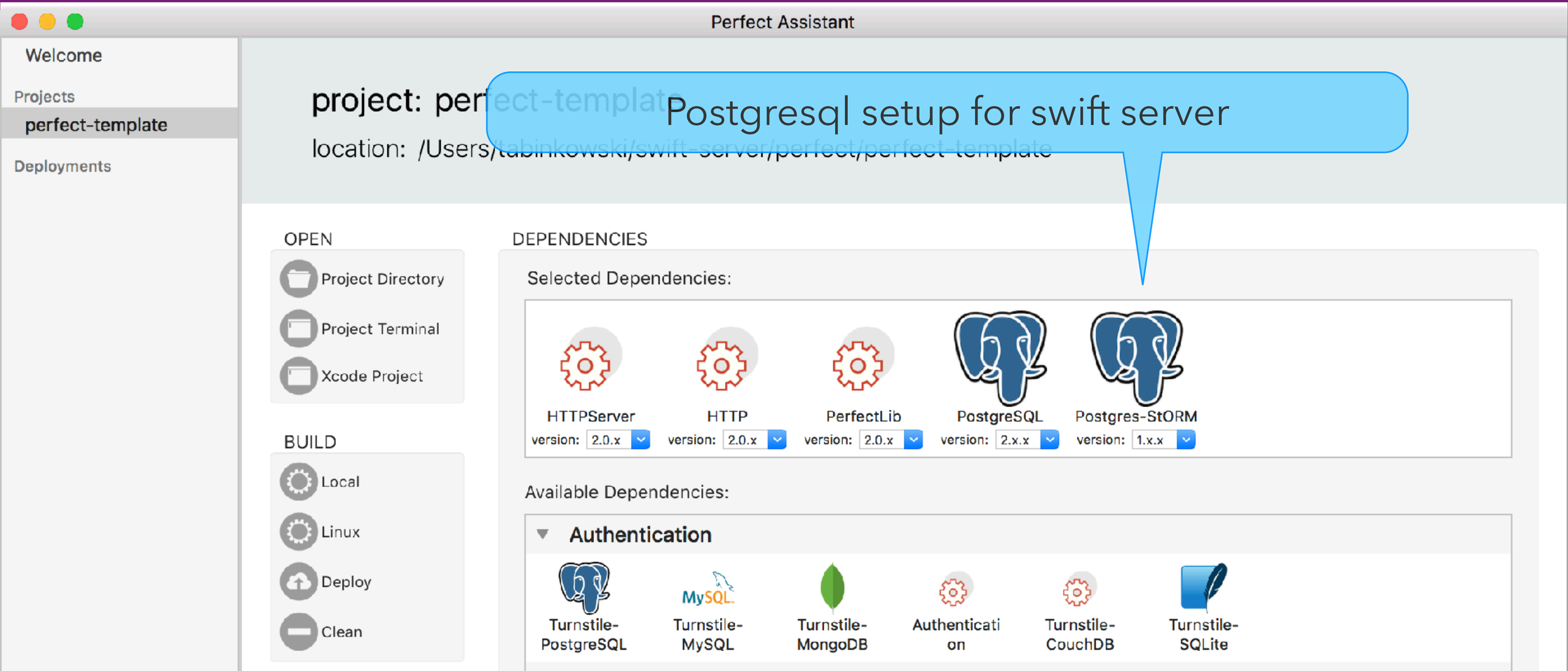
PERFECT ASSISTANT



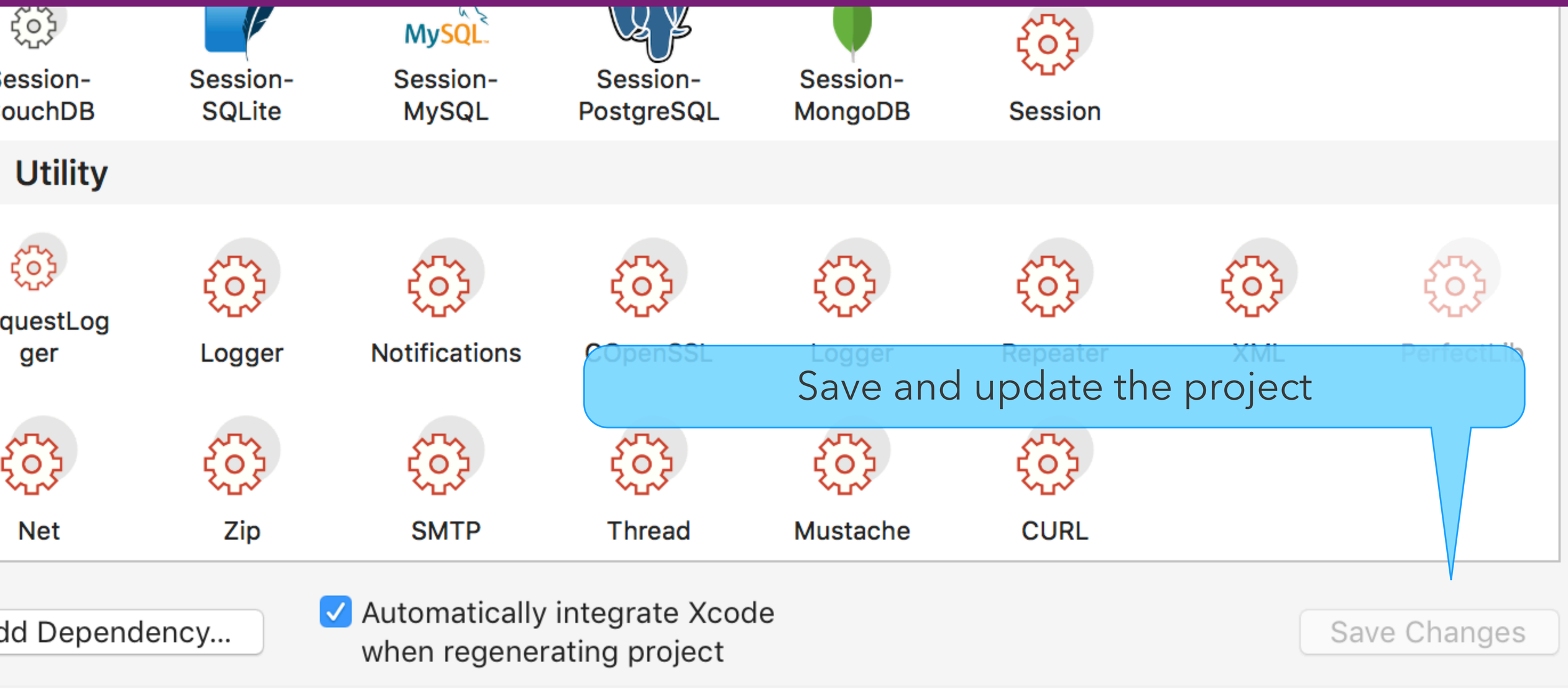
PERFECT ASSISTANT



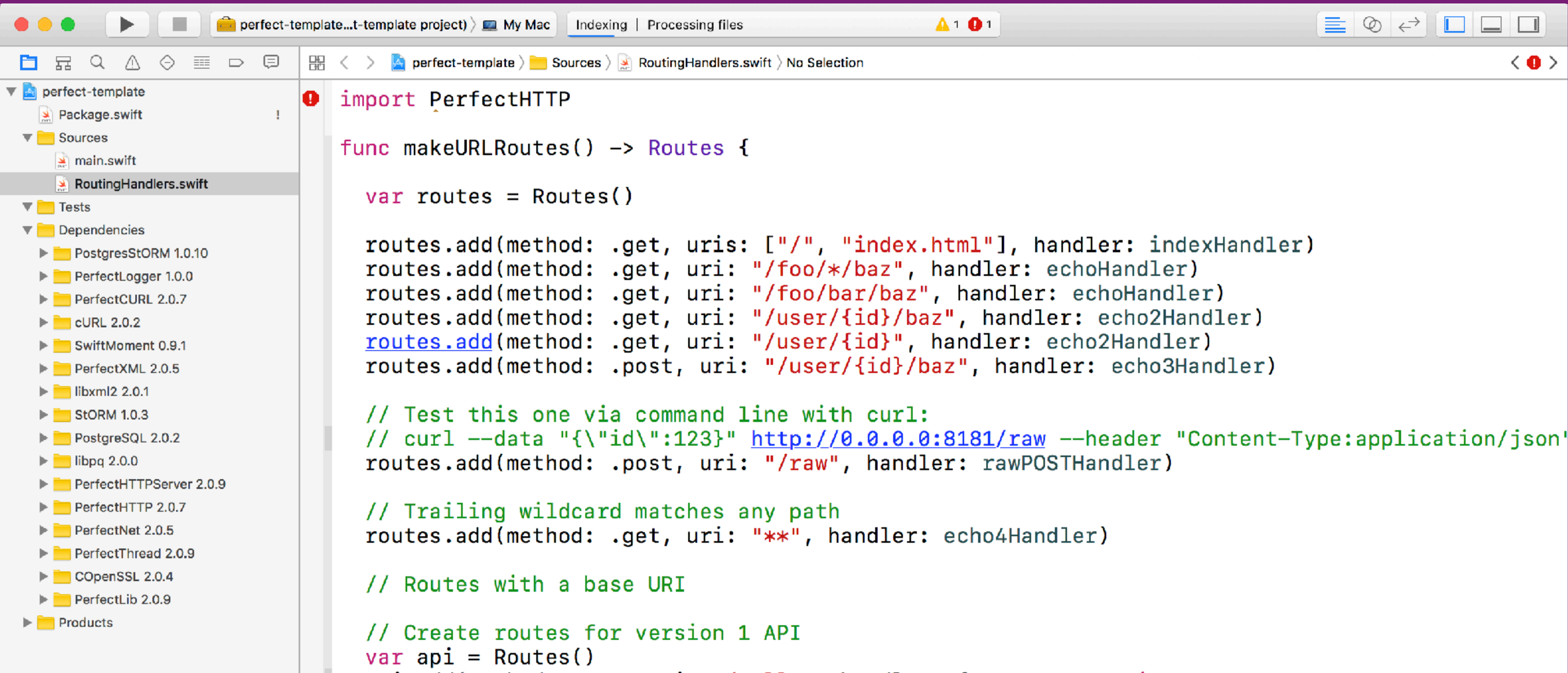
PERFECT ASSISTANT



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
  - Perfect

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

SWIFT PACKAGE MANAGER

FOR REFERENCE

SWIFT PACKAGE MANAGER

- The Swift Package Manager is a tool for managing the distribution of Swift code
- Officially part of Swift

Swift

ABOUT SWIFT
BLOG
DOWNLOAD
GETTING STARTED
DOCUMENTATION
MIGRATING TO SWIFT 4
SOURCE CODE
COMMUNITY
CONTRIBUTING
CONTINUOUS INTEGRATION
SOURCE COMPATIBILITY

FOCUS AREAS

ABI STABILITY
SERVER APIS *(WORK GROUP)*

PROJECTS

Package Manager

The Swift Package Manager is a tool for managing the distribution of Swift code. It's integrated with the Swift build system to automate the process of downloading, compiling, and linking dependencies.

The Package Manager is included in Swift 3.0 and above.

Conceptual Overview

This section describes the basic concepts that motivate the functionality of the Swift Package Manager.

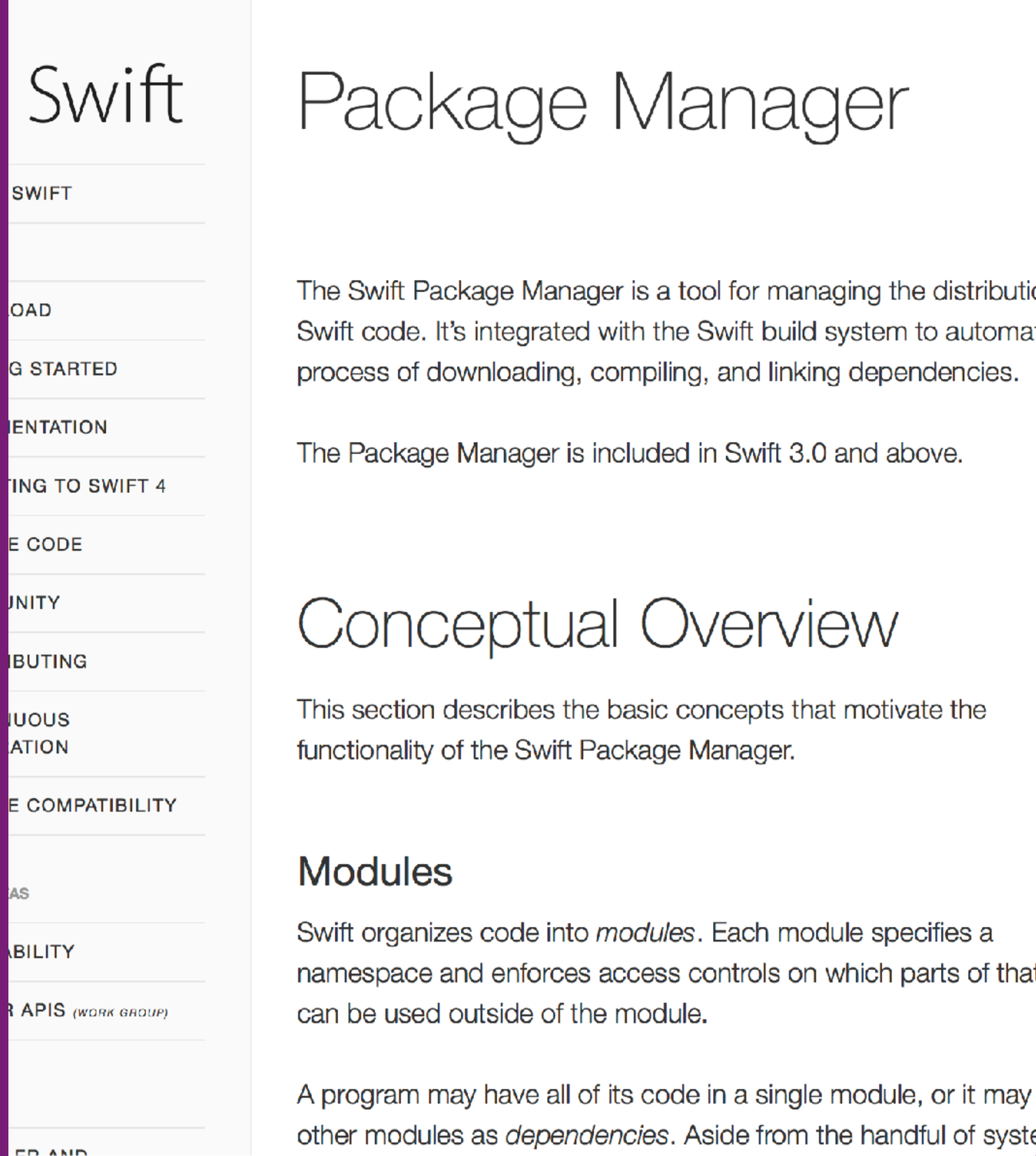
Modules

Swift organizes code into *modules*. Each module specifies a namespace and enforces access controls on which parts of that code can be used outside of the module.

A program may have all of its code in a single module, or it may import

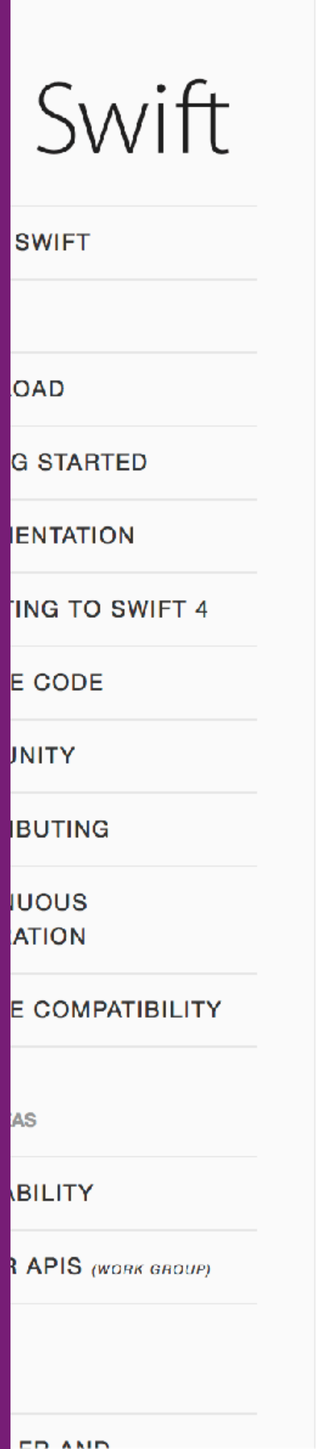
SWIFT PACKAGE MANAGER

- Consistent and convenient
  - Share and reuse code
  - Build libraries and executables
  - Manage dependendies



SWIFT PACKAGE MANAGER

- Modules
  - A group of related code
  - Share namespace and access controls
  - All of our current code has been in a single module



Package Manager

The Swift Package Manager is a tool for managing the distributi
Swift code. It's integrated with the Swift build system to automa
process of downloading, compiling, and linking dependencies.

The Package Manager is included in Swift 3.0 and above.

Conceptual Overview

This section describes the basic concepts that motivate the functionality of the Swift Package Manager.

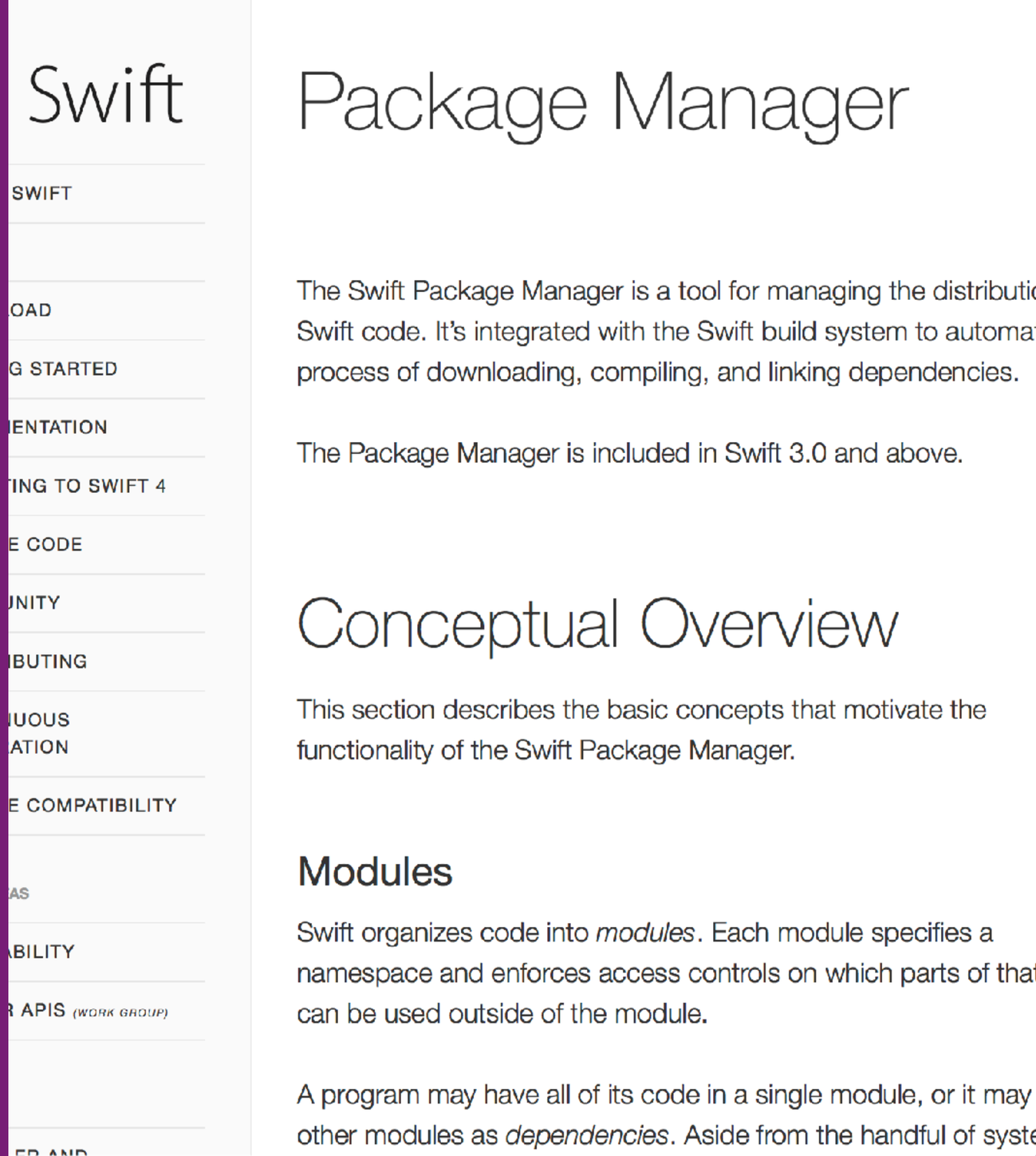
Modules

Swift organizes code into *modules*. Each module specifies a namespace and enforces access controls on which parts of tha
can be used outside of the module.

A program may have all of its code in a single module, or it may other modules as *dependencies*. Aside from the handful of syst

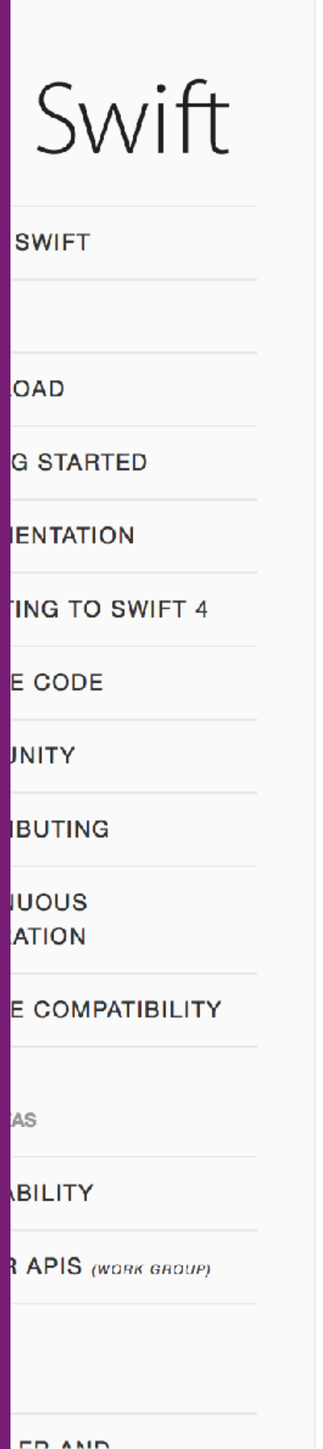
SWIFT PACKAGE MANAGER

- Packages
  - Contains source code that will be compile to a module/ library
  - Contains a manifest file `Package.swift` to define it



SWIFT PACKAGE MANAGER

- Executable
  - Package designed to run as a program
  - Contains `main.swift` file



Package Manager

The Swift Package Manager is a tool for managing the distributi
Swift code. It's integrated with the Swift build system to automa
process of downloading, compiling, and linking dependencies.

The Package Manager is included in Swift 3.0 and above.

Conceptual Overview

This section describes the basic concepts that motivate the functionality of the Swift Package Manager.

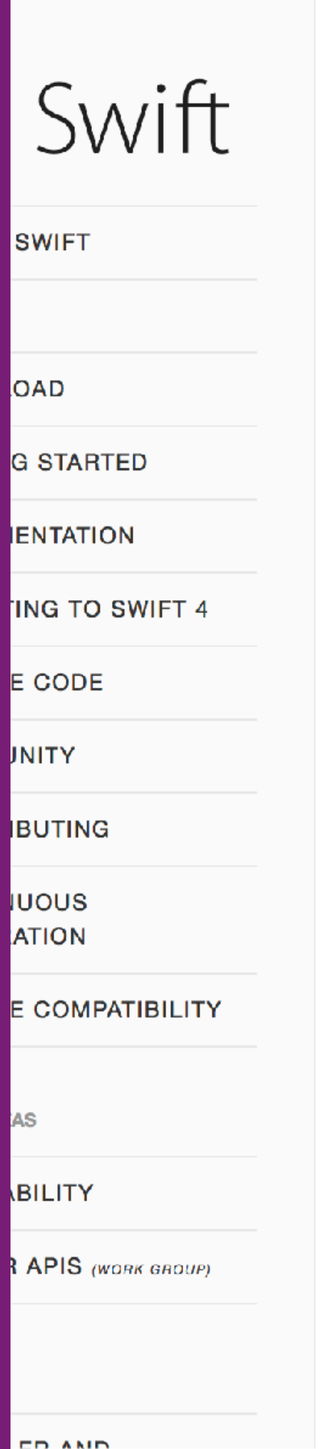
Modules

Swift organizes code into *modules*. Each module specifies a namespace and enforces access controls on which parts of tha
can be used outside of the module.

A program may have all of its code in a single module, or it may
other modules as *dependencies*. Aside from the handful of syst

SWIFT PACKAGE MANAGER

- Products
  - Target of a package or executable



Package Manager

The Swift Package Manager is a tool for managing the distributi
Swift code. It's integrated with the Swift build system to automa
process of downloading, compiling, and linking dependencies.

The Package Manager is included in Swift 3.0 and above.

Conceptual Overview

This section describes the basic concepts that motivate the functionality of the Swift Package Manager.

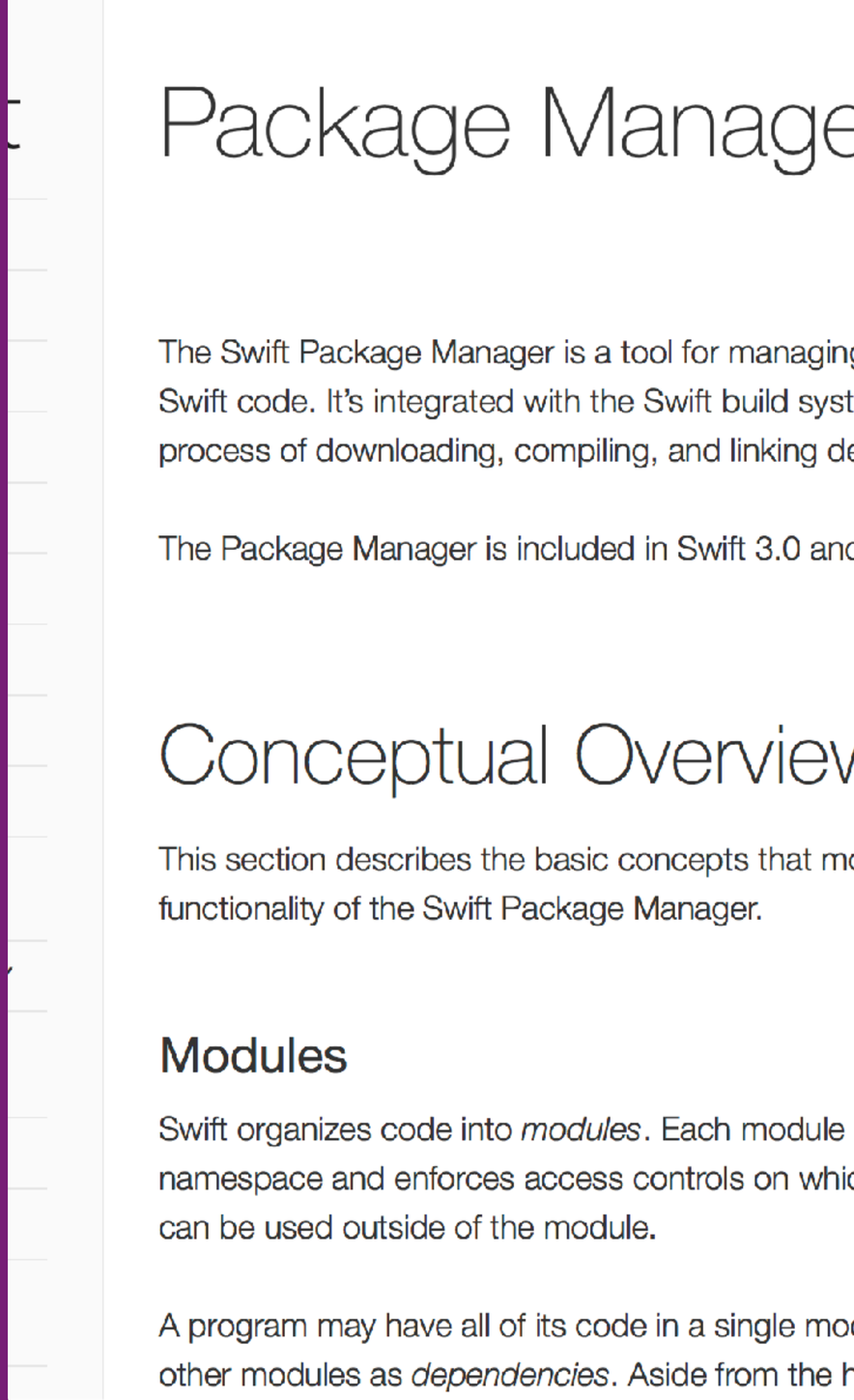
Modules

Swift organizes code into *modules*. Each module specifies a namespace and enforces access controls on which parts of tha can be used outside of the module.

A program may have all of its code in a single module, or it may other modules as *dependencies*. Aside from the handful of syst

SWIFT PACKAGE MANAGER

- Dependencies
  - Modules that are required for cod in the package
  - Not all programs will have dependencies
  - Package manager coordinates all of the dependencies of dependencies of dependencies...



Package Manage

The Swift Package Manager is a tool for managing
Swift code. It's integrated with the Swift build syst
process of downloading, compiling, and linking de

The Package Manager is included in Swift 3.0 and

Conceptual Overview

This section describes the basic concepts that mo
functionality of the Swift Package Manager.

Modules

Swift organizes code into *modules*. Each module
namespace and enforces access controls on whic
can be used outside of the module.

A program may have all of its code in a single mo
other modules as *dependencies*. Aside from the h

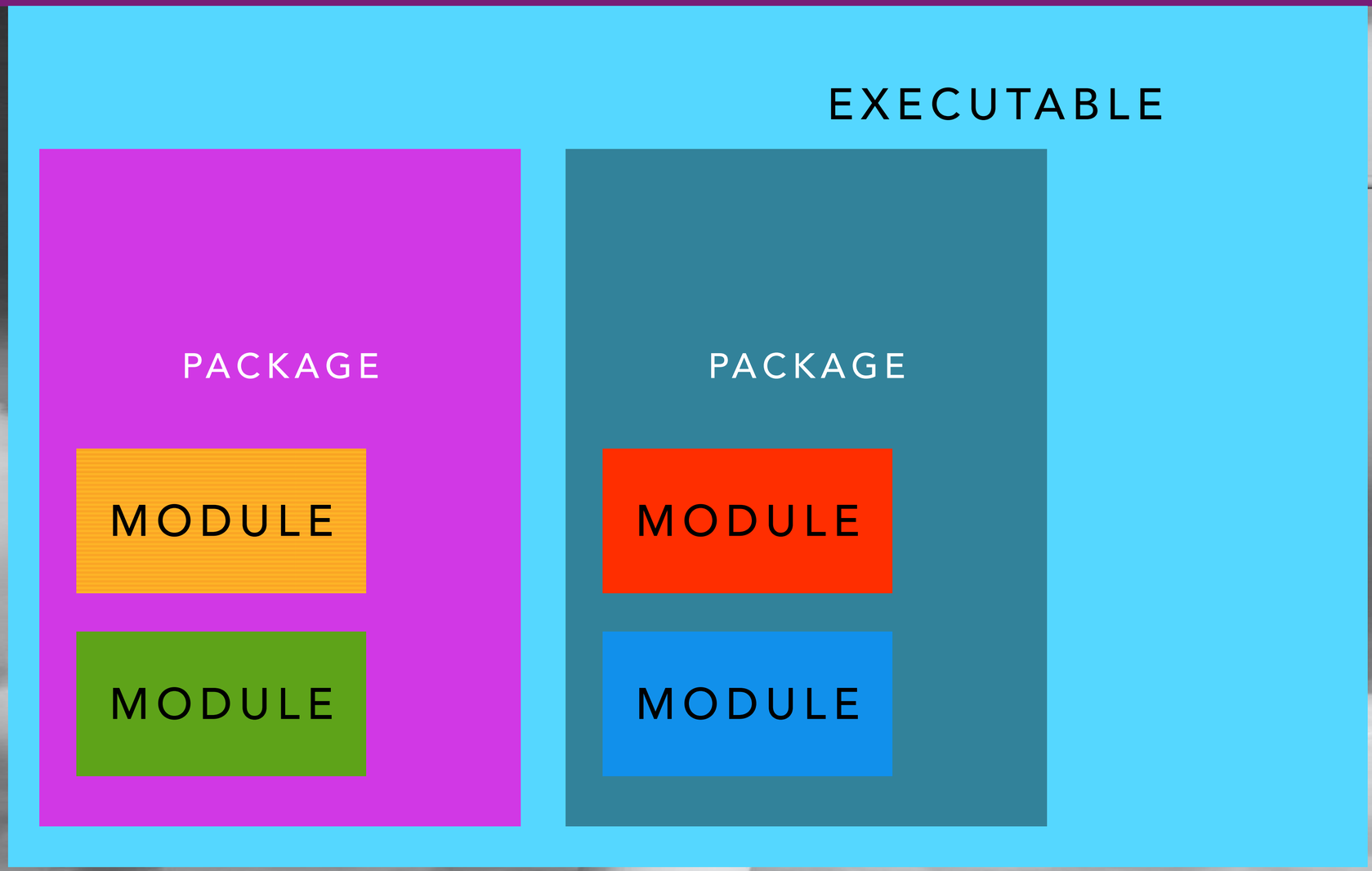
SWIFT PACKAGE MANAGER

SWIFT PACKAGE MANAGER

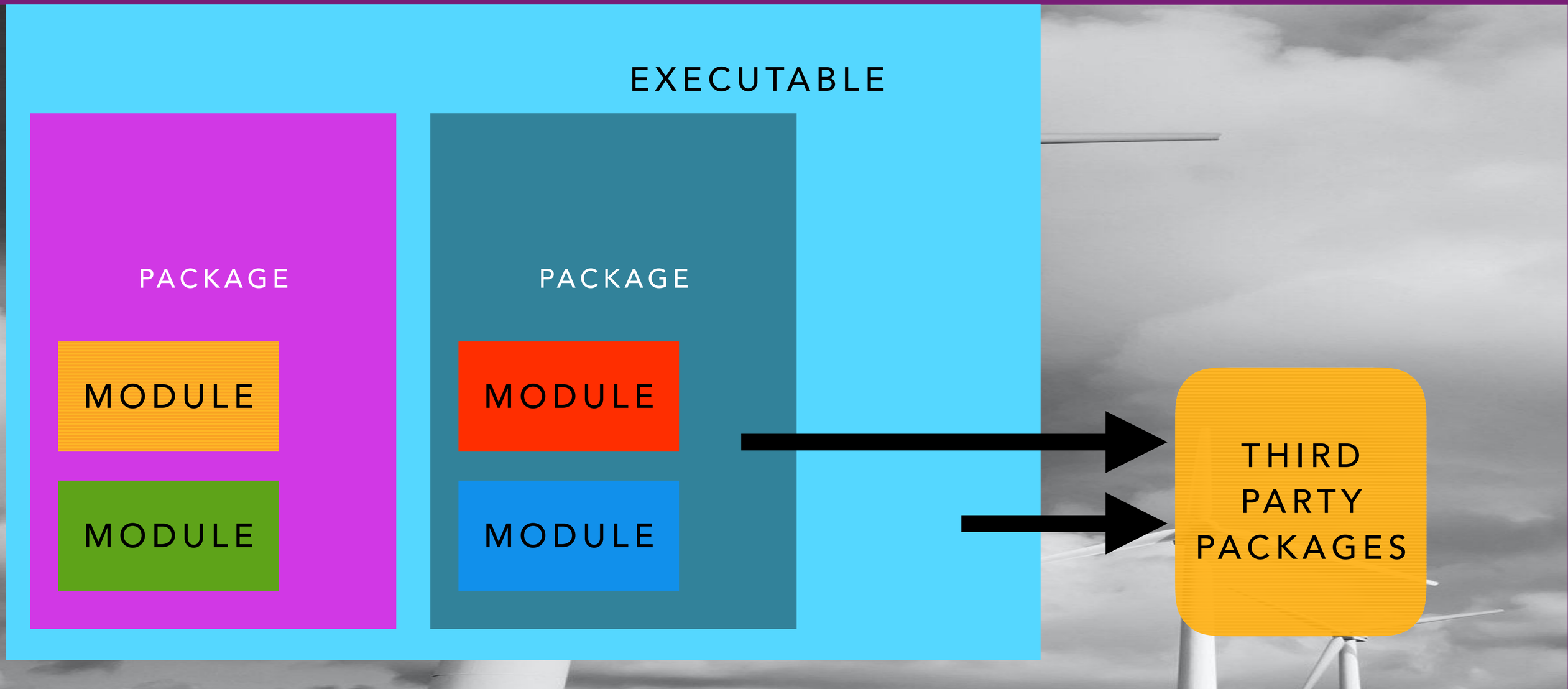
SWIFT PACKAGE MANAGER

SWIFT PACKAGE MANAGER

SWIFT PACKAGE MANAGER



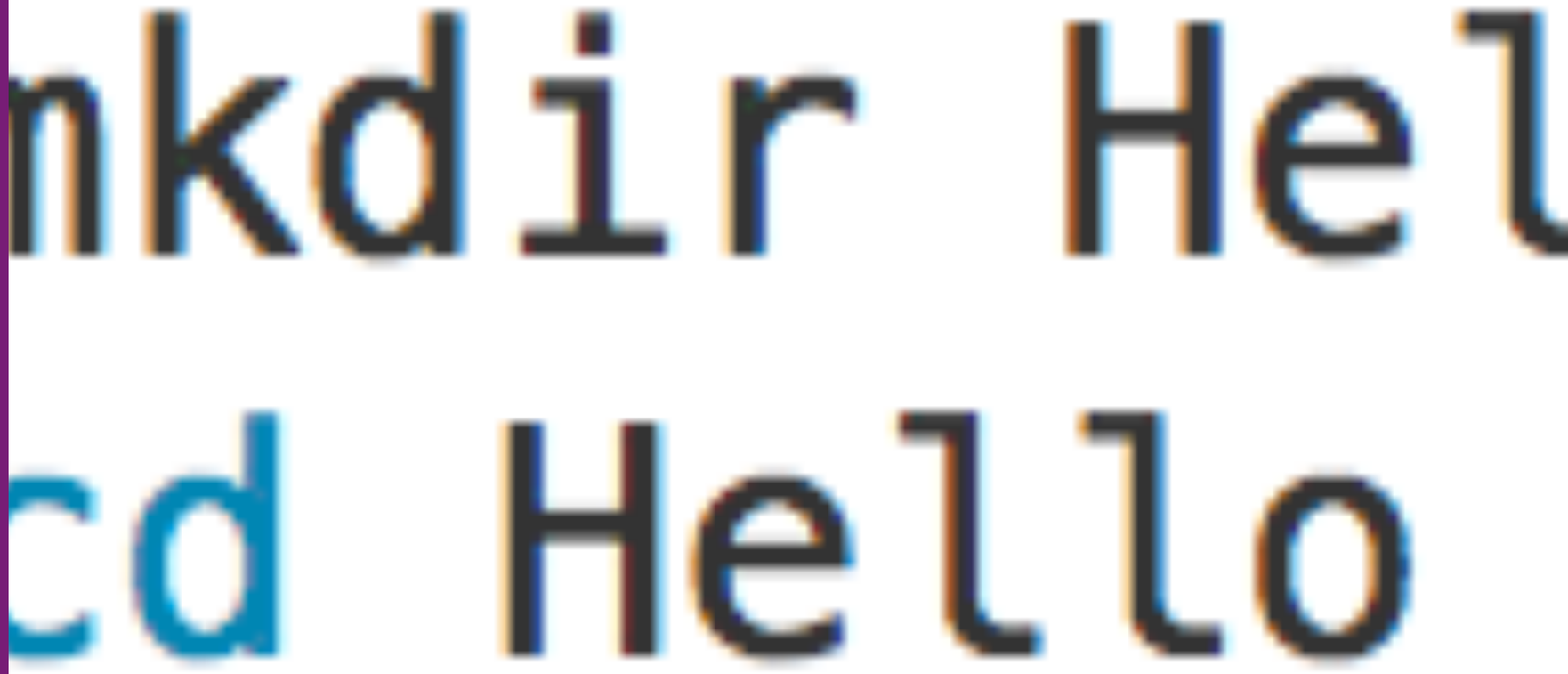
SWIFT PACKAGE MANAGER



CREATE A PACKAGE

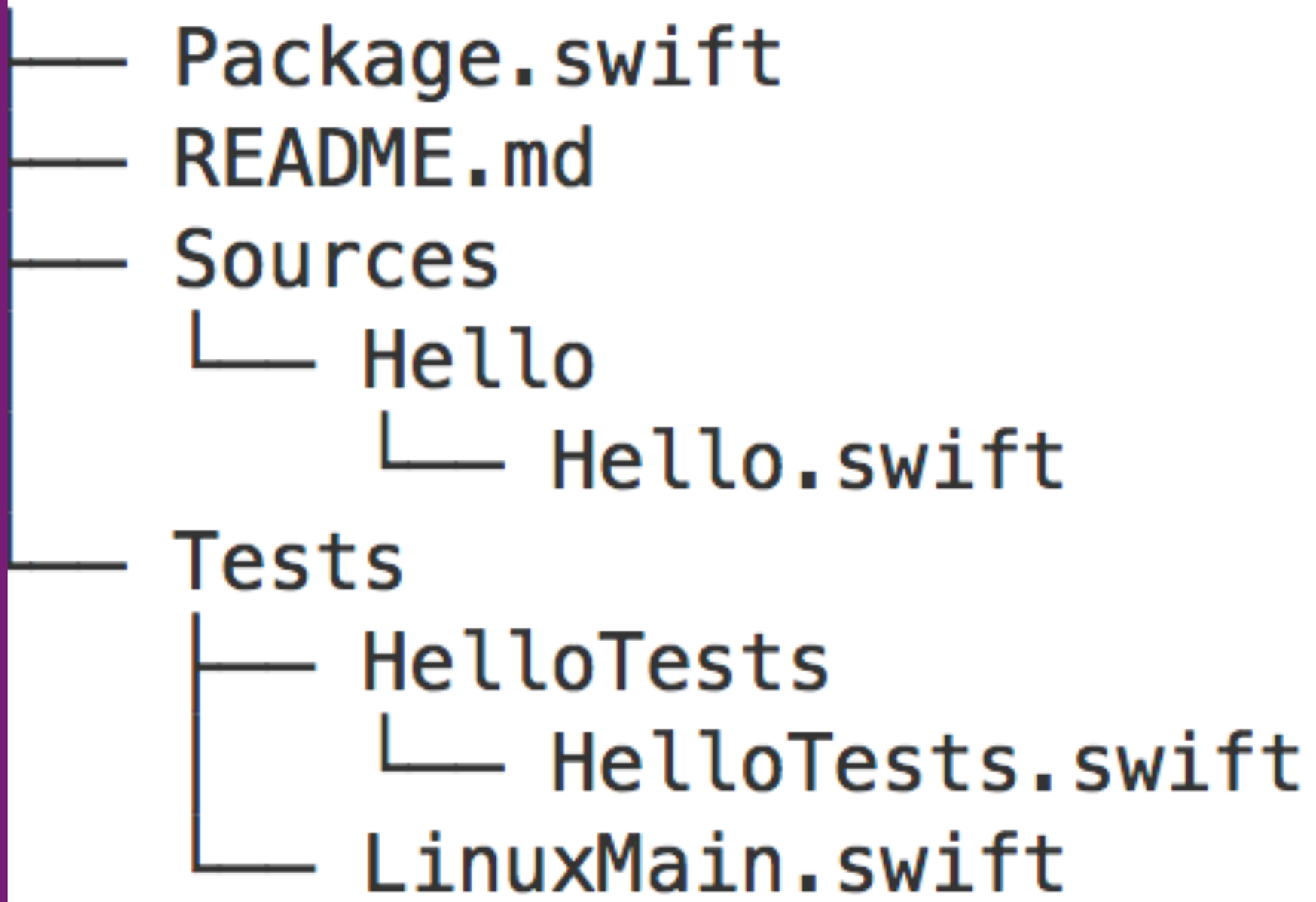
CREATE A PACKAGE

- Create a directory
- `cd` to directory
- Run `swift package init`



CREATE A PACKAGE

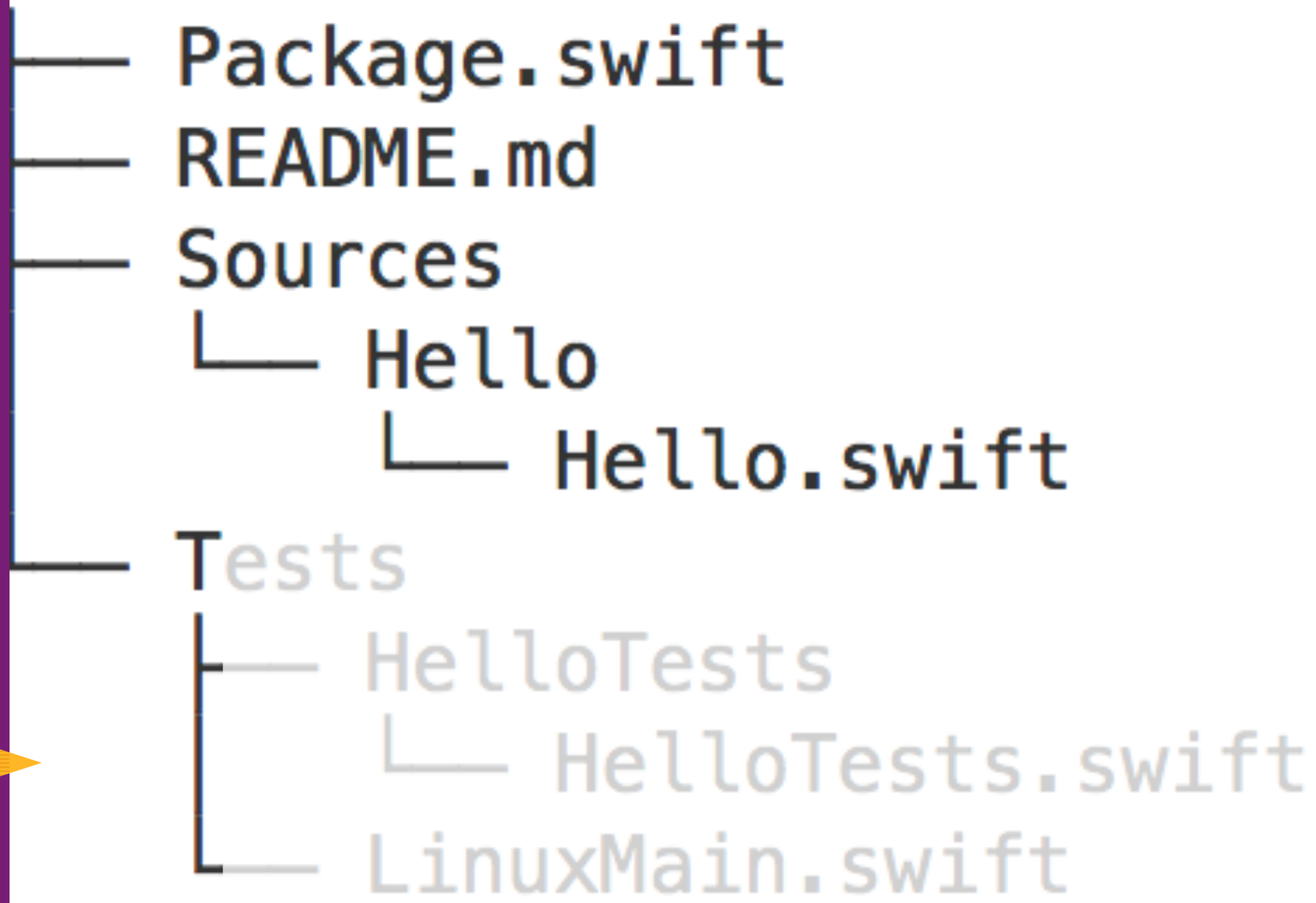
- Creates a directory/file structure
- Name of folder is name of package



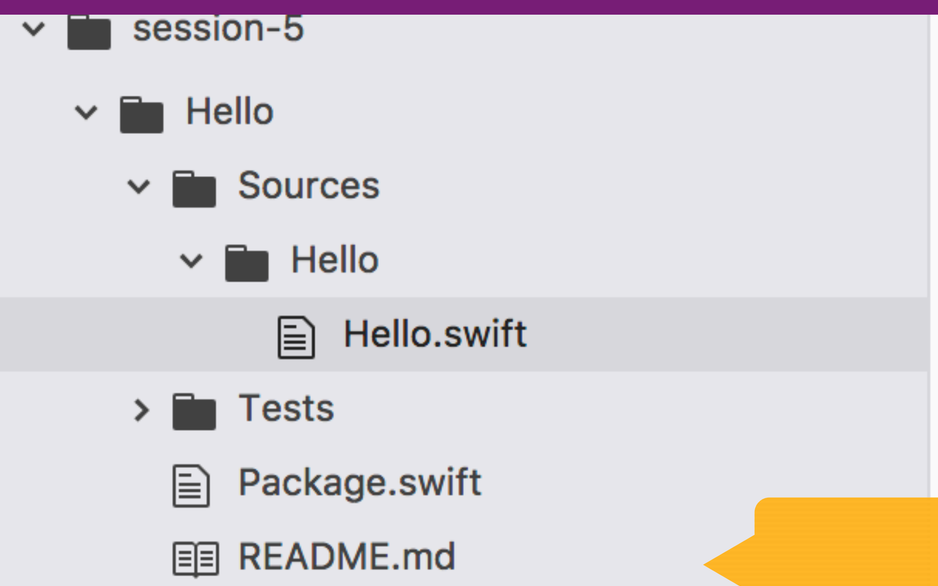
CREATE A PACKAGE

- Creates a directory/file structure
- Name of folder is name of package

IGNORE TESTS FOR NOW



CREATE A PACKAGE

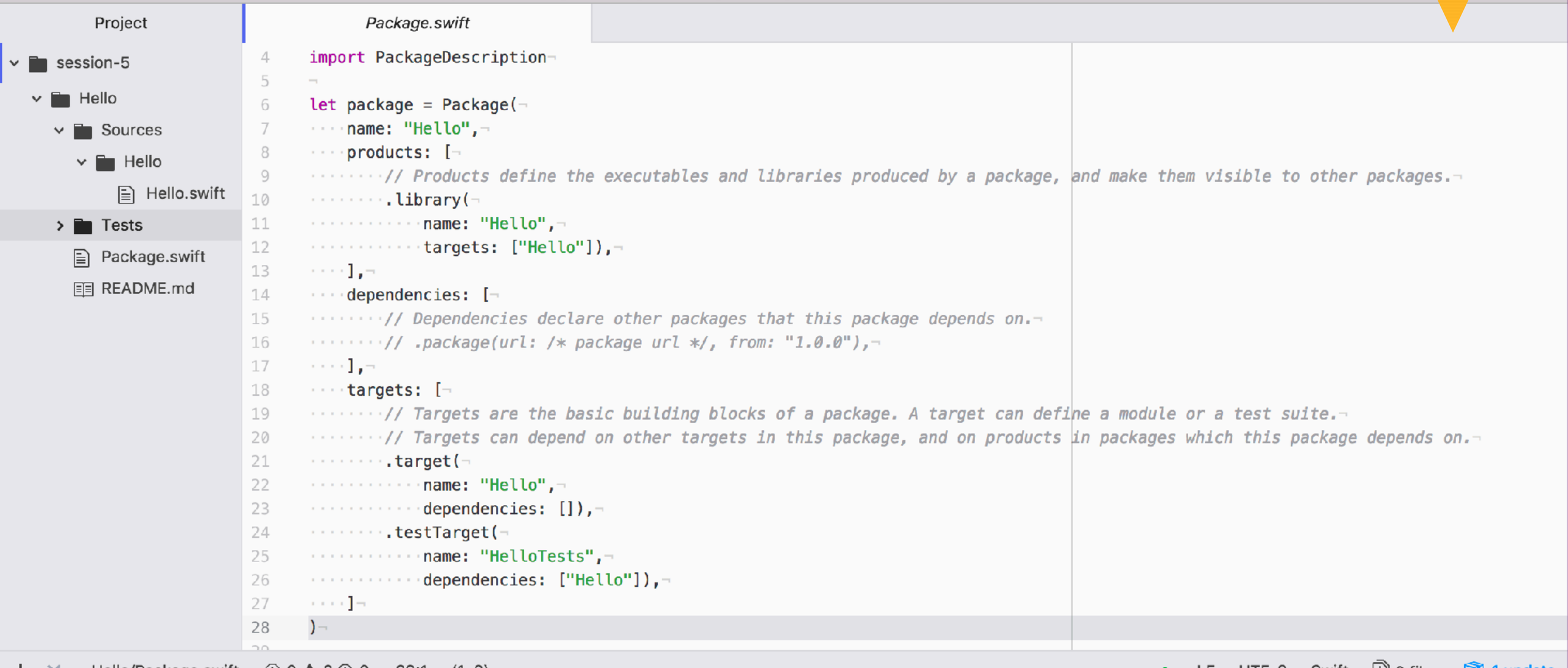



```
struct Hello {
    var text = "Hello, World!"
}

```

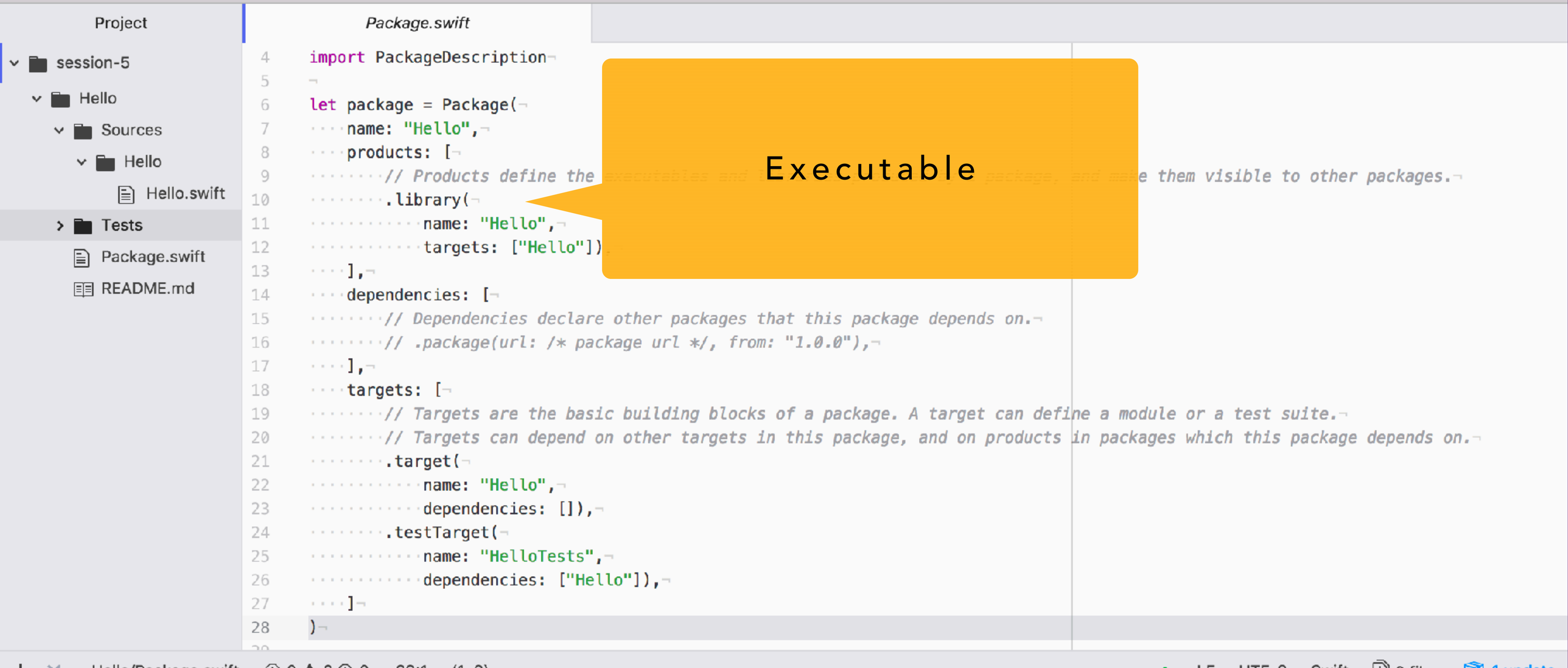
Source code for your package

CREATE A PACKAGE

Manifest file describing the package



CREATE A PACKAGE



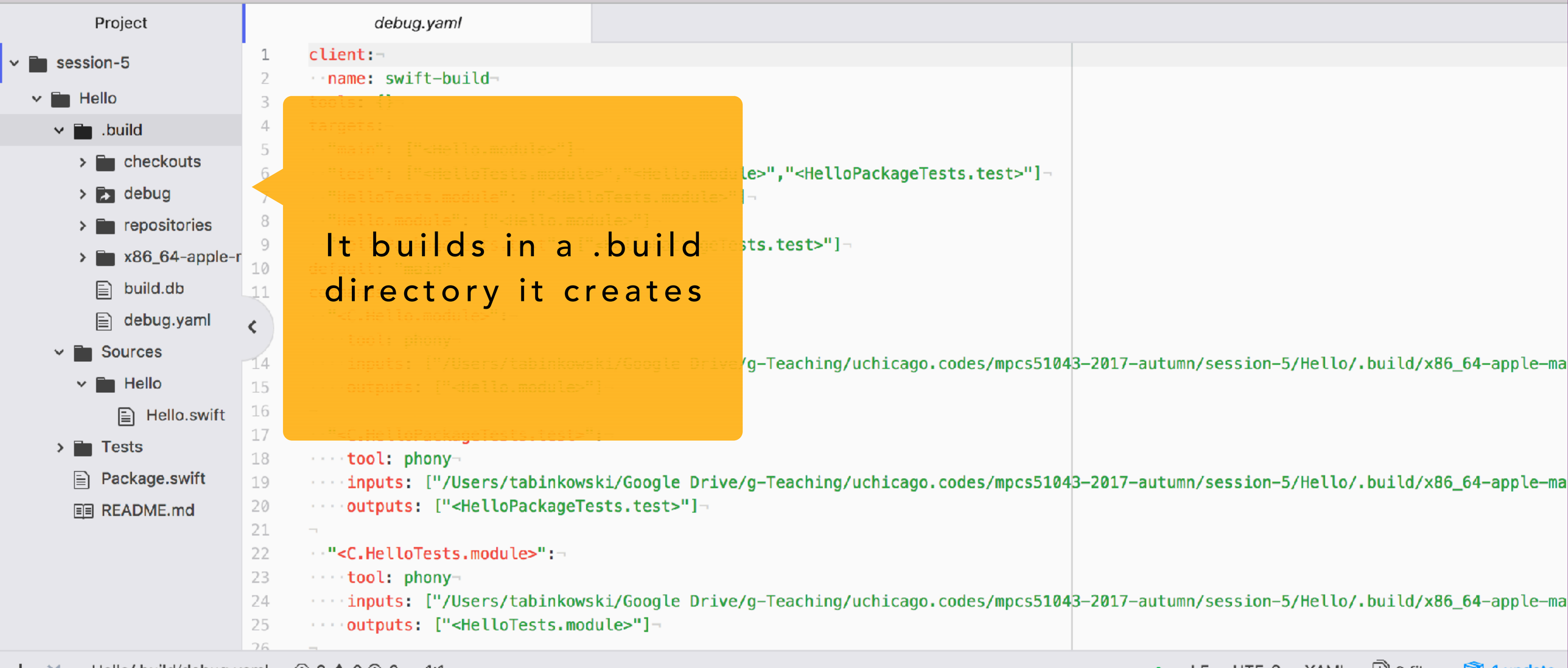
CREATE A PACKAGE

CREATE A PACKAGE

CREATE A PACKAGE

- Build a package
- Download, resolve and compile dependencies

CREATE A PACKAGE



CREATE A PACKAGE

- Packages need to be under git control
- Package management will match the tag

```
% git init
% git add .
% git commit -m "Initial "
% git tag 1.0.0
```

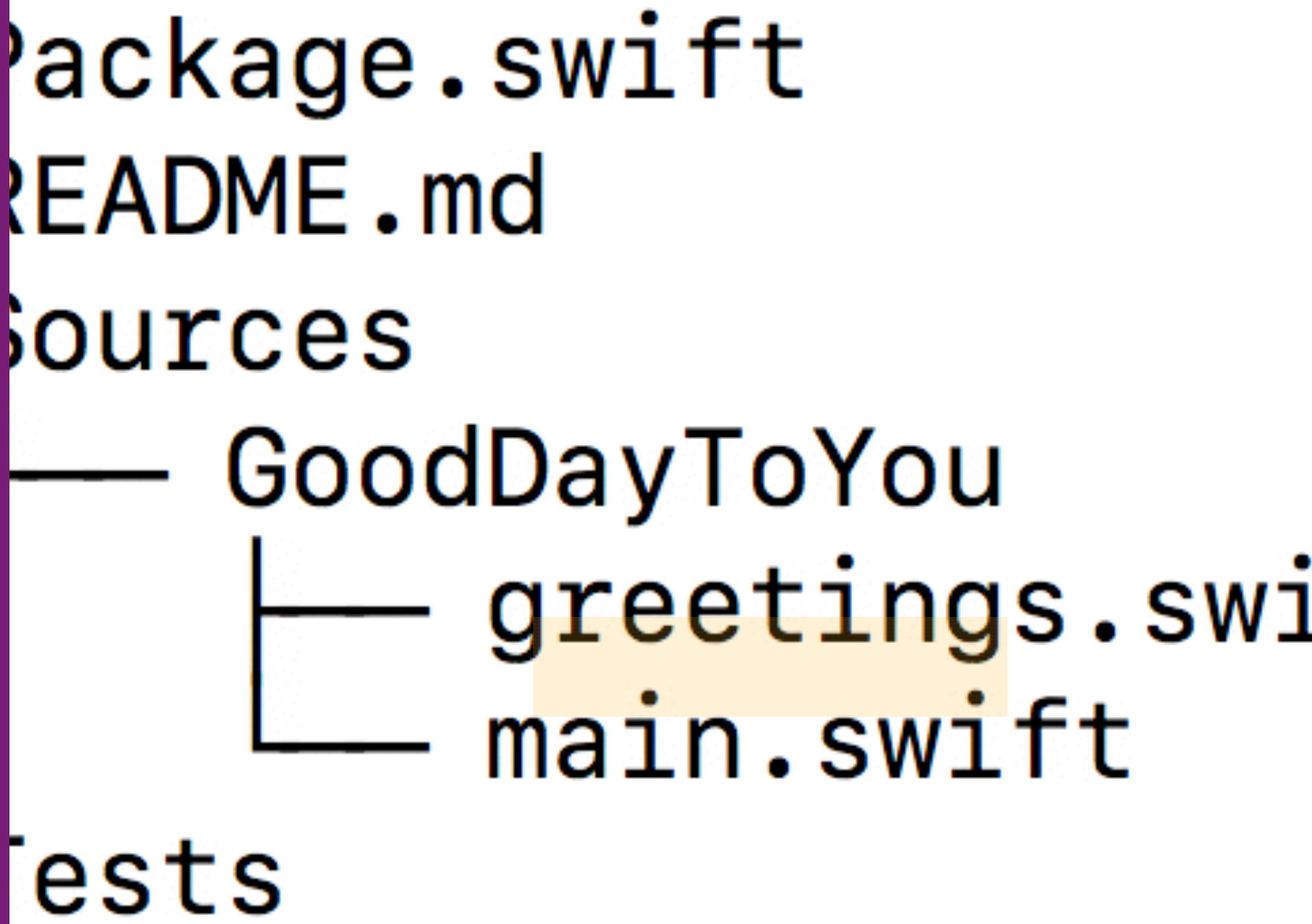
CREATE A PACKAGE

- We have a package containing code modules
- Now what? 

BUILD AN EXECUTABLE

BUILD AN EXECUTABLE

- Same structure as package except contains a main.swift file
- Entry point to run program

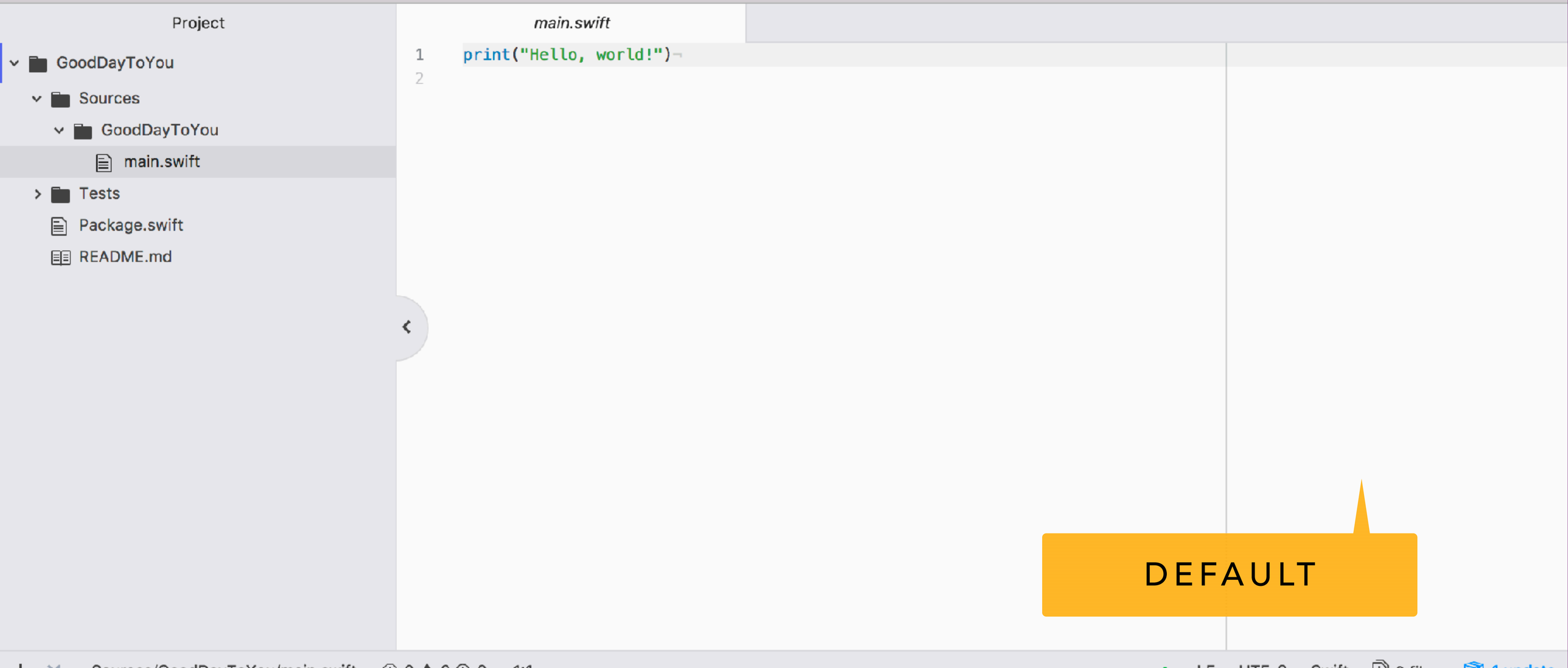


BUILD AN EXECUTABLE

- Create an executable package

```
% mkdir GoodDayToYou
% cd Hello
% swift package init --type
  executable
```

BUILD AN EXECUTABLE



BUILD AN EXECUTABLE

- Create an executable package

```
# Build it
% swift build
Compile Swift Module 'GoodDayToYou' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/
debug/GoodDayToYou

# Run it
% swift run
% swift run GoodDayToYou
```

BUILD AN EXECUTABLE

- Runs from .build directory
  - Copy it out to use it

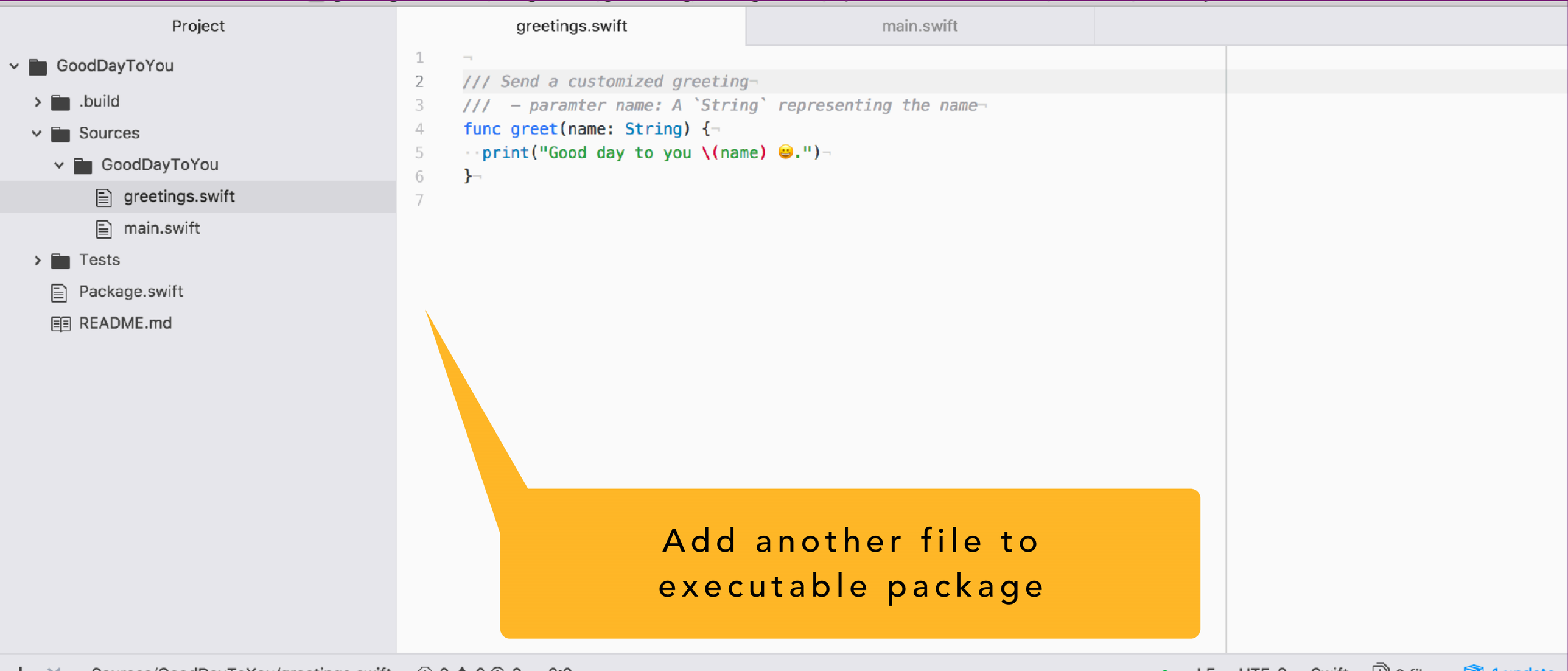
```
# Build it
% swift build
Compile Swift Module 'GoodDayToYou' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/
debug/GoodDayToYou

# Run it
% swift run
% swift run GoodDayToYou
```

This is the location of the executable

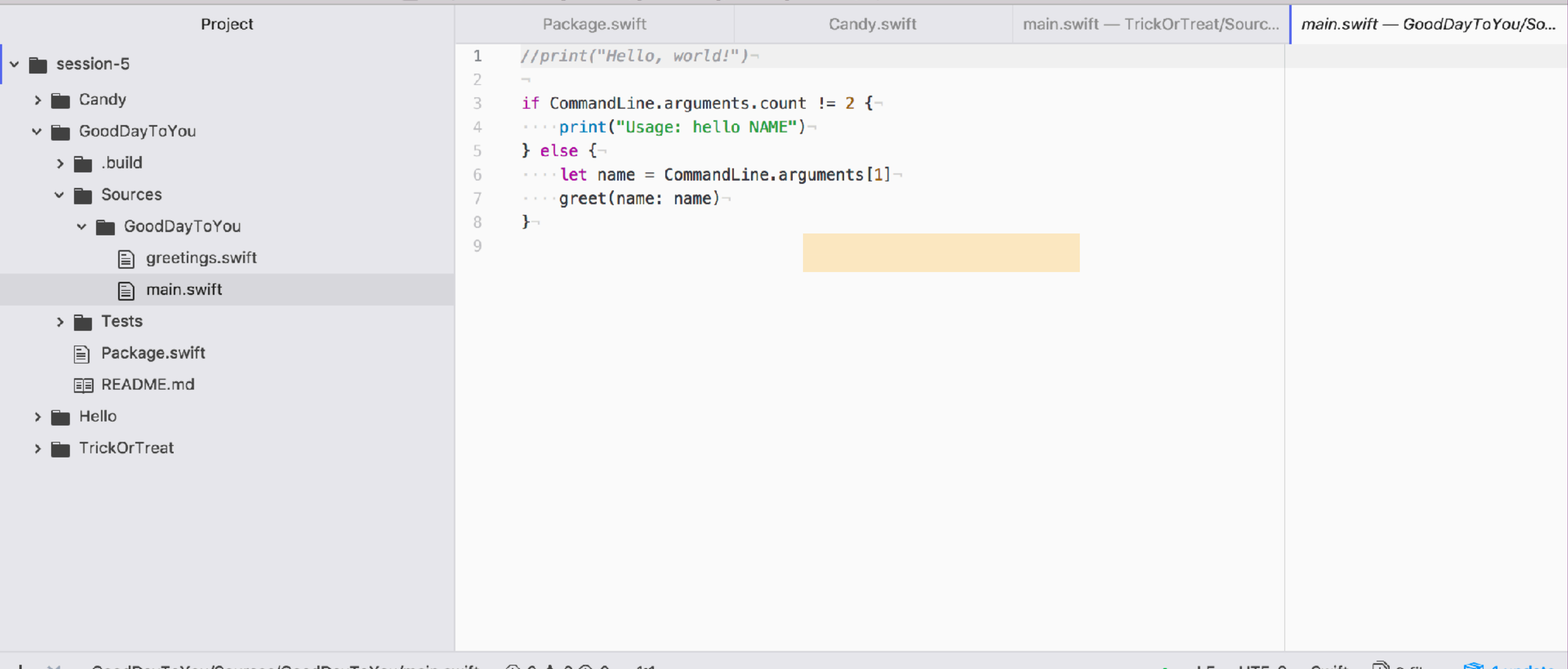
CREATE A PACKAGE

```
580 % ./.build/x86_64-apple-macosx10.10/debug/GoodDayToYou
Hello, world!
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/GoodDayToYou
581 %
```

CREATE A PACKAGE

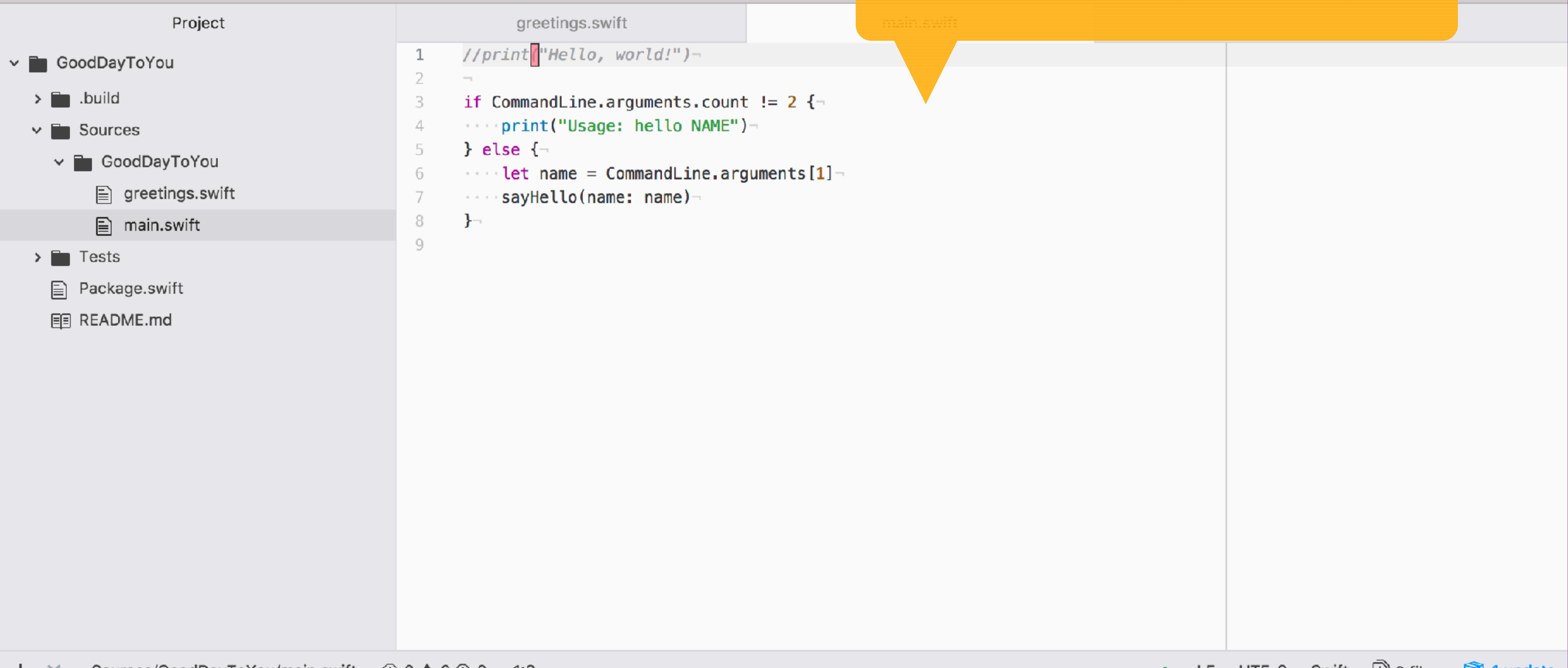


CREATE A PACKAGE



CREATE A PACKAGE

No need to import anythinging because they are from the same module



CREATE A PACKAGE

```
017-autumn/session-5/GoodDayToYou
585 % swift run
Usage: hello NAME
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2
017-autumn/session-5/GoodDayToYou
586 % swift run
Usage: hello NAME
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2
017-autumn/session-5/GoodDayToYou
587 %
```

Needed commandline argument

CREATE A PACKAGE

```
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/GoodDayToYou
589 % swift run GoodDayToYou Andrew
Good day to you Andrew 😀.
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/GoodDayToYou
590 %
```

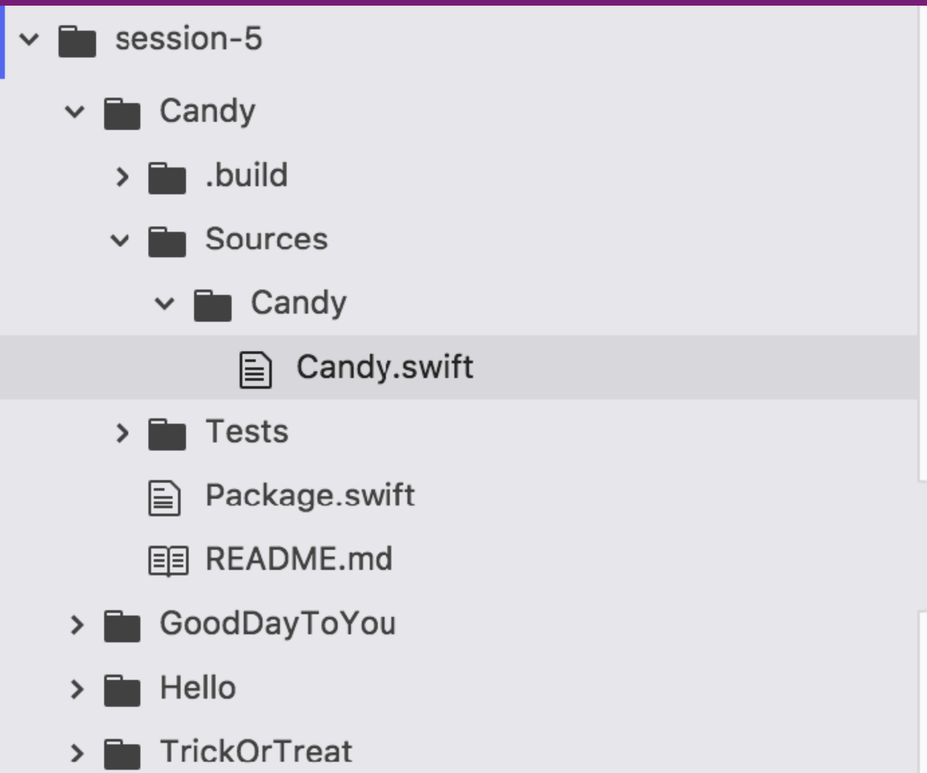
EXECUTABLES WITH PACKAGES

EXECUTABLES WITH PACKAGES

- Create an executable package that uses a dependent package
- Typical behavior

```
# Build it
% swift build
Compile Swift Module 'GoodDayToYou' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/
debug/GoodDayToYou

# Run it
% swift run
% swift run GoodDayToYou
```

EXECUTABLES WITH PACKAGES

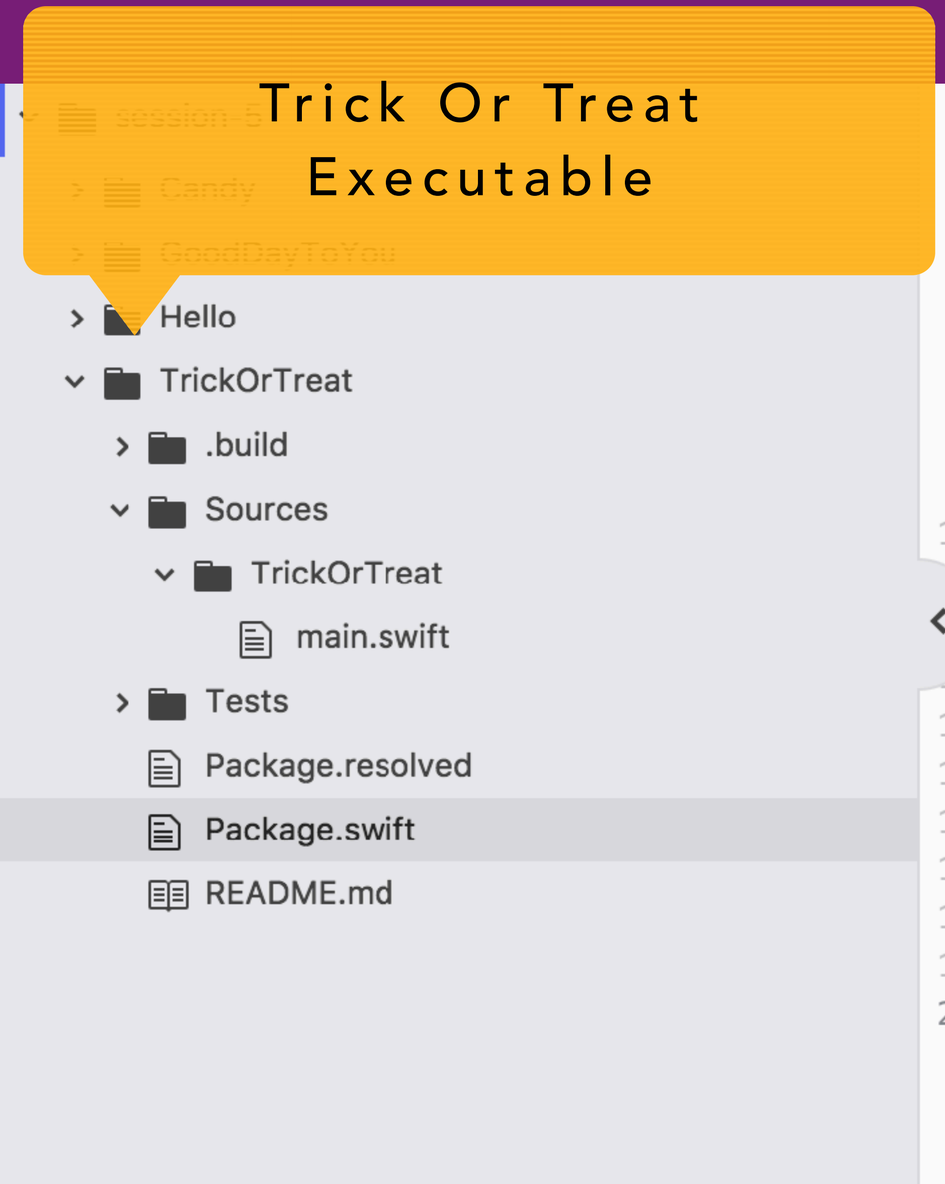



```swift
public struct Candy {
  public let name: String
  public let calories: Int
  public var eat: Bool?

  public init (name: String, calories: Int) {
    self.name = name
    self.calories = calories
  }

}
```

Candy package

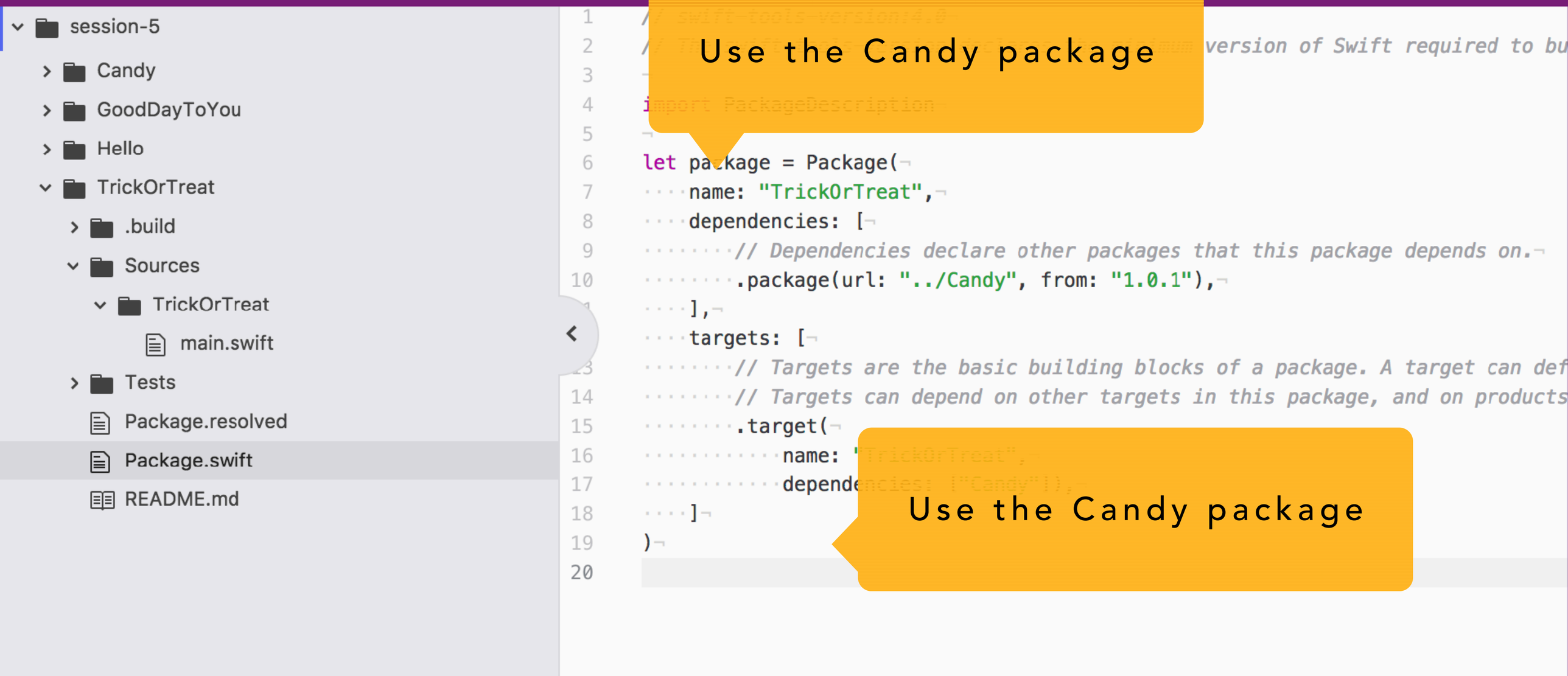
EXECUTABLES WITH PACKAGES

Trick Or Treat Executable

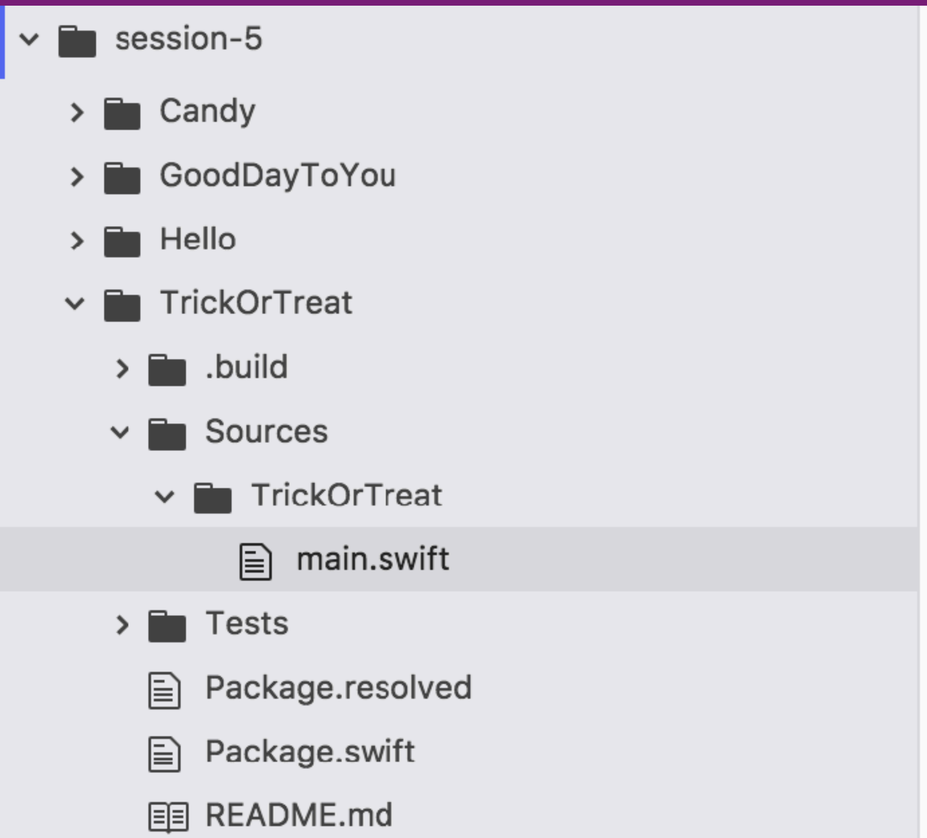



```
// swift-tools-version:4.0
// The swift-tools-version declares the minimum version of Swift required to bu

import PackageDescription

let package = Package(
    name: "TrickOrTreat",
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(url: "../Candy", from: "1.0.1"),
    ],
    targets: [
        // Targets are the basic building blocks of a package. A target can def
        // Targets can depend on other targets in this package, and on products
        .target(
            name: "TrickOrTreat",
            dependencies: ["Candy"]),
    ]
)
```

EXECUTABLES WITH PACKAGES



EXECUTABLES WITH PACKAGES




```swift
import Candy

let kitkat = Candy(name: "Kit Kat", calories: 200)

print("🎃 Trick or Treat!")
print("Have a \(kitkat.name)")
print("😃")

```

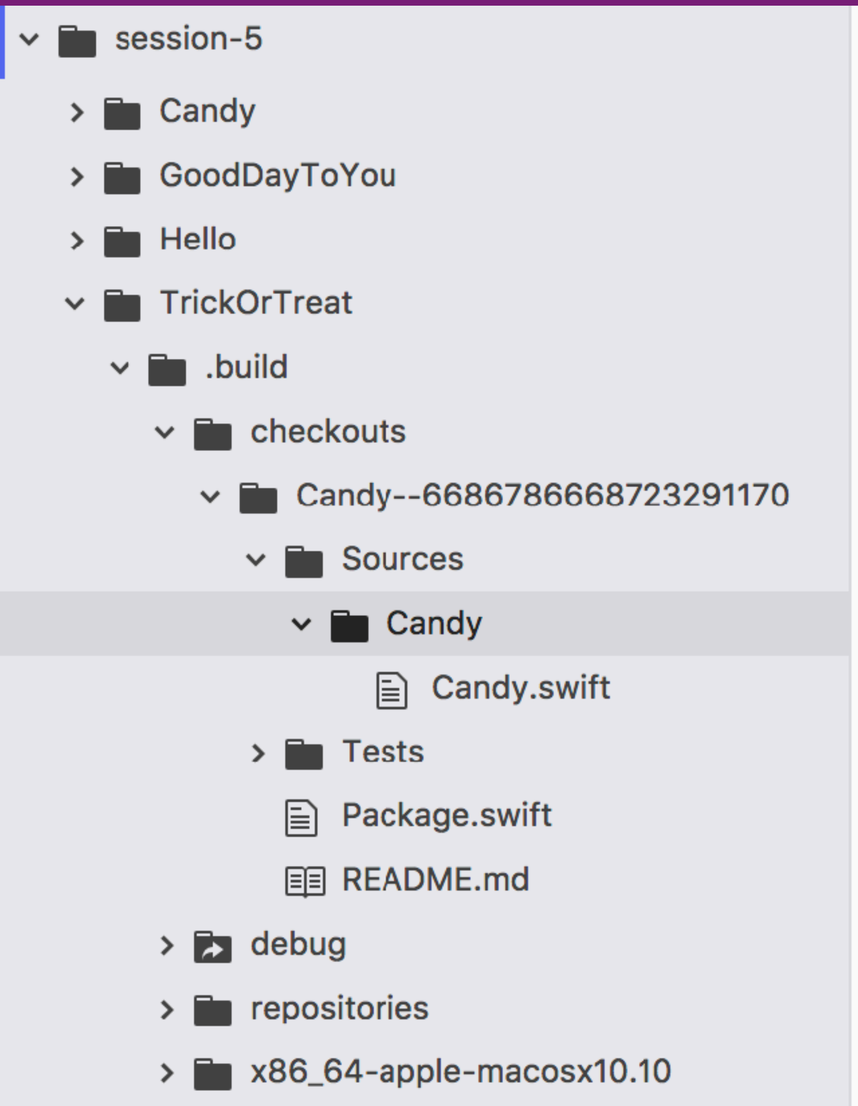
Import Candy

EXECUTABLES WITH PACKAGES

Build the executable

```
511 % swift build
Fetching /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Candy
Cloning /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Candy
Resolving /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Candy at 1.0.1
Compile Swift Module 'Candy' (1 sources)
Compile Swift Module 'TrickOrTreat' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/debug/TrickOrTreat
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/TrickOrTreat
512 %
```

EXECUTABLES WITH PACKAGES




```
// swift-tools-version:4.0
// The swift-tools-version declares the minimum version of Swift required to build

import PackageDescription

let package = Package(
    name: "Candy",
    products: [
        // Products define the executables and libraries produced by a package, and
        .library(
            name: "Candy",
            targets: ["Candy"]),
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        // .package(url: /* package url */, from: "1.0.0"),
    ],
    targets: [
        // Targets are the basic building blocks of a package. A target can define
        // Targets can depend on other targets in this package, and on products in
        .target(
            name: "Candy",
            dependencies: []),
```

WITH XCODE

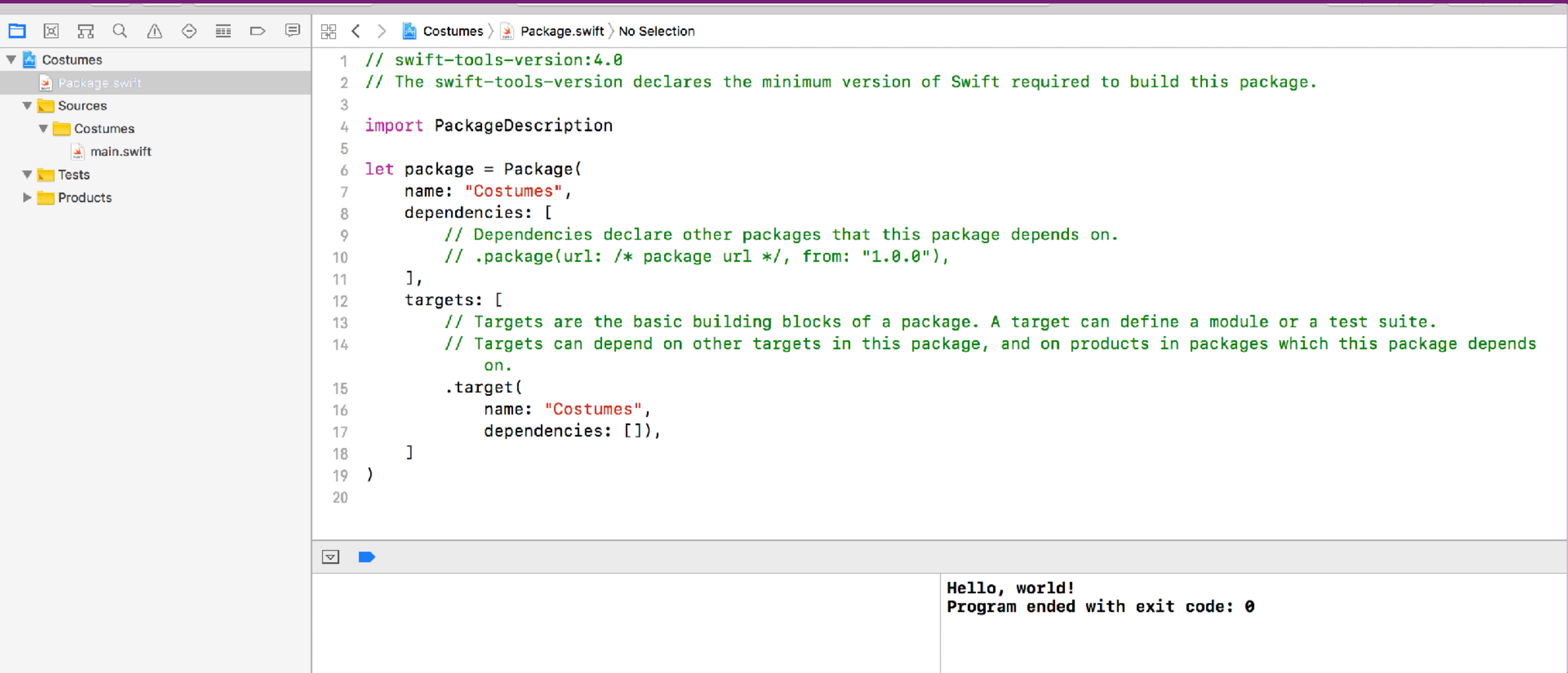
WITH XCODE

Costumes — -bash — 93×14

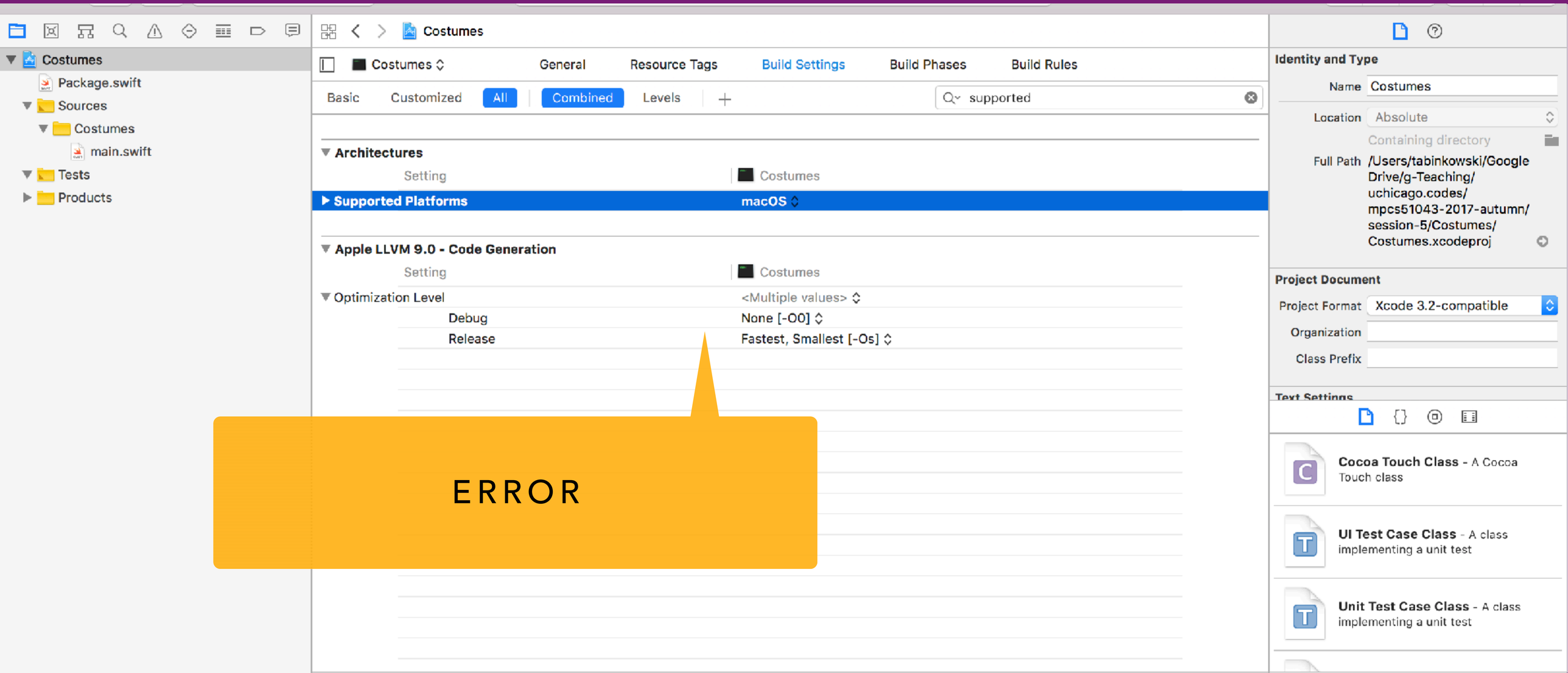
es/firebase/functions — -bash | ...1033-2017-autumn-playground — -bash | ...mn/session-5/GoodDayToYou — -bash | ...-autumn/session-5/Costumes

```
wift package init --type=executable
g executable package: Costumes
g Package.swift
g README.md
g Sources/
g Sources/Costumes/main.swift
g Tests/
wski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-aut
Costumes
wift package generate-xcodeproj
ed: ./Costumes.xcodeproj
wski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-aut
Costumes
```

WITH XCODE


Costumes
Package.swift
No Selection
Costumes
Package.swift
Sources
Costumes
main.swift
Tests
Products
// swift-tools-version:4.0
// The swift-tools-version declares the minimum version of Swift required to build this package.
import PackageDescription
let package = Package(
name: "Costumes",
dependencies: [
// Dependencies declare other packages that this package depends on.
// .package(url: /* package url */, from: "1.0.0"),
],
targets: [
// Targets are the basic building blocks of a package. A target can define a module or a test suite.
// Targets can depend on other targets in this package, and on products in packages which this package depends on.
.target(
name: "Costumes",
dependencies: []),
]
)
Hello, world!
Program ended with exit code: 0

WITH XCODE


Costumes
Package.swift
Sources
Costumes
main.swift
Tests
Products
Costumes
General
Resource Tags
Build Settings
Build Phases
Build Rules
Basic
Customized
All
Combined
Levels
supported
Architectures
Setting
Costumes
Supported Platforms
macOS
Apple LLVM 9.0 - Code Generation
Setting
Costumes
Optimization Level
<Multiple values>
Debug
None [-O0]
Release
Fastest, Smallest [-Os]
ERROR
Identity and Type
Name
Costumes
Location
Absolute
Containing directory
Full Path
/Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Costumes/Costumes.xcodeproj
Project Document
Project Format
Xcode 3.2-compatible
Organization
Class Prefix
Text Settings
Cocoa Touch Class - A Cocoa Touch class
UI Test Case Class - A class implementing a unit test
Unit Test Case Class - A class implementing a unit test

THIRD PARTY PACKAGES

THIRD PARTY PACKAGES

build passing | pod v4.5.1 | Carthage compatible | platform ios | osx | tvos | watchos | twitter @AlamofireSF | chat on gitter

Alamofire is an HTTP networking library written in Swift.

- Features
- Component Libraries
- Requirements
- Migration Guides
- Communication
- Installation
- Usage
  - **Intro -** Making a Request, Response Handling, Response Validation, Response Caching
  - **HTTP -** HTTP Methods, Parameter Encoding, HTTP Headers, Authentication

THIRD PARTY PACKAGES

```
$ pod install
```

Carthage

Carthage is a decentralized dependency manager that builds your dependencies and provides you with binary frameworks.

You can install Carthage with Homebrew using the following command:

```
$ brew update
$ brew install carthage
```

To integrate Alamofire into your Xcode project using Carthage, specify it in your `Cartfile`:

```
github "Alamofire/Alamofire" ~> 4.5
```

Run `carthage update` to build the framework and drag the built `Alamofire.framework` into your Xcode project.

THIRD PARTY PACKAGES

THIRD PARTY PACKAGES

awesome-lists ios linux

2,591 commits **2** branches **0** releases **387** contributors CC0-1.0

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

READMEbot [auto] [ci skip] Generate README & Database.json Latest commit edc8e53 a day ago

.github	Update Hi.swift	16 days ago
.gitignore	fix push script, now disabled	4 months ago
.travis.yml	re-ordered scripts	4 months ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	4 months ago
Dangerfile	Create Dangerfile	a month ago
LICENSE	Initial commit	3 years ago
README.md	[auto] [ci skip] Generate README & Database.json	a day ago
contents.json	Merge pull request #1063 from eneko/feature/add-sourcedocs	a day ago
database.json	[auto] [ci skip] Generate README & Database.json	a day ago

README.md

THIRD PARTY PACKAGES

Code | Issues 2 | Pull requests 6 | Projects 0 | Wiki | Insights

A collection of functions for statistical calculation written in Swift.

statistics swift

344 commits | **4** branches | **26** releases | **3** contributors | MIT

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

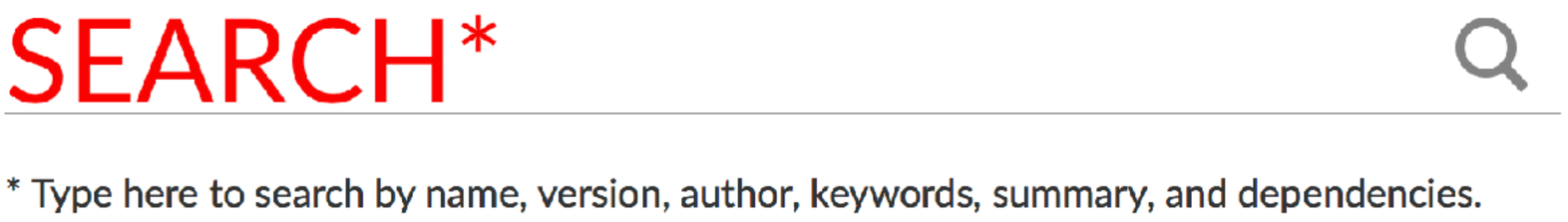

Evgenii Neumerzhitckii Update pod version — Latest commit f4ad79c 28 days ago

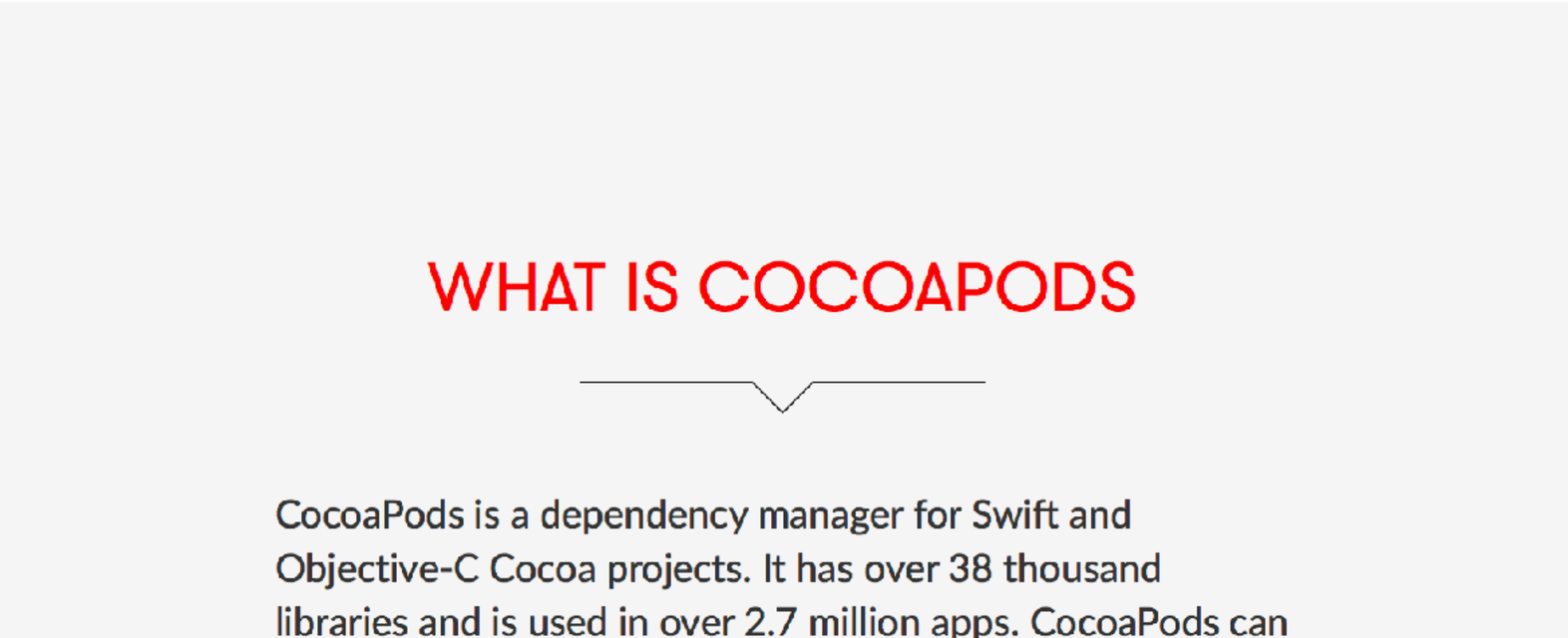
Distrib	Update to Swift 3.1	7 months ago
Graphics	Add kurtosis	10 months ago
SigmaSwiftStatistics-Mac	Update to swift 2.0	2 years ago
SigmaSwiftStatistics-Watch	Add target for Watch	2 years ago
SigmaSwiftStatistics-tvOS	Add watchOS support	2 years ago
SigmaSwiftStatistics.xcodeproj	Update to Swift 4.0	5 months ago
SigmaSwiftStatistics	Update to Swift 3.1	7 months ago
SigmaSwiftStatisticsTests	Add rank function	9 months ago

OTHER PACKAGE MANAGERS

OTHER PACKAGE MANAGERS

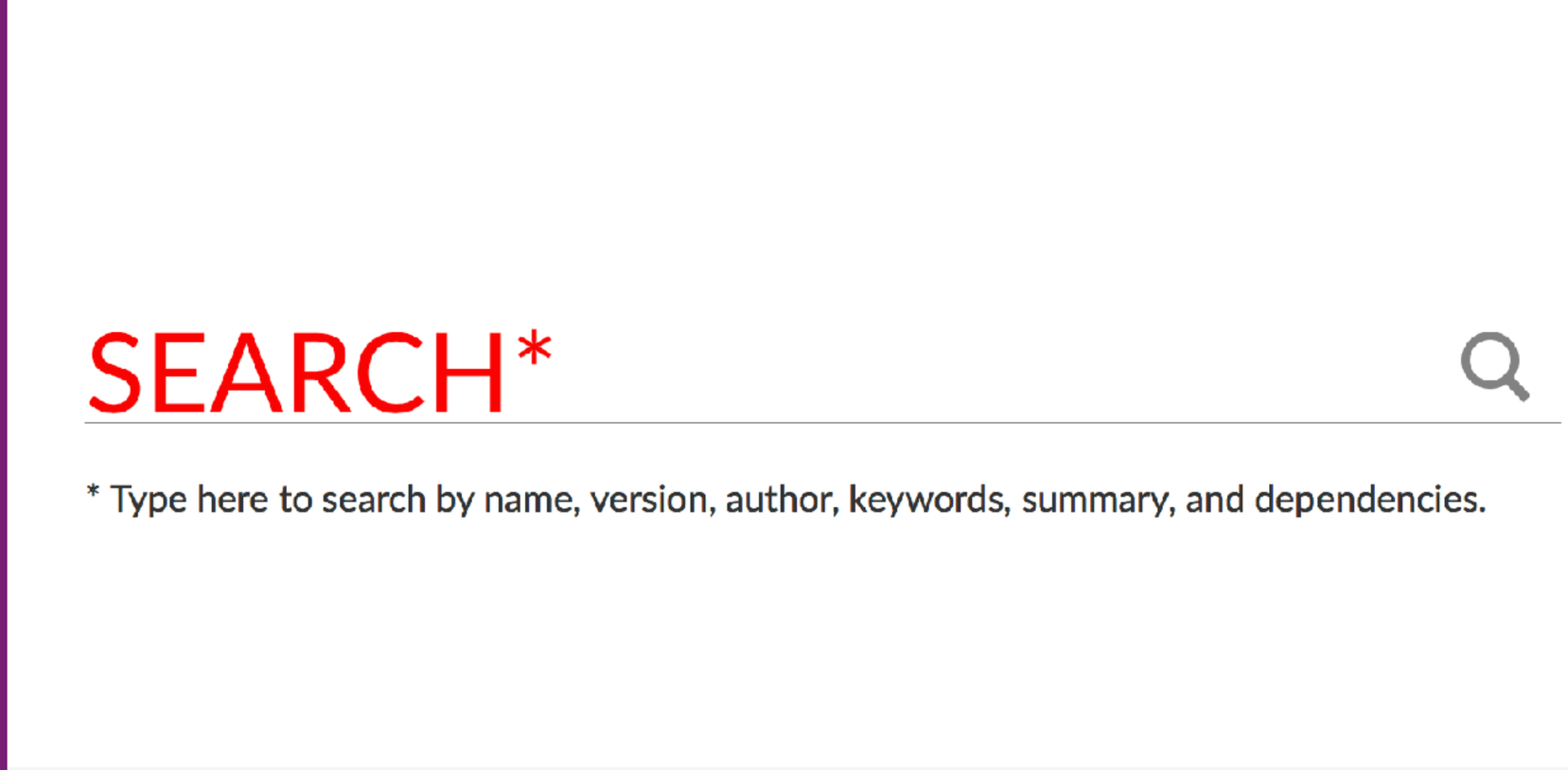
- Package managers for Swift (iOS)
  - CocoaPods
  - Carthage





OTHER PACKAGE MANAGERS

- Cocoa pods is a centralized repository
- Used heavily throughout industry



OTHER PACKAGE MANAGERS

- https://www.youtube.com/watch?v=iEAjvNRdZa0

OTHER PACKAGE MANAGERS

- Downloads everything to your project and creates a workspace
- Similar to SPM

OTHER PACKAGE MANAGERS

- Decentralized
  - No point of failure
- Relies on xcode tools

Carthage

license MIT release v0.26.2

Carthage is intended to be the simplest way to add frameworks to your Cocoa application.

The basic workflow looks something like this:

1. Create a Cartfile that lists the frameworks you'd like to use in your project.
2. Run Carthage, which fetches and builds each framework you've listed.
3. Drag the built `.framework` binaries into your application's Xcode project.

Carthage builds your dependencies and provides you with binary frameworks, but you retain full control over your project structure and setup. Carthage does not automatically modify your project files or your build settings.

Differences between Carthage and CocoaPods

CocoaPods is a long-standing dependency manager for Cocoa. So why was Carthage created?

Firstly, CocoaPods (by default) automatically creates and updates an Xcode workspace for your application and all dependencies. Carthage builds framework binaries using `xcodebuild`, but leaves the responsibility of integrating them up to the user. CocoaPods' approach is easier to use, while Carthage's is flexible and unintrusive.

The goal of CocoaPods is listed in its README as follows:

> ... to improve discoverability of, and engagement in, third party open-source libraries, by creating a more centralized ecosystem.

By contrast, Carthage has been created as a *decentralized* dependency manager. There is no central list of projects

OTHER PACKAGE MANAGERS

- Downloads libraries to local machine, compiles and copies library to your project

Carthage

license MIT release v0.26.2

Carthage is intended to be the simplest way to add frameworks to your Cocoa application.

The basic workflow looks something like this:

1. Create a Cartfile that lists the frameworks you'd like to use in your project.
2. Run Carthage, which fetches and builds each framework you've listed.
3. Drag the built `.framework` binaries into your application's Xcode project.

Carthage builds your dependencies and provides you with binary frameworks, but you retain full control over your project structure and setup. Carthage does not automatically modify your project files or your build settings.

Differences between Carthage and CocoaPods

CocoaPods is a long-standing dependency manager for Cocoa. So why was Carthage created?

Firstly, CocoaPods (by default) automatically creates and updates an Xcode workspace for your application and all dependencies. Carthage builds framework binaries using `xcodebuild`, but leaves the responsibility of integrating them up to the user. CocoaPods' approach is easier to use, while Carthage's is flexible and unintrusive.

The goal of CocoaPods is listed in its README as follows:

> ... to improve discoverability of, and engagement in, third party open-source libraries, by creating a more centralized ecosystem.

By contrast, Carthage has been created as a *decentralized* dependency manager. There is no central list of projects

OTHER PACKAGE MANAGERS

- Not as popular, but some people prefer their approach

Carthage

license MIT release v0.26.2

Carthage is intended to be the simplest way to add frameworks to your Cocoa application.

The basic workflow looks something like this:

1. Create a Cartfile that lists the frameworks you'd like to use in your project.
2. Run Carthage, which fetches and builds each framework you've listed.
3. Drag the built `.framework` binaries into your application's Xcode project.

Carthage builds your dependencies and provides you with binary frameworks, but you retain full control over your project structure and setup. Carthage does not automatically modify your project files or your build settings.

Differences between Carthage and CocoaPods

CocoaPods is a long-standing dependency manager for Cocoa. So why was Carthage created?

Firstly, CocoaPods (by default) automatically creates and updates an Xcode workspace for your application and all dependencies. Carthage builds framework binaries using `xcodebuild`, but leaves the responsibility of integrating them up to the user. CocoaPods' approach is easier to use, while Carthage's is flexible and unintrusive.

The goal of CocoaPods is listed in its README as follows:

> ... to improve discoverability of, and engagement in, third party open-source libraries, by creating a more centralized ecosystem.

By contrast, Carthage has been created as a *decentralized* dependency manager. There is no central list of projects

OTHER PACKAGE MANAGERS

- Swift Package Manager will take these over eventually
- Many popular frameworks on iOS use CocoaPods
- The most popular use all three

SERVERLESS ARCHITECTURES

SERVERLESS ARCHITECTURES

- Serverless architectures can mean many things
  - BaaS
  - PaaS
- Function as a service (Faas) is the extreme realization of this architecture

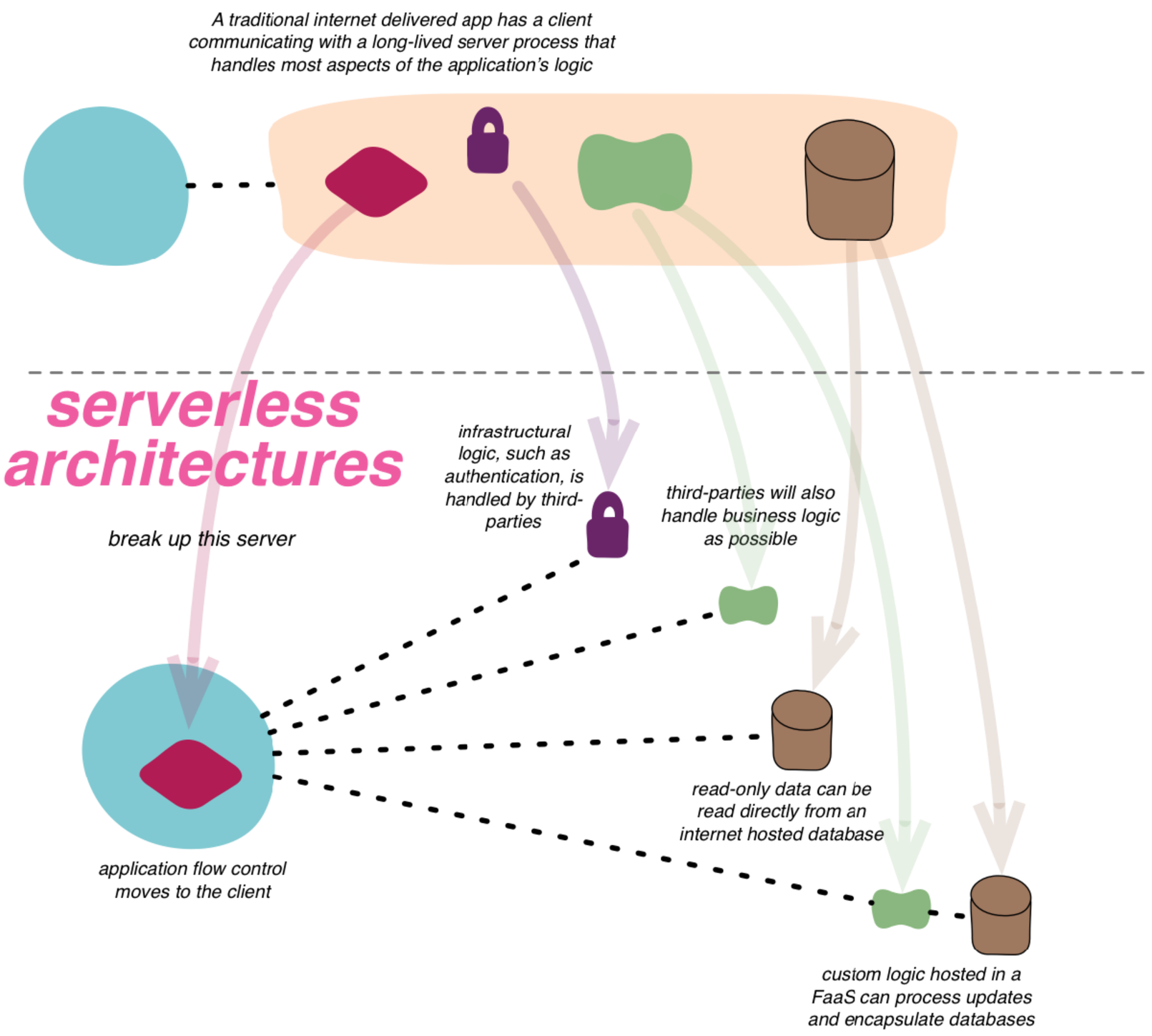
I don't think anyone really calls it this

SERVERLESS ARCHITECTURES

- Despite what you may think, serverless architecture is really about moving as much behavior to the front end as possible
- Remove the need for traditional "always on" server

SERVERLESS ARCHITECTURES

- Traditional vs. serverless architectures

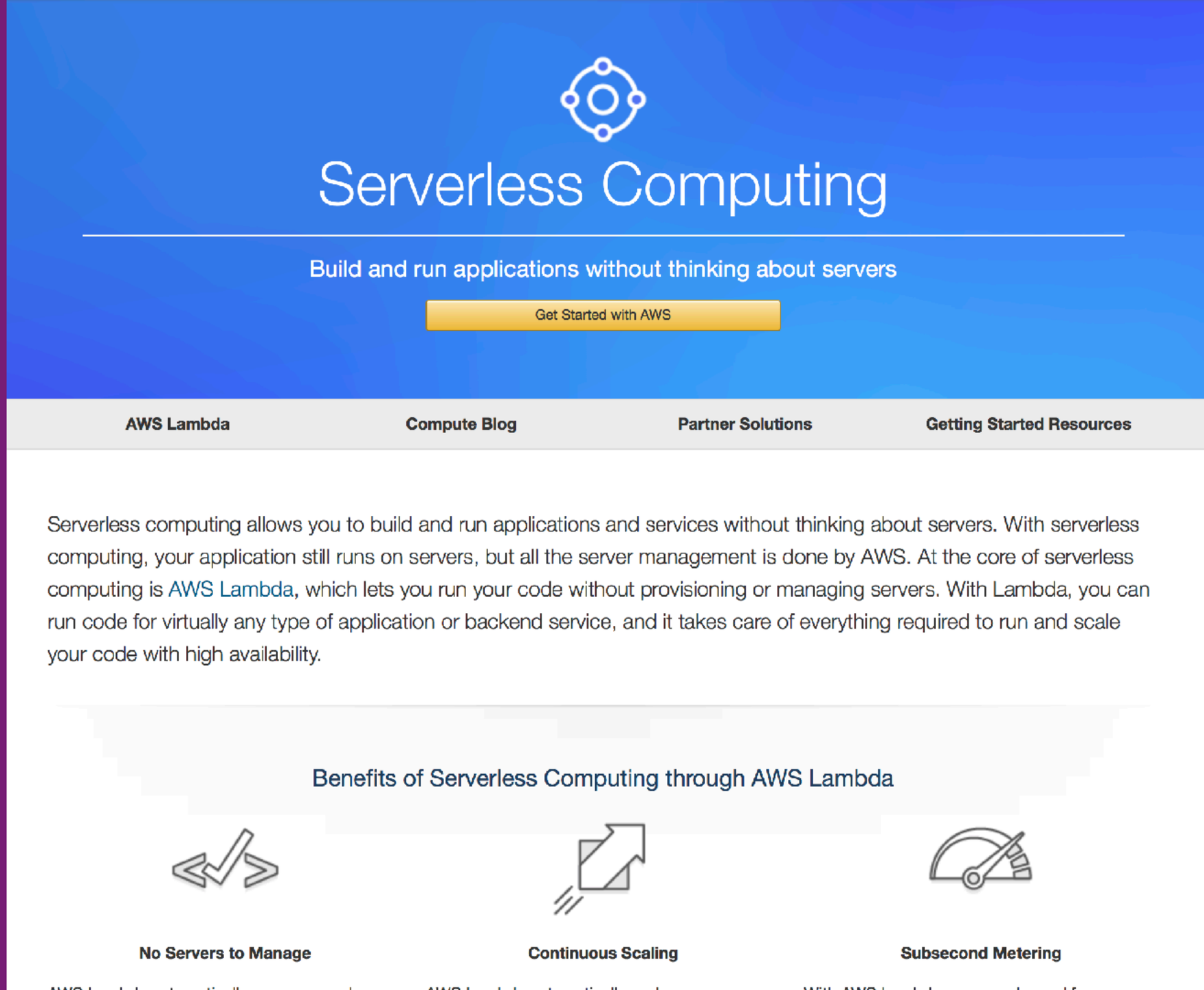


SERVERLESS ARCHITECTURES

- Serverless applications have to rely on third-party services to accomplish tasks that are typically consolidated by a server
  - Each service might be provided by a different provider (eg. authentication with Google, database with Azure) or by a single provider (eg. Firebase)
  - The fact that some services are by the same provider is more of an administrative convenience...serverless is about abstraction

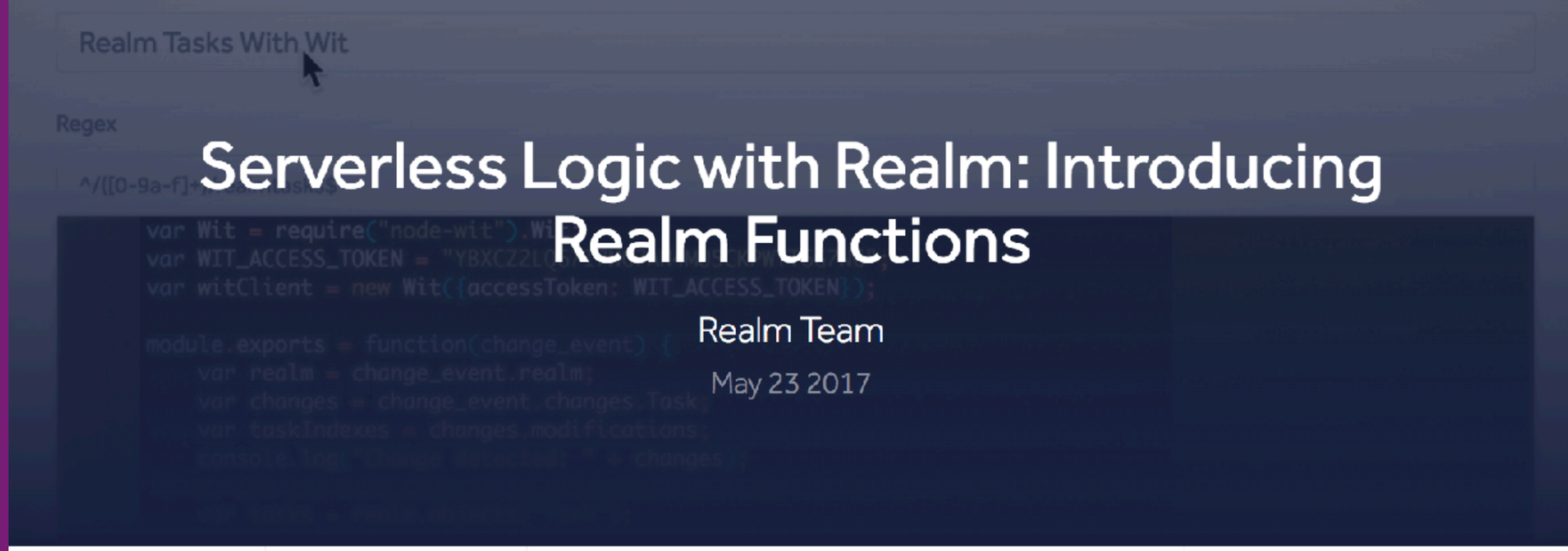
SERVERLESS ARCHITECTURES

- Key players
  - AWS Lambda
  - Microsoft
  - Google Cloud Functions
  - Firebase Cloud Functions



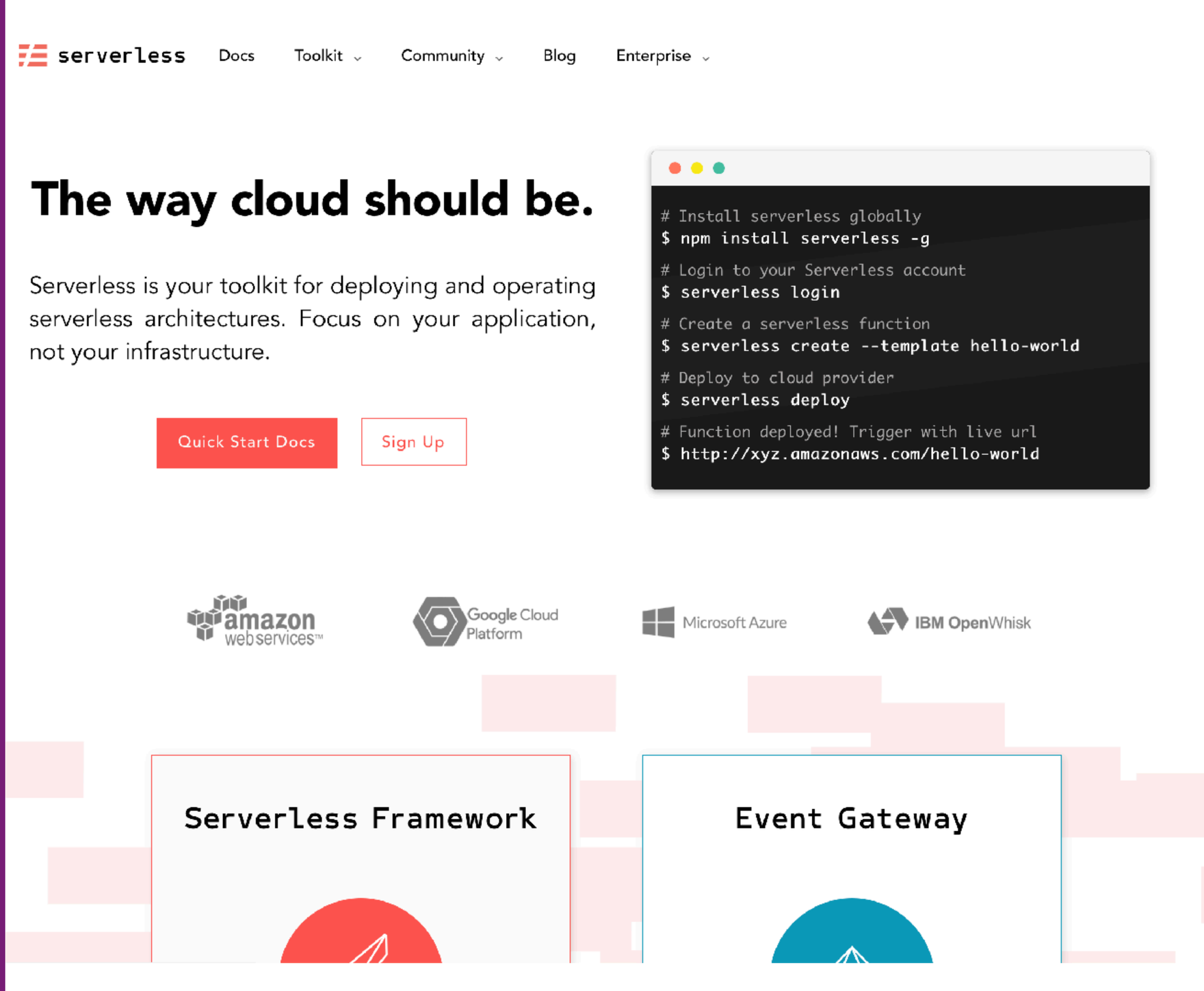
SERVERLESS ARCHITECTURES

- Serverless logic tied to services
  - Parse (RIP)
  - Realm
  - Firebase



SERVERLESS ARCHITECTURES

- Frameworks for server less
  - Write once, deploy everywhere



SERVERLESS ARCHITECTURES

- A major consideration
  - The runtime of server less functions can be extremely limited
  - Server only active during process (pay per time)
  - Data can be returned in the response
  - Data saves requires an outgoing request

All about tradeoffs

SERVERLESS ARCHITECTURES

- Benefits
  - Costs (pay for what you use)
  - Reduced complexity (of running a server)
  - Scaling

SERVERLESS ARCHITECTURES

- Costs
  - Conceptual
  - Third-party dependency
  - Control (whims of big companies)
  - Local development and testing

AWS LAMBDA

AWS LAMBDA

Compute

EC2
EC2 Container Service
Lightsail
Elastic Beanstalk
Lambda
Batch

Developer Tools

CodeStar
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
X-Ray

Analytics

Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
Data Pipeline
QuickSight

Application Services

Step Functions
SWF
API Gateway
Elastic Transcoder

Storage

S3
EFS
Glacier
Storage Gateway

Management Tools

CloudWatch
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Trusted Advisor
Managed Services

Artificial Intelligence

Lex
Polly
Rekognition
Machine Learning

Messaging

Simple Queue Service
Simple Notification Service
SES

Database

RDS
DynamoDB
ElastiCache

Internet Of Things

AWS IoT

Business Productivity

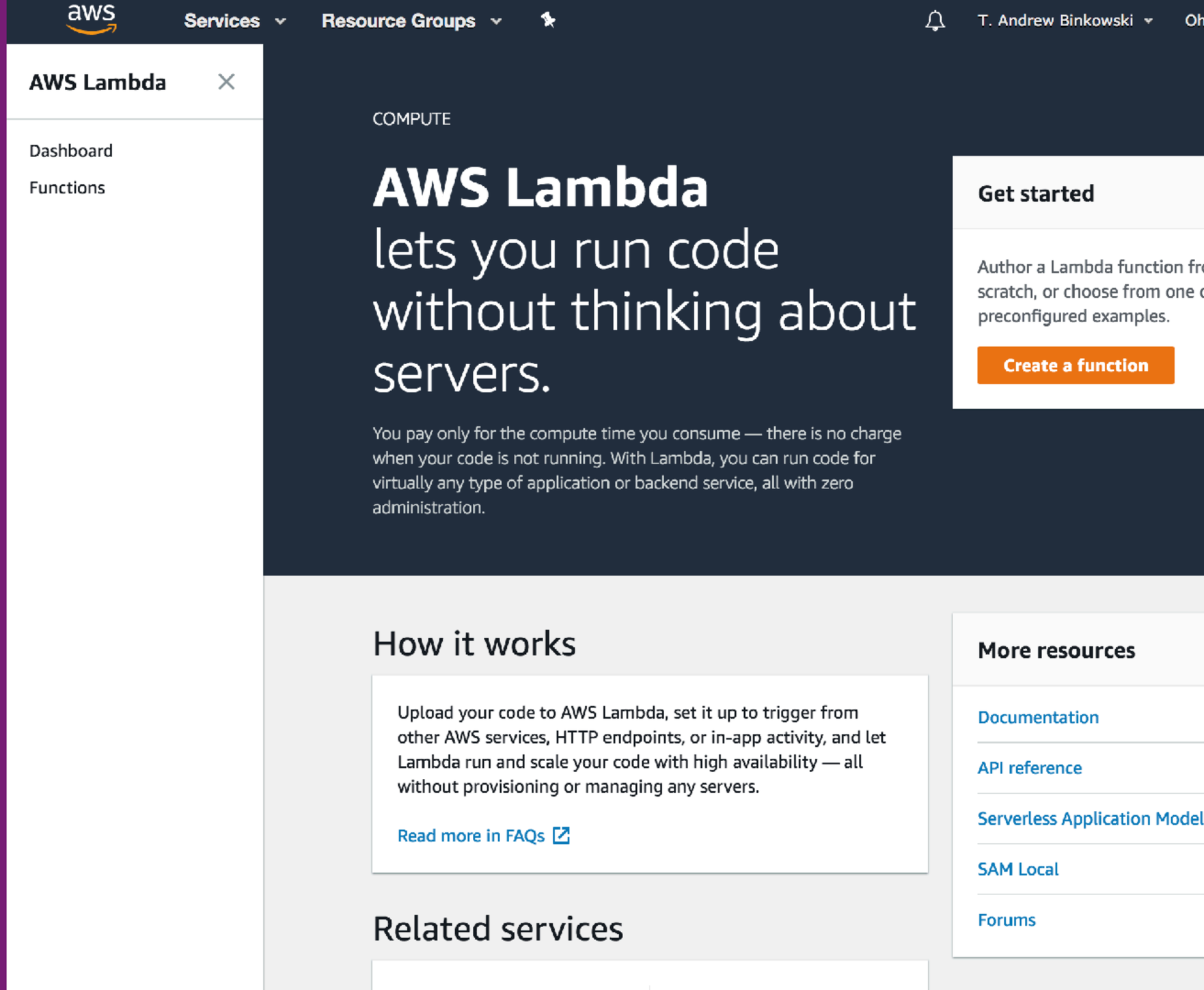
WorkDocs
WorkMail
Amazon Chime

Desktop & App Streaming

AWS has a lot of offerings

AWS LAMBDA

- AWS Lambda serverless architecture



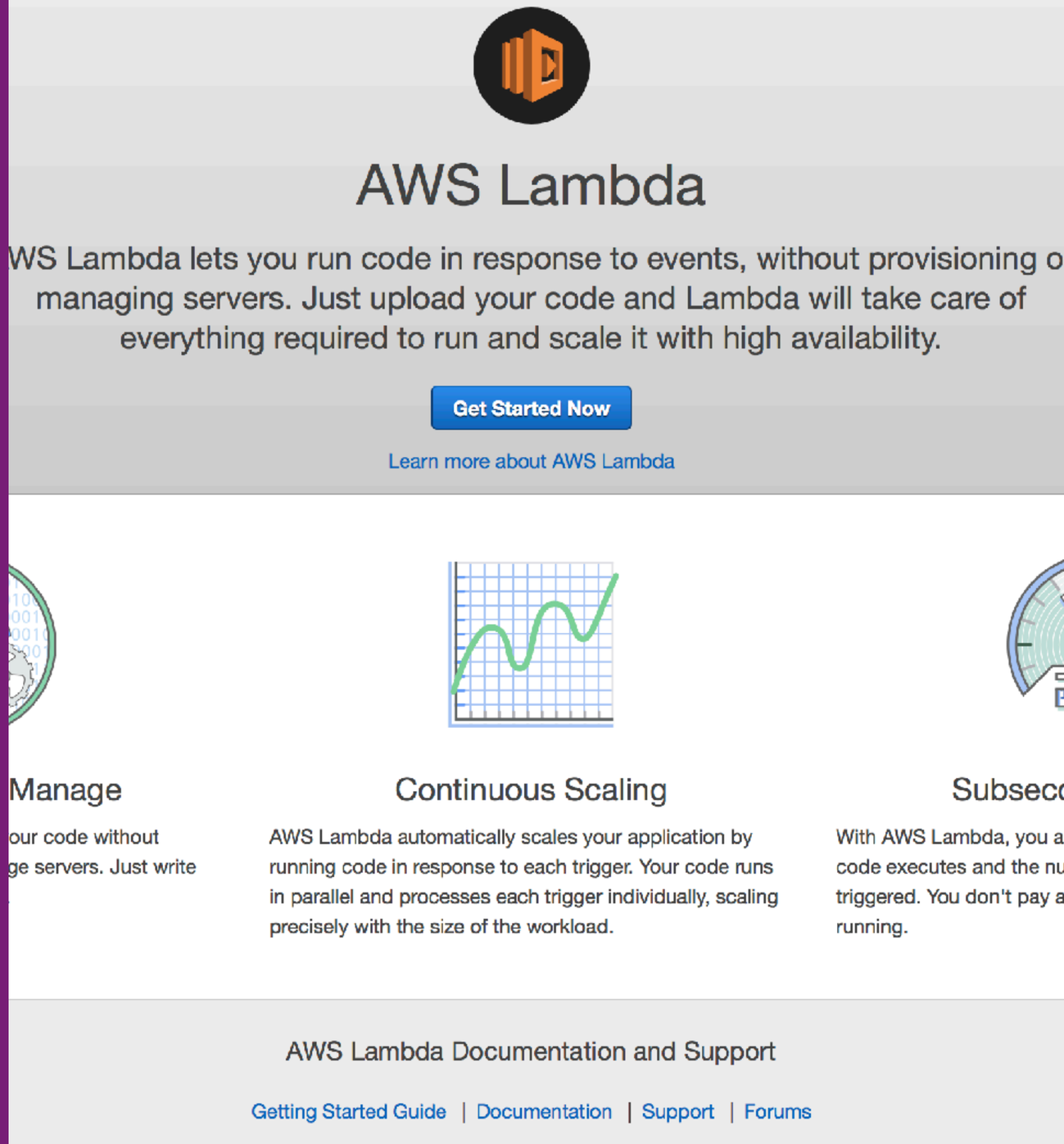
AWS LAMBDA

- The Lambda free tier includes 1M free requests per month and 400,000 GB-seconds of compute time per month
- The memory size you choose for your Lambda functions determines how long they can run in the free tier
- The Lambda free tier does not automatically expire at the end of your 12 month AWS Free Tier term, but is available forever

Memory (MB)	Free tier seconds per month	Price per 100ms ($)
128	3,200,000	0.000000208
192	2,133,333	0.000000313
256	1,600,000	0.000000417
320	1,280,000	0.000000521
384	1,066,667	0.000000625
448	914,286	0.000000729
512	800,000	0.000000834
576	711,111	0.000000938
640	640,000	0.000001042
704	581,818	0.000001146
768	533,333	0.000001250
832	492,308	0.000001354
896	457,143	0.000001459
960	426,667	0.000001563
1024	400,000	0.000001667
1088	376,471	0.000001771
1152	355,556	0.000001875
1216	336,842	0.000001980

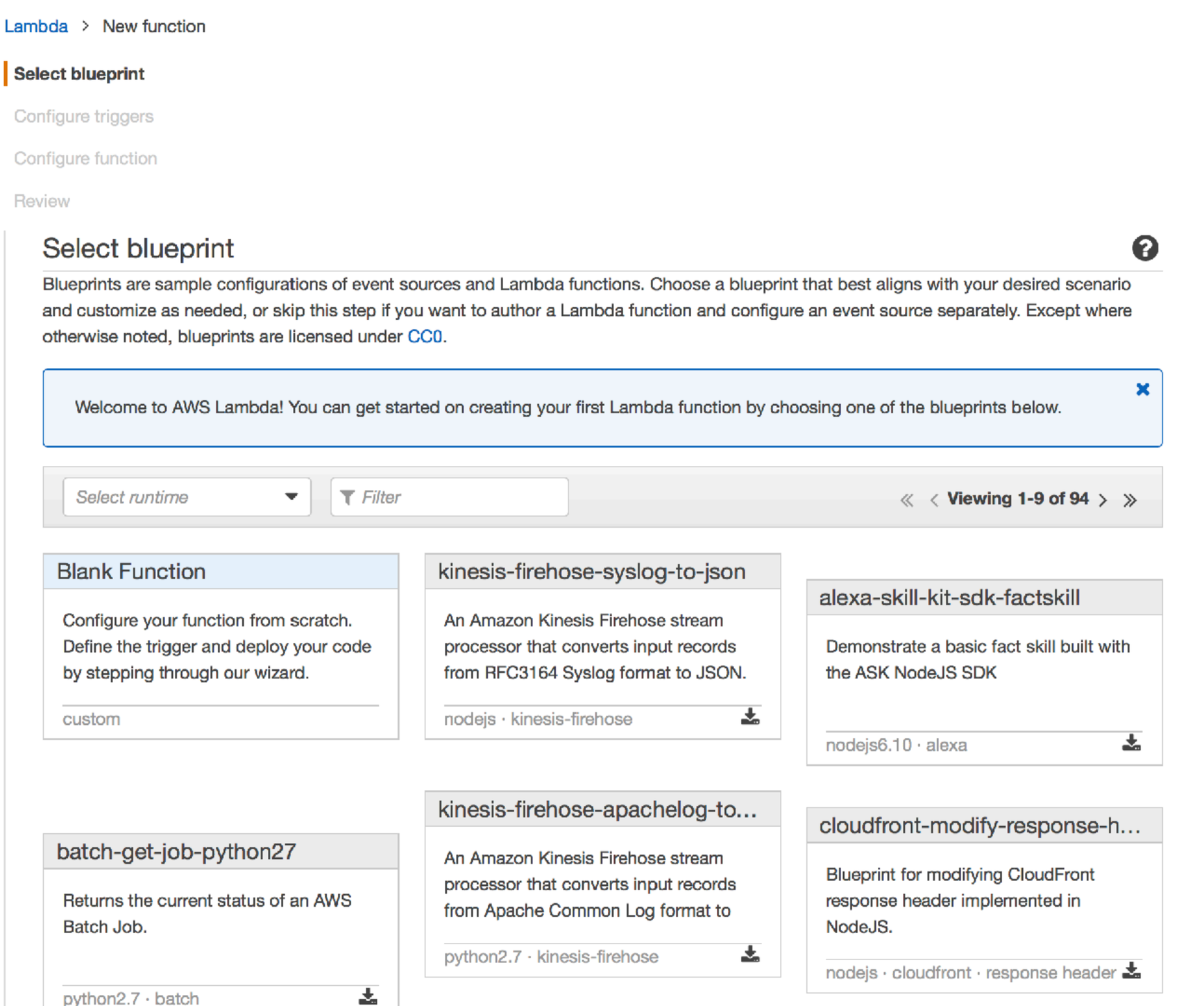
AWS LAMBDA

- Create an AWS account
- Create admin account (optional)
- Download AWS CLI
  - Homebrew or pip
- Create and deploy



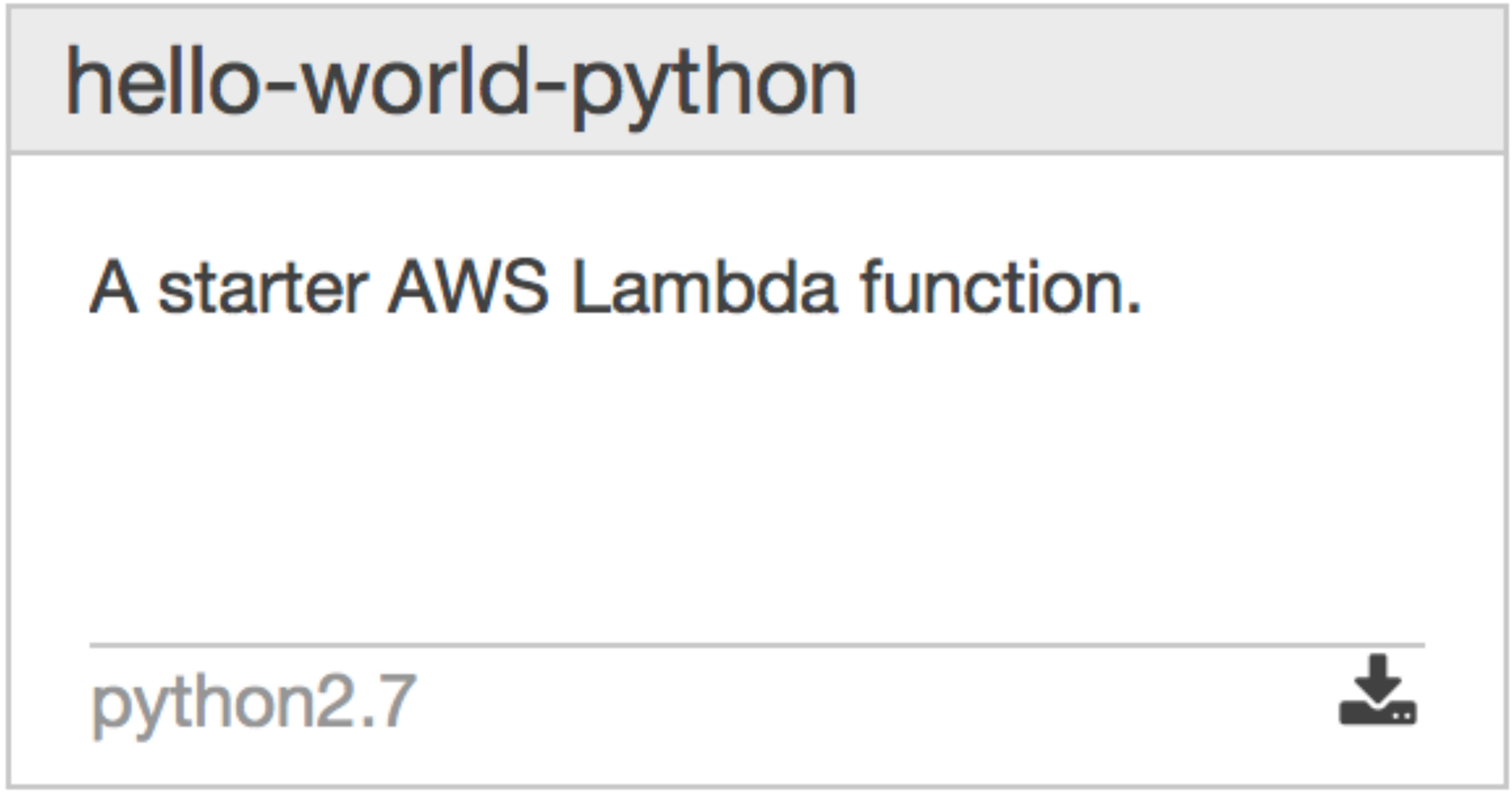
AWS LAMBDA

- Blueprints are sample configurations of event sources and Lambda functions
- Choose a blueprint that best aligns with your desired scenario and customize



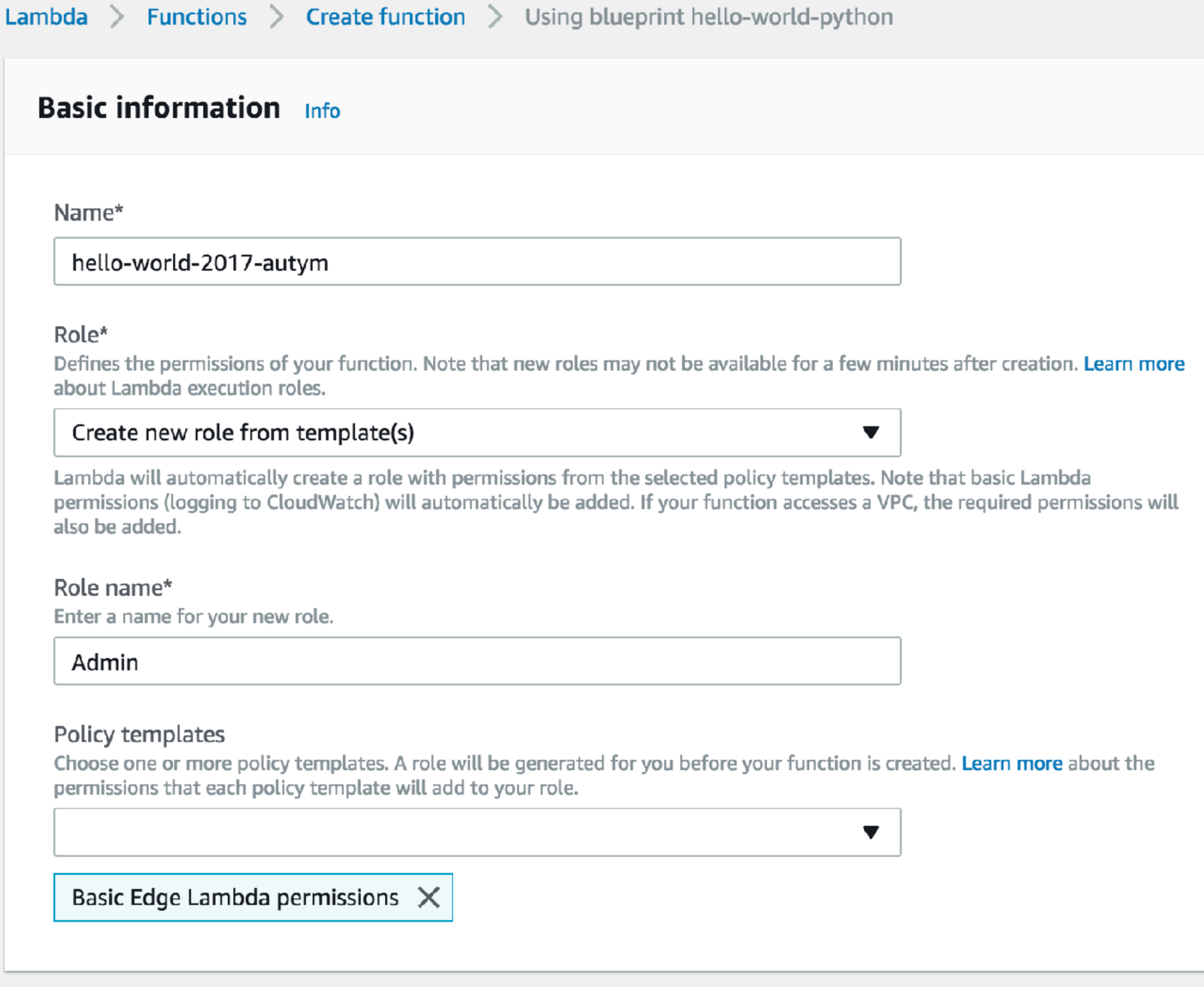
AWS LAMBDA

- Warning: the configurations for the blueprints don't all work out of the box



AWS LAMBDA

- Create functions
  - In-browser editor
  - External gzip and push



AWS LAMBDA

- Create functions
  - Set roles
  - Set permissions

Enter a name that describes the purpose of your function.

hello-aws

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function.

Python 3.8

Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ **Choose or create an execution role**

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the **IAM console**.

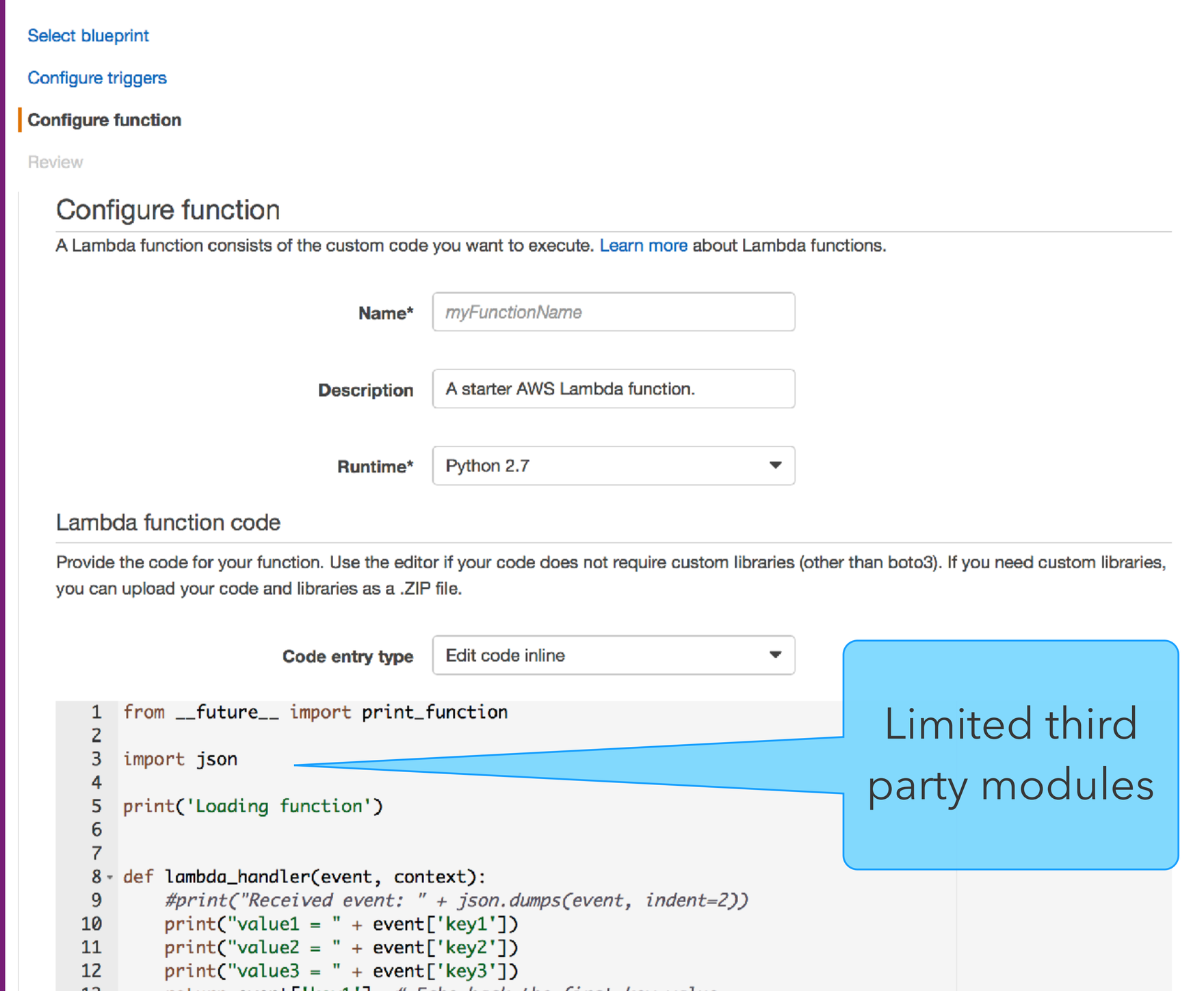
- (•) Create a new role with basic Lambda permissions
- ( ) Use an existing role
- ( ) Create a new role from AWS policy templates

ⓘ Role creation might take a few minutes. The new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Lambda will create an execution role named hello-aws-role-8kti9fmr, with permission to upload logs to Amazon CloudWatch Logs.

AWS LAMBDA

- Functions can be written in JS, Python, Java, and Go



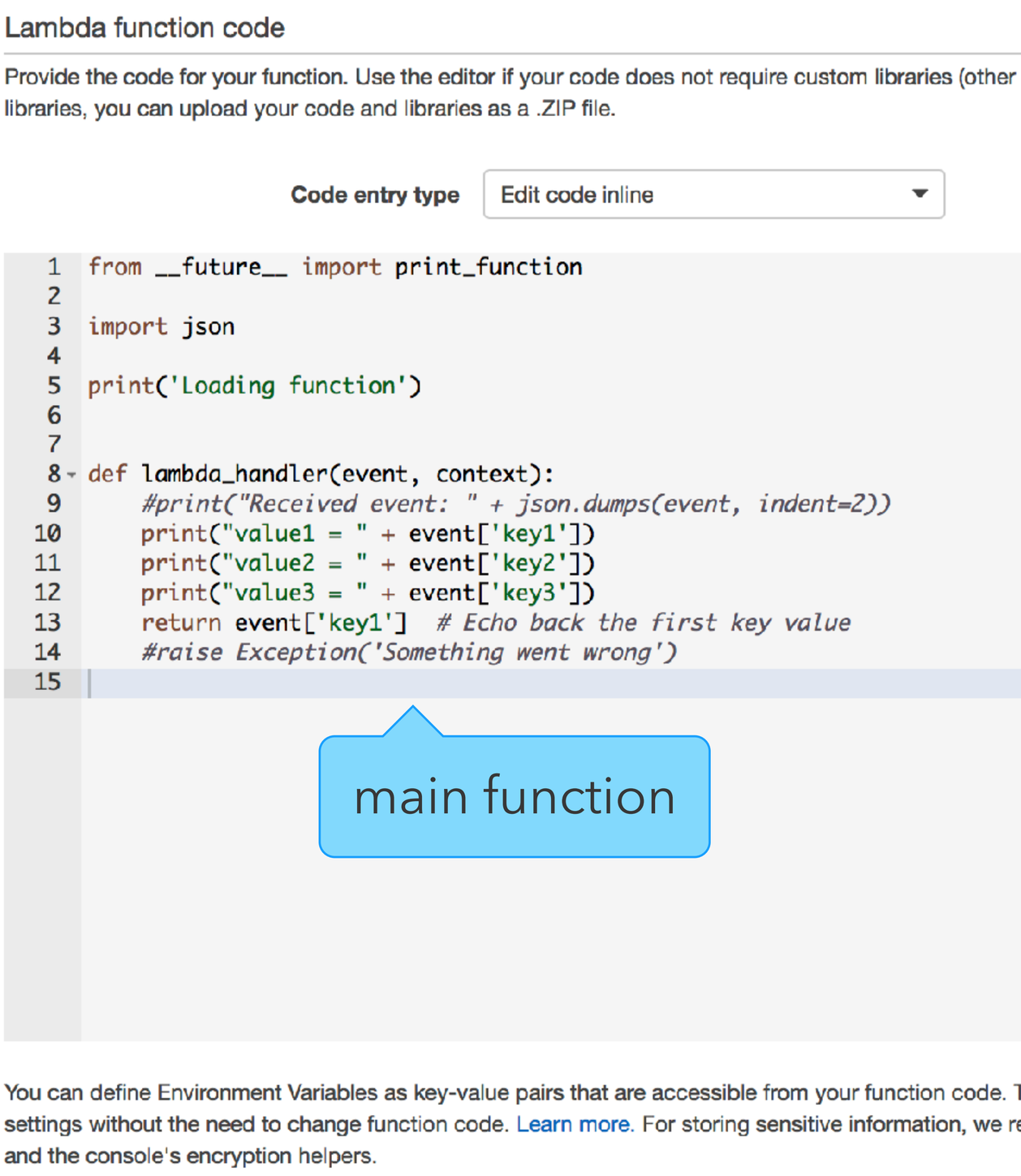
AWS LAMBDA

- Triggers define when you function runs
  - Event based to other AWS Services (push to S3; filter photo)
  - HTTP API
  - Mobile API
  - Scheduled Events (cron)

Functions can be triggered by different events

AWS LAMBDA

- Function handler is the main function called
  - `lambda_function.lambda_handler` is the default console function
  - Name doesn't matter



AWS LAMBDA

- Parameters
  - Event - data passed in
  - Context - runtime information

AWS LAMBDA

- Return values depends on how you set up the function
- Functions need to return properly to test

unction.py

```
from __future__ import print_function

import json

print('Loading function')


def lambda_handler(event, context):
    return {"statusCode": 200,  \
        "headers": {"Content-type": "application/json"}, \
        "body": "{\"count\": 5, \"message\": \"hello\"}", \
        "isBase64Encoded": "false"}
```

AWS LAMBDA

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. Learn more about how Lambda pricing works.

Memory (MB)* 128

Timeout* 0 min 3 sec

Sufficient for Hello World

AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's retry policy and DLQs. **Please ensure your role has appropriate permissions to access the DLQ resource.**

DLQ Resource Select resource

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. Learn more about accessing VPCs within Lambda. **Please ensure your role has**

AWS LAMBDA

Lambda > Functions > hello-world-2017-autym

ARN - arn:aws:lambda:us-east-2:735044932848:function:hello-world-2017-autym

hello-world-2017-autym

Qualifiers ▼ | Actions ▼ | test1 ▼ | Test

Test

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution.

```
"value1"
```

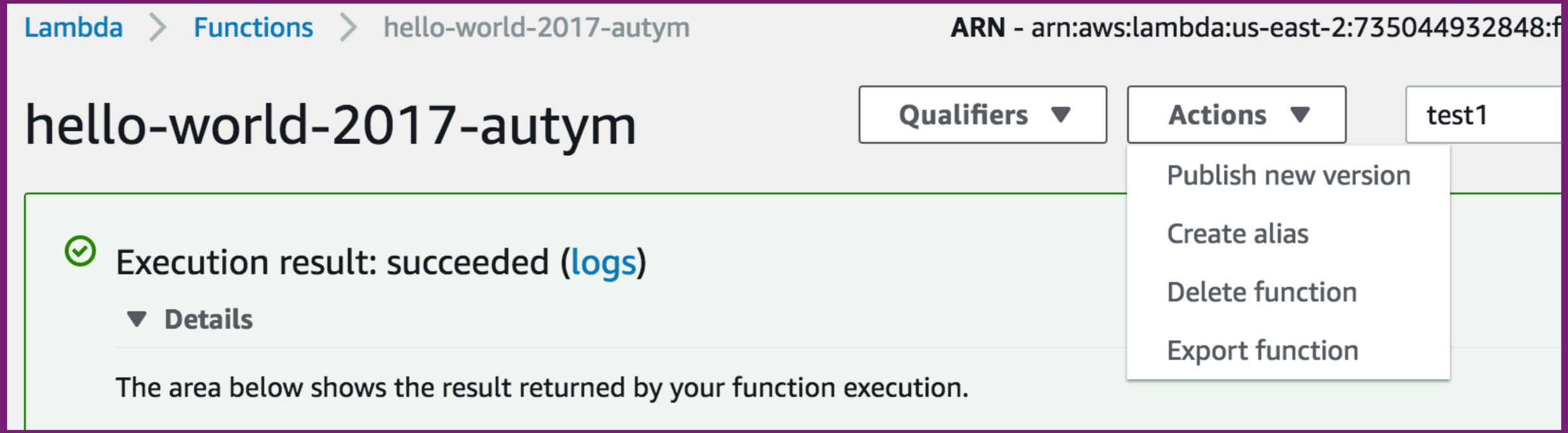
Summary

Code SHA-256

6chaTsf/AFve6WlJ3HrQFLvl
t7xggyqTeTa8DvnxQeA=

Request ID

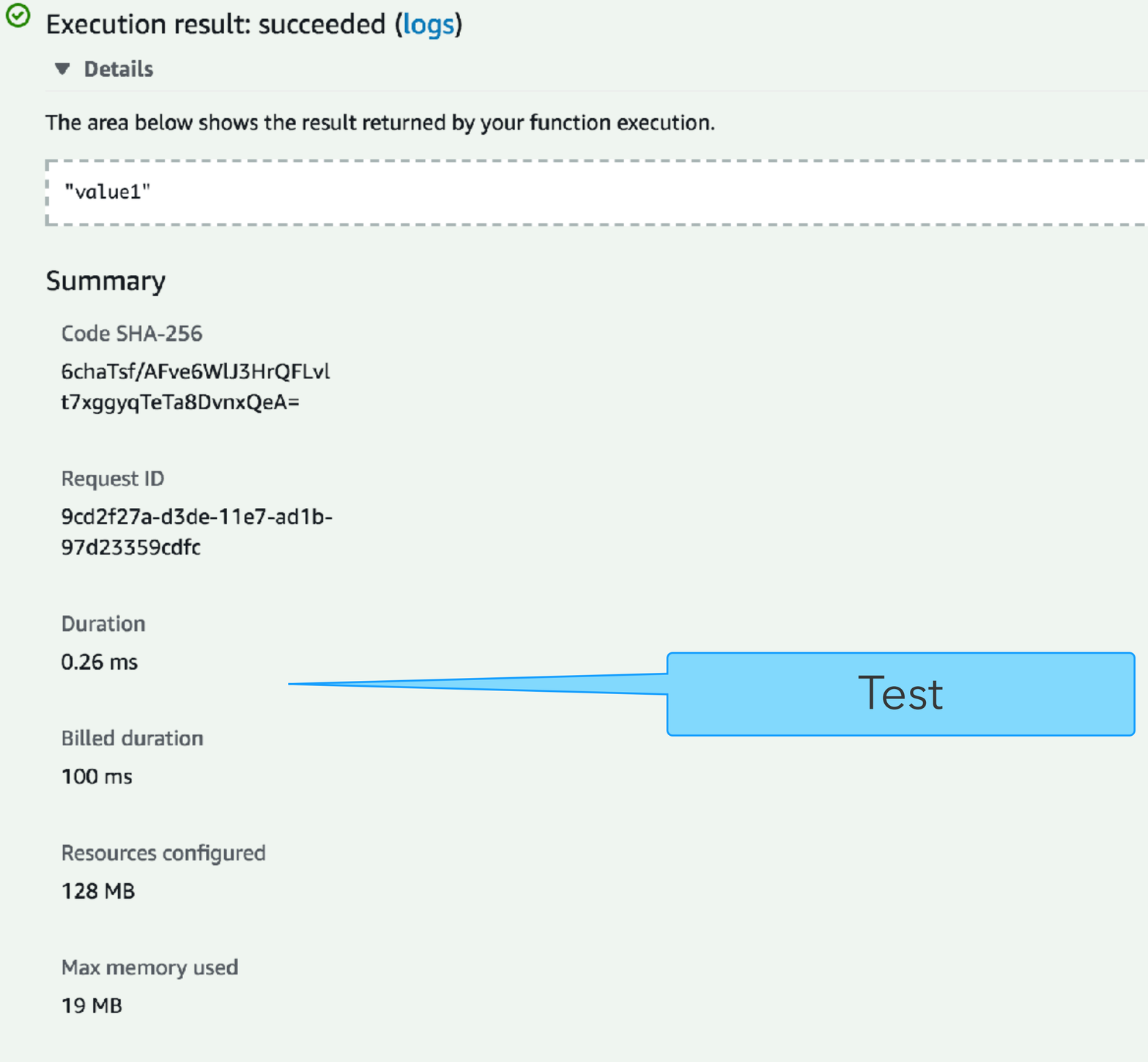
AWS LAMBDA



- Review your Lambda function before deploying
- You can test and edit

AWS LAMBDA

- Test function
  - Response
  - Log output
  - Runtime summary



AWS LAMBDA

Print statements go to logging

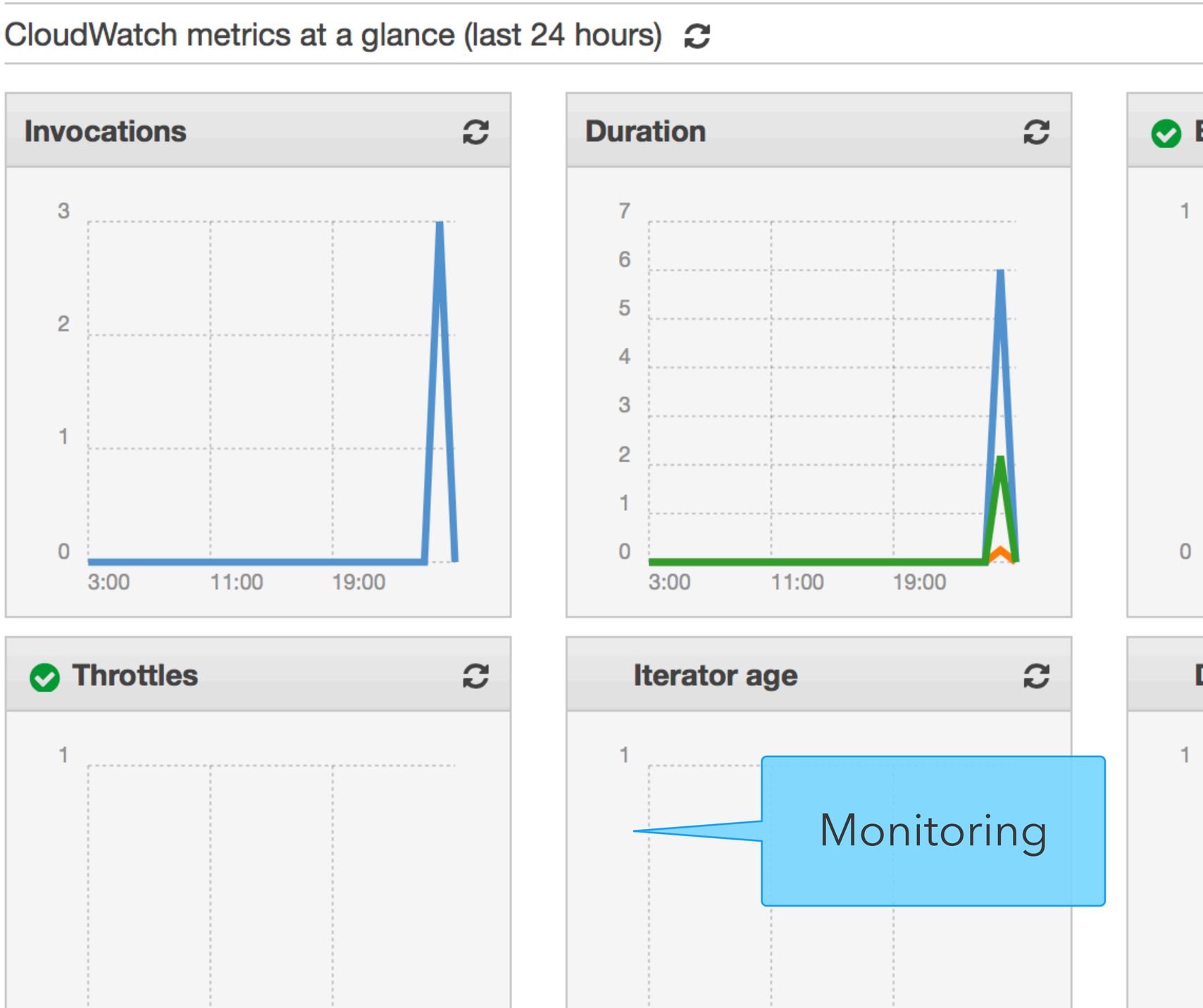
Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

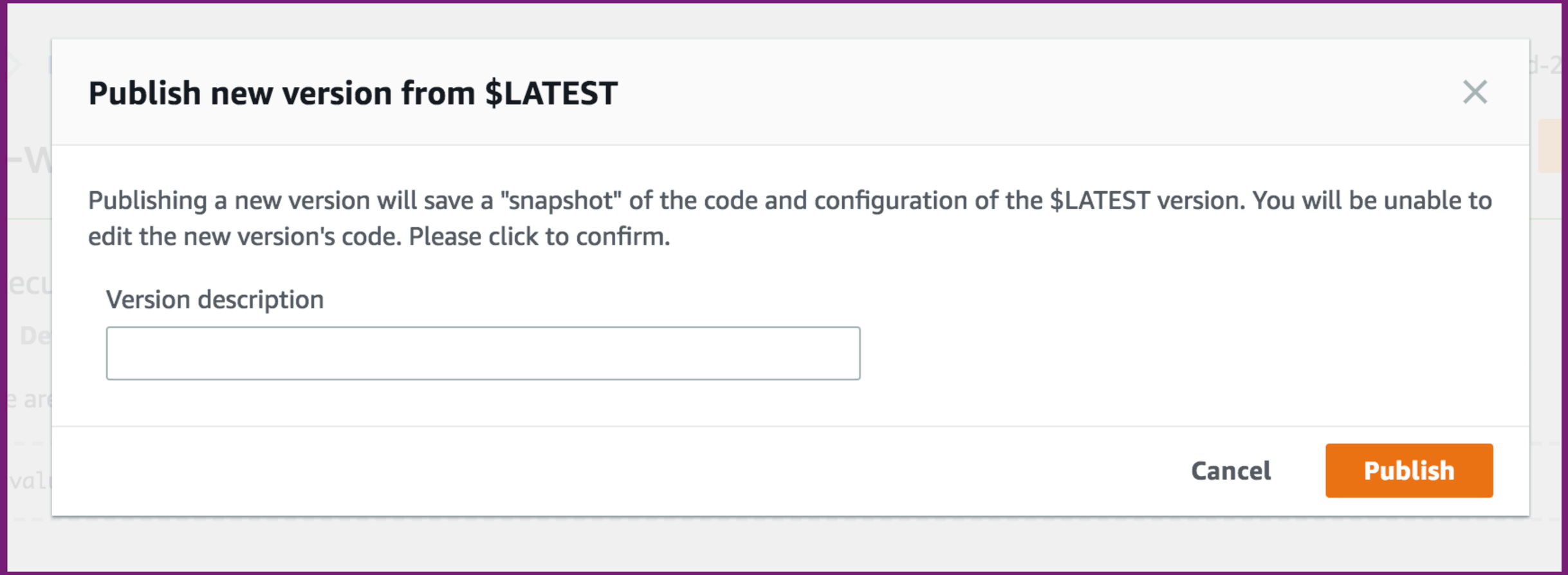
```
START RequestId: 9cd2f27a-d3de-11e7-ad1b-97d23359cdfc Version: $LATEST
value1 = value1
value2 = value2
value3 = value3
END RequestId: 9cd2f27a-d3de-11e7-ad1b-97d23359cdfc
REPORT RequestId: 9cd2f27a-d3de-11e7-ad1b-97d23359cdfc  Duration: 0.26 ms       Billed Duration: 100 ms         Memory Size:
128 MB  Max Memory Used: 19 MB
```

AWS LAMBDA

- Monitor you function



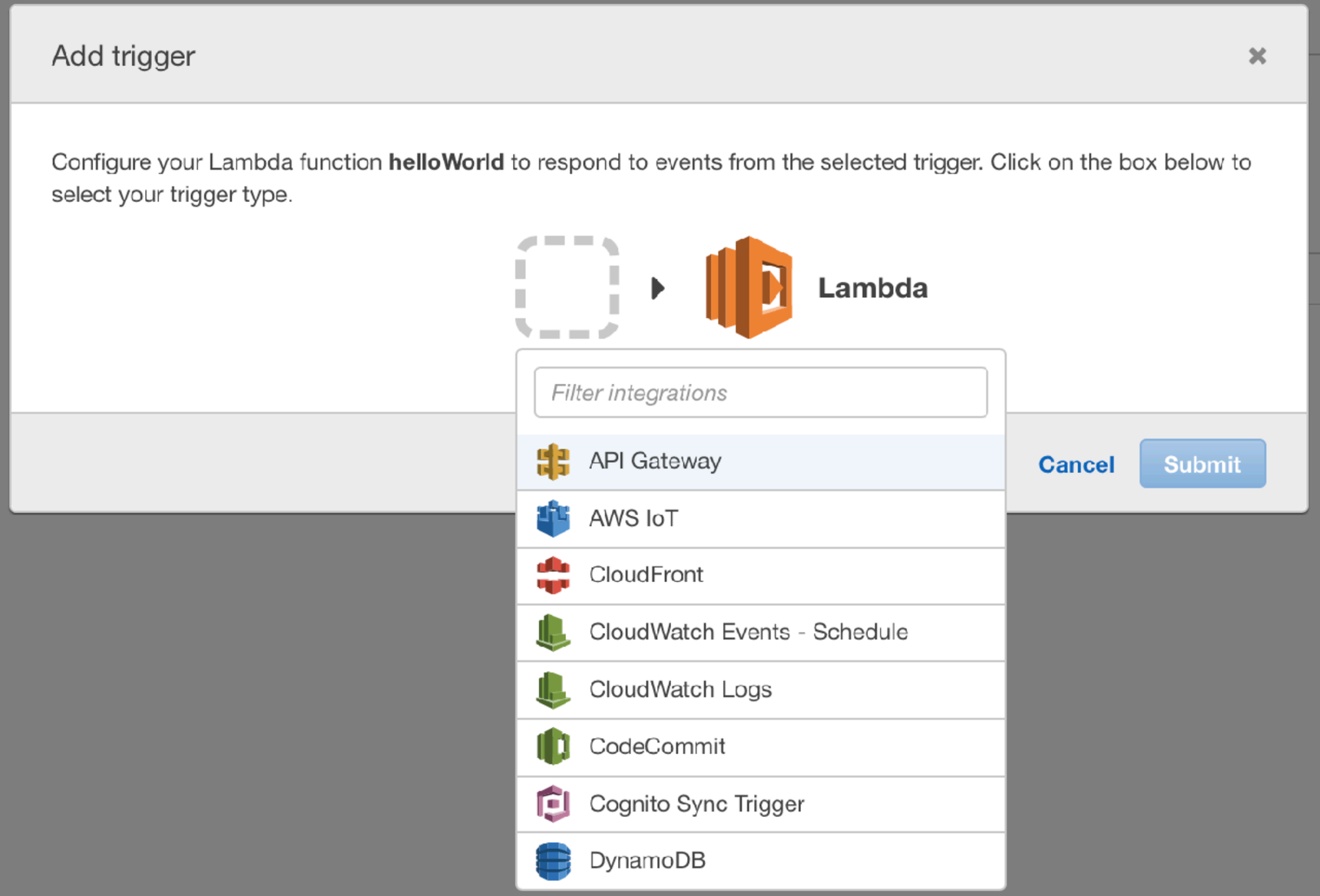
AWS LAMBDA



- Publish

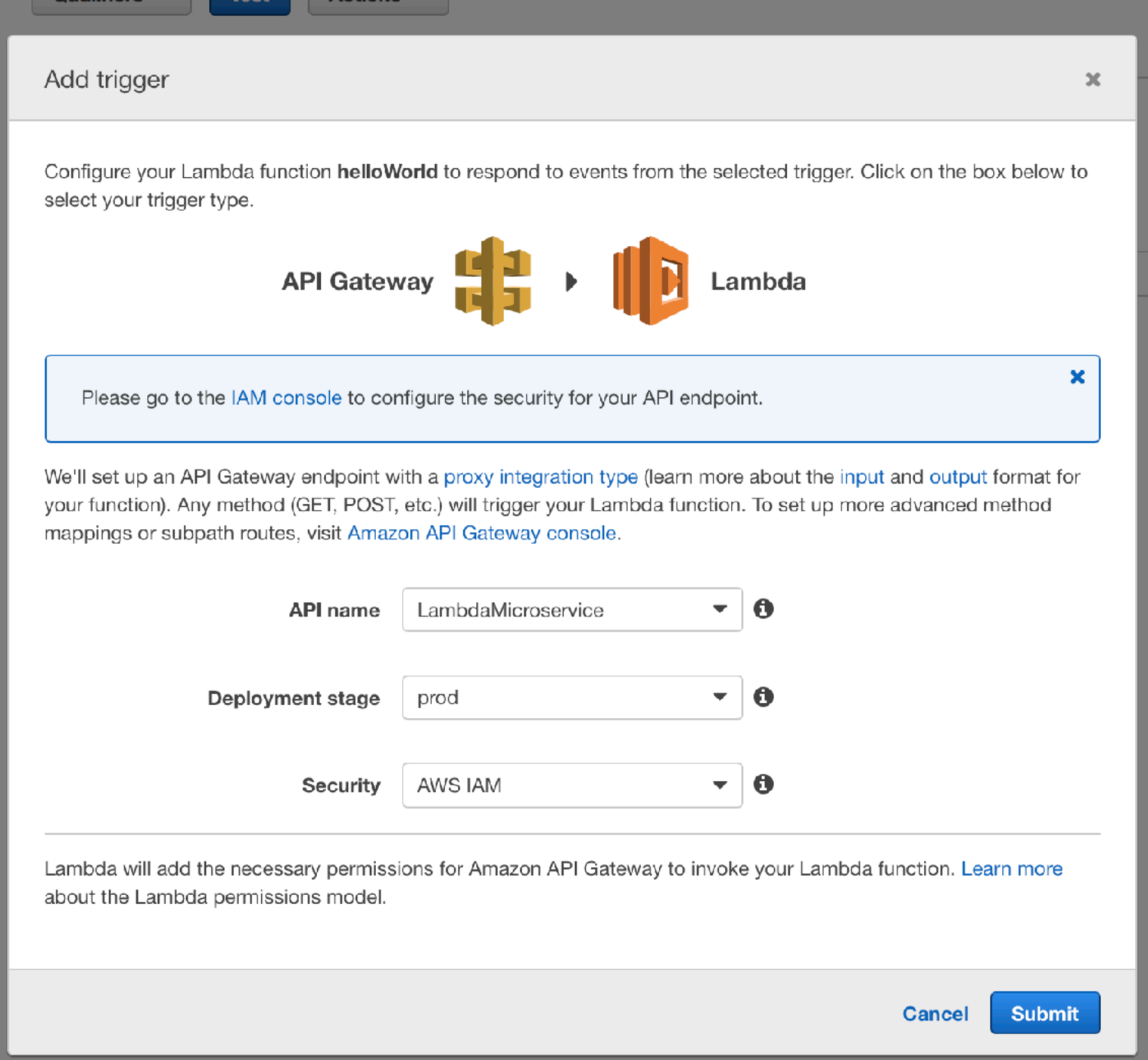
AWS LAMBDA

- Set up a trigger

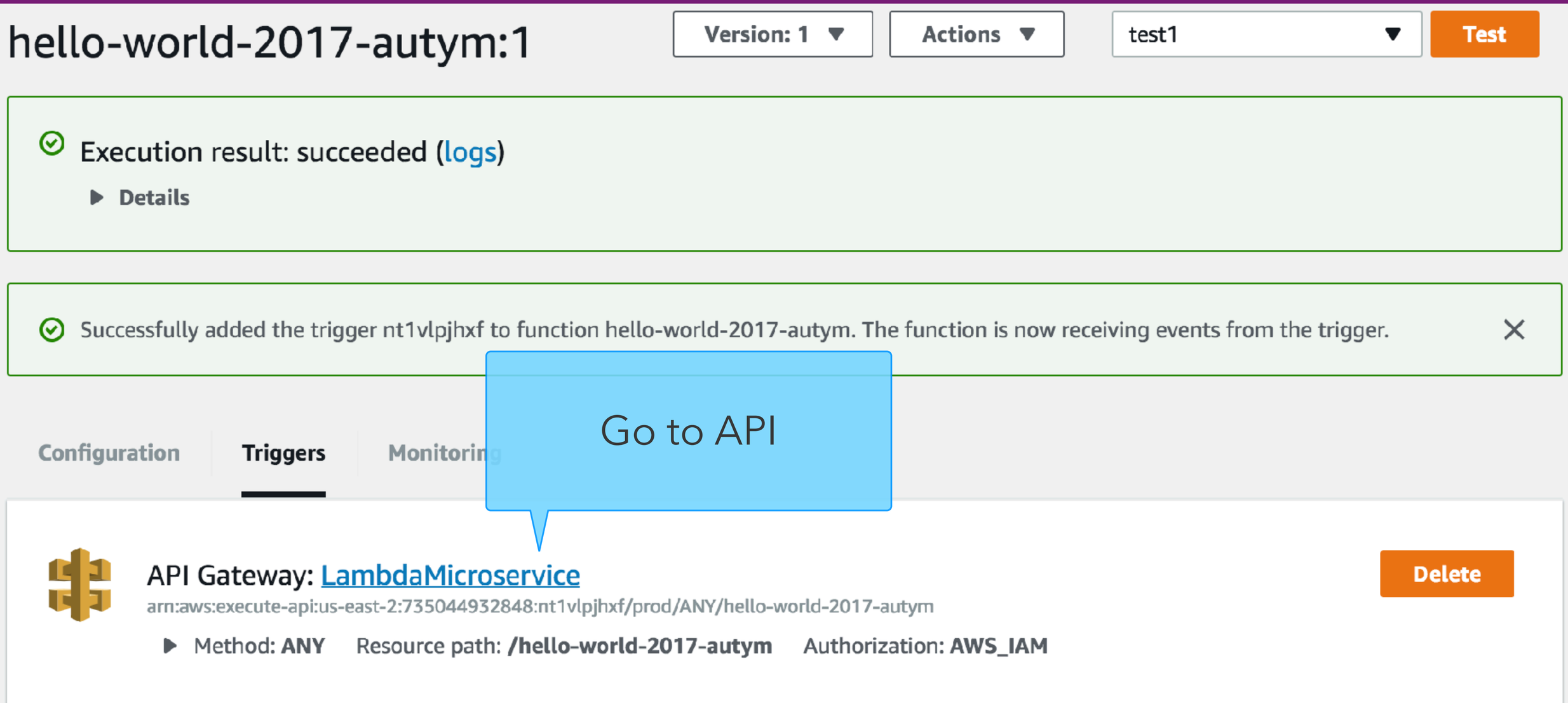


AWS LAMBDA

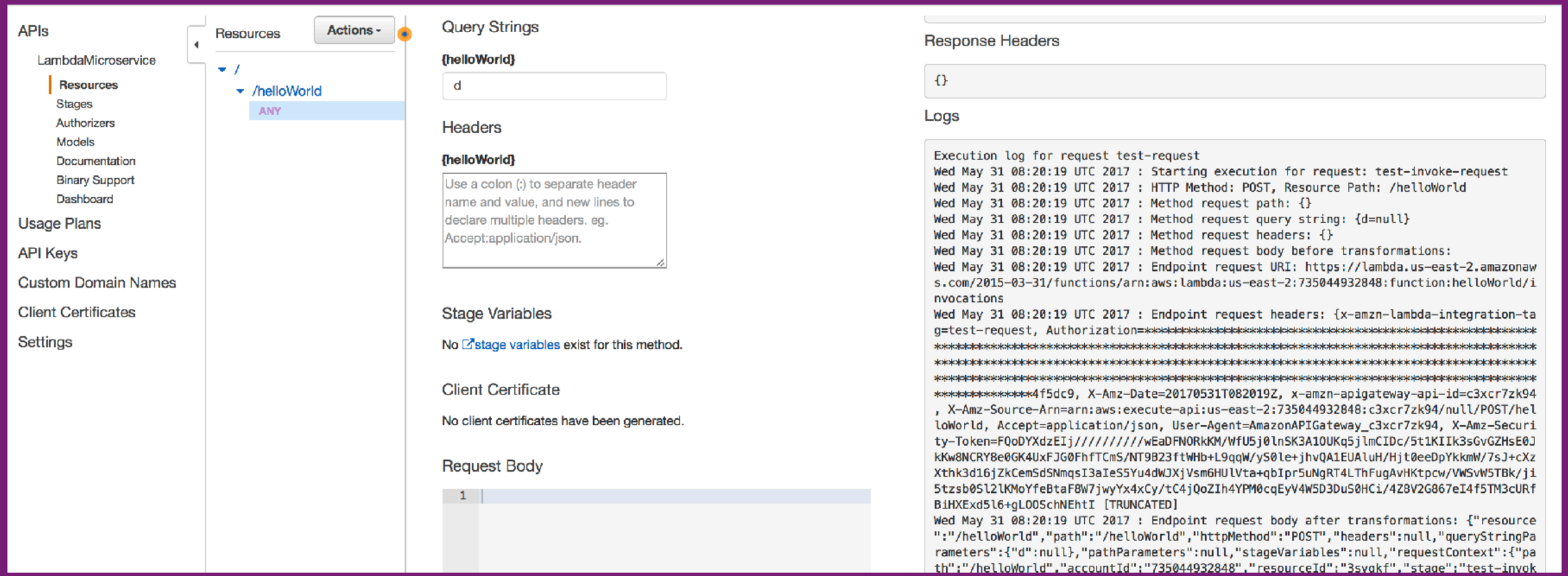
- More security hoops to jump through
- In my experience, you may need to delete the trigger after updating



AWS LAMBDA


hello-world-2017-autym:1
Version: 1
Actions
test1
Test
Execution result: succeeded (logs)
Details
Successfully added the trigger nt1vlpjhxf to function hello-world-2017-autym. The function is now receiving events from the trigger.
Configuration
Triggers
Monitoring
Go to API
API Gateway: LambdaMicroservice
arn:aws:execute-api:us-east-2:735044932848:nt1vlpjhxf/prod/ANY/hello-world-2017-autym
Delete
Method: ANY
Resource path: /hello-world-2017-autym
Authorization: AWS_IAM

AWS LAMBDA



- Test your API (note that to be successful, need to return proper code)

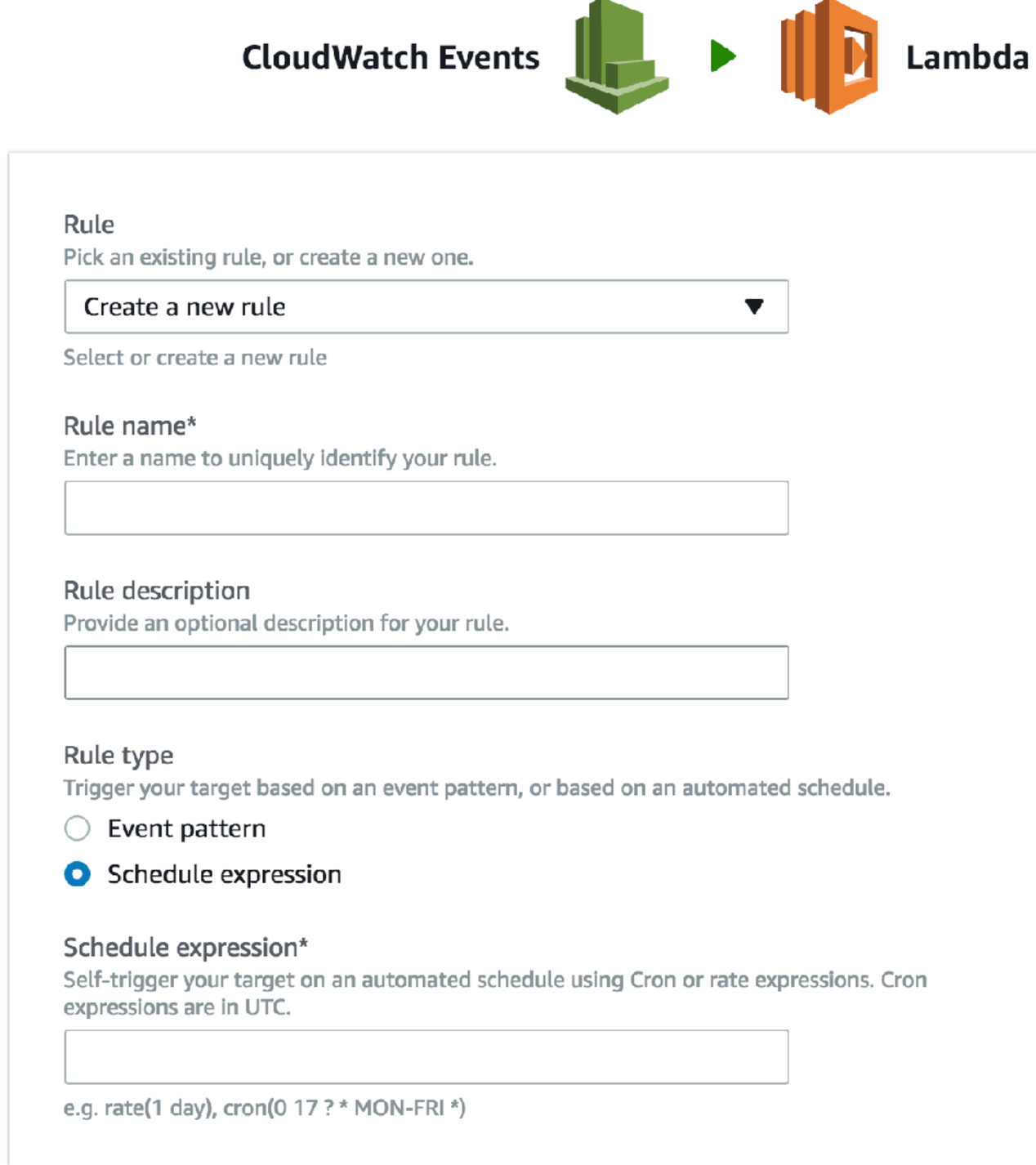
AWS LAMBDA

```
import json

print('Loading function')


def lambda_handler(event, context):
    return {"statusCode": 200,  \
        "headers": {"Content-type": "application/json"}, \
        "body": "{\"count\": 5, \"message\": \"hello\"}", \
        "isBase64Encoded": "false"}
```

- Test your API (note that to be successful, need to return proper code)

AWS LAMBDA

- CloudWatch trigger
  - Schedule
  - AWS Event



AWS LAMBDA

- Limits to consider for lambda

AWS Lambda Resource Limits

Resource	Default Limit
Ephemeral disk capacity ("/tmp" space)	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds
Invoke request body payload size (RequestResponse)	6 MB
Invoke request body payload size (Event)	128 K
Invoke response body payload size (RequestResponse)	6 MB

The following table lists the Lambda account limits per region.

AWS Lambda Account Limits Per Region

Resource	Default Limit
Concurrent executions (see Lambda Function Concurrent Executions).	1000

FIREBASE CLOUD FUNCTIONS VS LAMBDA

🔥FIREBASE CLOUD FUNCTIONS

- https://youtu.be/vr0Gfvp5v1A

🔥FIREBASE CLOUD FUNCTIONS

- Integrates with Firebase in different ways
  - Realtime Database Triggers
  - Firebase Authentication Triggers
  - Firebase Analytics Triggers
  - Cloud Storage Triggers
  - Cloud Pub/Sub Triggers
  - HTTP Triggers

🔥FIREBASE
CLOUD FUNCTIONS

- Overall the functionality and offerings are comparable
  - Deploy functions in the cloud
  - Integrations with other offering (aws and Google ecosystem)
- Devil is in the details

CREATING FUNCTIONS

CREATING FUNCTIONS

- Lambda
  - Nice testing environment
  - In-browser editor
  - Language support
  - More trigger variety (cron)
  - Ton of boilerplate code

CREATING FUNCTIONS

- Firebase
  - Limited (now) to JS
  - Functions can be tied to other Firebase/Google events
  - Fewer triggers

The functions you write can respond to events generated by these other Firebase and Google Cloud features:

- Realtime Database Triggers
- Firebase Authentication Triggers
- Google Analytics for Firebase Triggers
- Cloud Storage Triggers
- Cloud Pub/Sub Triggers
- HTTP Triggers

DEPLOYING

DEPLOYING

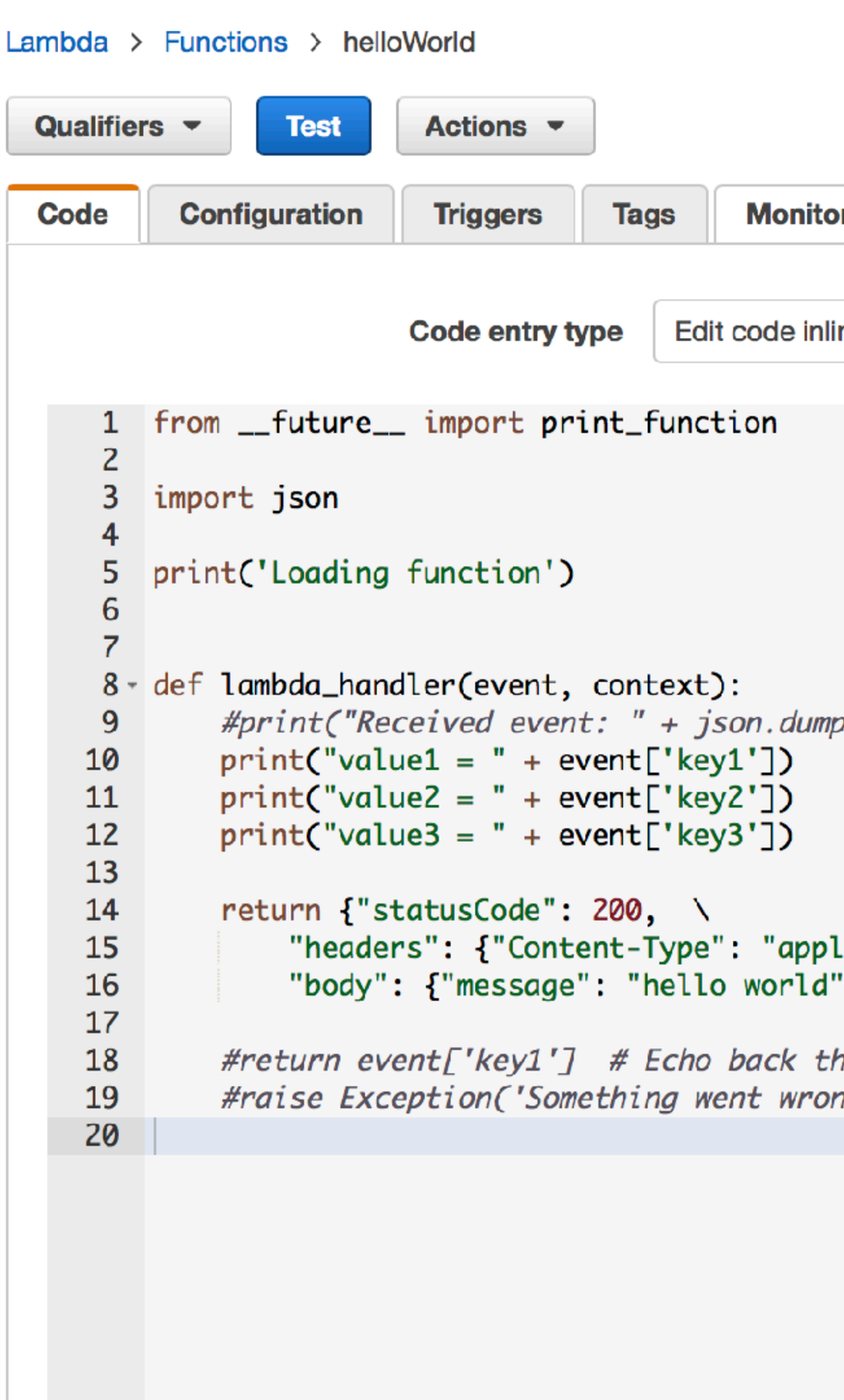
- Deployment and setup is similar
- Lamda does have the in-brower development and deployment

```
aws deploy

firebase deploy
```

TESTING

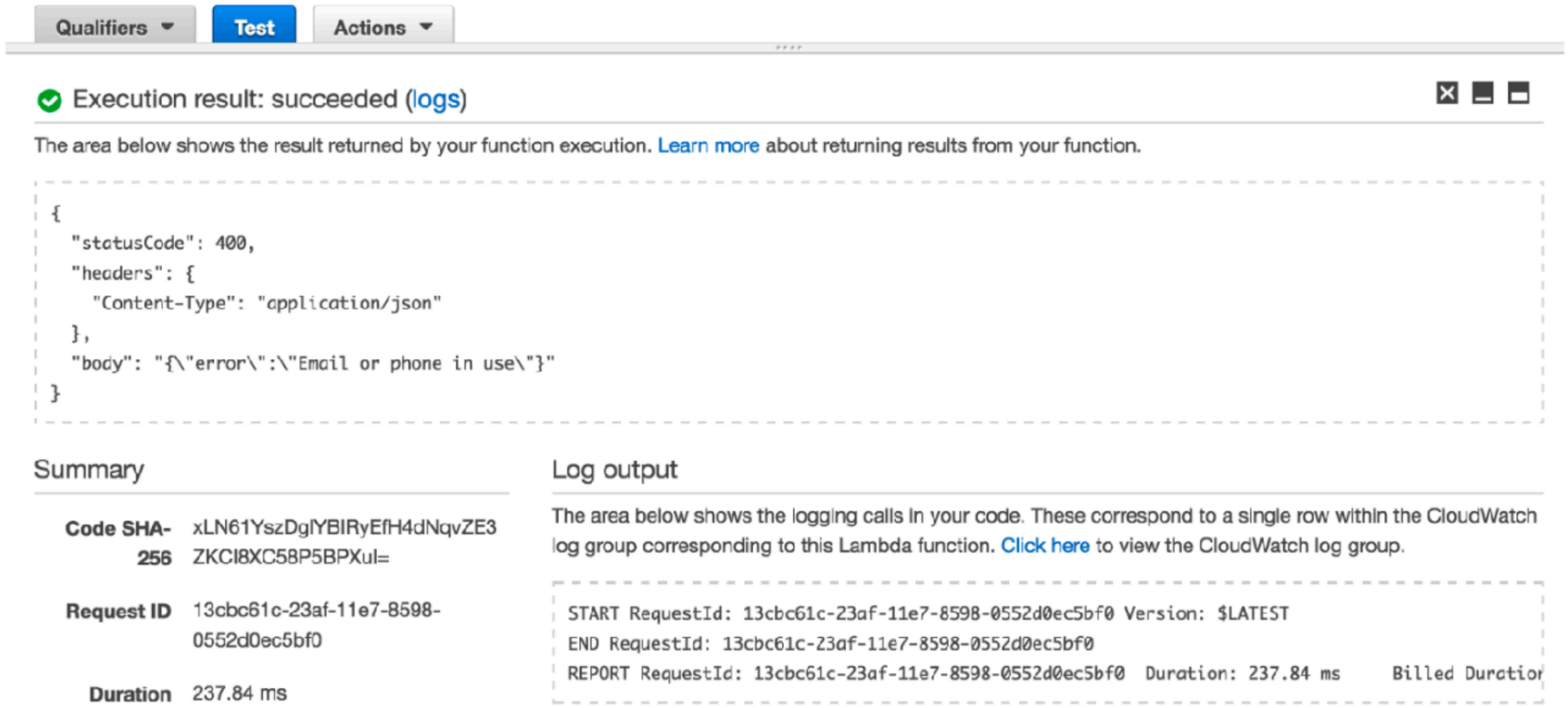
TESTING

- Lambda
  - Great testing environment
  - Test functions and API



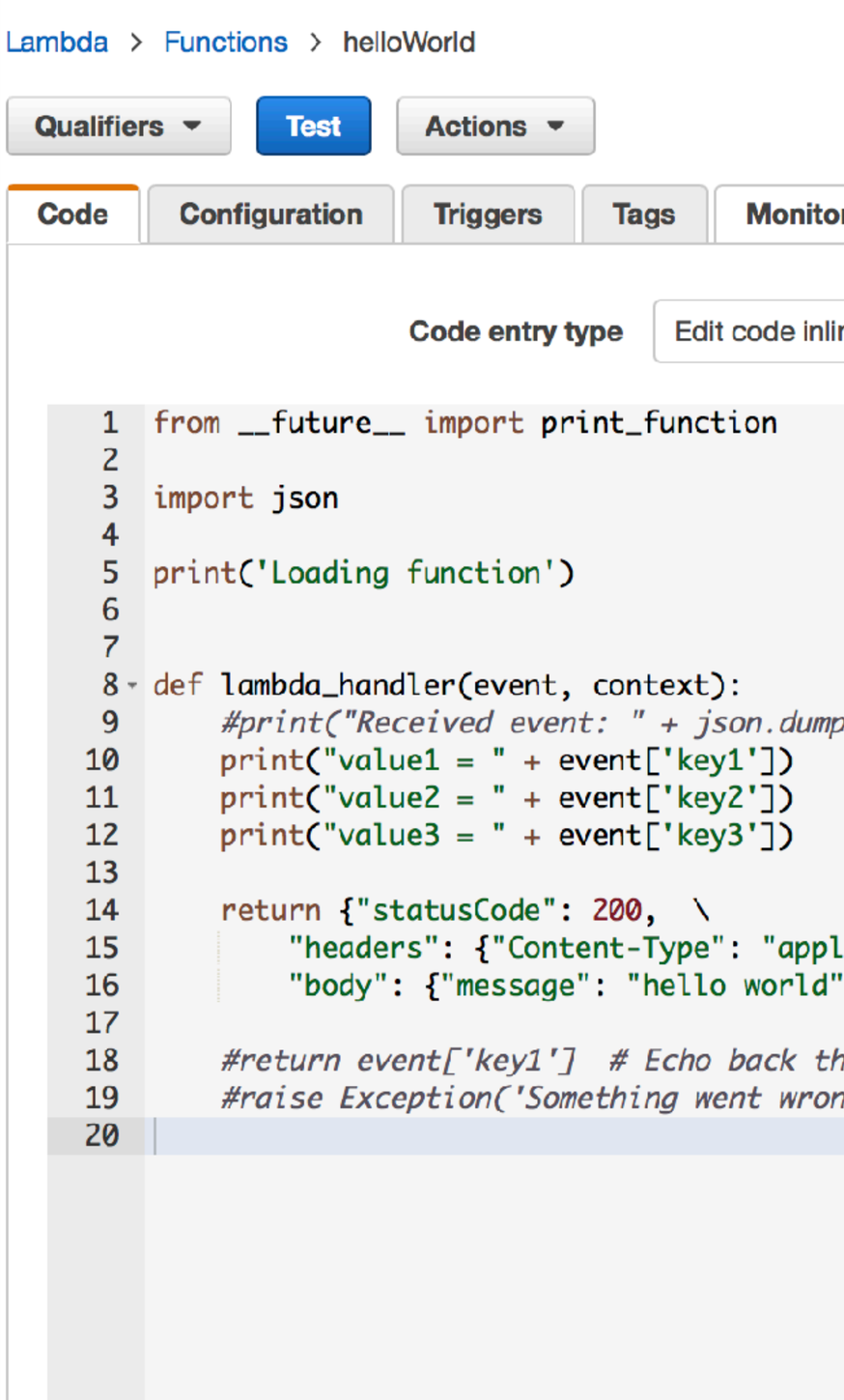
TESTING

- Lambda
  - Debug console and logging provides useful information



TESTING

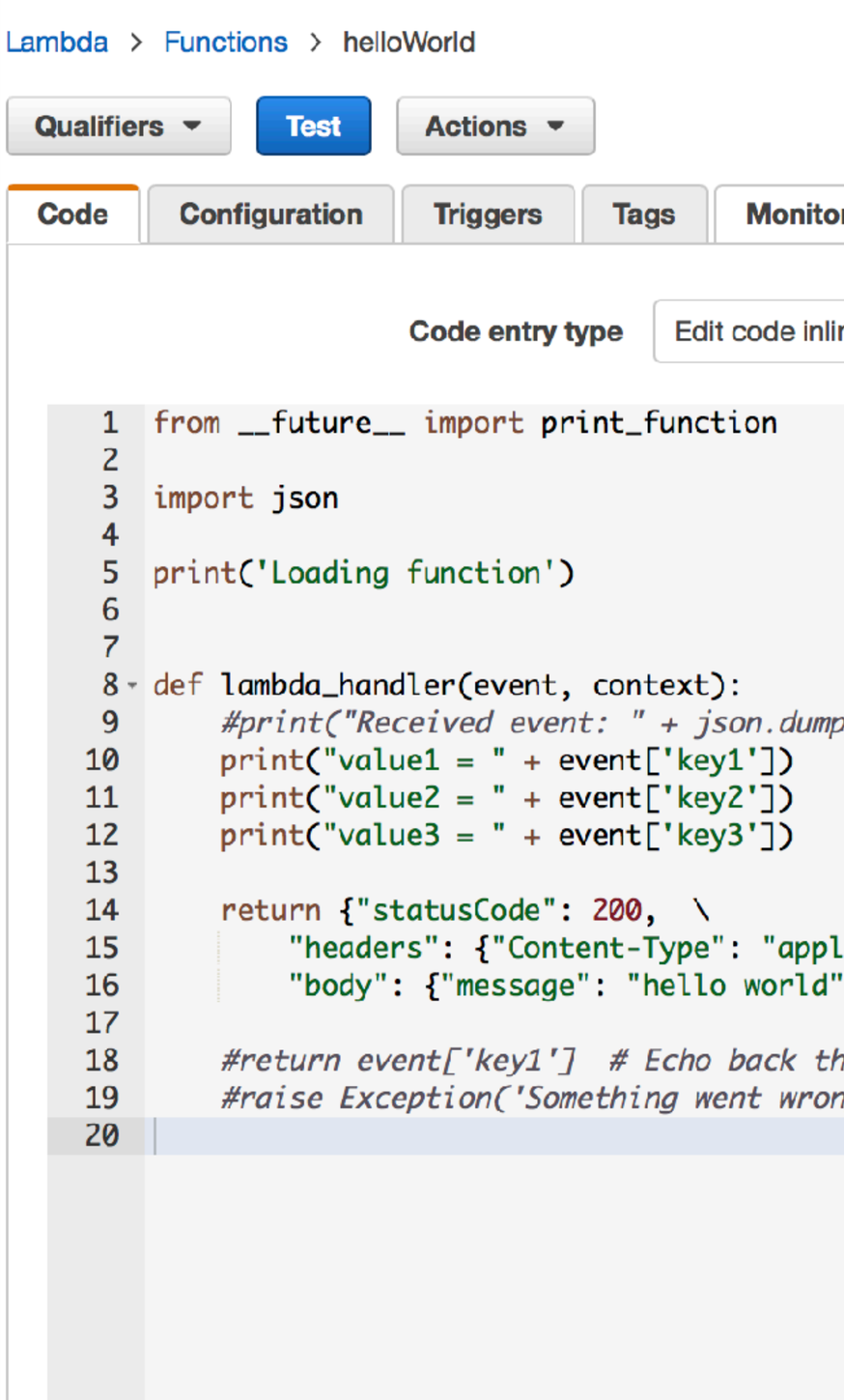
- Firebase
  - Local development server
  - Deploy/test cycle is slow
  - Console only shows logged information



PRICING

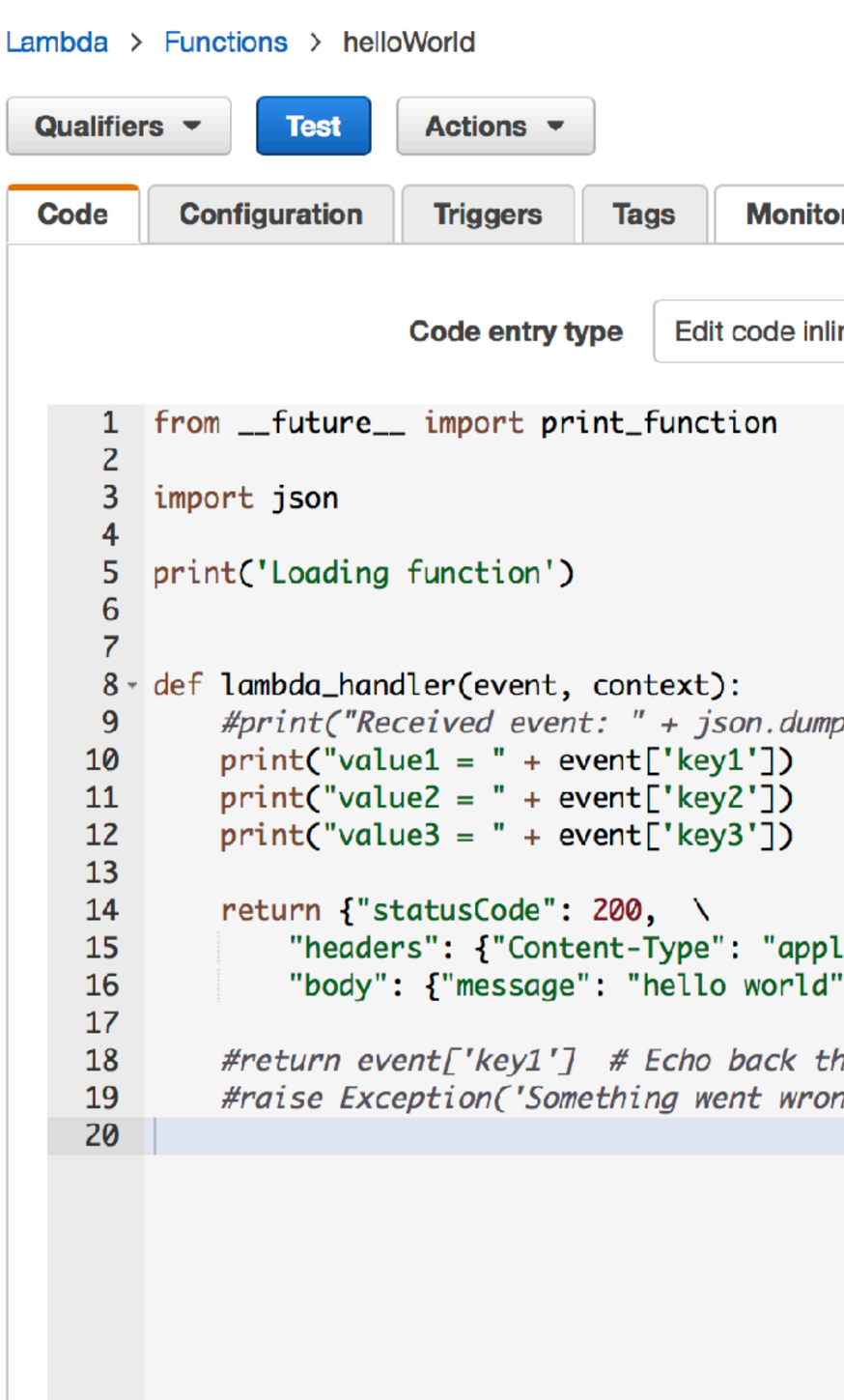
PRICING

- Pricing metrics
  - Number of invocations
  - Duration of invocation
  - Hardware



PRICING

- Cloud Functions
  - $.40 per million (2M free)
- Lambda
  - $.20 per millions (1M free)



ODDS AND ENDS

STATUS AND DOCUMENTATION

- Cloud functions
  - In beta
  - Up and coming
  - Updated documentation

CLOUD FUNCTIONS BETA

A serverless environment to build and connect cloud services

VIEW DOCUMENTATION

VIEW CONSOLE

Serverless Applications on Google's Infrastructure

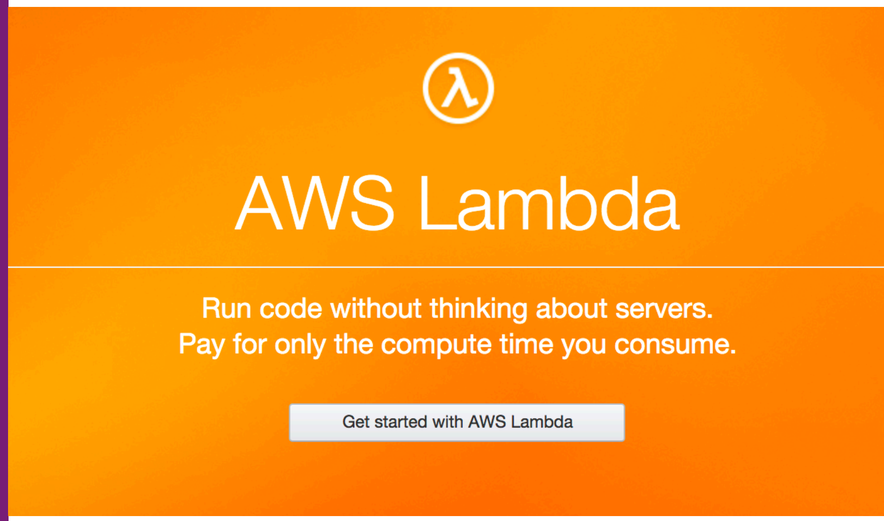
Cloud computing has made possible fully serverless models of computing where logic can be spun up on-de
in response to events originating from anywhere. Construct applications from bite-sized business logic billed
the nearest 100 milliseconds, only while your code is running. Serve users from zero to planet-scale, all witho
managing any infrastructure.

Microservices Over Monolith

Developer agility comes from building syst
functionality focused on doing one thing w
services at the level of a single function, n
VMs.

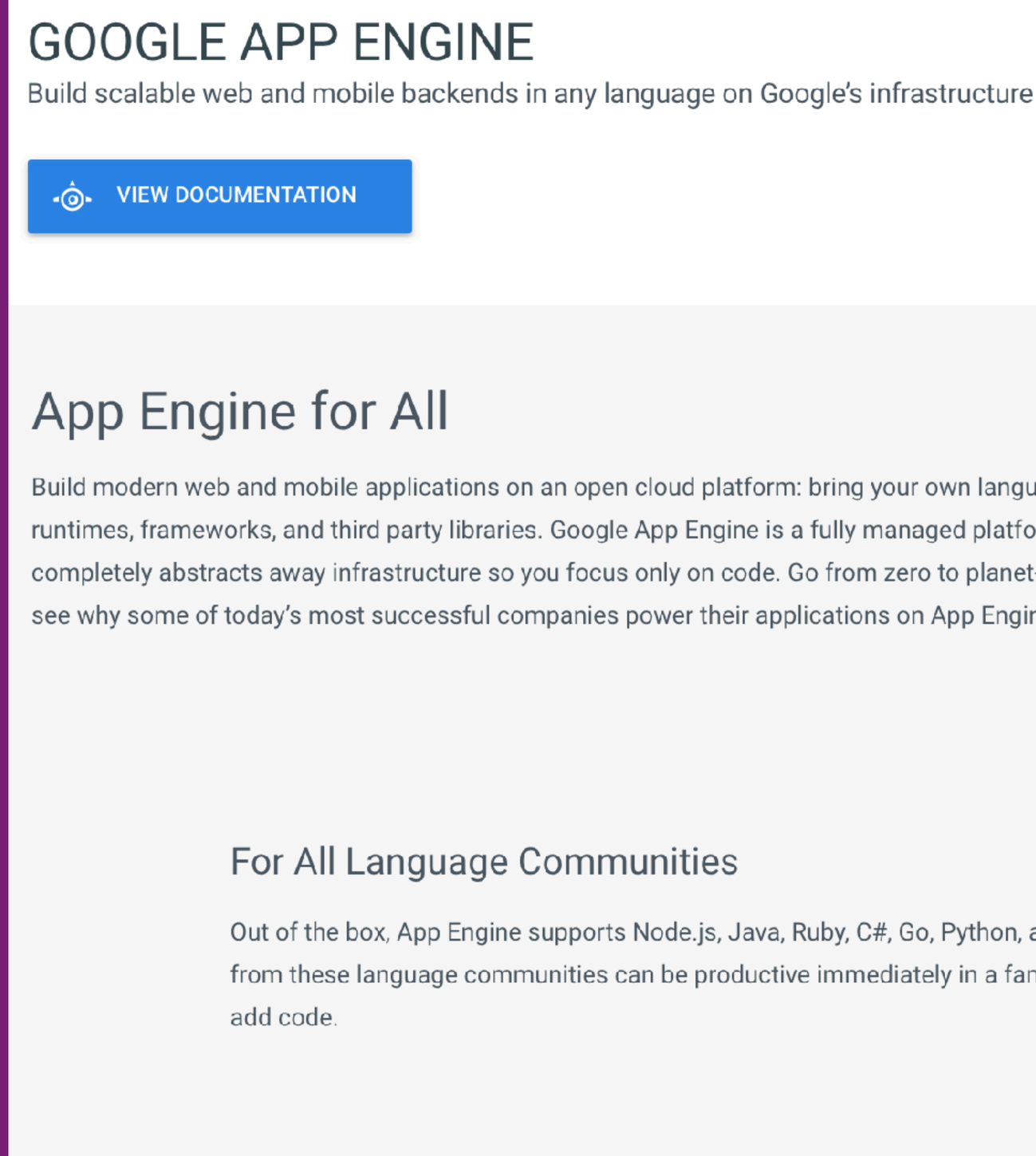
STATUS AND DOCUMENTATION

- Lambda
  - Generally available
  - Documentation is outdated in many circumstances
  - Stack Overflow has correct answers to old problems



STATUS AND DOCUMENTATION

- App Engine
  - More control
  - 3rd party modules
  - Any language



THE UNIVERSITY OF CHICAGO

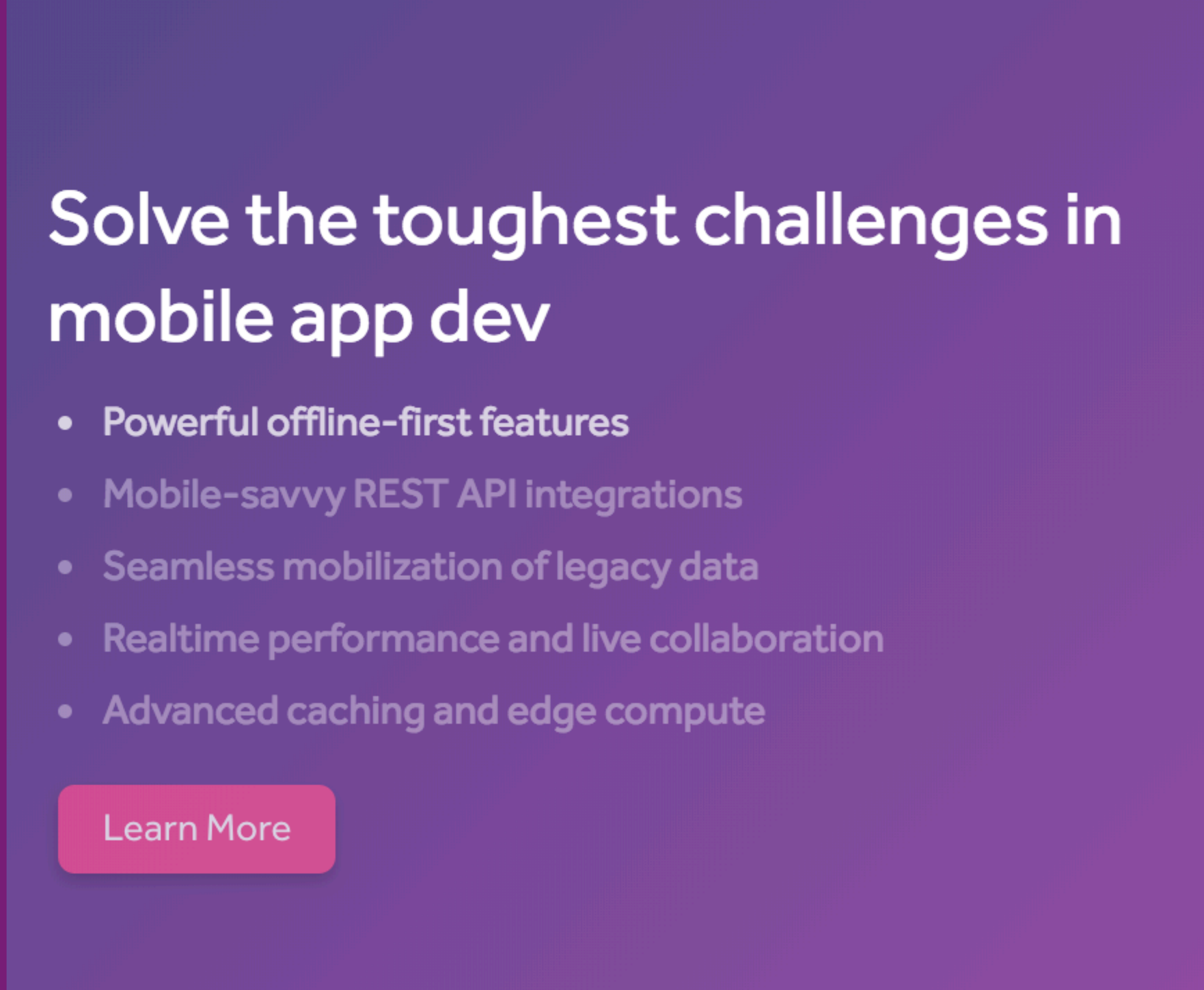
MPCS 51033 • AUTUMN 2019 • SESSION 8

BACKENDS FOR MOBILE APPLICATIONS

REALM

REALM

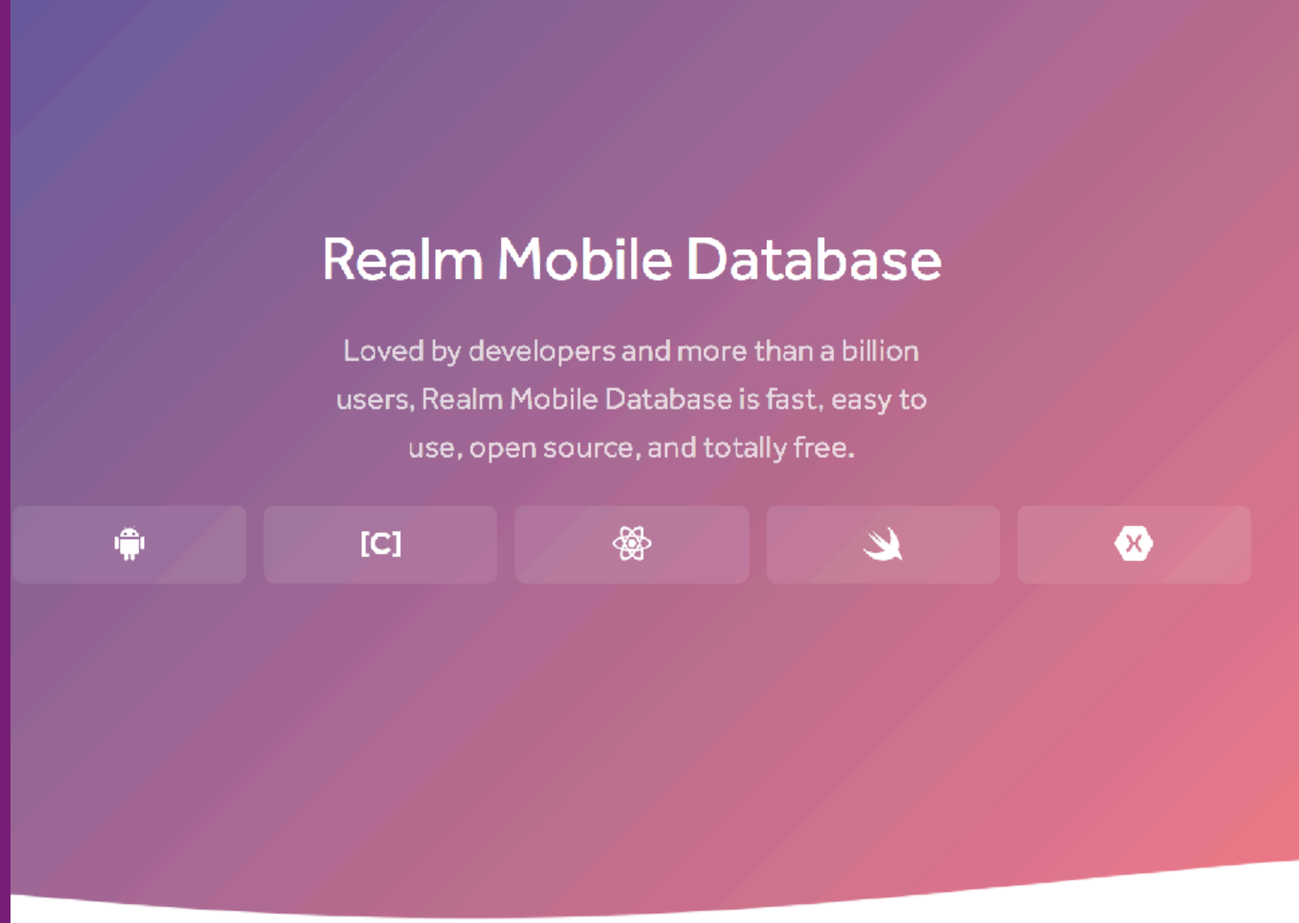
- Mobile first database
- Cross platform
- Full "Platform" for mobile app backends



IN THE BEGINNING…

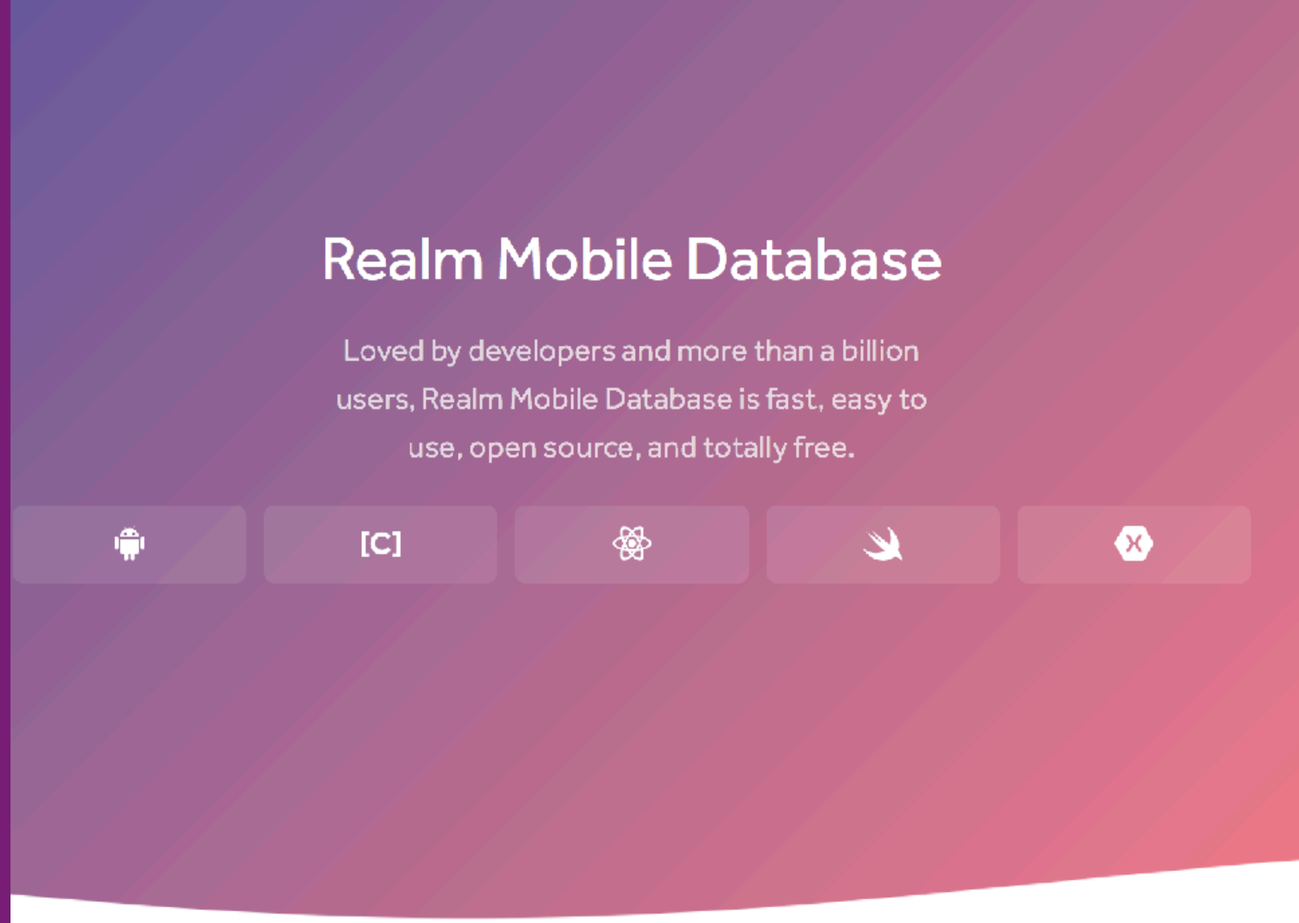
REALM

- Initially on device only
- Billed as a CoreData alternative
  - But easy to use
- Faster than SQLite



REALM

- UI Components tuned for the Realm database
  - UITableViewController/ NSFetchedResultsController



REALM

- Simple **local** object persistence
- Thread safe
- Encryption built in

```
class Dog: Object {
  dynamic var name = ""
  dynamic var age = 0
}

let dog = Dog()
dog.name = "Rex"
dog.age = 1

let realm = try! Realm()
try! realm.write {
  realm.add(dog)
}

let pups = realm.objects(Dog.self).filter("age < 2")
```

REALM

- On-Device limitation until...

Realm Mobile Platform

A flexible platform for creating offline-first, reactive mobile apps effortlessly.

Download the free Developer Edition

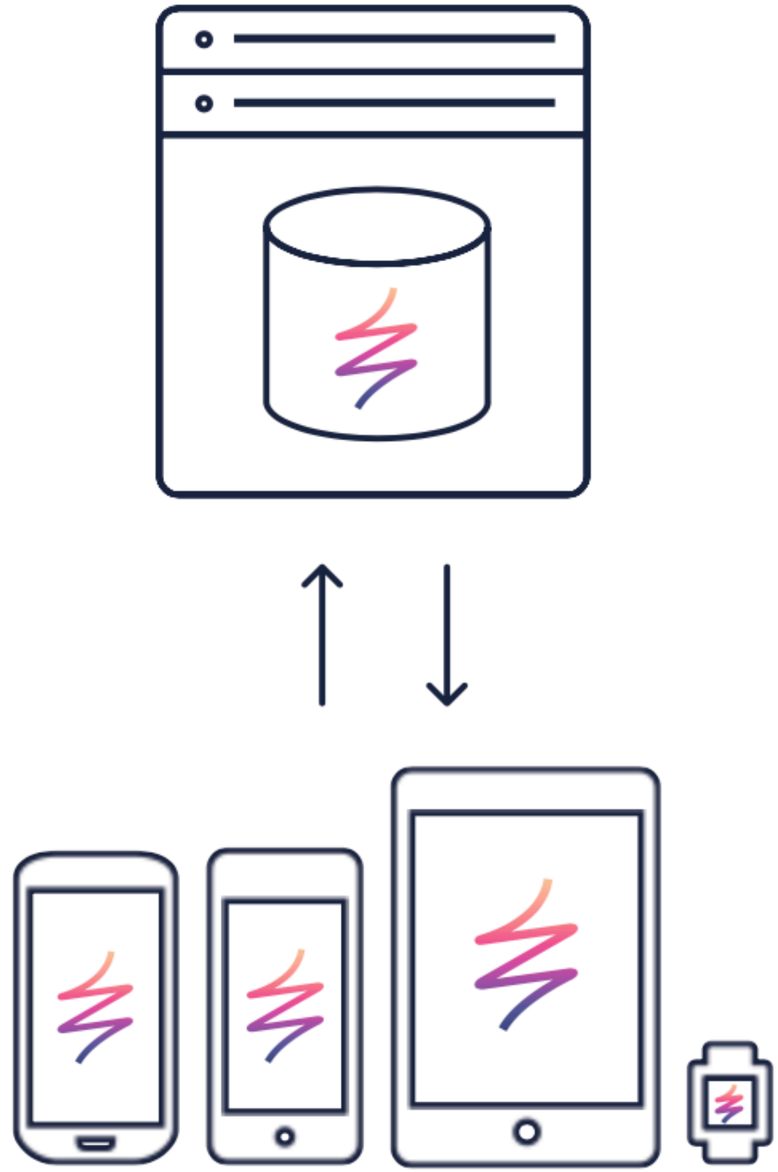
macOS

The perfect backend for the next generation of reactive mobile apps

The Realm Mobile Platform delivers automatic and seamless realtime data sync and powerful event handling between server and devices. You never need to think

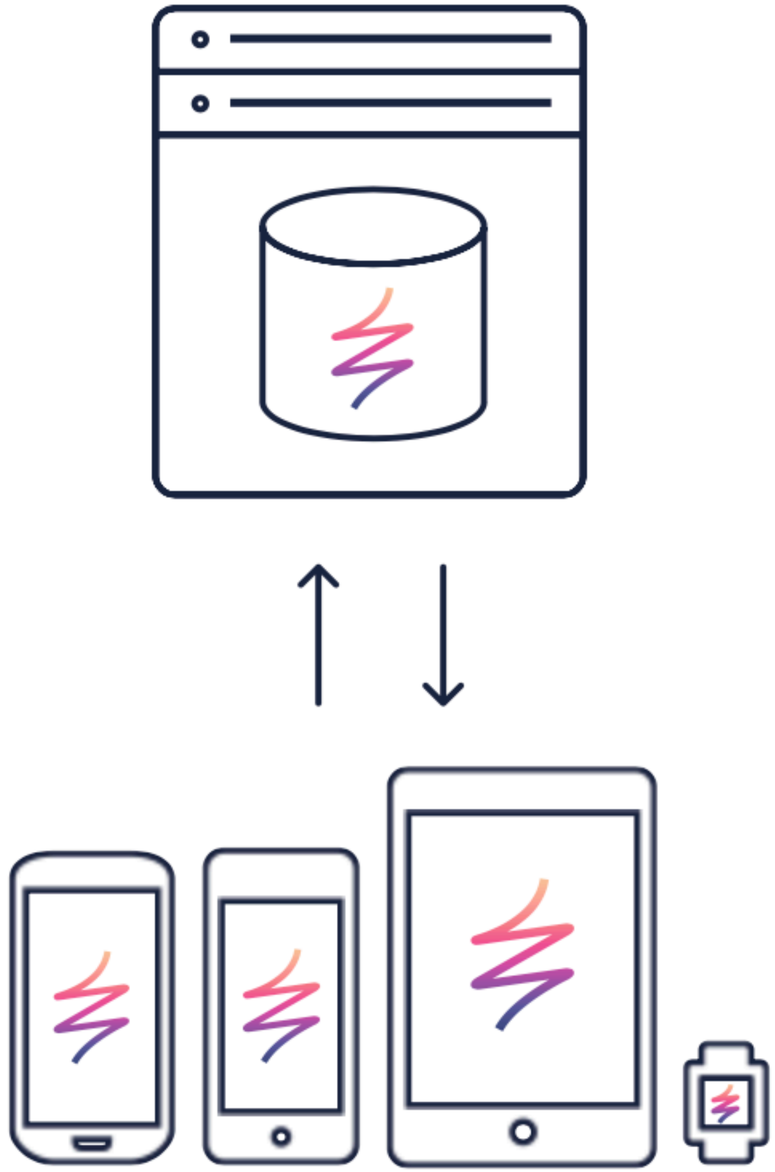
REALM

- Sync engine that runs on any cloud
- Nginx node.js server
- Public or private clouds



REALM

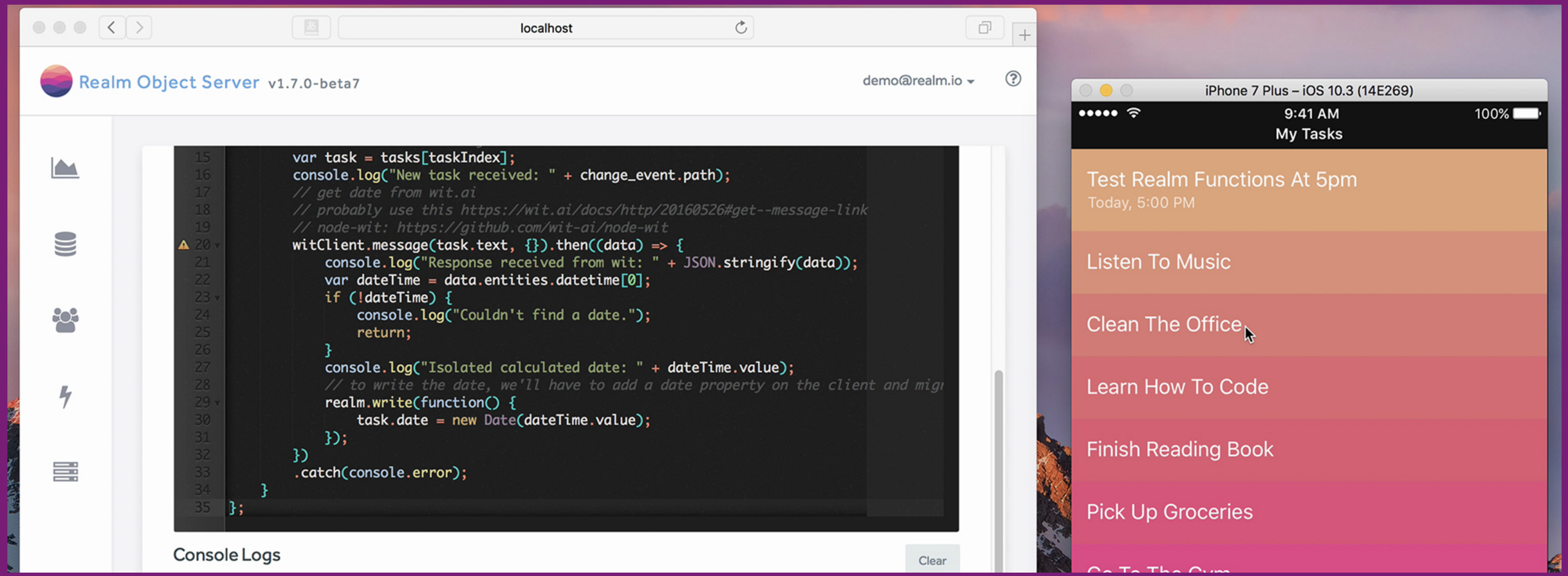
- Local cache and sync
- Live objects with observer model (like Firebase)
- Subscriptions (like CloudKit)



REALM

- But limitation of no server side logic, until....

REALM



- https://news.realm.io/news/serverless-logic-with-realm-introducing-realm-functions/

REALM PLATFORM

REALM PLATFORM

- Database
- Platform
- Studio

Realm Platform 2.0

Everything you love about the Realm Platform just got better:

- New Realm Studio: easily manage your Realms, users, and config
- Easy and simple NPM install
- Improved, fully pluggable authentication system
- A host of stability, performance, and usability enhancements

Watch the webinar or try 2.0 today

REALM PLATFORM

- Realm Object Server (ROS)
  - Realtime sync objects (no REST API)
  - Objects are live (no transport layer)
  - No networking code (that you write)

REALM PLATFORM

```
curl -s https://raw.githubusercontent.com/realm/realm-object-server/master/install.sh | bash

ros init my-app

cd my-app/
npm start
```

- Supports Ubuntu, Mac, Windows
  - Public/private clouds
- Run install script on server to download and configure

REALM PLATFORM

- Supports Ubuntu, Mac, Windows
- Run install script on server to download and configure

DOES MORE THAN SYNC

Realm Database

Our fast and reactive database is superior to SQLite-based alternatives as an embedded "live object" database on the device. And when you connect it to Realm Object Server, it becomes a distributed database providing automatic, realtime data synchronization.

Realm Sync

At the heart of Realm Object Server, automatically synchronizing data objects across all devices and the servers in realtime, is Realm Sync. It handles conflict resolution and offline states seamlessly — and your data is safe with TLS/SSL and AES-256 encryption.

Realm Studio

Functioning as your dashboard and your cockpit, Realm Studio gives you control over your data, platform functions, users, and configuration. Featuring an efficient, task-oriented UI, it's built on Electron and it works across every major platform.

Re m Connect

m Object Server, ect converts ex g REST APIs and data sources to live objects, freeing you to focus on features rather

Event handling

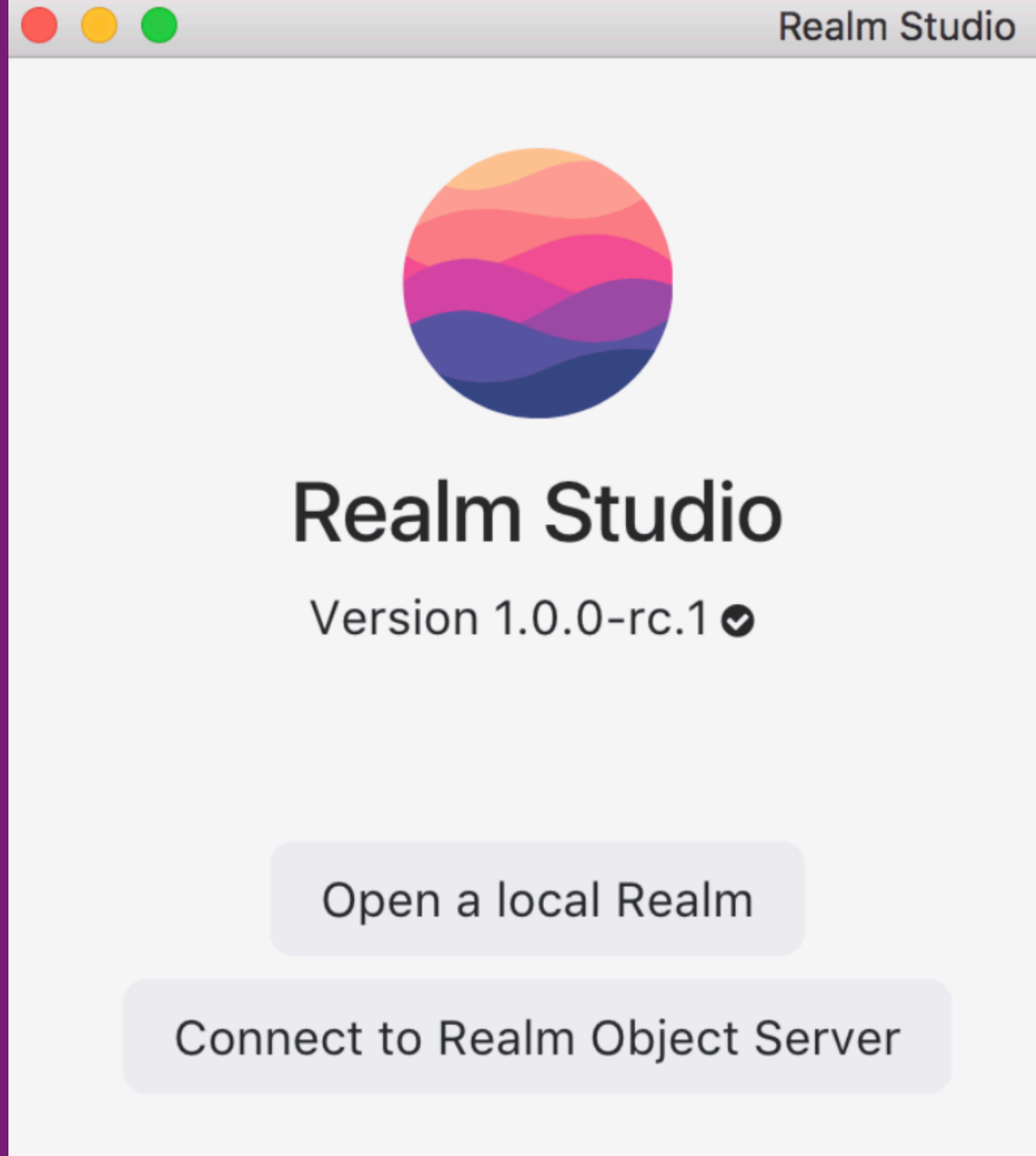
Realm Object Server's event handling functionality means you can easily build server-side features with simple JavaScript. When synced data

Permissions and authentication

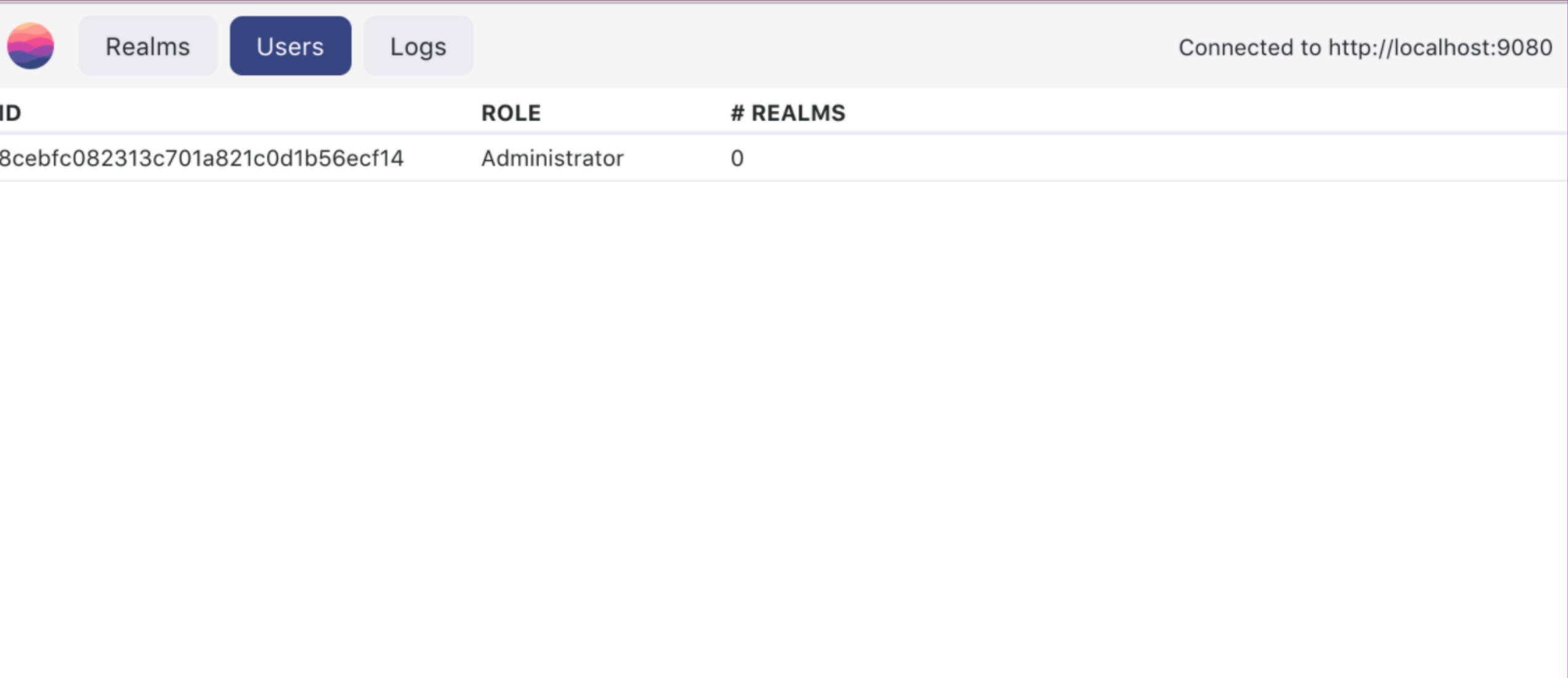
Log in users with our authentication systems, or customize authentication to work with your existing

REALM PLATFORM

- Realm Studio app
  - Manage server data
    - Add, delete, edit
  - View logs

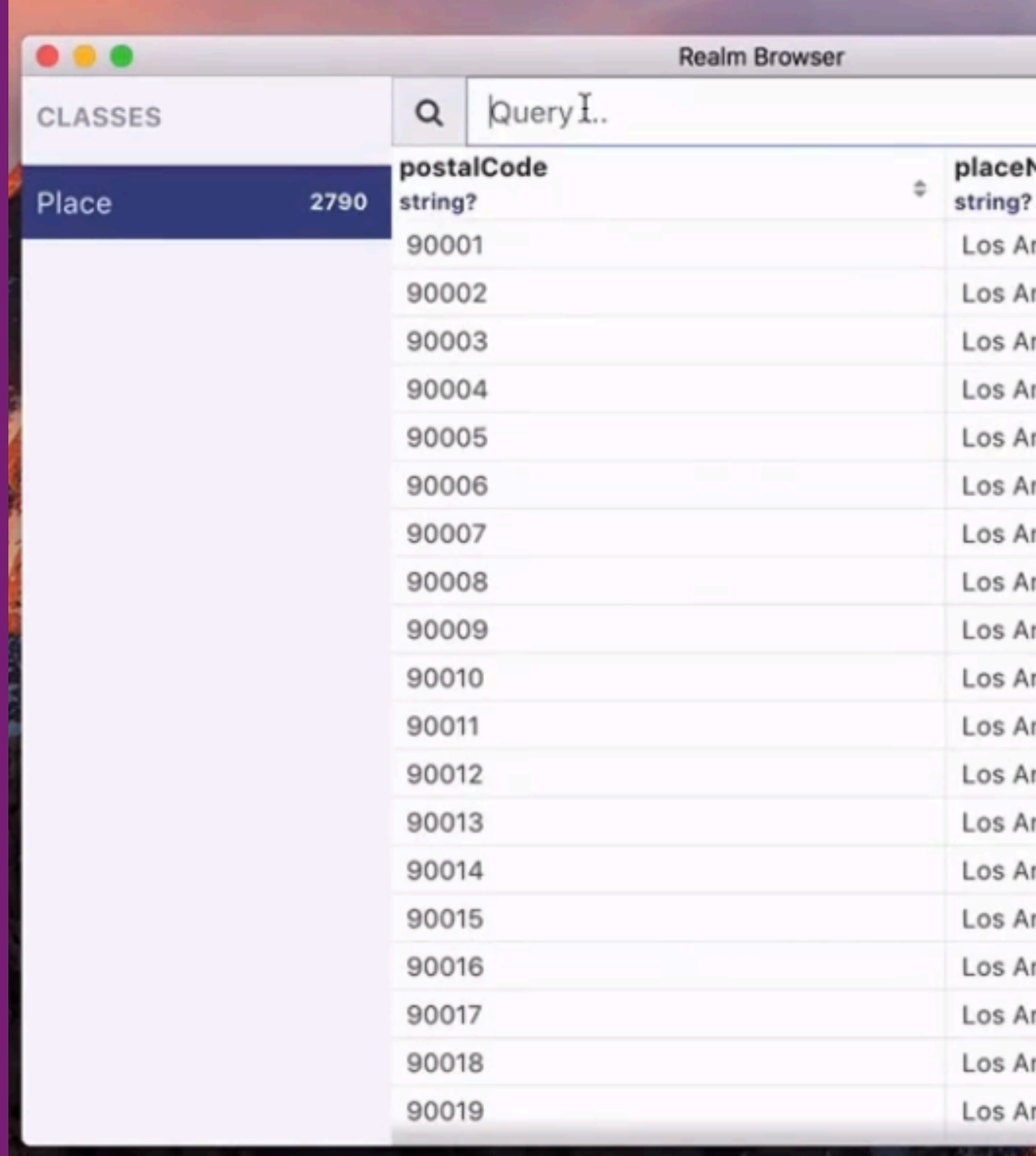


REALM PLATFORM



REALM PLATFORM

- How does the sync engine work?
  - App Engine, explicit calls
  - Firebase 🤷‍♀️
  - CloudKit, subscriptions



REALM PLATFORM

- Realm uses notifications that are sent when it observes a change
  - Realms
  - Collections
  - Objects

```
// Observe Realm Notifications
let token = realm.observe { notification, realm in
    viewController.updateUI()
}

// later
token.invalidate()
```

REALM PLATFORM

- Collections

```
// Observe Results Notifications
notificationToken = results.observe { [weak self] (changes: RealmCollectionChange) in
    guard let tableView = self?.tableView else { return }
    switch changes {
    case .initial:
        // Results are now populated and can be accessed without blocking the UI
        tableView.reloadData()
    case .update(_, let deletions, let insertions, let modifications):
        // Query results have changed, so apply them to the UITableView
        tableView.beginUpdates()
        tableView.insertRows(at: insertions.map({ IndexPath(row: $0, section: 0) }),
                             with: .automatic)
        tableView.deleteRows(at: deletions.map({ IndexPath(row: $0, section: 0)}),
                             with: .automatic)
        tableView.reloadRows(at: modifications.map({ IndexPath(row: $0, section: 0) }),
                             with: .automatic)
        tableView.endUpdates()
    case .error(let error):
        // An error occurred while opening the Realm file on the background worker thread
        fatalError("\(error)")
    }
}
```

REALM PLATFORM

- Objects

```
class StepCounter: Object {
    @objc dynamic var steps = 0
}

let stepCounter = StepCounter()
let realm = try! Realm()
try! realm.write {
    realm.add(stepCounter)
}
var token : NotificationToken?
token = stepCounter.observe { change in
    switch change {
    case .change(let properties):
        for property in properties {
            if property.name == "steps" && property.newValue as! Int > 1000 {
                print("Congratulations, you've exceeded 1000 steps.")
                token = nil
            }
        }
    case .error(let error):
        print("An error occurred: \(error)")
    case .deleted:
        print("The object was deleted.")
    }
}
```

REALM PLATFORM

- Realm takes adventage of Key-Value-Observing (KVO)
- KVO is a mechanism enables objects to register for asynchronous notifications driven by changes in another object's properties

```
import Foundation

//: # Santa object subclass of NSObject #
@objcMembers  // Activate objective-c's introspection capabiliti
class Santa: NSObject {

  // The dynamic declaration modifier is required whenever you r
  // make use of Objective-C's dynamism
  dynamic var string: String

  override init() {
    string = "🎅"
    super.init()
  }
}

//: # Print original Santa #
let santa = Santa()
print("Original: \(santa.string)")

//: # Set up an observer #
// "\." is shorthand for Santa (in this case); new Key Path in S
let observation = santa.observe(\.string) { (santa, change) in
  print("We observed a change in a property.  The new values is
}

//: # Make an observable change #
// We are chaing this...watch what happens
print("Getting read to change....")
santa.string = "santa"
```

REALM PLATFORM

```
import Foundation

//: # Santa object subclass of NSObject #
@objcMembers  // Activate objective-c's introspection capabilities
class Santa: NSObject {

  // The dynamic declaration modifier is required whenever you need to
  // make use of Objective-C's dynamism
  dynamic var string: String

  override init() {
    string = "🎅"
    super.init()
  }
}

//: # Print original Santa #
let santa = Santa()
print("Original: \(santa.string)")
```

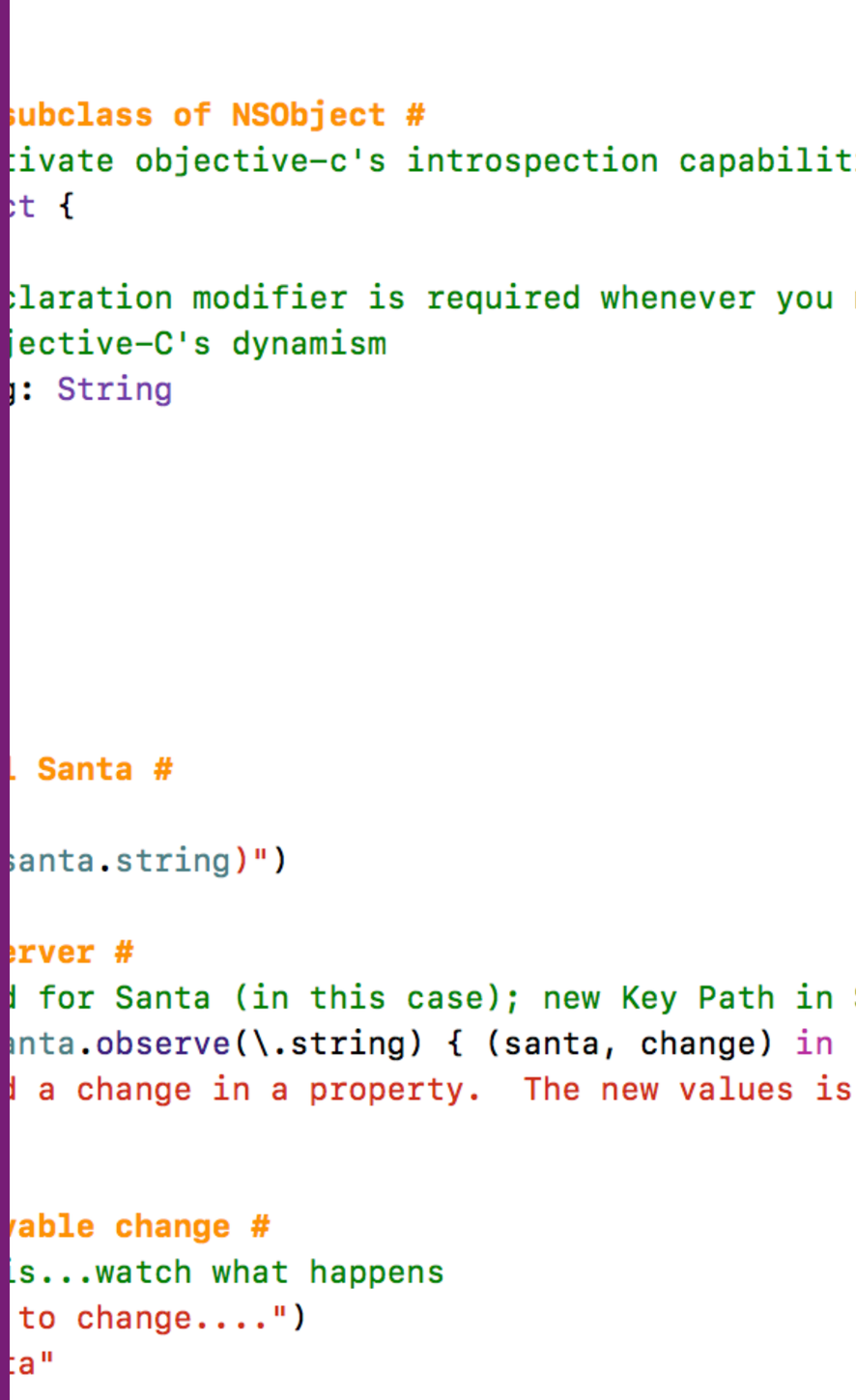
REALM PLATFORM

- When a change happens on device, it triggers update to server
- When a change happens on server, it triggers update on device

```
import Foundation

//: # Santa object subclass of NSObject #
@objcMembers  // Activate objective-c's int
class Santa: NSObject {

  // The dynamic declaration modifier is re
  // make use of Objective-C's dynamism
  dynamic var string: String

  override init() {
    string = "🎅🏻"
    super.init()
  }
}

//: # Print original Santa #
let santa = Santa()
print("Original: \(santa.string)")

//: # Set up an observer #
// "\." is shorthand for Santa (in this cas
let observation = santa.observe(\.string) {
  print("We observed a change in a property
}

//: # Make an observable change #
// We are chaing this...watch what happens
print("Getting read to change....")
santa.string = "santa"
```

REALM PLATFORM

- Similar to CloudKit subscriptions in that changes that match a predictate trigger a notification to be sent to all clients
    - Very limited amount of work setting it up and handling the notifications in CloudKit



REALM PLATFORM

- https://academy.realm.io/posts/learning-path-build-a-realtime-swift-app-with-realm/

Build a Realtime Swift App with Realm

Get hands-on with the Realm Platform, build a realtime Santa-tracking app, and learn ne...

REALM PLATFORM

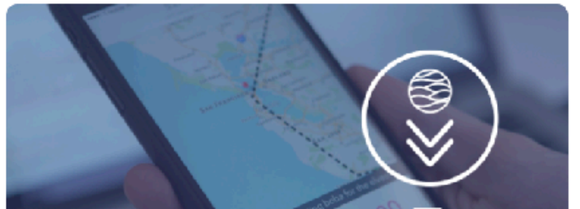
- https://academy.realm.io/posts/learning-path-build-a-realtime-swift-app-with-realm/

Track Santa with Realm: Part 1

Learn how to build a Swift app with the Realm Platform by tracking Santa on his global tour!

Track Santa with Realm: Part 2

Build on your app by completing the data models and making your UI reactive to data changes.

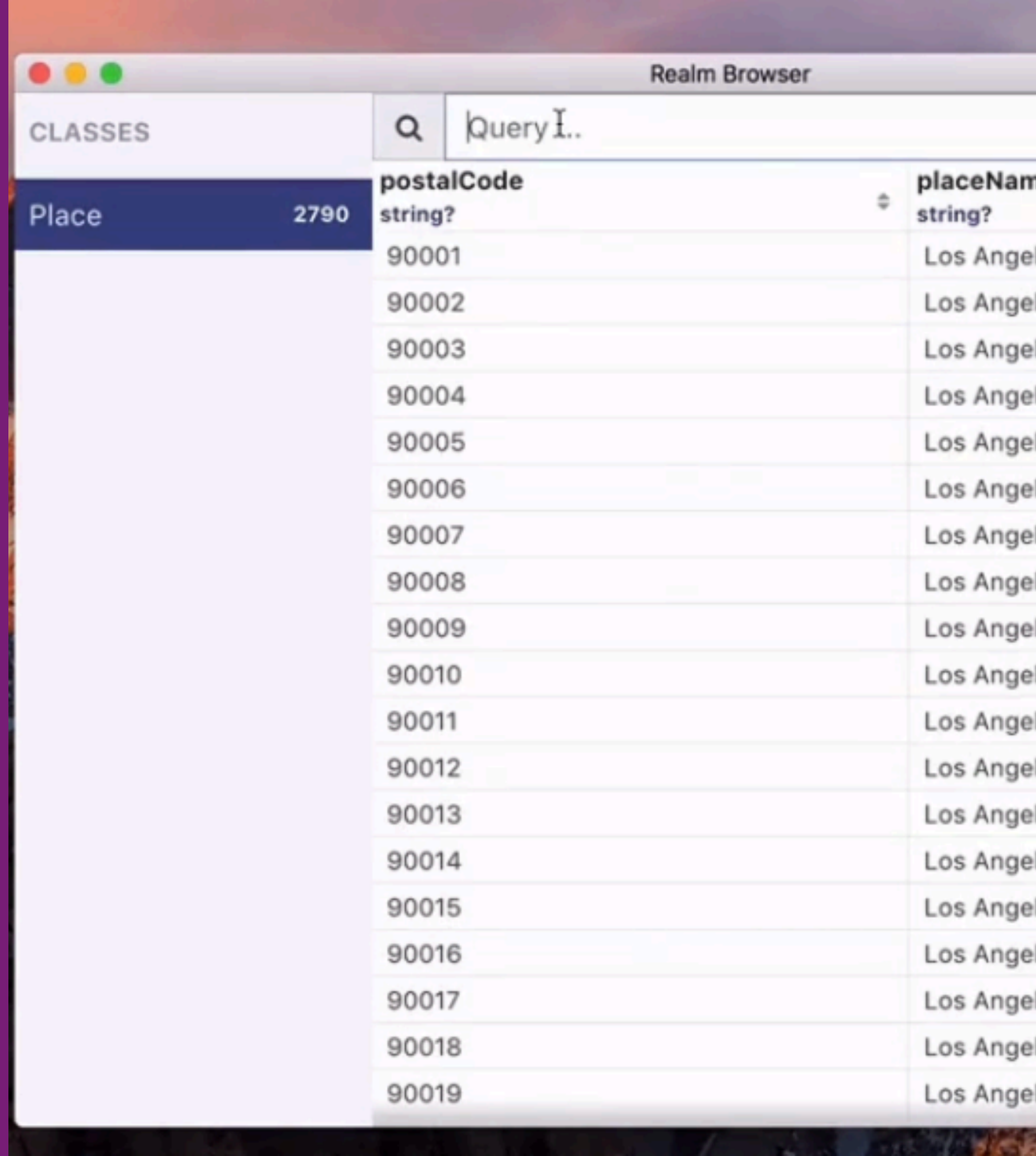


Track Santa with Realm: Part 3

TO REALM OR NOT TO REALM

TO REALM

- Realm checks off all the boxes for a mobile backend
  - Local persistence
  - Sync
  - SDK with networking
  - Authentication



REALM PLATFORM

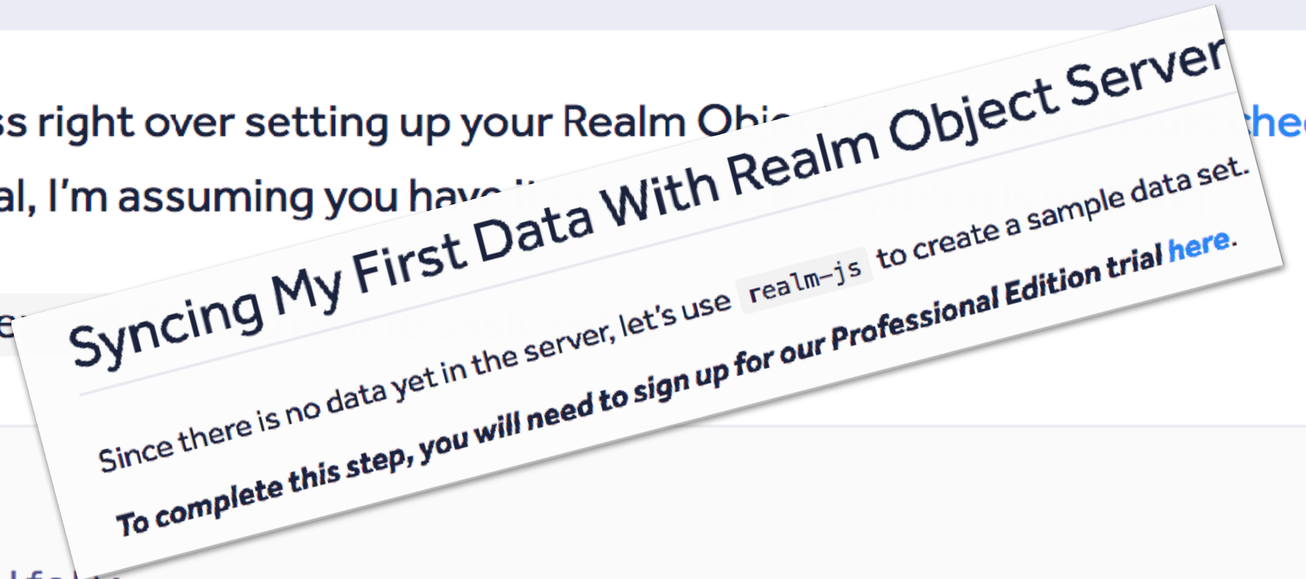
2. Handling the data on the server

This is the cool part. Where the magic happens. Granted, this example isn't very *exciting* magic, but hey, it's magic.

> Server-side data access is only available through the Professional or Enterprise Editions of the Realm Platform. You can read more about those, or sign up for a free trial, on the Pricing page.

So I'm going to gloss right over setting up your Realm Obj… …heck out the… …how to do that. For this tutorial, I'm assuming you hav…

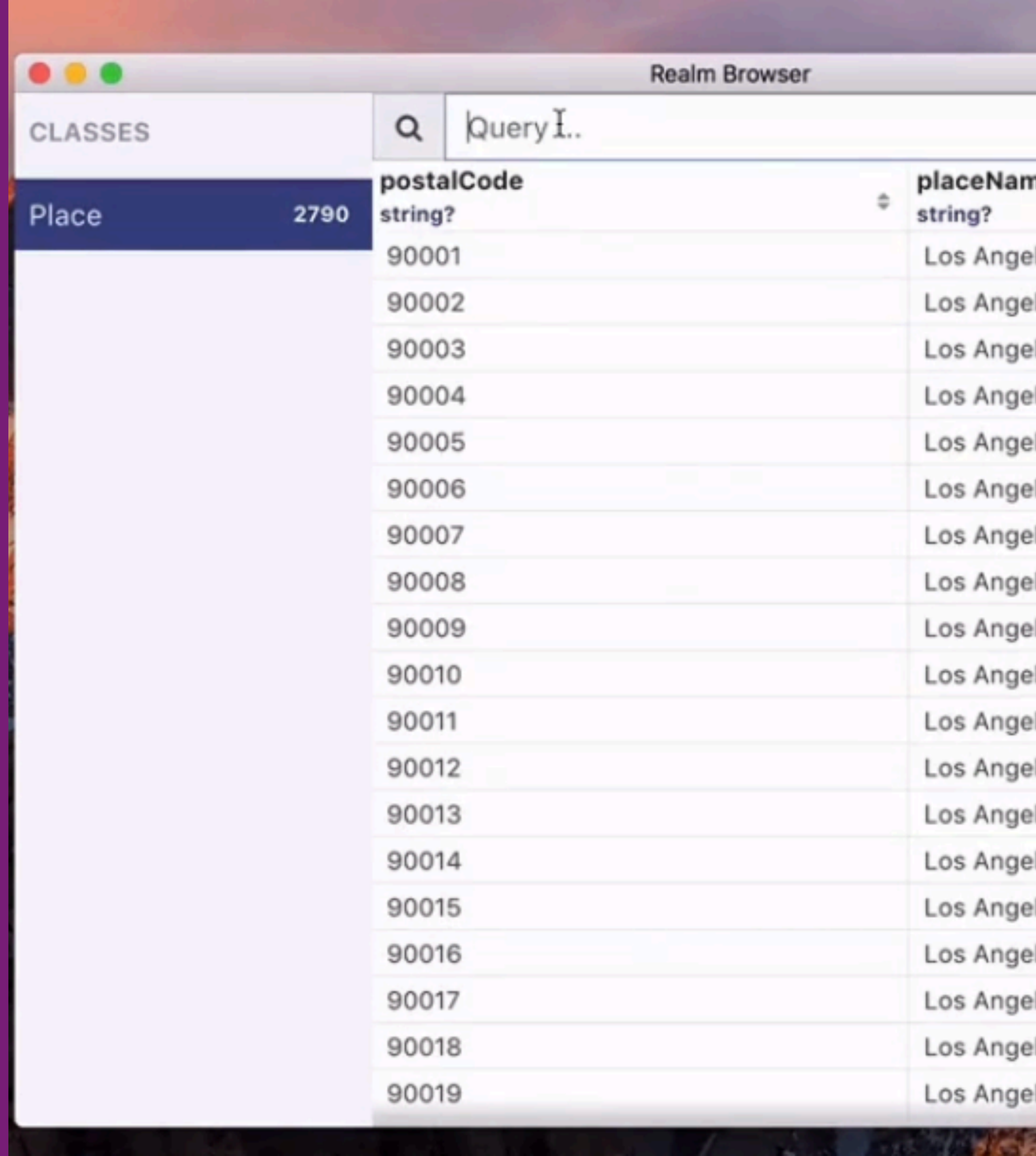
We'll open our `inde…`




```
'use strict';
```

TO REALM

- Professional edition
  - Event handling (functions)
  - Server access
  - Data migration tools
  - Horizontal Scaling?



REALM PLATFORM

Developer Edition

Free

Enable realtime two-way data sync for mobile devices and servers.

No cost to use, even in production for commercial use cases. Read the terms.

Get the download

Professional Edition

Free Trial

Powerful features like simplified integrations, event handling, server-side data access.

14-day free trial; pricing starts at $1,750 per month.

Download free tri

Enterprise Edition

Free Demo

Enterprise scale, availability, and integration with existing infrastructure/third-party APIs.

Pricing scales with use case.

Contact us for d o

TO REALM

- A hybrid of Firebase and CloudKit
  - Less blackbox
  - Requires some "fiddling" on the server
- Privacy (end-to-end control)
- Cross-platform