



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

Dec 3, 2015

## Swift is Open Source

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

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

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

- It all started here

## Swift.org

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

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

# 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

Standard Library

Standard Library

System Libraries/Frameworks

System Libraries

Darwin

Linux

- Something is missing...

# OPEN SOURCE SWIFT

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



# Swift

---

[ABOUT SWIFT](#)

---

[BLOG](#)

---

[DOWNLOAD](#)

---

[GETTING STARTED](#)

---

[DOCUMENTATION](#)

---

[SOURCE CODE](#)

---

[COMMUNITY](#)

---

[CONTRIBUTING](#)

---

[CONTINUOUS  
INTEGRATION](#)

---

---

[PROJECTS](#)

---

[COMPILER AND  
STANDARD LIBRARY](#)

---

[PACKAGE MANAGER](#)

---

[CORE LIBRARIES](#)

---

[REPL AND DEBUGGER](#)

# Welcome to Swi

Swift is now open source!

We are excited by this new chapter in the stor unveiled the Swift programming language, it q the fastest growing languages in history. Swift software that is incredibly fast and safe by des open source, you can help make the best gen programming language available everywhere.

For students, learning Swift has been a great i modern programming concepts and best pra is now open, their Swift skills will be able to be broader range of platforms, from mobile devic the cloud.

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

***– The Swift Team***

# STATE OF SWIFT ON THE SERVER

# STATE OF SWIFT ON THE SERVER

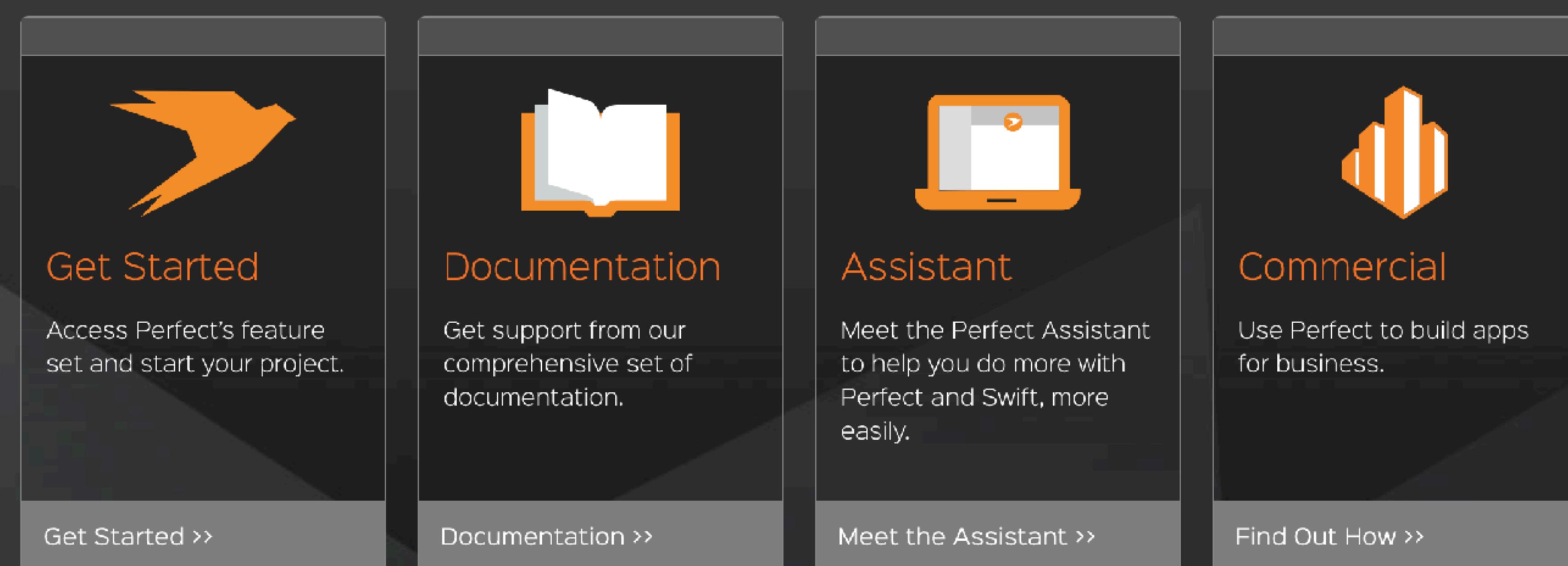
- Many Swift web frameworks in development
- Most popular (15,000+ stars)



Zewo

# STATE OF SWIFT ON THE SERVER

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



The screenshot shows the homepage of perfect.org. At the top, there's a navigation bar with links for "WHAT IS PERFECT?", "DOCUMENTATION", "DEVELOPER RESOURCES", and "EVENTS". Below this, there are four main sections: "Get Started" (with an orange bird icon), "Documentation" (with an orange book icon), "Assistant" (with an orange laptop icon), and "Commercial" (with an orange cloud icon). Each section has a brief description and a "Get Started >>" or "Find Out How >>" button at the bottom.

- Get Started**  
Access Perfect's feature set and start your project.  
[Get Started >>](#)
- Documentation**  
Get support from our comprehensive set of documentation.  
[Documentation >>](#)
- Assistant**  
Meet the Perfect Assistant to help you do more with Perfect and Swift, more easily.  
[Meet the Assistant >>](#)
- Commercial**  
Use Perfect to build apps for business.  
[Find Out How >>](#)



This part of the screenshot shows two calls-to-action on an orange background. On the left, there's a video camera icon with the text "TAKE ME TO THE LEARNING CENTRE". On the right, there's text that says "Show me Ray Wenderlich's video tutorials >>".

## What is Perfect?

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

# STATE OF SWIFT ON THE SERVER

- Perfect
  - A complete framework
  - Runs 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

The future of web development.

GET STARTED

JOIN CHAT



```
import Vapor

let app = try Application()
let router = try app.make(Router.self)

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

try app.run()
```



Non-blocking, event-driven architecture built on top of Apple's



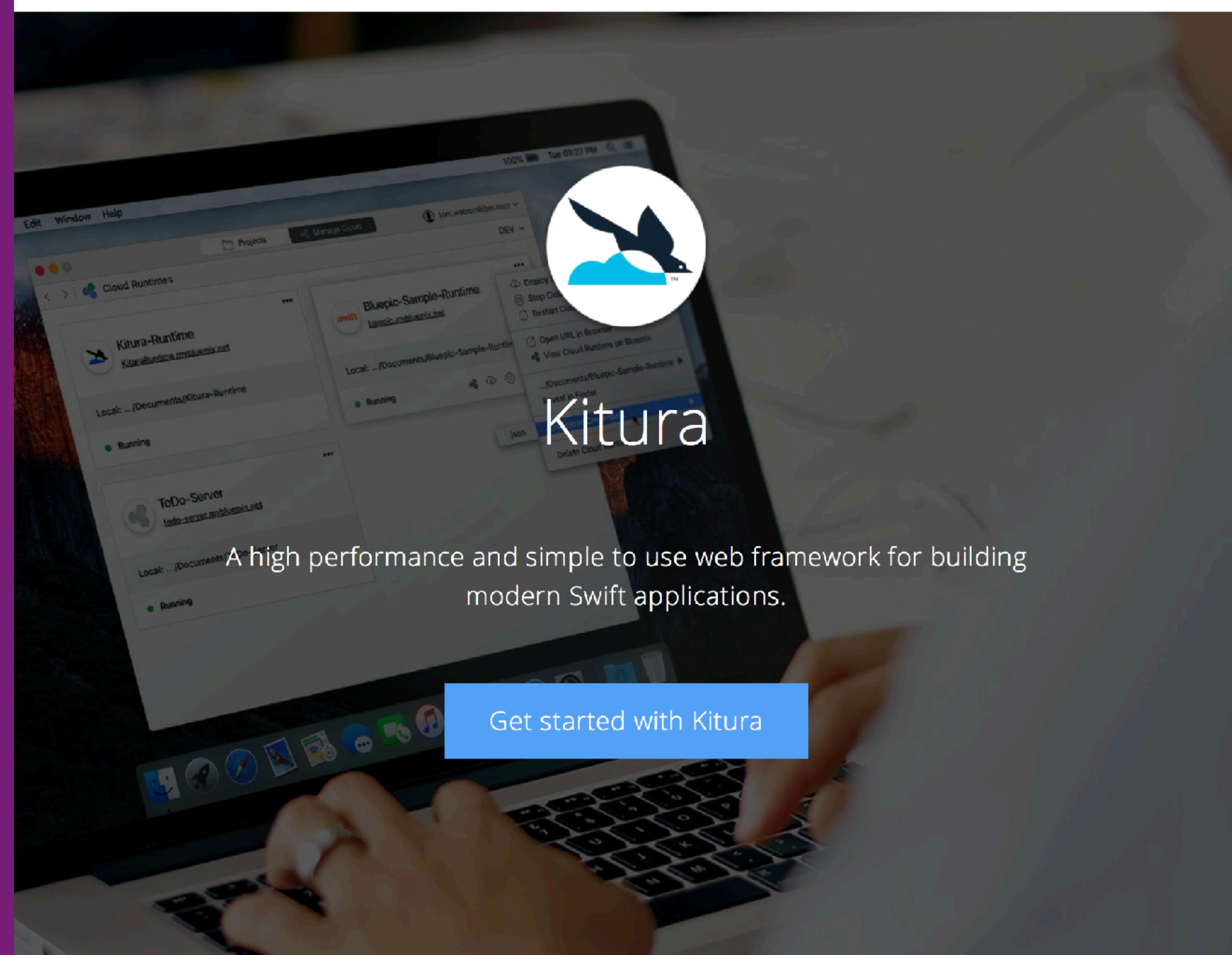
Written in Swift, the powerful programming language that is also



Expressive, protocol-oriented design with a focus on type safety.

# 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

open-swift / S4

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs

## HTTP standards for Swift

264 commits 2 branches 27 releases 12 contributors

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

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

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

S4 - HTTP

# STATE OF SWIFT ON THE SERVER

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

apple / swift-nio

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 yesterday

.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 generate optimal code... ...

Tests HTTP1TestServer: close accepted channel on 'stop' (#1244) ...

dev make-single-file-spm: support lines ending in 'n' (#1250) ...

docker EventLoopFuture: save one allocation per future (#1224) ...

docs Add an optimization tips document. (#1024) ...

scripts perf tests: switch printed metric to mean (#1244) ...

.gitignore assert EventLoopGroup::syncShutdownGracefully is not null (#1223) ...

.mailmap update contributors (#1225) ...

CODE\_OF\_CONDUCT.md update conduct email group (#1120) ...

CONTRIBUTING.md Make clear generate\_linux\_tests must be run (#113) ...

**SwiftNIO: A simple guide to async on the server**

By Joannis Orlando Feb 28 2019 · Article (20 mins) · Intermediate

Download Materials

5/5 ★★★★★ 2 Ratings

Version Swift 4.2, macOS 10.14, Xcode 10

An important topic in server-side Swift is asynchronous programming. This tutorial teaches you how to work with two important aspects of async programming: futures and promises, using SwiftNIO.



VAPOR

# VAPOR

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

vapor / vapor

Code Issues 0 Pull requests 1 Pulse Graphs

A server-side Swift web framework. <https://vapor.codes>

vapor swift server-side-swift web-framework server framework

3,196 commits 6 branches 172 releases 77 contributors MIT

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

tanner0101 committed on GitHub Merge pull request #1012 from vapor/readme-update ... Latest commit ca5cb62 6 hours ago

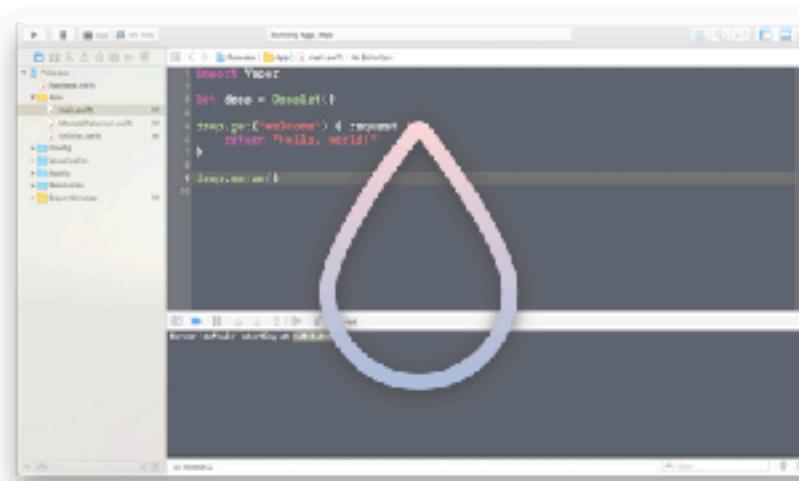
File	Description	Time
Documents	Adding another break line	3 months ago
Sources	Sessions Middleware doesn't require Foundation anymore	10 hours ago
Tests	Merge branch 'master' into sessionsMiddlewareImprovements	8 hours ago
Utilities	allow xcode 8.1	8 months ago
.codecov.yml	remove unnecessary files	a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

# VAPOR

- Great documentation and resources



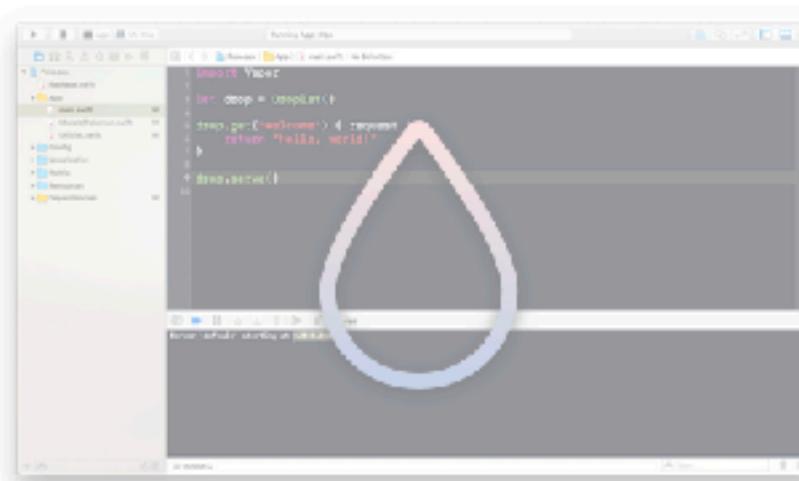
vapor.university



## Look Ma! Server Side Swift Using Vapor

A detailed introduction to using Vapor and a SQLite database.

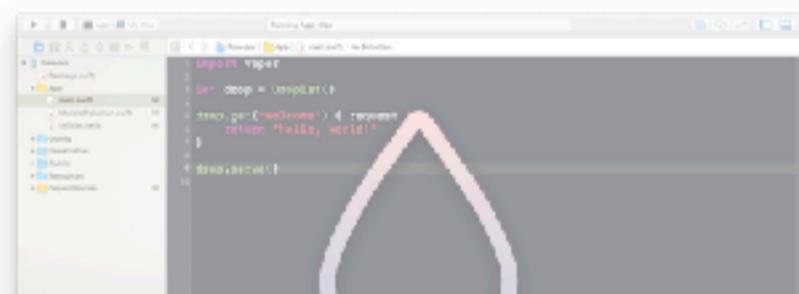
15m • Easy • Mohammad Azam



## 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 — bash — 49x20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
[530 % curl -sL check.vapor.sh | bash
    ✓ Compatible
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
531 % ]
```

# VAPOR

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

```
vapor — git-remote-https ▾ brew install vapor/tap/vapor — 49x20  
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor  
[534 % brew install vapor/tap/vapor  
█
```

# VAPOR

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

```
[504 % vapor --help
Usage: vapor command

Join our team chat if you have questions, need help,
or want to contribute: http://vapor.team

Commands:
  new    Creates a new Vapor application from a template.
        Use --template=repo/template for github templates
        Use --template=full-url-here.git for non github templates
        Use --web to create a new web app
        Use --auth to create a new authenticated API app
        Use --api (default) to create a new API
  build   Compiles the application.
  run     Runs the compiled application.
  fetch   Fetches the application's dependencies.
  update  Updates your dependencies.
  clean   Cleans temporary files--usually fixes
        a plethora of bizarre build errors.
  test    Runs the application's tests.
  xcode   Generates an Xcode project for development.
        Additionally links commonly used libraries.
  version Displays Vapor CLI version
  cloud   Commands for interacting with Vapor Cloud.
  heroku  Commands to help deploy to Heroku.
  provider Commands to help manage providers.

Use `vapor command --help` for more information on a command.
tabinkowski:Users/tabinkowski
505 %
```

# VAPOR

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

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

# VAPOR

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

📁 vapor — -bash — 50×20



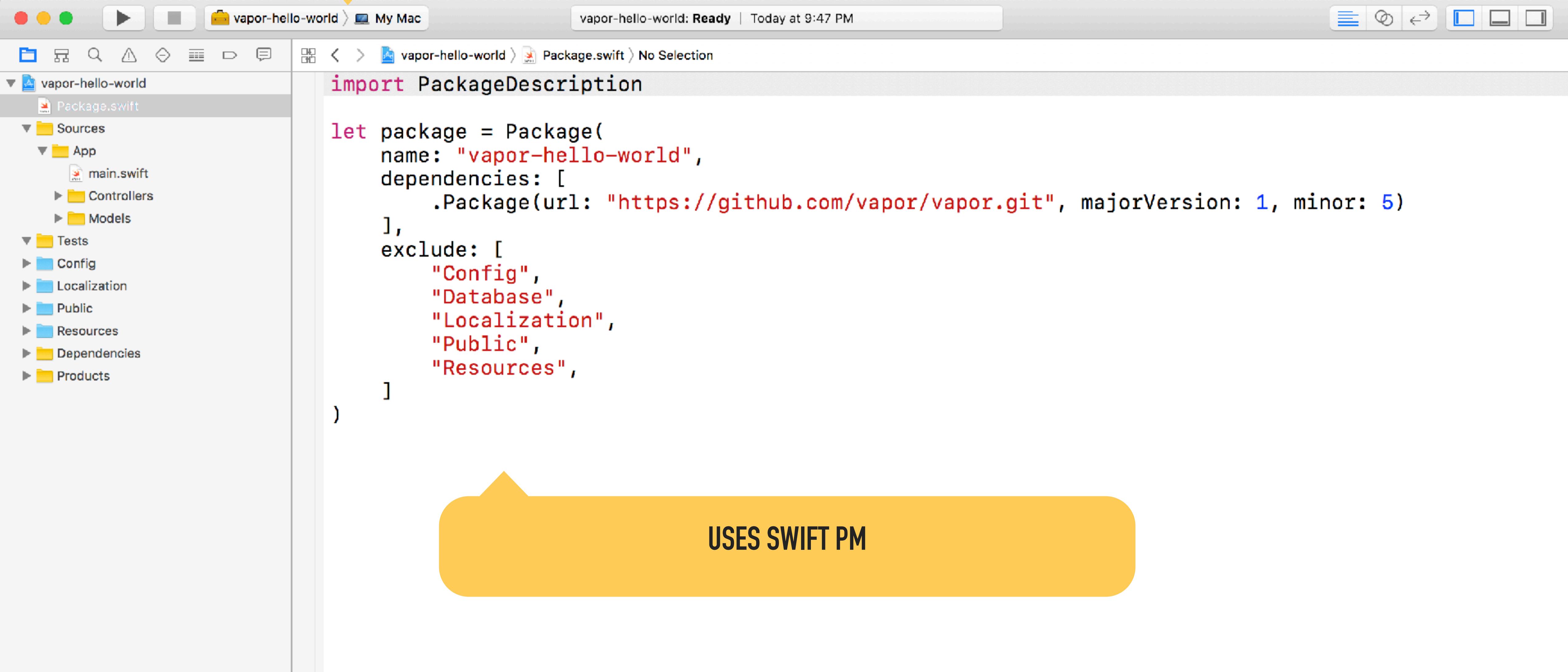
# VAPOR

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

```
vapor-hello-world — swift-package • vapor xcode — 50x20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor
[557 % cd vapor-hello-world/
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[558 % vapor xcode
No .build folder, fetch may take a while...
Fetching Dependencies [•]
```

# VAPOR

BUILDS A FRAMEWORK AND APP



The screenshot shows the Xcode interface with a Vapor project named "vapor-hello-world". The left sidebar displays the project structure, including Sources (App, Controllers, Models), Tests, Config, Localization, Public, Resources, Dependencies, and Products. The main editor area shows the contents of Package.swift:

```
import PackageDescription

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

A yellow speech bubble at the top right contains the text "BUILDS A FRAMEWORK AND APP". A yellow callout at the bottom right contains the text "USES SWIFT PM".

# VAPOR

The screenshot shows the Xcode interface with a Vapor project named "vapor-hello-world". The left sidebar displays the project structure, including Sources (App, 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, TLS 1.1.2), and a Package.swift file. The main editor window shows the contents of Package.swift:

```
import PackageDescription

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

A yellow callout bubble with the word "DEPENDENCIES" points to the Dependencies folder in the project structure.

# VAPOR

RUN APP

The screenshot shows a Mac OS X desktop environment with an Xcode window open. The title bar reads "Running App : App". The main area displays the code for "main.swift" in the "vapor-hello-world" project. The code imports Vapor and defines a Droplet instance that handles a GET request by returning a localized welcome message. Below the code editor, the output pane shows the server starting at port 8080.

```
import Vapor

let drop = Droplet()

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

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

# VAPOR

- Builds all the package dependencies and links them

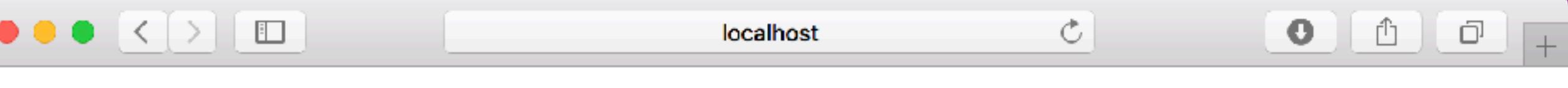
The current cipher key "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA=" is secure.  
Update cipher.key in Config/crypto.json before using in production.  
Use `openssl rand -base64 32` to generate a random string.  
Database prepared  
No command supplied, defaulting to serve...  
Starting server on 0.0.0.0:8080  
GET /  
GET /  
GET /hello/  
GET /hello

**RUN THE MAIN APP**

```
1 import Vapor
2
3
4         return json
5     }
6
7     get("plaintext") { req in
8         return "Hello, world!"
9     }
10
11    // response to requests to /info domain
12    // with a description of the request
13    get("info") { req in
14        return req.description
15    }
16
17    get("description") { req in return req.description }
18
19
20
21    try resource("posts", PostController.self)
22
23
24
25
26
```

# VAPOR

- Server runs on <http://localhost:8080>



It works.

# VAPOR

- File structure is designed to make testing easier 🙌

The screenshot shows a Mac OS X desktop environment with a Vapor project open in a file browser and a code editor.

**File Browser:** The left pane displays the project structure:

- helloworld (project folder)
  - Package.swift
  - Sources
    - App
      - Controllers
        - TodoController.swift
      - Models
        - Todo.swift
        - app.swift
        - boot.swift
        - configure.swift
  - routes.swift

**Code Editor:** The right pane shows the contents of the routes.swift file:

```
import Vapor

/// Register your app's routes here
public func routes(_ router: Router) throws {
    // Basic "Hello, world!" examples
    router.get("hello") { req in
        return "Hello, world!"
    }

    // Basic "Hello, [name]!" examples
    router.get("hello/:name") { req in
        let name = req.parameters.get("name", defaultValue: "World")
        return "Hello, \(name)!"
    }

    // Example of a controller
    let todoController = TodoController()
    router.get("todos") { req in
        return todoController.index(req)
    }
    router.get("todos/:id") { req in
        return todoController.show(req)
    }
    router.post("todos") { req in
        return todoController.create(req)
    }
    router.put("todos/:id") { req in
        return todoController.update(req)
    }
    router.delete("todos/:id") { req in
        return todoController.delete(req)
    }
}
```

# VAPOR

The image shows a screenshot of a macOS desktop environment. On the left, there is a file tree window showing the project structure of a Vapor application. The structure includes a `Package.swift` file at the root, a `Sources` folder containing an `App` folder which has `Controllers` and `Models` subfolders. Inside `Controllers` is a `TodoController.swift` file, and inside `Models` is a `Todo.swift` file. There is also an `app.swift` file at the root level. A preview pane below the file tree displays the text "Hello, world!". On the right, there is a code editor window displaying Swift code for route registration. The code defines a `routes(\_ router: Router)` function that registers three basic routes: a "It works" endpoint, a "Hello, world!" endpoint, and a "Goodbye" endpoint.

```
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       }
18
```

# VAPOR

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

```
import Vapor

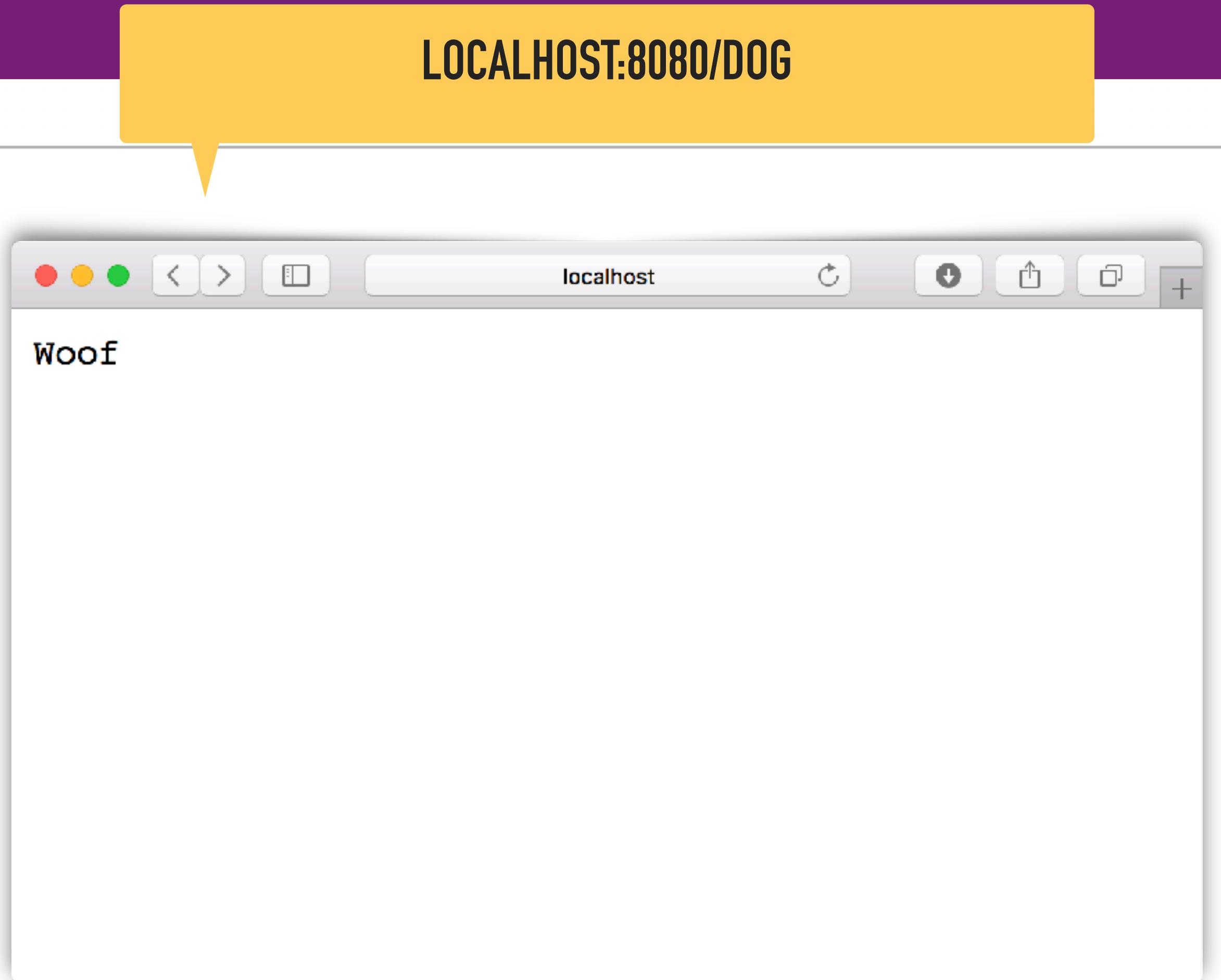
let drop = Droplet()

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

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

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

drop.run()
```



# VAPOR

LOCALHOST:8080/GOODBYE

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

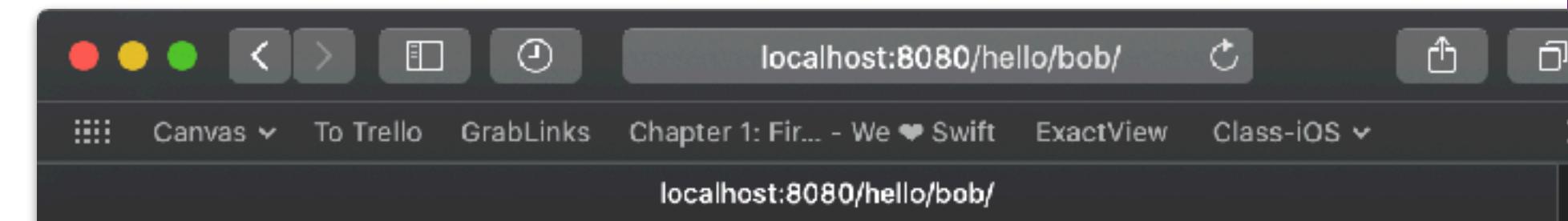
LOCALHOST:8080/USER/LOLA

```
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



Log in to your account



Email address



Password

Log In

New to Heroku? [Sign Up](#)

[Log in via SSO](#)

[Forgot your password?](#)

# DEPLOY VAPOR APP TO HEROKU

# Install tools

```
brew install heroku
```

# Set credentials

```
heroku login
```

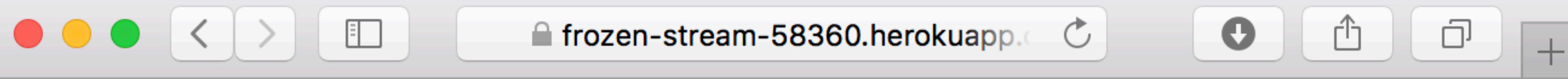
# Upload

```
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 POSTGRESOL

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

The world's most advanced open source database.

**PostgreSQL**

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

**LATEST RELEASES**

11<sup>th</sup> May 2017

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

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

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

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

**FEATURED USER**

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

Ram Mohan, CTO, [Afilias](#)

» [Case Studies](#) | [More Quotes](#) | [Featured Users](#)

**LATEST NEWS**

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

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

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

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

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

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

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

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

**UPCOMING EVENTS**

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

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

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

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

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

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

**UPCOMING TRAINING**

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

» [More](#) | [RSS](#)

**SHORTCUTS**

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

**SUPPORT US**

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



# VAPOR WITH POSTGRESOL

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

vapor / fluent

Code Issues Pull requests Wiki Pulse Graphs

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

vapor orm swift database sql nosql

608 commits 3 branches 97 releases 31 c

Branch: master New pull request Create new file

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

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

\* Fluent

# VAPOR WITH POSTGRESOL

- Import package
- Configure droplet
- Add configuration file

The world's most advanced open source database.

**PostgreSQL**

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

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

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

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

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

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



**FEATURED USER**

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

Ram Mohan, CTO, [Afilias](#)

» [Case Studies](#) | [More Quotes](#) | [Featured Users](#)

**LATEST NEWS**

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

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

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

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

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

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

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

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

**UPCOMING EVENTS**

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

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

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

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

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

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

**UPCOMING TRAINING**

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

» [More](#) | [RSS](#)

**LATEST RELEASES**

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

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

**SHORTCUTS**

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

**SUPPORT US**

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

# VAPOR WITH POSTGRESQL

```
# Install postgres  
brew install postgres
```

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

```
# Create a database  
createdb dogs
```

# VAPOR WITH POSTGRESOL

```
vapor-hello-world — psql — 107x20
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

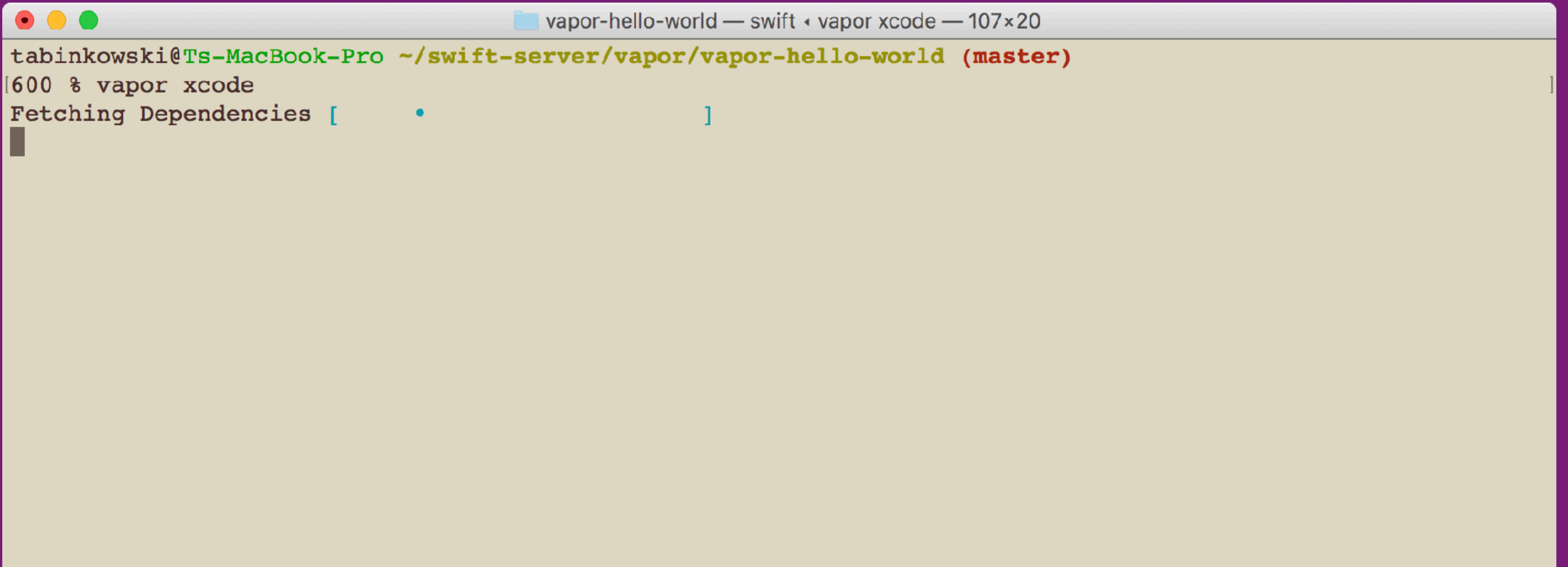
Finished running App : App

```
import PackageDescription

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

Add a provider

# VAPOR WITH POSTGRESOL



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

```
vapor-hello-world — swift • vapor xcode — 107x20
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[600 % vapor xcode
Fetching Dependencies [ • ]
```

- Update the project and rebuild your framework in Xcode

# VAPOR WITH POSTGRESOL

< >  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 POSTGRESOL

The screenshot shows a Mac OS X desktop with an Xcode interface. The title bar reads "vapor-hello-world | Build Succeeded". The left sidebar shows the project structure:

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

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

Database configuration file

Add a new file

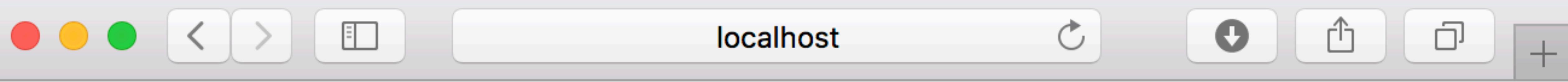
# VAPOR WITH POSTGRESQL

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

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

Print out database version

# VAPOR WITH POSTGRESQL



IT'S BORING, BUT IT WORKS

# VAPOR WITH POSTGRESQL

The screenshot shows the Xcode IDE interface with a Swift file named `main.swift` open. The code implements a Vapor application to handle dog data.

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

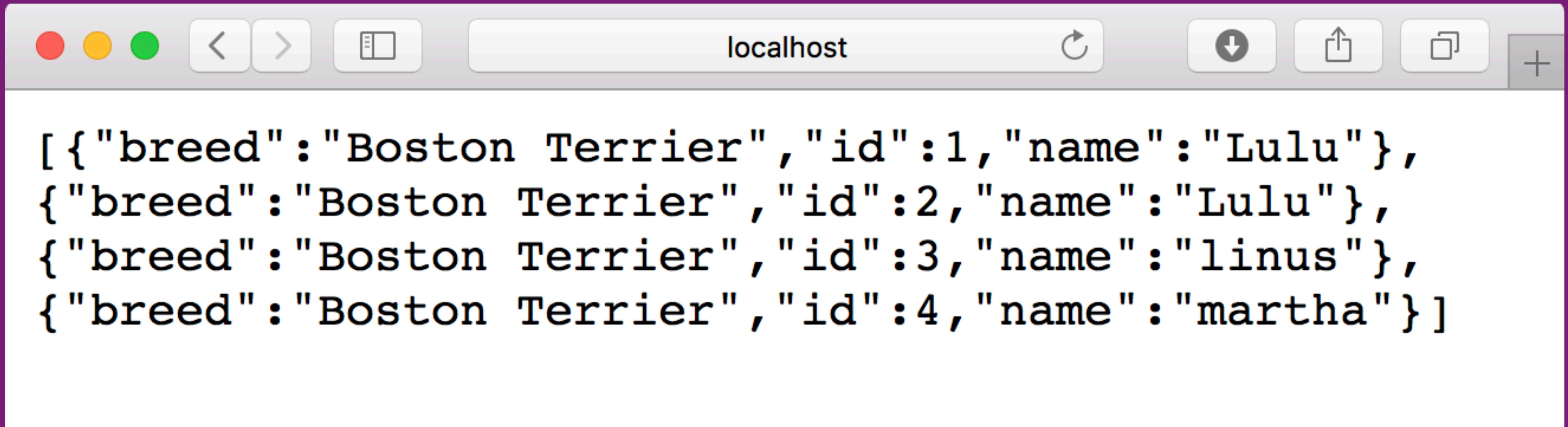
// Add provider
do {
    try drop.addProvider(VaporPostgreSQL.Provider.self)
} catch {
    assertionFailure("Error adding provider: \(error)")
}

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)
}
```

Annotations highlight specific parts of the code:

- A yellow callout points to the line `drop.preparations.append(Dog.self)` with the text **ADD DOG MODEL**.
- A yellow callout points to the line `let dog = [Dog(name:"Lulu", breed: "Boston Terrier"), Dog(name:"Snoopy", breed: "Beagle"), Dog(name:"Fifi", breed: "Poodle")]` with the text **CREATE DATA**.

# 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 db
  - \c - switch db
  - \d - list tables

```
(5 rows)

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

dogs=#
```

DEPLOY TO HEROKU  
WITH POSTGRESQL

# DEPLOY TO HEROKU WITH POSTGRESQL

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

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

# DEPLOY TO HEROKU WITH POSTGRESQL

Add postgres to your heorku app

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

New database

# DEPLOY TO HEROKU WITH POSTGRESQL

```
vapor hello world - bash - 85x15
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[613 % heroku addons:create heroku-postgresql:hobby-dev
Creating heroku-postgresql:hobby-dev on ⚙ frozen-stream-58360... free
Database has been created and is available
! This database is empty. If upgrading, you can transfer
! data from another database with pg:copy
Created postgresql-vertical-49786 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
[614 % heroku config
==== frozen-stream-58360 Config Vars
DATABASE_URL: postgres://hzgtyzqowaztwa:4ad9847249b30c97f1835744ca06b1bae2b9012422c21d32d
4dad5e5088def04@ec2-23-23-227-188.compute-1.amazonaws.com:5432/d9hapt1pulfgrf
tabinkowski@Ts-MacBook-Pro ~/swift-server/vapor/vapor-hello-world (master)
615 %
```

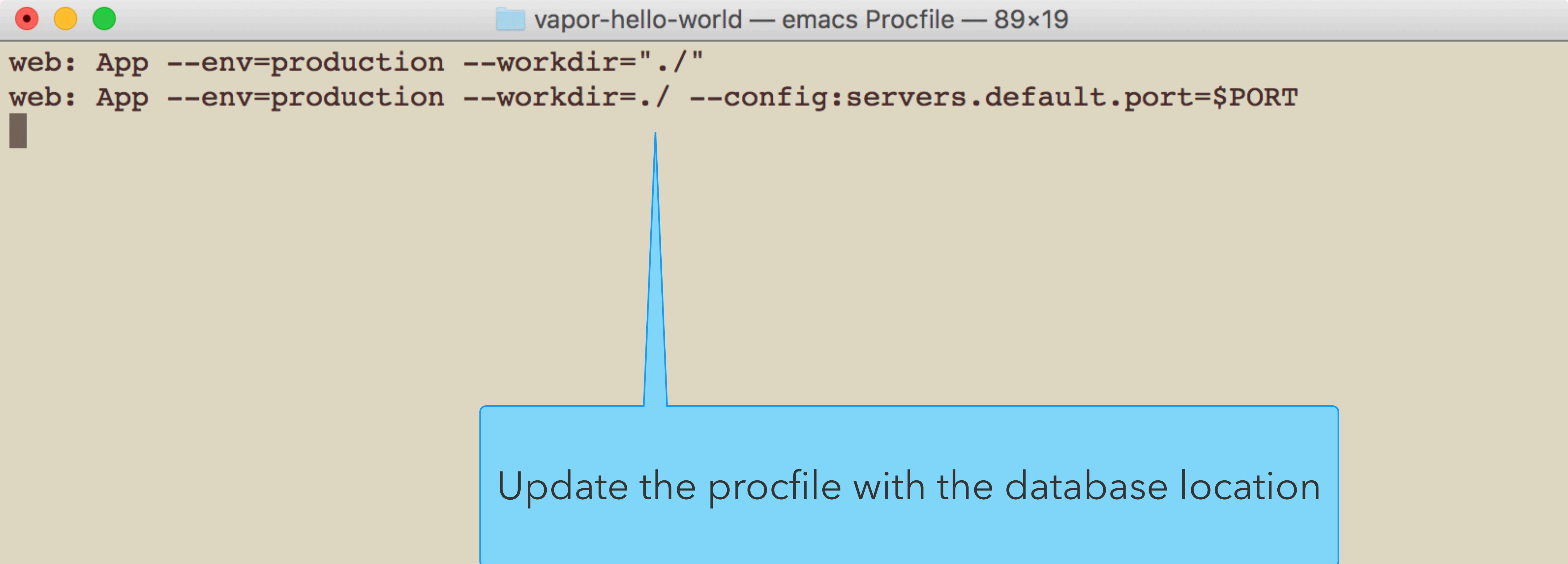
Database location

# DEPLOY TO HEROKU WITH POSTGRESQL

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

Update the procfile with the database location

# DEPLOY TO HEROKU WITH POSTGRESQL



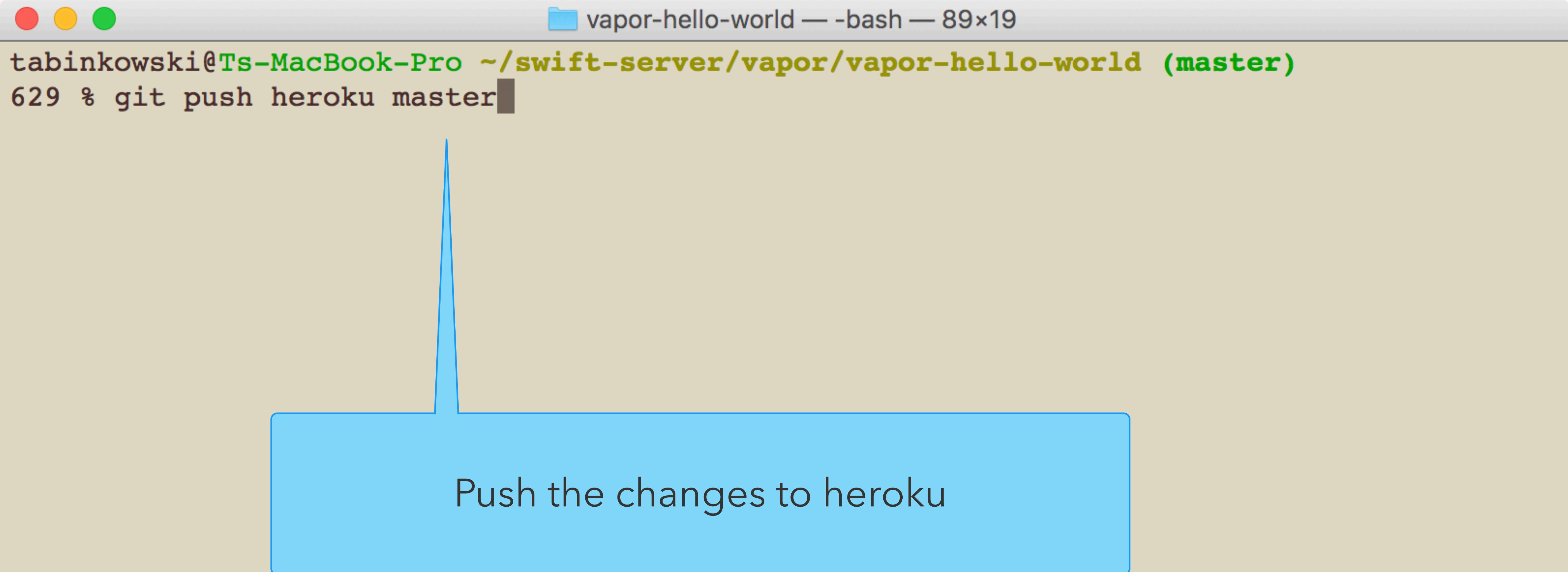
A screenshot of a terminal window titled "vapor-hello-world — emacs Procfile — 89x19". The window contains two lines of code:

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

A blue callout box with a white border and a blue arrow pointing upwards from the bottom right towards the terminal window contains the text:

Update the procfile with the database location

# DEPLOY TO HEROKU WITH POSTGRESQL

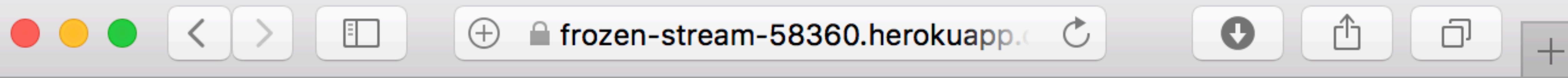


A terminal window with a dark purple header bar. The window title is "vapor-hello-world — -bash — 89x19". The terminal shows the command "git push heroku master" being typed by a user named "tabinkowski". The background of the slide features a large blue arrow pointing upwards from the bottom left towards the terminal window.

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

Push the changes to heroku

# DEPLOY TO HEROKU WITH POSTGRESQL

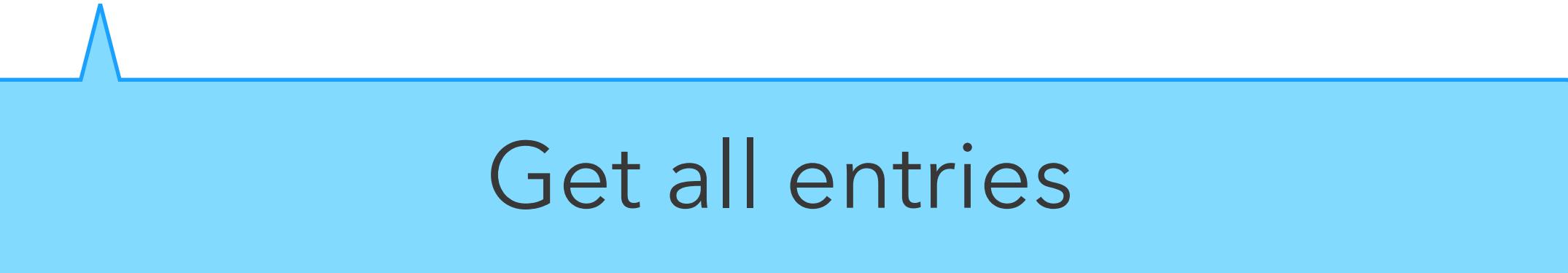


```
[ {"breed": "Boston Terrier", "id": 1, "name": "martha"}, {"breed": "Boston Terrier", "id": 2, "name": "martha"} ]
```

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

# VAPOR WITH POSTGRESQL

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



Get all entries

# VAPOR WITH POSTGRESQL

```
// 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 POSTGRESOL

```
//  
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

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

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

# PERFECT

- Requirements
  - Xcode 8 Mac
  - Linux toolchain

The screenshot shows the homepage of perfect.org. At the top left is the Perfect logo (a stylized orange bird) and the URL "perfect.org". At the top right is a three-line menu icon. The main content area is divided into four sections:

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

# 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 — 70x19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
644 % swift package generate-xcodeproj
generated: ./perfect-hello-world.xcodeproj
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
645 %
```

Swift package manager to create an Xcode project

# PERFECT

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

# PERFECT

- Working command line application

A screenshot of a macOS terminal window titled "pe...ct) > My Mac". The title bar also shows "Finished running perfect-hello-world : perfect-hello.". The window has three panes: a left pane showing a file tree for a "perfect-hello-world" project, a middle pane showing the contents of "main.swift", and a bottom pane showing the terminal output.

The file tree shows:

- perfect-hello-world
  - Package.swift
  - Sources
    - main.swift
    - Tests
    - Products

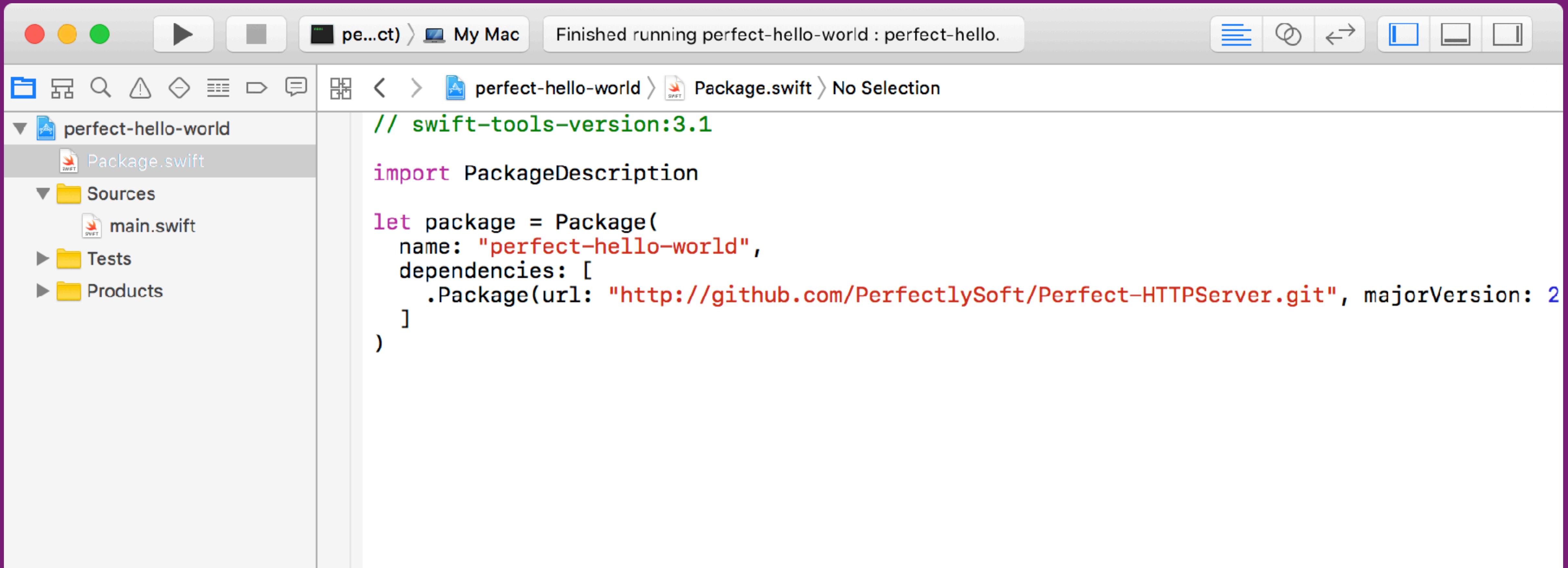
The "main.swift" file contains the following code:

```
print("Hello, world!")
```

The terminal output pane shows:

```
Hello, world!
Program ended with exit code: 0
```

# PERFECT



The screenshot shows the Xcode IDE interface. The top bar displays the title "pe...ct) > My Mac" and the status message "Finished running perfect-hello-world : perfect-hello.". The main window has a toolbar with various icons. The left sidebar shows the project structure under "perfect-hello-world": "Sources" contains "main.swift", and "Tests" and "Products" are also listed. The right pane shows the "Package.swift" file content:

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

let package = Package(
    name: "perfect-hello-world",
    dependencies: [
        .Package(url: "http://github.com/PerfectlySoft/Perfect-HTTPServer.git", majorVersion: 2)
    ]
)
```

- Add Perfect as a project dependency

# PERFECT

```
perfect-hello-world — swift-package update ▶ git-remote-https — 70×19
tabinkowski@Ts-MacBook-Pro ~/swift-server/perfect/perfect-hello-world
[648 % swift package update
Fetching http://github.com/PerfectlySoft/Perfect-HTTPServer.git
Fetching https://github.com/PerfectlySoft/Perfect-HTTP.git
[
```

- Update the package manager

# PERFECT

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

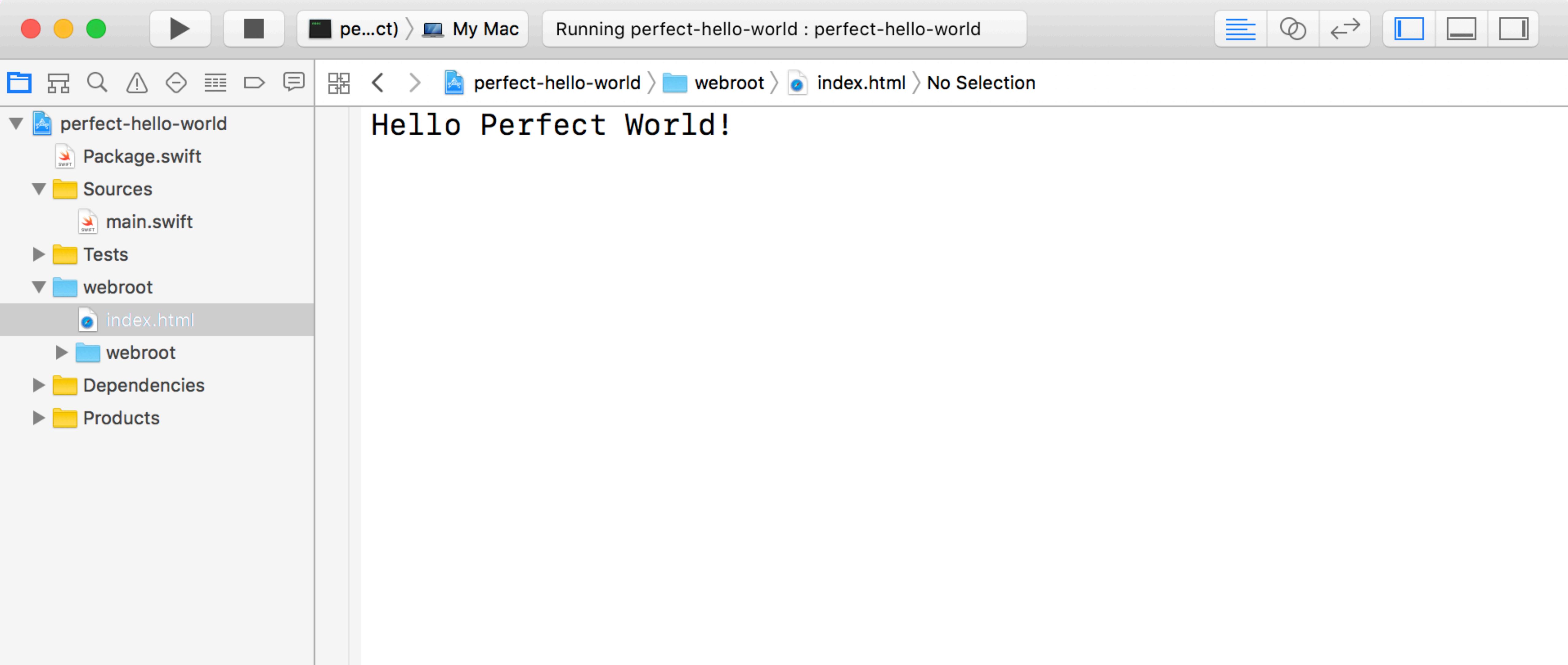
- Update the Xcode project

# PERFECT

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

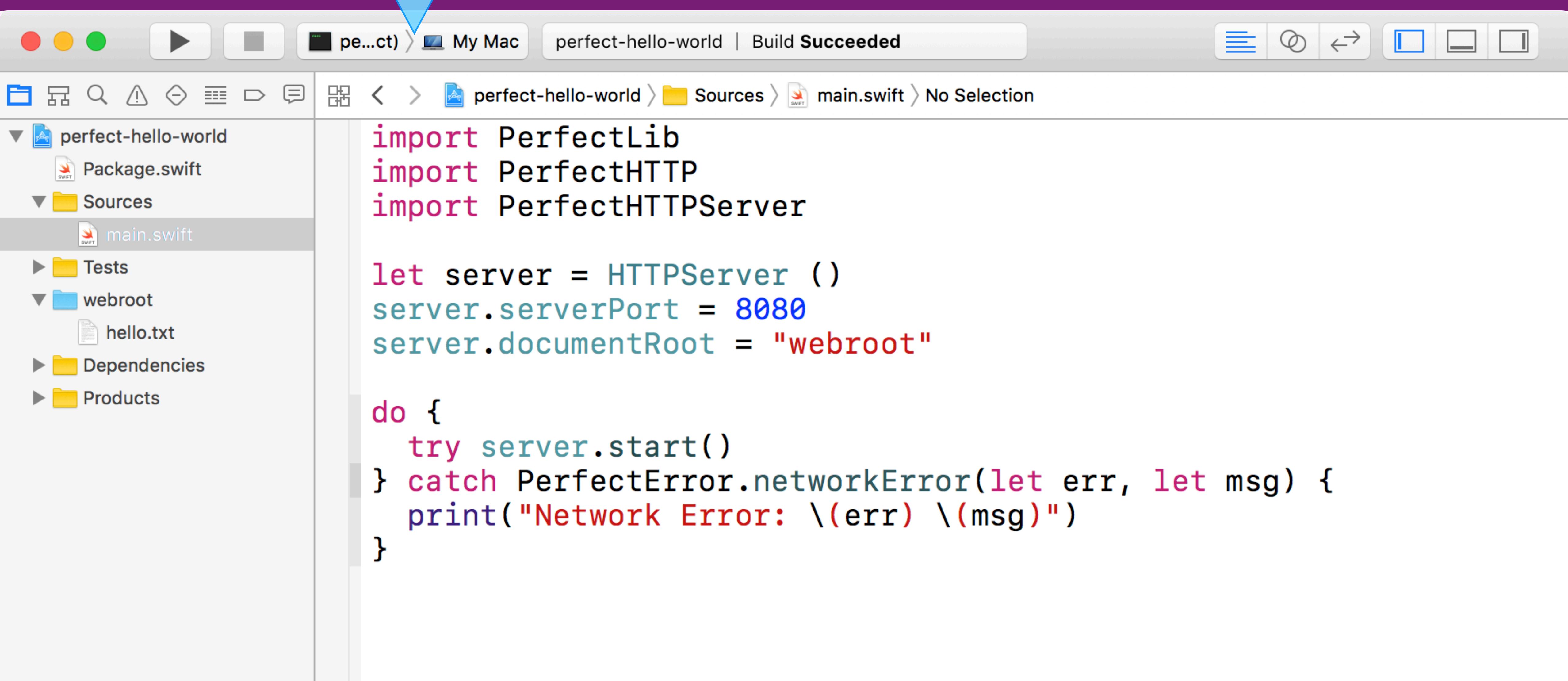
- Create a root directory for the web server

# PERFECT



# PERFECT

## Create a server



The screenshot shows a Mac OS X desktop with an Xcode interface. A blue speech bubble is positioned above the title bar, containing the text "Create a server". The Xcode window has a title bar with standard OS X controls (red, yellow, green buttons) and a tab labeled "perfect-hello-world | Build Succeeded". The main area shows a file browser on the left and a code editor on the right. The file browser shows a project structure:

- perfect-hello-world (project folder)
  - Package.swift
  - Sources
    - main.swift
  - Tests
  - webroot
    - hello.txt
  - Dependencies
  - Products

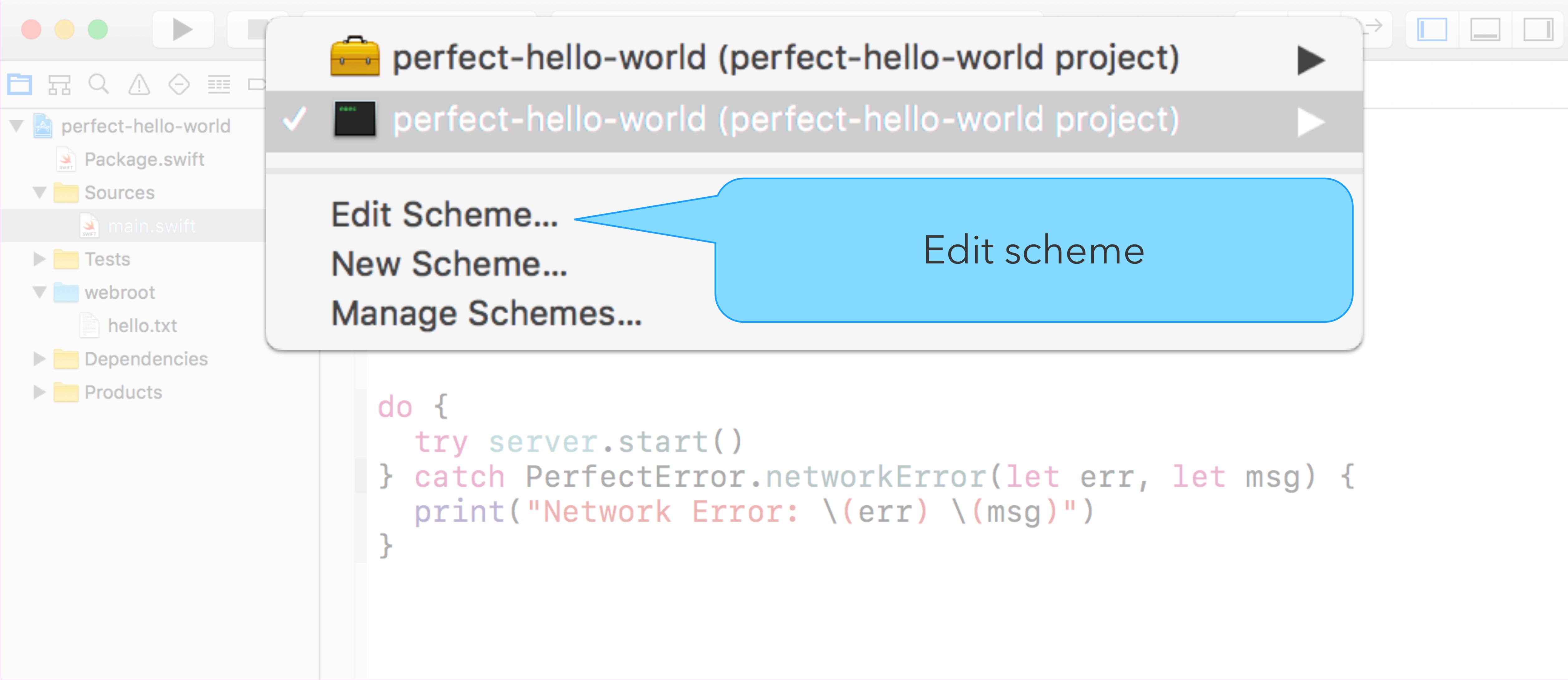
The code editor displays the contents of the main.swift file:

```
import PerfectLib
import PerfectHTTP
import PerfectHTTPServer

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

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

# PERFECT



# PERFECT

## Options

Build  
1 target

Run  
Debug

Test  
Debug

Profile  
Release

Analyze  
Debug

Archive  
Release

Info Arguments Options Diagnostics

Core Location  Allow Location Simulation

Default Location

Application Data

Routing App Coverage File

GPU Frame Capture

Metal API Validation

Persistent State  Launch application without state restoration

Document Versions  Allow debugging when using document Versions Browser

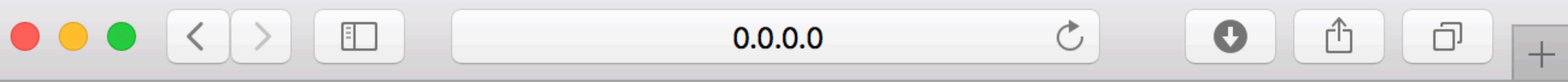
Working Directory  Use custom working directory:

Localization Debugging  Show non-localized strings

Application Language

Working Directory

# PERFECT



Hello Perfect World!

# PERFECT

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

var routes = Routes()

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

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

# PERFECT

The screenshot shows a development environment with two main windows. On the left is a file browser and code editor for a Swift project. The file `main.swift` contains the following code:

```
var routes = Routes()

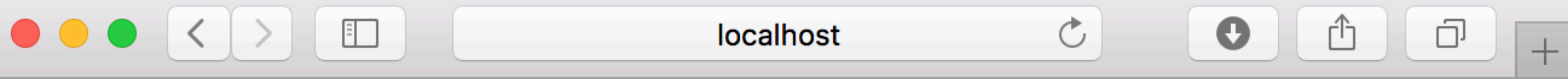
// Route "/"
routes.add(method: .get, uri: "/", handler: {
    request, response in
    response.setBody(string: "Bark!")
    response.completed()
})
server.addRoutes(routes)
```

On the right is a web browser window titled `localhost` displaying the word `Bark!`.

# PERFECT

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

PERFECT



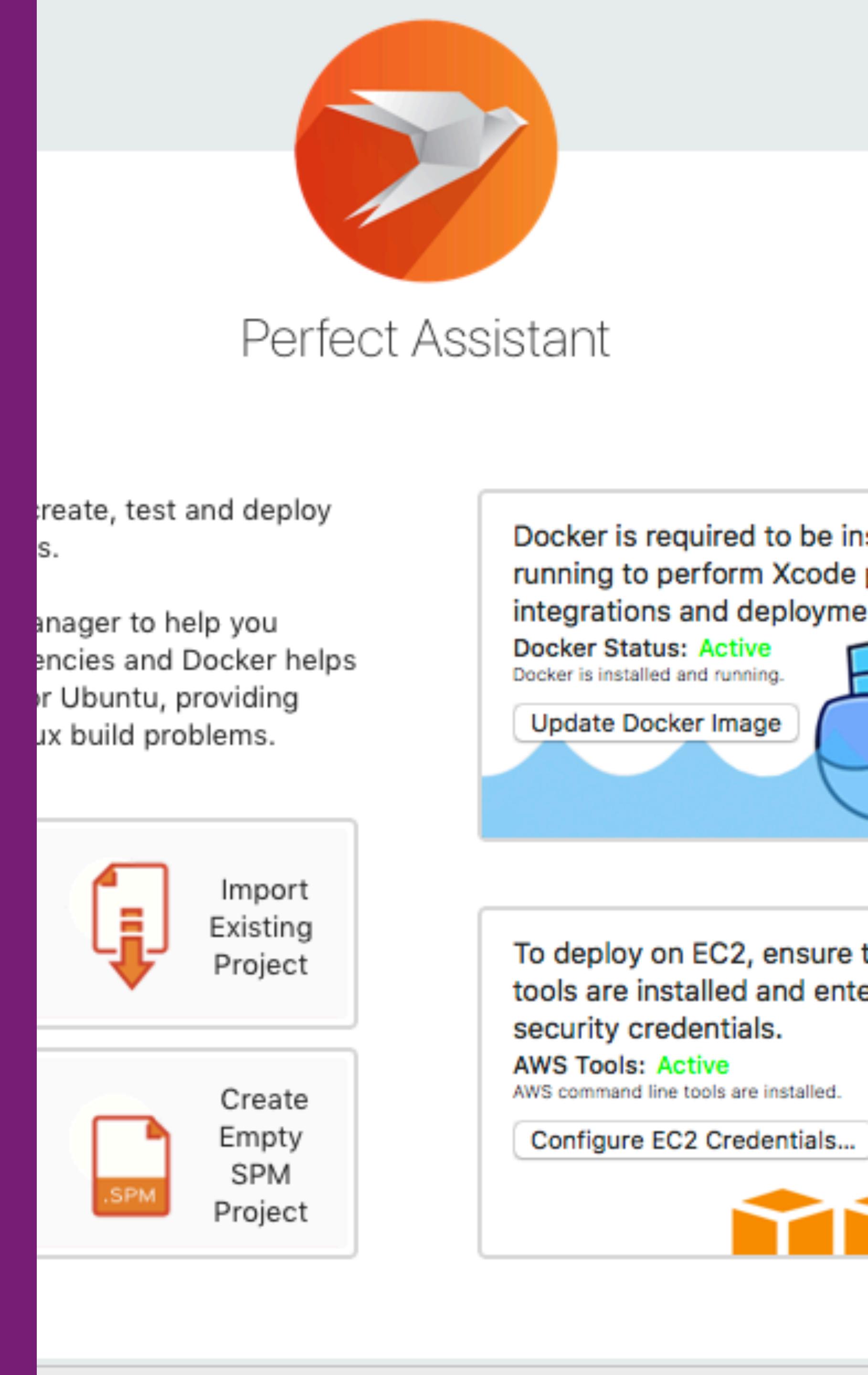
**lassie says Bark!**

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

PERFECT ASSISTANT

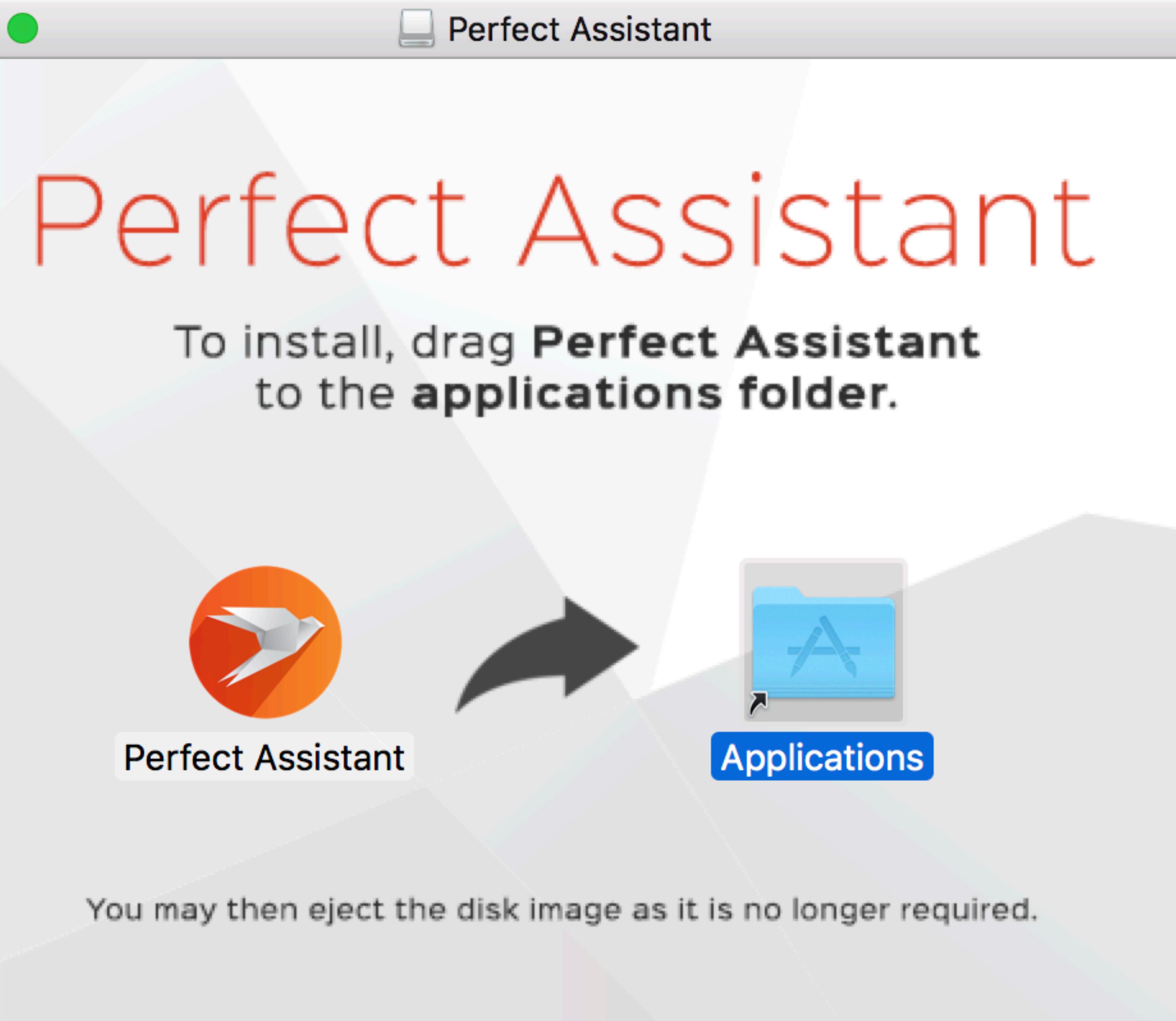
# PERFECT ASSISTANT

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



# PERFECT ASSISTANT

- Download and install



# PERFECT ASSISTANT

- Download and install

Perfect Assistant

Welcome

Projects

Deployments

Welcome to Perfect Assistant



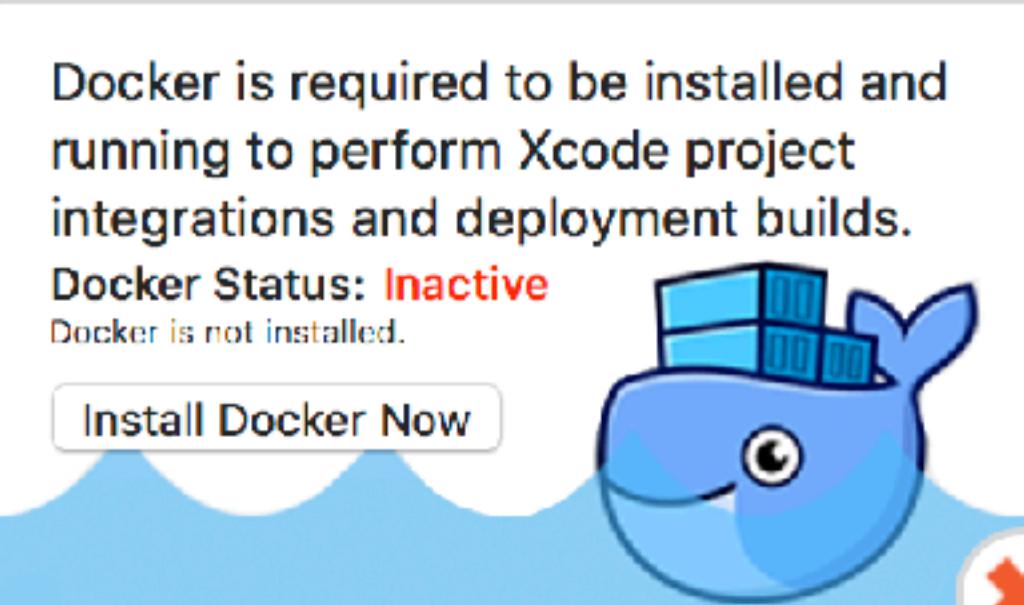
This tool will help you easily create, test and deploy your server-side swift projects.

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

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

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

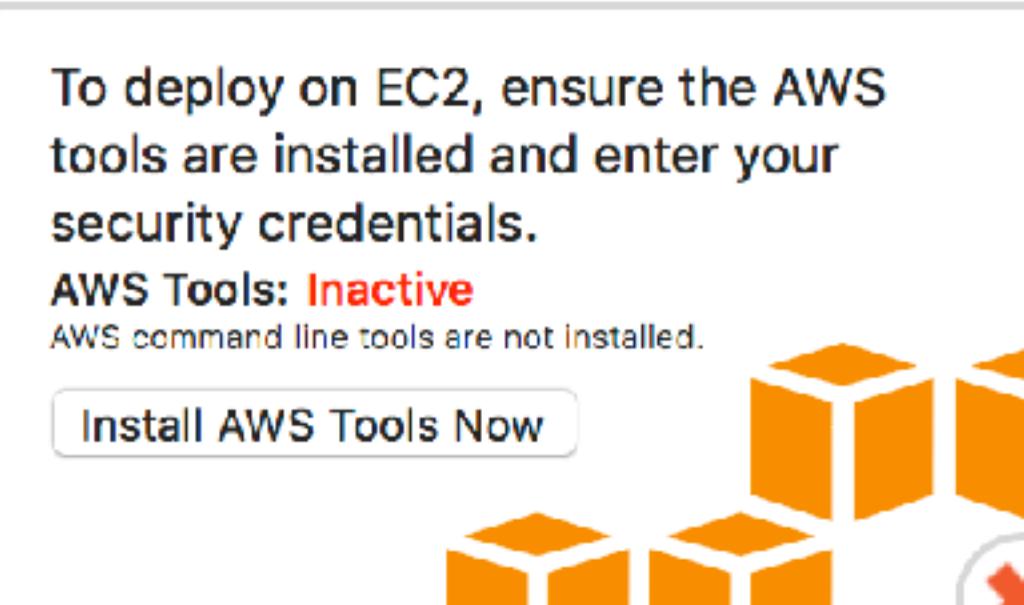
Install Docker Now



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

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

Install AWS Tools Now



Create New Project

Import Existing Project

Create HTTP Server Project

Create Empty SPM Project

+ ⌂

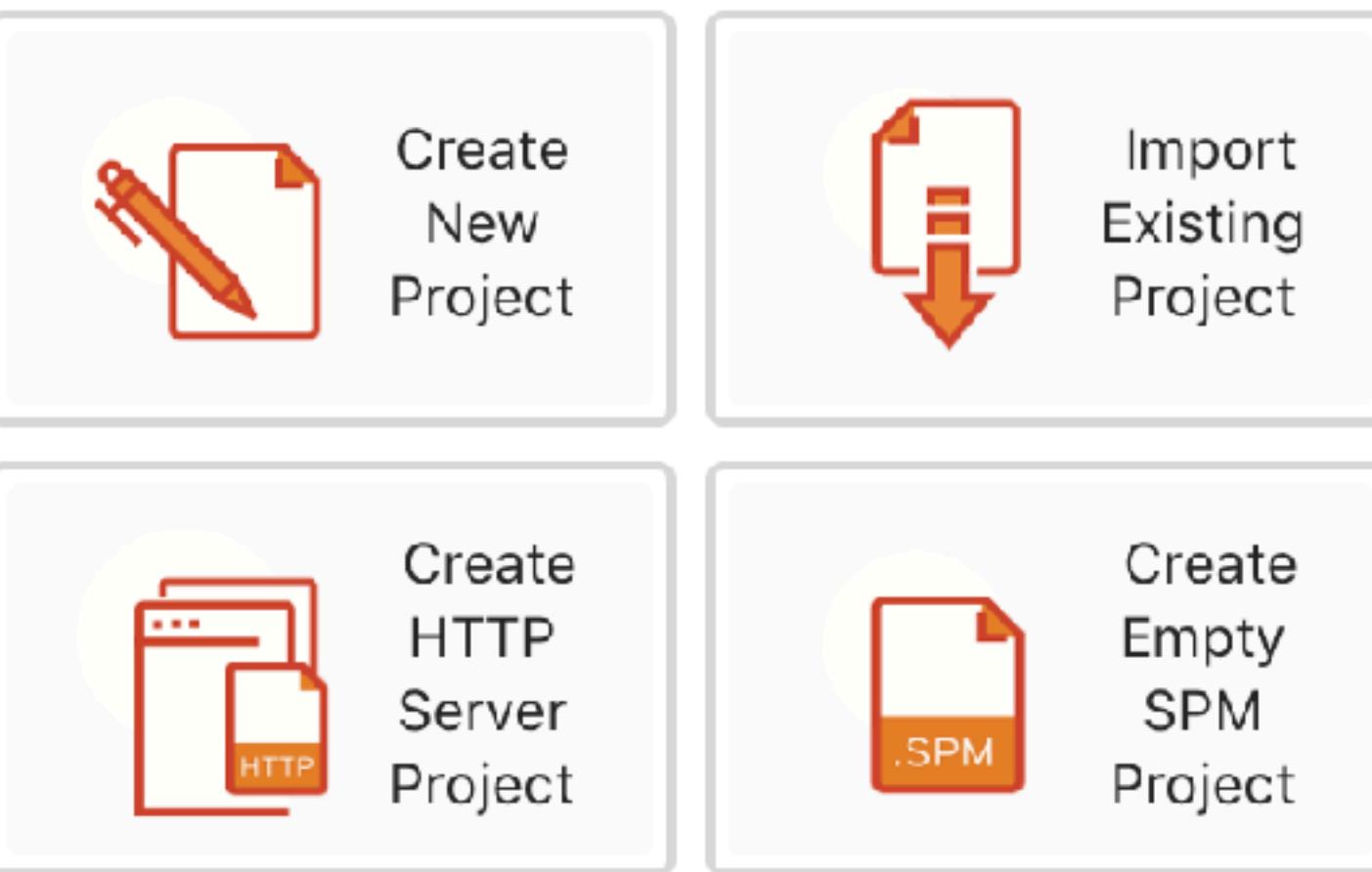
# PERFECT ASSISTANT

- Deploy using Docker

## Perfect Assistant

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

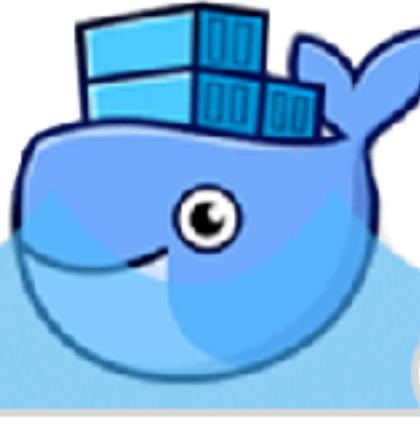
It leverages Swift Package Manager to help you manage your project dependencies and Docker helps to build your project locally for Ubuntu, providing feedback in Xcode of any Linux build problems.



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

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

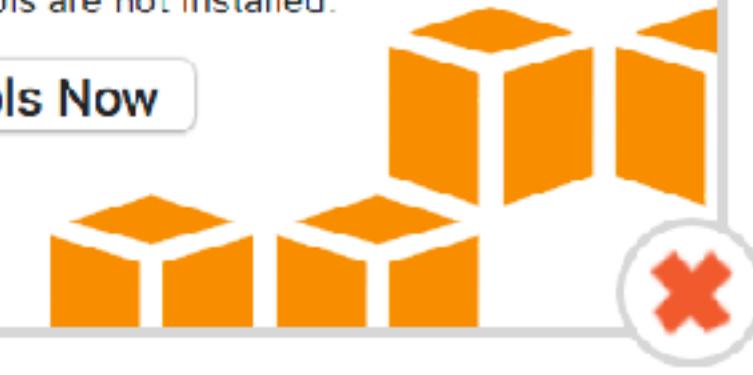
[Install Docker Now](#)



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

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

[Install AWS Tools Now](#)



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

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

[Open Cloud Console...](#)



# PERFECT ASSISTANT

Perfect Assistant

Welcome Projects Deployments

## New Project

Project Deployment

Choose one of the starter project templates below.

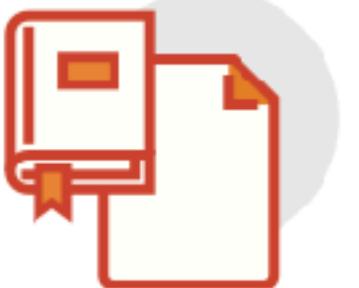
**Basic**



Perfect Template App



Empty Executable Project



Empty Library Project



Custom Repository URL



Perfect Template App Engine

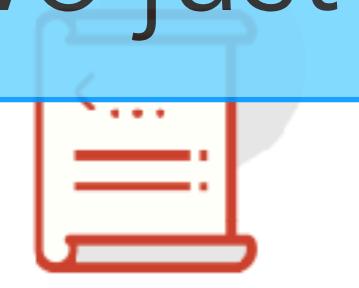
**Examples**



Perfect: Upload



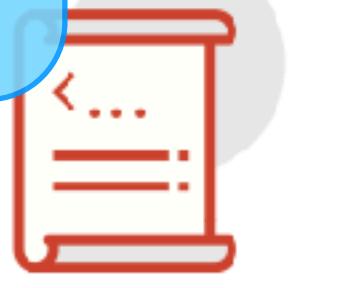
Perfect: URL



Perfect:



Perfect:



Perfect:

What we just did.

# PERFECT ASSISTANT

Perfect Assistant

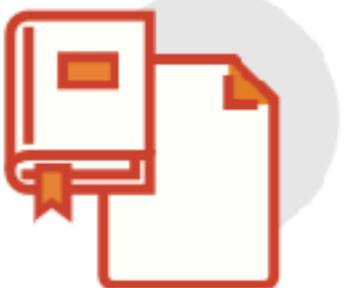
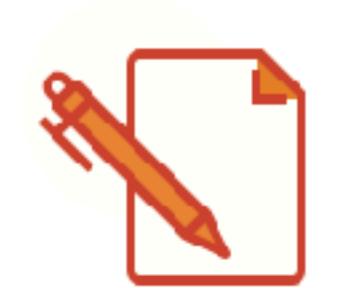
Welcome Projects Deployments

## New Project

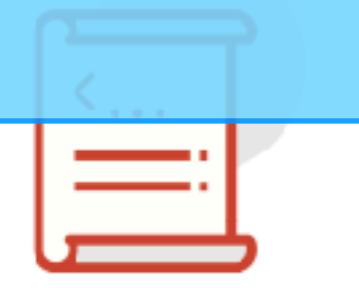
Project Deployment

Choose one of the starter project templates below.

**Basic**

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

**Examples**

-  Perfect: Upload
-  Perfect: URL
-  Perfect: Web Content
-  Perfect: Email
-  Perfect: File

Create a project

# PERFECT ASSISTANT

Perfect Assistant

Welcome Projects Deployments

## New Project

Project Deployment

Choose one of the starter project templates below.

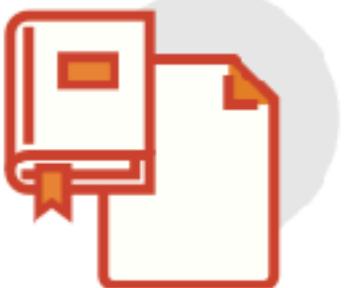
**Basic**



Perfect Template App



Empty Executable Project



Empty Library Project



Custom Repository URL



Perfect Template App Engine

**Examples**



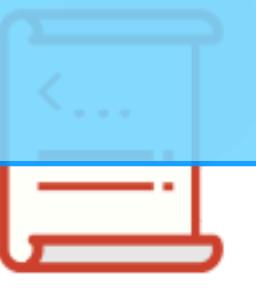
Perfect: Upload



Perfect: URL



Perfect: Web Content



Perfect:



Perfect:

Yeah!!

# PERFECT ASSISTANT

Perfect  
Template App

Empty  
Executable  
Project

Empty Library  
Project

Custom  
Repository URL

Perfect  
Template App  
Engine

## Examples



Perfect: Upload  
Enumerator



Perfect: URL  
Routing



Perfect:  
WebSockets  
Server



Perfect:  
Turnstile with  
SQLite



Perfect:  
Turnstile with  
PostgreSQL



Perfect:  
Weather



Perfect: Polling



Perfect: URL  
Streamer



Perfect: Blog  
Mustache



Perfect: JSON  
API

Yeah!!

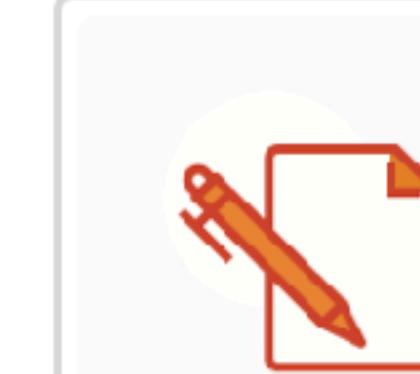
Cancel

Previous

Next

# PERFECT ASSISTANT

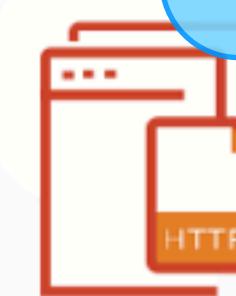
Feedback in Xcode or any Linux build problems.



Create  
New  
Project



Import  
Existing  
Project



Create  
HTTP  
Server  
Project



Create  
Empty  
SPM  
Project

Create a new project

Install AWS Tools Now



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

AWS command line tools are not installed.



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

Clear

Project saved.

Creating working directory and files

Backed up existing package file

Created package file

# PERFECT ASSISTANT

Perfect Assistant

Welcome

Projects perfect-template

Deployments

project: perfect-template

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

Add dependencies for your project

OPEN

- Project Directory
- Project Terminal
- Xcode Project

BUILD

- Local
- Linux
- Deploy
- Clean

DEPENDENCIES

Selected Dependencies:

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

Available Dependencies:

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

Database Connector

# PERFECT ASSISTANT

Available Dependencies:

- MySQL
- CouchDB-StORM
- SQLite-StORM
- Postgres-StORM
- MongoDB-StORM

▼ Server

- WebRedirects
- HTTPServer
- WebSockets
- HTTP
- FastCGI

Build and test (mac and linux)

▼ Session Management

- Session-CouchDB
- Session-SQLite
- Session-MYSQL
- Session-PostgreSQL
- Session-MongoDB
- Session

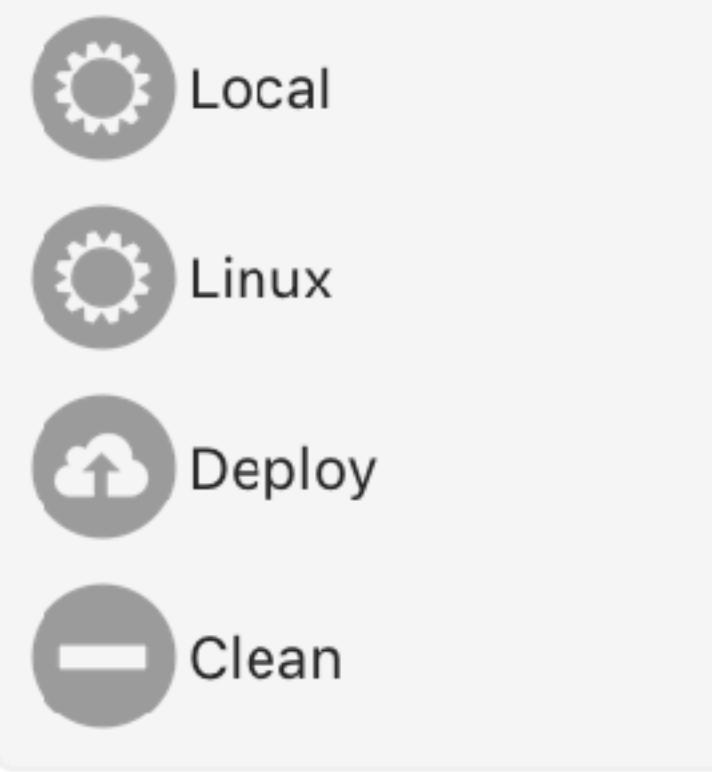
BUILD

- Local
- Linux
- Deploy
- Clean

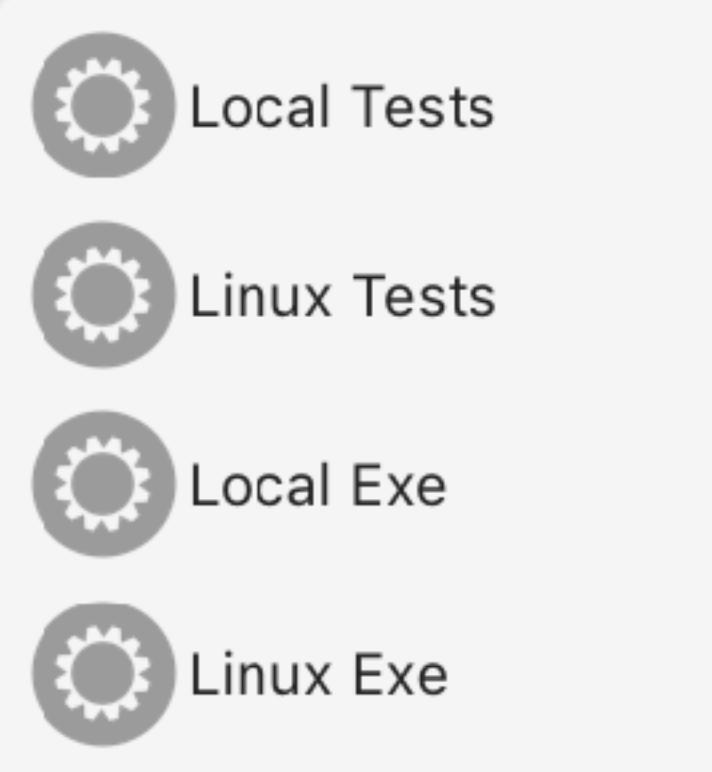
RUN

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

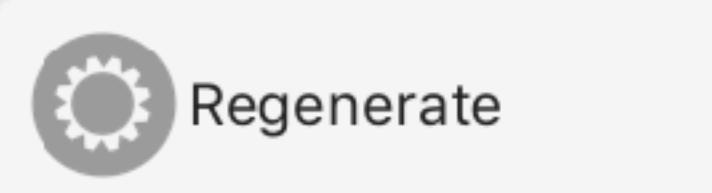
# PERFECT ASSISTANT



## RUN



## XCODE PROJECT



## Available Dependencies:

### Authentication



Turnstile-  
PostgreSQL



Turnstile-  
MySQL



Turnstile-  
MongoDB



Authenticati  
on



Turnstile-  
CouchDB



Turnstile-  
SQLite

### Database Connector



PostgreSQL



MySQL



CouchDB



FileMaker



Redis



SQLite



MongoDB

### ORM



MySQL-  
StORM



CouchDB-  
StORM



SQLite-  
StORM



Postgres-  
StORM



MongoDB-  
StORM

<https://github.com/SwiftORM/CouchDB-StORM.git>

### Server

# PERFECT ASSISTANT

Perfect Assistant

Welcome

Projects  
perfect-template

Deployments

project: perfect-template Postgresql setup for swift server

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

OPEN

- Project Directory
- Project Terminal
- Xcode Project

DEPENDENCIES

Selected Dependencies:

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

BUILD

- Local
- Linux
- Deploy
- Clean

Available Dependencies:

▼ Authentication

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

# PERFECT ASSISTANT



Session-  
ouchDB



Session-  
SQLite



Session-  
MySQL



Session-  
PostgreSQL



Session-  
MongoDB



Session

## Utility

questLog  
ger



Logger



Notifications



OpenSSL



Logger



Repeater



XML



PerfectLib



Net



Zip



SMTP



Thread



Mustache



CURL

Save and update the project

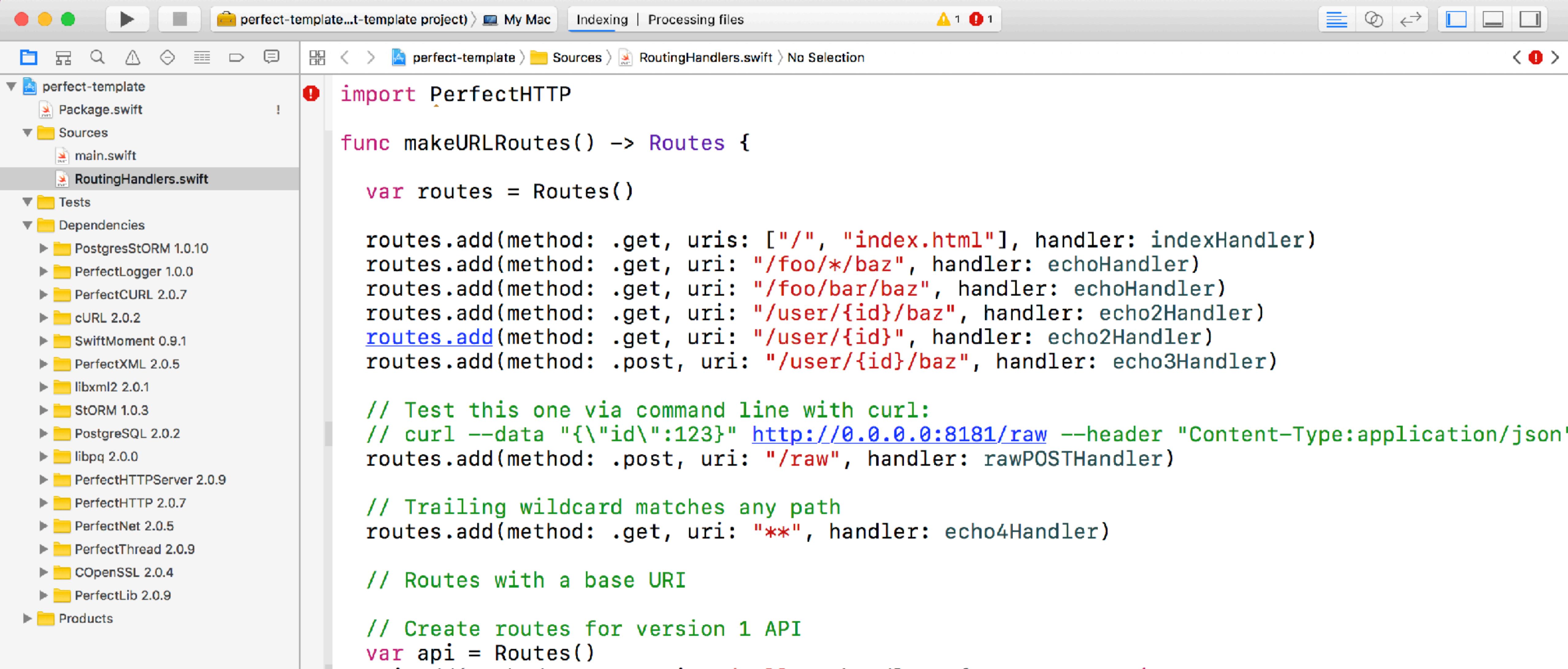


Automatically integrate Xcode  
when regenerating project

Add Dependency...

Save Changes

# PERFECT ASSISTANT



The screenshot shows the Xcode IDE interface with a Swift project named "perfect-template". The project structure on the left includes files like Package.swift, main.swift, and RoutingHandlers.swift. The RoutingHandlers.swift file is open in the editor, displaying code for setting up routes using the PerfectHTTP framework. The code defines a function makeURLRoutes() that adds various routes like GET /, GET /foo/\*baz, and POST /raw. It also includes comments for testing with curl and notes about trailing wildcards and base URIs. The Xcode status bar at the top indicates indexing and processing files.

```
import PerfectHTTP

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

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

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

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

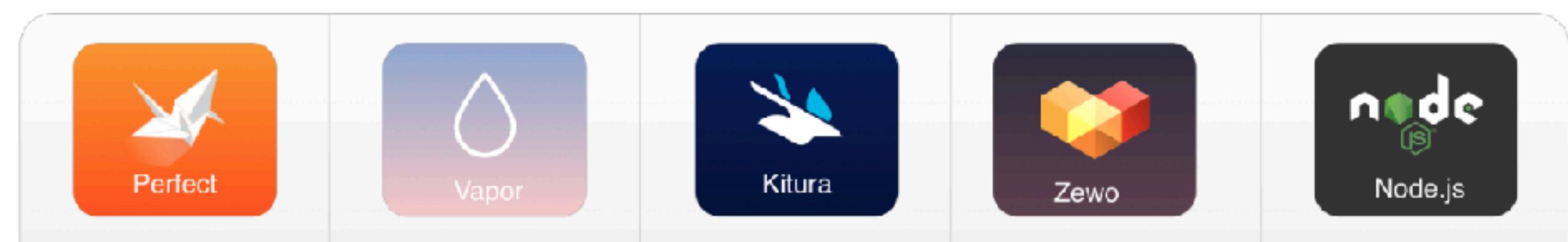
    // Routes with a base URI
    // Create routes for version 1 API
    var api = Routes()
```

# THE FUTURE OF SWIFT ON THE SERVER

# PERFORMANCE

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

## Results Summary



### 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

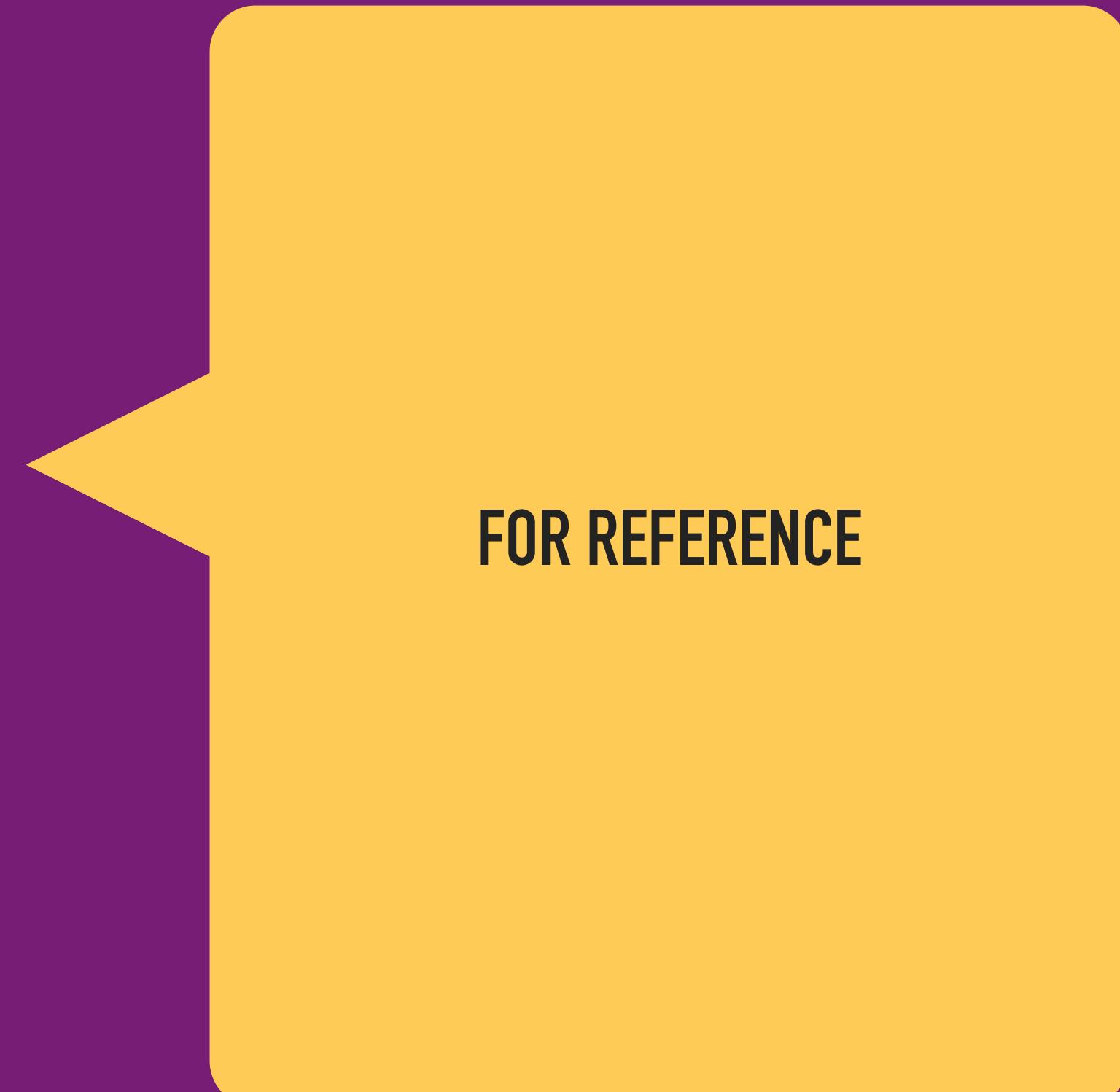
# FUTURE OF SWIFT ON THE SERVER

- 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 dependencies

# Swift

SWIFT

OAD

G STARTED

ENTATION

TING TO SWIFT 4

E CODE

UNITY

IBUTING

VIOUS

ATION

E COMPATIBILITY

AS

ABILITY

R APIs (WORK GROUP)

ER AND

# 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 automatically handle 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 namespace can be used outside of the module.

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

# 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

# Swift

SWIFT

OAD

G STARTED

ENTATION

INTING TO SWIFT 4

E CODE

UNITY

IBUTING

VIOUS

ATION

E COMPATIBILITY

AS

ABILITY

R APIs (WORK GROUP)

ER AND

# 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 automatically handle 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 namespace can be used outside of the module.

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

# 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

SWIFT

OAD

G STARTED

ENTATION

INTING TO SWIFT 4

E CODE

UNITY

IBUTING

VIOUS

ATION

E COMPATIBILITY

AS

ABILITY

R APIs (WORK GROUP)

ER AND

# 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 automatically handle 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 namespace can be used outside of the module.

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

# SWIFT PACKAGE MANAGER

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

# Swift

SWIFT

OAD

G STARTED

ENTATION

INTING TO SWIFT 4

E CODE

UNITY

IBUTING

VIOUS

ATION

E COMPATIBILITY

AS

ABILITY

R APIs (WORK GROUP)

ER AND

# 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 depend on other modules as *dependencies*. Aside from the handful of system

# SWIFT PACKAGE MANAGER

- Products
  - Target of a package or executable

# Swift

SWIFT

OAD

G STARTED

ENTATION

INTING TO SWIFT 4

E CODE

UNITY

IBUTING

VIOUS

ATION

E COMPATIBILITY

AS

ABILITY

R APIs (WORK GROUP)

ER AND

# 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 automatically handle 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 namespace can be used outside of the module.

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

# SWIFT PACKAGE MANAGER

- Dependencies

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

# Package Manager

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

The Package Manager is included in Swift 3.0 and later versions.

## Conceptual Overview

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

### Modules

Swift organizes code into *modules*. Each module defines a namespace and enforces access controls on which functions and types can be used outside of the module.

A program may have all of its code in a single module or it may depend on other modules as *dependencies*. Aside from the basic

# SWIFT PACKAGE MANAGER



MODULE

# SWIFT PACKAGE MANAGER

PACKAGE

MODULE



# SWIFT PACKAGE MANAGER

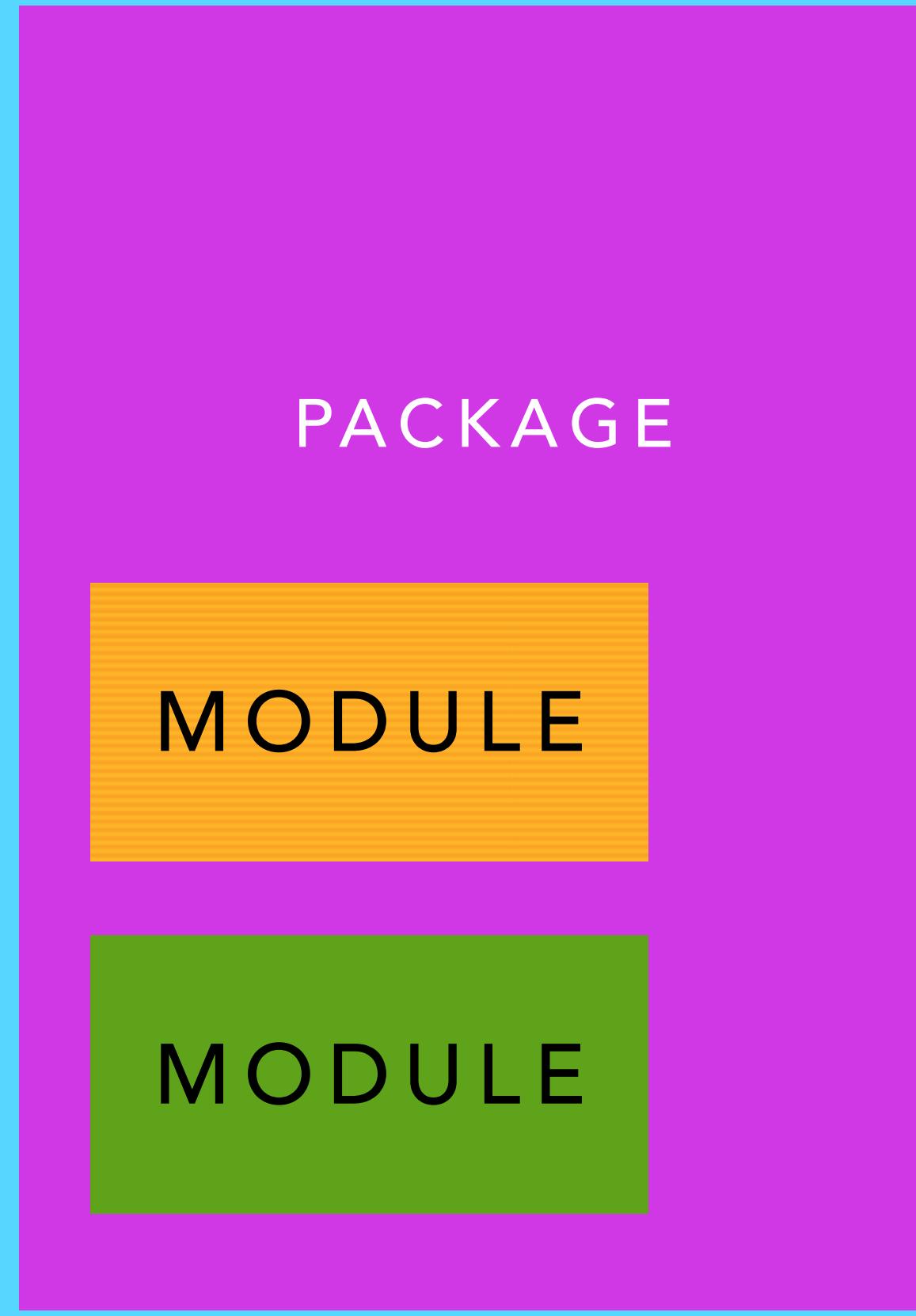
PACKAGE

MODULE

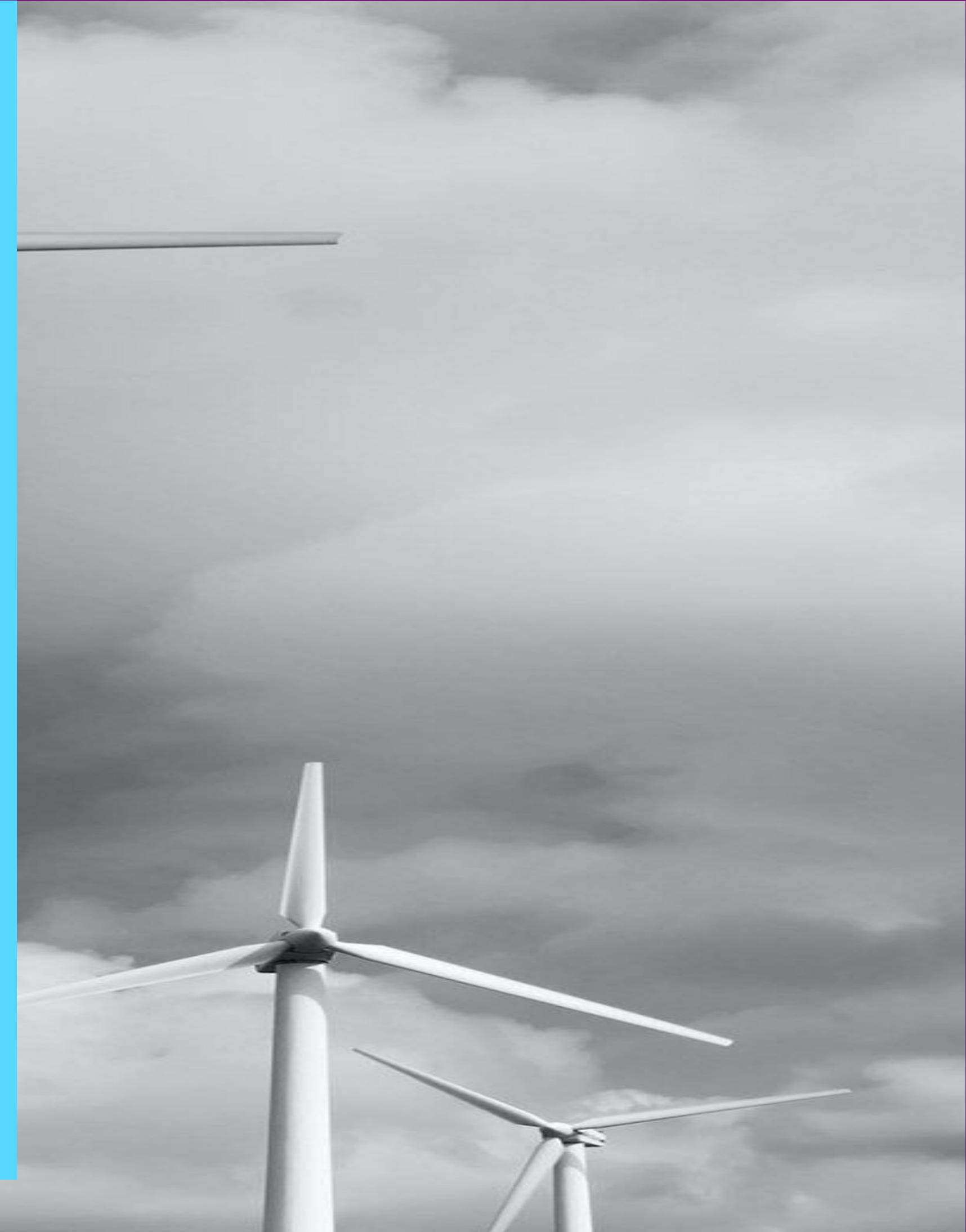
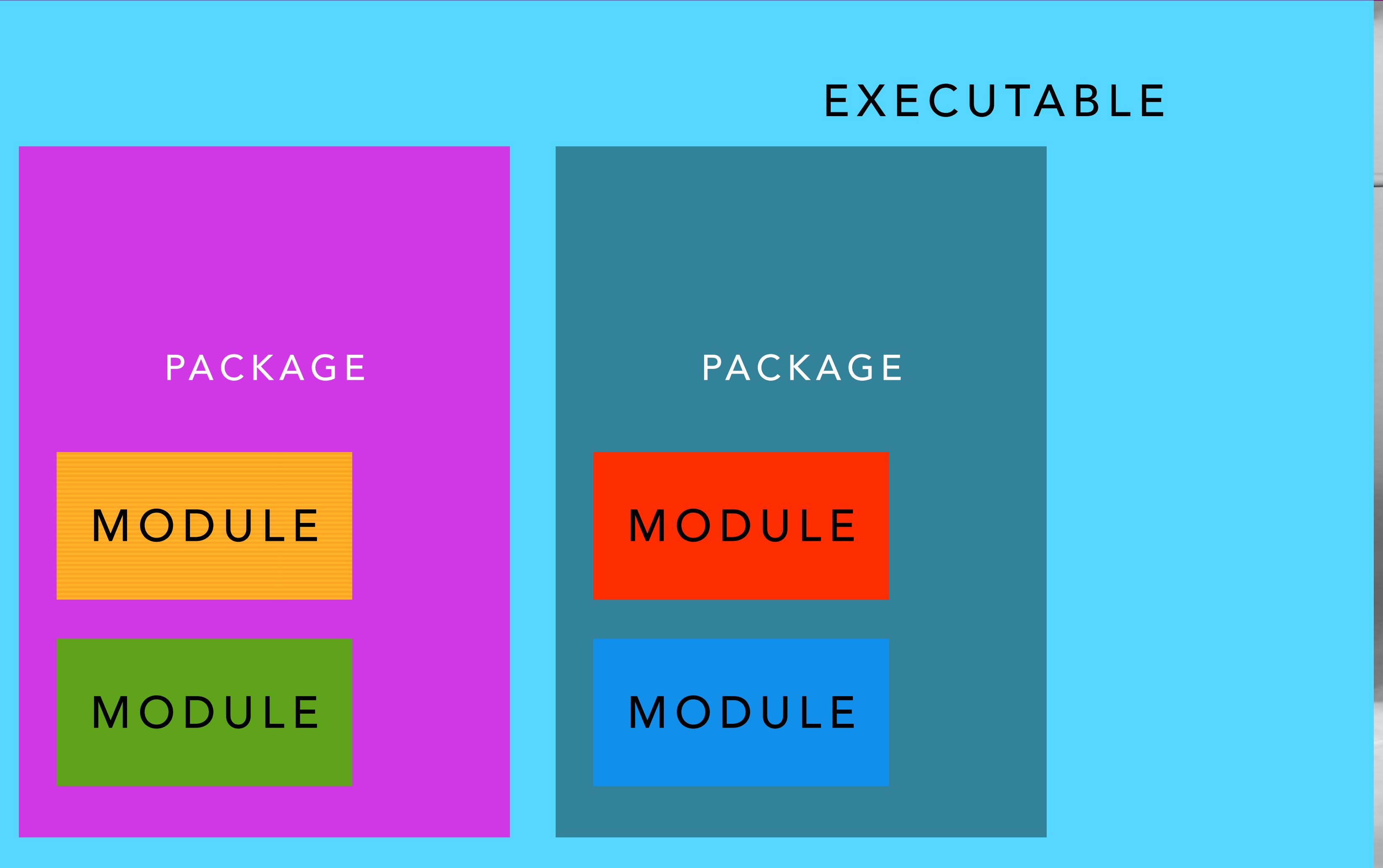
MODULE



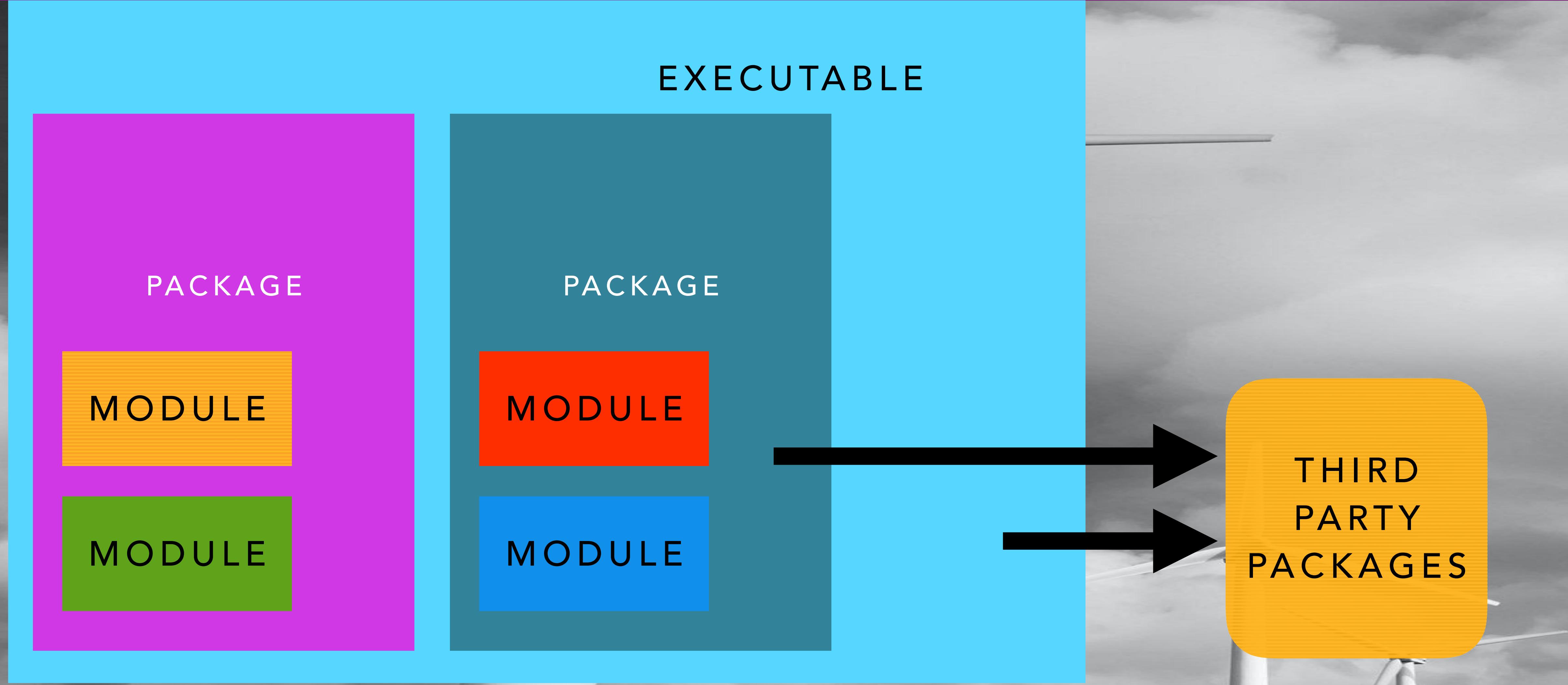
# SWIFT PACKAGE MANAGER



# SWIFT PACKAGE MANAGER



# SWIFT PACKAGE MANAGER



**CREATE A  
PACKAGE**

# CREATE A PACKAGE

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

```
mkdir Hello  
cd Hello
```

## CREATE A PACKAGE

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

```
└── Package.swift
└── README.md
└── Sources
    └── Hello
        └── Hello.swift
└── Tests
    └── HelloTests
        └── HelloTests.swift
    └── LinuxMain.swift
```

# CREATE A PACKAGE

- Creates a directory/file
- IGNORE TESTS FOR NOW
- Name of folder is name of package

```
└── Package.swift
└── README.md
└── Sources
    └── Hello
        └── Hello.swift
└── Tests
    └── HelloTests
        └── HelloTests.swift
└── LinuxMain.swift
```

# CREATE A PACKAGE

The screenshot shows a Xcode workspace with the following structure:

- A project named "session-5" containing:
  - A "Hello" folder with:
    - A "Sources" folder containing a "Hello" folder with a "Hello.swift" file.
  - A "Tests" folder.
  - A "Package.swift" file.
  - A "README.md" file.

The "Hello.swift" file contains the following Swift code:

```
1 struct Hello {  
2     ...  
3     var text = "Hello, World!"  
4 }
```

A yellow callout bubble with a black border and arrow points from the text "Source code for your package" to the "Hello.swift" file in the Xcode interface.

Source code  
for your  
package

# CREATE A PACKAGE

# Manifest file describing the package

The screenshot shows the Xcode interface with the Project Navigator on the left and the Editor on the right.

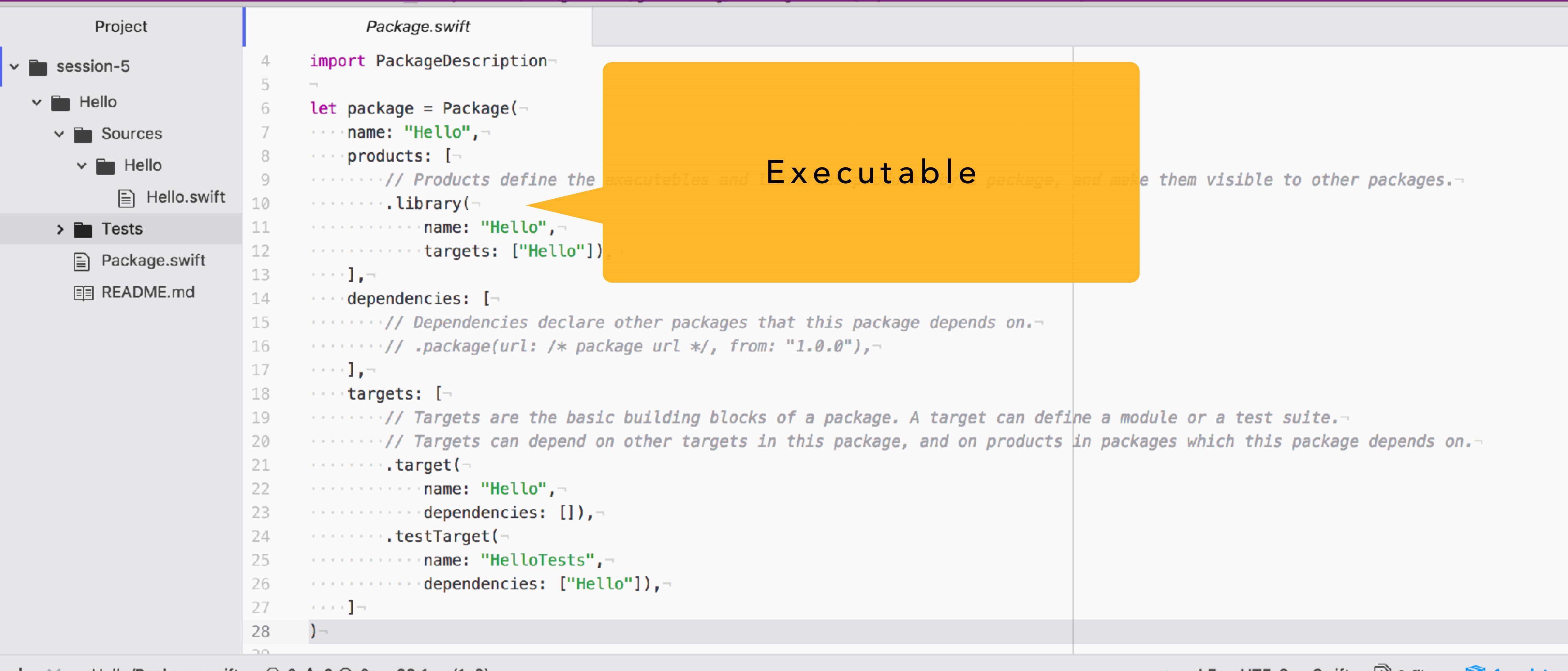
**Project Navigator:**

- Project: session-5
- Folder: Hello
- Folder: Sources
  - Folder: Hello
    - File: Hello.swift
- Folder: Tests
- File: Package.swift
- File: README.md

**Editor (Package.swift):**

```
4 import PackageDescription
5
6 let package = Package(
7   name: "Hello",
8   products: [
9     // Products define the executables and libraries produced by a package, and make them visible to other packages.
10    .library(
11      name: "Hello",
12      targets: ["Hello"]),
13  ],
14  dependencies: [
15    // Dependencies declare other packages that this package depends on.
16    // .package(url: /* package url */, from: "1.0.0"),
17  ],
18  targets: [
19    // Targets are the basic building blocks of a package. A target can define a module or a test suite.
20    // Targets can depend on other targets in this package, and on products in packages which this package depends on.
21    .target(
22      name: "Hello",
23      dependencies: []),
24    .testTarget(
25      name: "HelloTests",
26      dependencies: ["Hello"]),
27  ]
28 )
```

# CREATE A PACKAGE



The screenshot shows the Xcode interface with a project named "session-5". The left sidebar shows files like "Hello.swift" and "Tests". The main editor window displays "Package.swift" with the following code:

```
4 import PackageDescription
5
6 let package = Package(
7   name: "Hello",
8   products: [
9     .library(
10       name: "Hello",
11       targets: ["Hello"]),
12     ],
13   dependencies: [
14     // Dependencies declare other packages that this package depends on.
15     // .package(url: /* package url */, from: "1.0.0"),
16     ],
17   targets: [
18     // Targets are the basic building blocks of a package. A target can define a module or a test suite.
19     // Targets can depend on other targets in this package, and on products in packages which this package depends on.
20     .target(
21       name: "Hello",
22       dependencies: []),
23     .testTarget(
24       name: "HelloTests",
25       dependencies: ["Hello"]),
26     ],
27   )
28 )
```

A yellow callout box points to the line ".library(name: "Hello", targets: ["Hello"])" with the text "Executable".

# CREATE A PACKAGE

The screenshot shows the Xcode interface with the Project Navigator on the left and the code editor on the right. The Project Navigator lists a project named 'session-5' containing a 'Hello' folder with 'Sources' and 'Hello' subfolders, and files 'Hello.swift', 'Tests', 'Package.swift', and 'README.md'. The 'Package.swift' file is selected in the code editor.

```
Project          Package.swift
4 import PackageDescription
5
6 let package = Package(
7   name: "Hello",
8   products: [
9     // Products define the executables and libraries produced by a package, and make them visible to other packages.
10    .library(
11      name: "Hello",
12      targets: ["Hello"]),
13  ],
14  dependencies: [
15    // Dependencies declare other packages that this package depends on.
16    // .package(url: /* package url */, from: "1.0.0"),
17  ],
18  targets: [
19    // Targets are the basic building blocks of a package. A target can define a module or a test suite.
20    // Targets can depend on other targets in this package, and on products in packages which this package depends on.
21    .target(
22      name: "Hello",
23      dependencies: []),
24    .testTarget(
25      name: "HelloTests",
26      dependencies: ["Hello"]),
27  ]
28 )
```

A yellow callout box with the word 'Dependencies' is positioned over the code block, pointing to the line 'dependencies: []'.

# CREATE A PACKAGE

The screenshot shows the Xcode interface with the Project Navigator on the left and the Editor on the right. The Project Navigator lists a project named 'session-5' containing a 'Hello' folder with 'Sources' and 'Hello' subfolders, and files 'Hello.swift', 'Package.swift', and 'README.md'. The 'Tests' folder is expanded. The Editor shows the 'Package.swift' file with the following code:

```
4 import PackageDescription
5
6 let package = Package(
7   name: "Hello",
8   products: [
9     // Products define the executables and libraries produced by a package, and make them visible to other packages.
10    .library(
11      name: "Hello",
12      targets: ["Hello"]),
13  ],
14  dependencies: [
15    // Dependencies declare other packages that this package depends on
16    // .package(url: /* package url */, from: "1.0.0"),
17  ],
18  targets: [
19    // Targets are the basic building blocks of a package. A target can define a module or a test suite.
20    // Targets can depend on other targets in this package, and on products in packages this package depends on
21    .target(
22      name: "Hello",
23      dependencies: []),
24    .testTarget(
25      name: "HelloTests",
26      dependencies: ["Hello"]),
27  ]
28 )
```

A yellow callout bubble points to the 'targets' section of the code, containing the line 'targets: ["Hello"]'. To the right of the callout, the word 'Targets' is written in large black text.

# CREATE A PACKAGE

- Build a package
- Download, resolve and compile dependencies

due | He

# CREATE A PACKAGE

The screenshot shows a code editor with a yellow callout box highlighting a specific section of the `debug.yaml` file. The callout box contains the text: "It builds in a `.build` directory it creates".

**Project Tree:**

- session-5
  - Hello
    - .build
      - checkouts
      - debug
      - repositories
      - x86\_64-apple-r
    - build.db
    - debug.yaml
  - Sources
    - Hello
      - Hello.swift
  - Tests
  - Package.swift
  - README.md

**debug.yaml Content:**

```
client:  
  - name: swift-build  
  - exec: []  
  - targets:  
    - "main": ["<Hello.module>"]  
    - "test": ["<HelloTests.module>","<Hello.module>","<HelloPackageTests.test>"]  
    - "HelloTests.module": ["<HelloTests.module>"]  
    - "Hello.module": ["<Hello.module>"]  
    - "HelloPackageTests.test": ["<HelloPackageTests.test>"]  
  - default: "main"  
  - tool: phony  
  - inputs: ["~/Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Hello/.build/x86_64-apple-ma...]  
  - outputs: ["<Hello.module>"]  
  - "HelloPackageTests.test":  
    - tool: phony  
    - inputs: ["~/Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Hello/.build/x86_64-apple-ma...]  
    - outputs: ["<HelloPackageTests.test>"]  
  - "  
    - <C.HelloTests.module>":  
      - tool: phony  
      - inputs: ["~/Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-autumn/session-5/Hello/.build/x86_64-apple-ma...]  
      - outputs: ["<HelloTests.module>"]
```

# 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

Package.swift

README.md

Sources

— GoodDayToYou



— greetings.swift

— main.swift

Tests

# BUILD AN EXECUTABLE

- Create an executable package

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

# BUILD AN EXECUTABLE

The screenshot shows the Xcode interface with a project named "GoodDayToYou". The "main.swift" file in the "Sources/GoodDayToYou" folder contains the following code:

```
1 print("Hello, world!")
```

The "Project" sidebar on the left lists the project structure:

- GoodDayToYou
  - Sources
    - GoodDayToYou
      - main.swift
  - Tests
  - Package.swift
  - README.md

A yellow callout bubble with the text "DEFAULT" points to the status bar at the bottom of the screen.

# 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

The screenshot shows the Xcode interface with a project named "GoodDayToYou". The project structure is as follows:

- GoodDayToYou**: The main project folder containing:
  - .build**: Build configuration files.
  - Sources**: The source code directory containing:
    - GoodDayToYou**: A folder containing:
      - greetings.swift**: The current file being edited.
      - main.swift**: Another Swift file in the package.
  - Tests**: Test cases for the package.
  - Package.swift**: The package descriptor file.
  - README.md**: A markdown file.

Add another file to  
executable package

# CREATE A PACKAGE

The screenshot shows a Xcode interface with the following details:

- Project Navigator:** On the left, under "session-5", there are several project components:
  - A folder named "Candy".
  - A folder named "GoodDayToYou" which contains ".build" and "Sources".
  - "Sources" contains a folder "GoodDayToYou" with files "greetings.swift" and "main.swift".
  - A folder "Tests".
  - A file "Package.swift".
  - A file "README.md".
  - Folders "Hello" and "TrickOrTreat".
- Code Editor:** The main area displays the "main.swift" file from the "GoodDayToYou" source code.

```
1 //print("Hello, world!")-
2 -
3 if CommandLine.arguments.count != 2 {-
4     ...print("Usage: hello NAME")-
5 } else {-
6     ...let name = CommandLine.arguments[1]-
```

The line "if CommandLine.arguments.count != 2 {" is highlighted with a yellow background.

```
7     ...greet(name: name)-
```

The line "}" is also highlighted with a yellow background.8 }-

The line "9" is visible below the closing brace.
- Tab Bar:** At the top, tabs for "Package.swift", "Candy.swift", "main.swift — TrickOrTreat/Sourc...", and "main.swift — GoodDayToYou/So..." are visible. The "main.swift — GoodDayToYou/So..." tab is currently selected.

# CREATE A PACKAGE

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

The screenshot shows the Xcode interface with a project named "GoodDayToYou". The Project Navigator on the left lists the project structure:

- GoodDayToYou (target)
- .build
- Sources
  - GoodDayToYou
    - greetings.swift
    - main.swift
- Tests
- Package.swift
- README.md

The main editor area displays the contents of the "greetings.swift" file:

```
1 //print("Hello, world!")-
2 -
3 if CommandLine.arguments.count != 2 {-
4 ...print("Usage: hello NAME")-
5 } else {-
6 ...let name = CommandLine.arguments[1]-
7 ...sayHello(name: name)-
8 }-
```

A yellow callout bubble points from the text "No need to import anything because they are from the same module" to the first line of the "greetings.swift" code, which contains the commented-out line `//print("Hello, world!")`.

# 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 cmdline  
argument

# CREATE A PACKAGE

```
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2
017-autumn/session-5/GoodDayToYou
[589 % swift run GoodDayToYou Andrew
Good day to you Andrew 😊.
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2
017-autumn/session-5/GoodDayToYou
590 % ]
```

# EXECUTABLES WITH PACKAGES

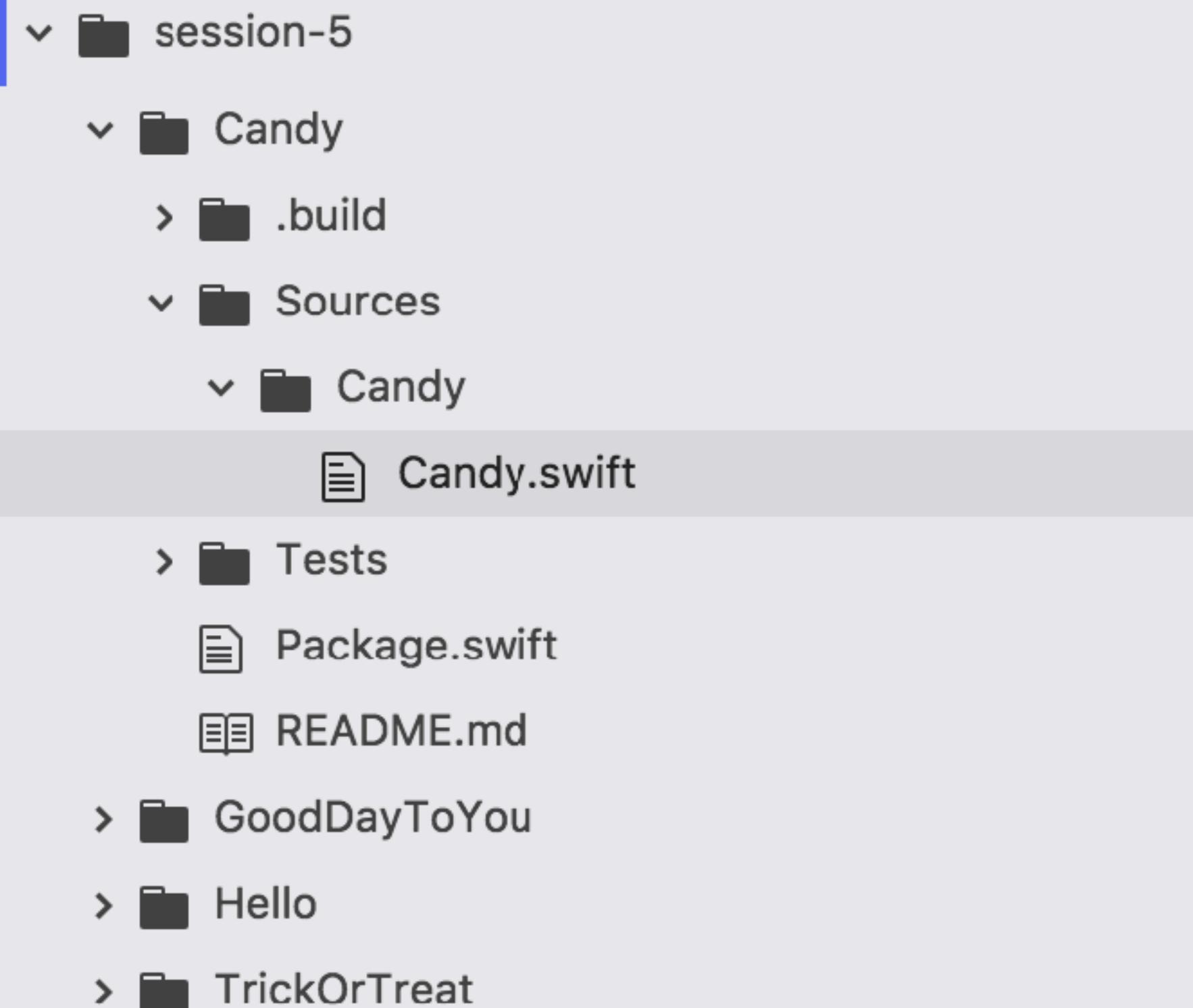
# EXECUTABLES WITH PACKAGES

- 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



```
1  public struct Candy {  
2      public let name: String  
3      public let calories: Int  
4      public var eat: Bool?  
5  }  
6  public init (name: String, calories: Int) {  
7      self.name = name  
8      self.calories = calories  
9  }  
10 }
```

Candy  
package

# EXECUTABLES WITH PACKAGES

## Trick Or Treat Executable

```
session-5
  +--- Candy
  +--- GoodDayToYou
  > - Hello
  +--- TrickOrTreat
    > - .build
    +--- Sources
      +--- TrickOrTreat
        - main.swift
    > - Tests
  - Package.resolved
  - Package.swift
  - README.md
```

```
1 // swift-tools-version:4.0
2 // The swift-tools-version declares the minimum version of Swift required to build
3
4 import PackageDescription
5
6 let package = Package(
7   name: "TrickOrTreat",
8   dependencies: [
9     // Dependencies declare other packages that this package depends on.
10    .package(url: "../Candy", from: "1.0.1"),
11  ],
12  targets: [
13    // Targets are the basic building blocks of a package. A target can define
14    // Targets can depend on other targets in this package, and on products
15    .target(
16      name: "TrickOrTreat",
17      dependencies: ["Candy"]),
18    ]
19 )
20
```

# EXECUTABLES WITH PACKAGES

```
session-5
  > Candy
  > GoodDayToYou
  > Hello
  < TrickOrTreat
    > .build
    < Sources
      < TrickOrTreat
        main.swift
      > Tests
      Package.resolved
      Package.swift
    README.md
```

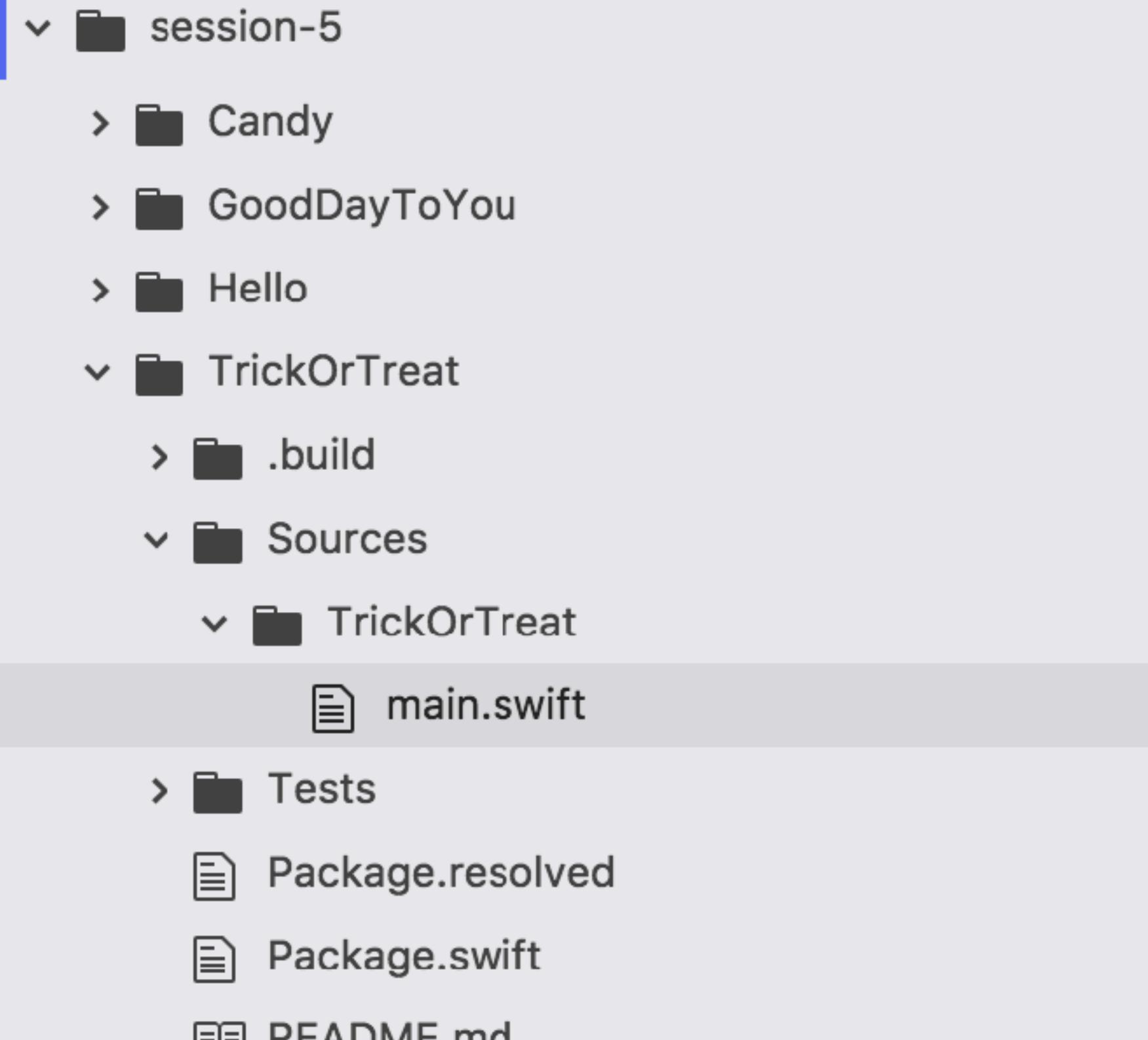
```
1 // swift-tools-version:5.0
2 // The swift-tools-version specifies the version of Swift required to build
3
4 import PackageDescription
5
6 let package = Package(
7   name: "TrickOrTreat",
8   dependencies: [
9     // Dependencies declare other packages that this package depends on.
10    .package(url: "../Candy", from: "1.0.1"),
11  ],
12  targets: [
13    // Targets are the basic building blocks of a package. A target can define
14    // Targets can depend on other targets in this package, and on products
15    .target(
16      name: "TrickOrTreat",
17      dependencies: ["Candy"]),
18    ]
19 )
20
```

version of Swift required to build

Use the Candy package

Use the Candy package

# EXECUTABLES WITH PACKAGES



```
1 import Candy
2
3 let kitkat = Candy(name: "Kit Kat", calories: 200)
4
5 print("🎃 Trick or Treat!")
6 print("Have a \(kitkat.name)")
7 print("😊")
```

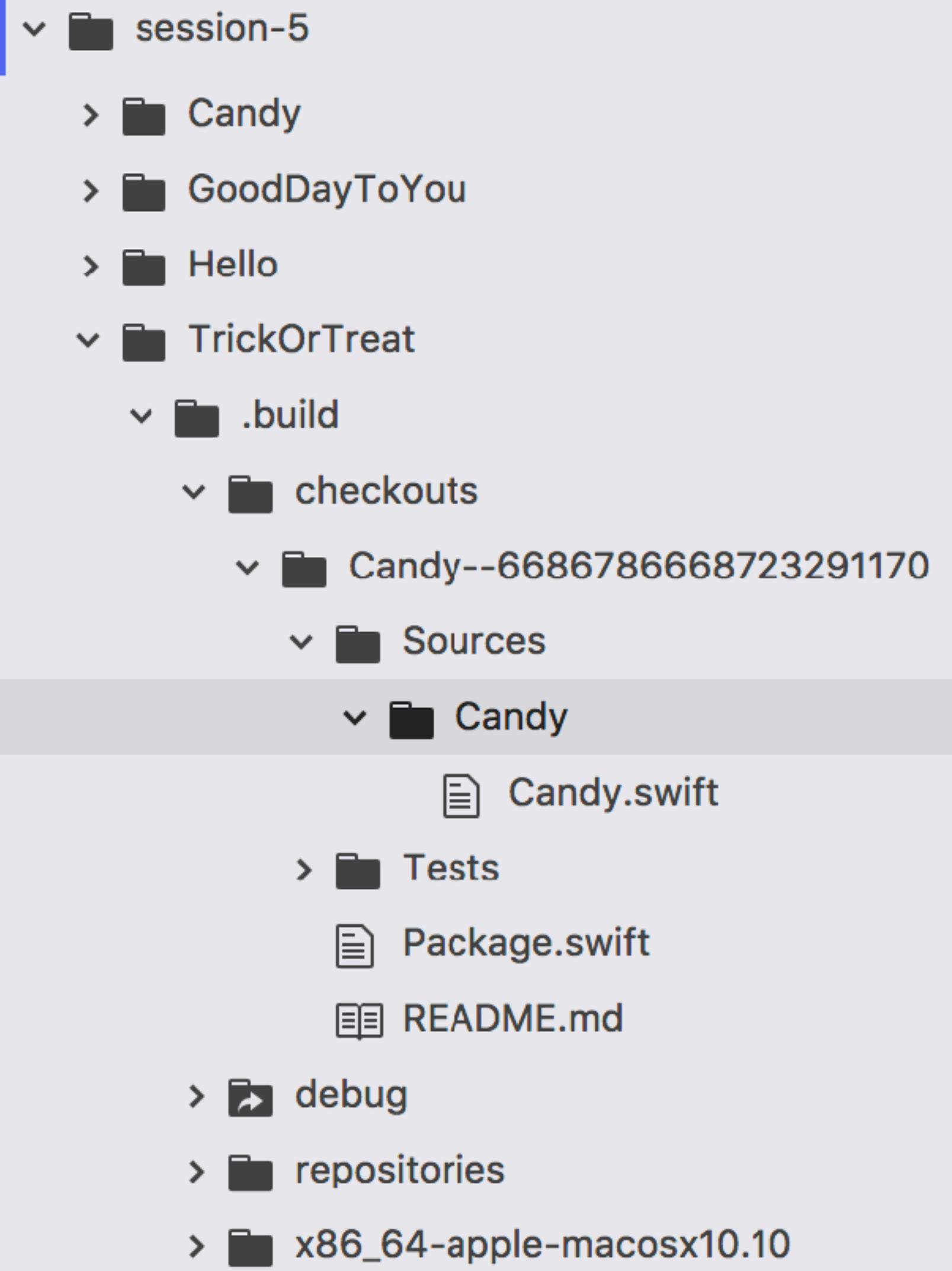
Import Candy

# EXECUTABLES WITH PACKAGES

Build the executable

```
[511 % swift build
Fetching /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mp
cs51043-2017-autumn/session-5/Candy
Cloning /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/mpc
s51043-2017-autumn/session-5/Candy
Resolving /Users/tabinkowski/Google Drive/g-Teaching/uchicago.codes/m
pcs51043-2017-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/m
pcs51043-2017-autumn/session-5/TrickOrTreat
512 % ]
```

# EXECUTABLES WITH PACKAGES



```
1 // swift-tools-version:4.0
2 // The swift-tools-version declares the minimum version of Swift required to build
3
4 import PackageDescription
5
6 let package = Package(
7   name: "Candy",
8   products: [
9     // Products define the executables and libraries produced by a package, and
10    .library(
11      name: "Candy",
12      targets: ["Candy"]),
13    ],
14   dependencies: [
15     // Dependencies declare other packages that this package depends on.
16     // .package(url: /* package url */, from: "1.0.0"),
17   ],
18   targets: [
19     // Targets are the basic building blocks of a package. A target can define
20     // Targets can depend on other targets in this package, and on products in
21     .target(
22       name: "Candy",
23       dependencies: []),
```

**WITH XCODE**

# WITH XCODE

```
Costumes — -bash — 93x14
es.firebaseio.functions — -bash ...1033-2017-autumn-playground — -bash ...mn/session-5/GoodDayToYou — -bash ...-autumn/session-5/Costumes
swift 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
swift package generate-xcodeproj
ed: ./Costumes.xcodeproj
wski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.codes/mpcs51043-2017-aut
Costumes
```

# WITH XCODE

The screenshot shows the Xcode interface with a Swift package structure. The left sidebar displays the project navigation with a selected file 'Package.swift'. The main editor area shows the contents of 'Package.swift' with syntax highlighting for Swift code. The bottom right corner shows the output window displaying the text 'Hello, world!' and 'Program ended with exit code: 0'.

```
// 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

The screenshot shows the Xcode interface with a project named "Costumes". The left sidebar lists files like Package.swift, main.swift, and test cases. The main area shows build settings for macOS, including architectures and code generation options. A large orange callout box with the word "ERROR" is overlaid on the bottom-left of the screen.

**Costumes**

General Resource Tags Build Settings Build Phases Build Rules

Basic Customized All Combined Levels + Q supported

Architectures Setting Costumes

Supported Platforms macOS

Apple LLVM 9.0 - Code Generation Setting Costumes

<Multiple values> ◊ None [-O0] ◊ Fastest, Smallest [-Os] ◊

Optimization Level Debug Release

**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

ERROR

# THIRD PARTY PACKAGES

# THIRD PARTY PACKAGES

## *Elegant Networking in Swift*

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:

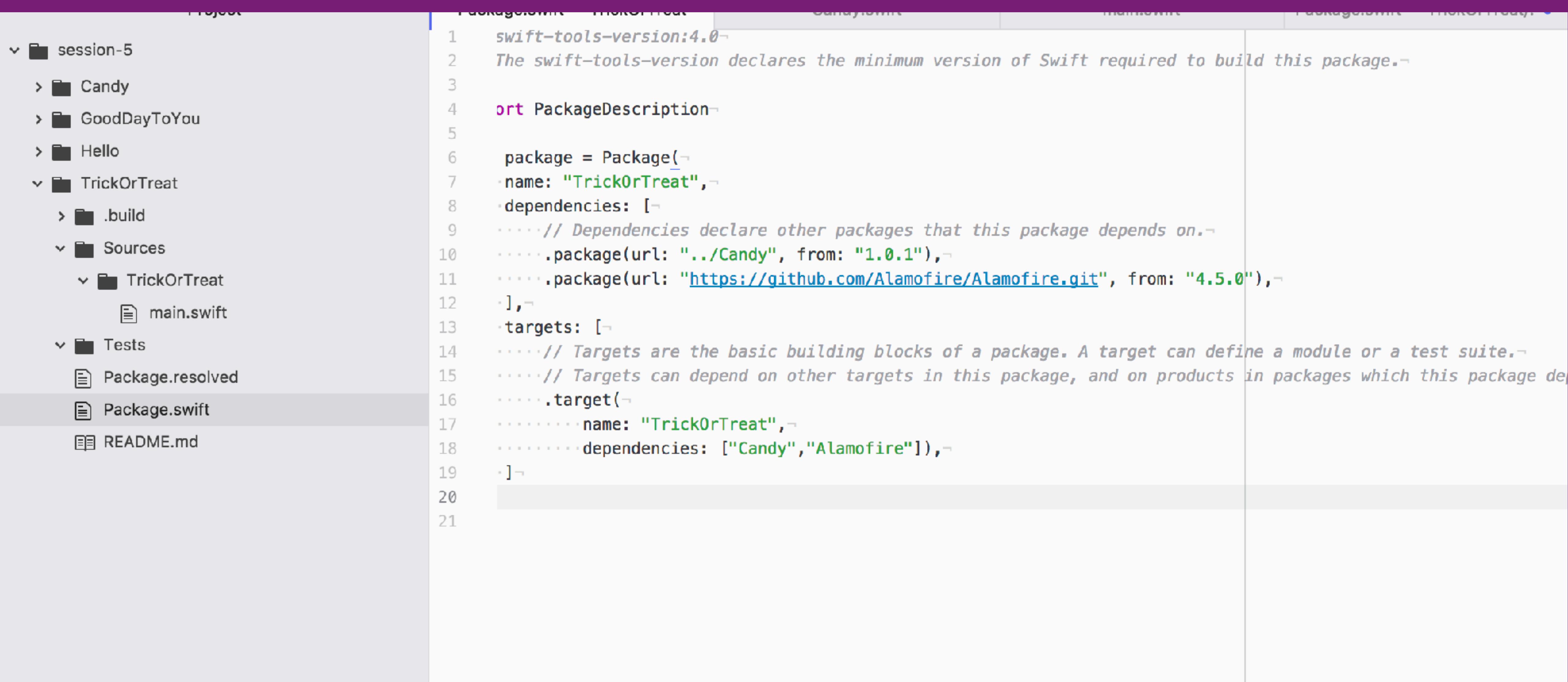
```
$ brew update  
$ brew install carthage
```

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



The screenshot shows a Xcode project interface with the following structure:

- Project** tab is selected.
- session-5** project is open.
- TrickOrTreat** is a selected package.
- Package.swift** file is open in the editor.
- Code Content:**

```
1 swift-tools-version:4.0
2 The swift-tools-version declares the minimum version of Swift required to build this package.
3
4 import PackageDescription
5
6 package = Package(
7   name: "TrickOrTreat",
8   dependencies: [
9     // Dependencies declare other packages that this package depends on.
10    .package(url: "../Candy", from: "1.0.1"),
11    .package(url: "https://github.com/Alamofire/Alamofire.git", from: "4.5.0"),
12  ],
13  targets: [
14    // Targets are the basic building blocks of a package. A target can define a module or a test suite.
15    // Targets can depend on other targets in this package, and on products in packages which this package depends on.
16    .target(
17      name: "TrickOrTreat",
18      dependencies: ["Candy", "Alamofire"]),
19  ]
20
21
```

# 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 ▾

File / Commit	Description	Date
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

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

## SEARCH\*



\* Type here to search by name, version, author, keywords, summary, and dependencies.

## WHAT IS COCOAPODS

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. It has over 38 thousand libraries and is used in over 2.7 million apps. CocoaPods can

## OTHER PACKAGE MANAGERS

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

## SEARCH\*



\* Type here to search by name, version, author, keywords, summary, and dependencies.

## WHAT IS COCOAPODS

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. It has over 38 thousand libraries and is used in over 2.7 million apps. CocoaPods can

# OTHER PACKAGE MANAGERS

- [https://  
www.youtube.com  
/watch?  
v=iEAjvNRdZa0](https://www.youtube.com/watch?v=iEAjvNRdZa0)



Introduction to CocoaPods (Route 85)

5 views

849

11

SHARE



## OTHER PACKAGE MANAGERS

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

## SEARCH\*

\* Type here to search by name, version, author, keywords, summary, and dependencies.



## WHAT IS COCOAPODS

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. It has over 38 thousand libraries and is used in over 2.7 million apps. CocoaPods can

# 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.

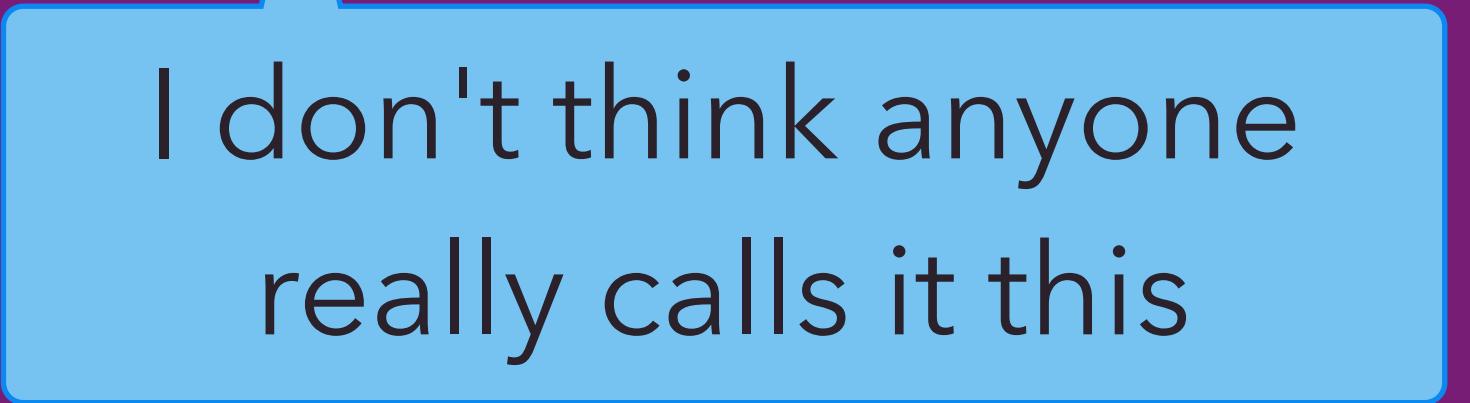
# OTHER PACKAGE MANAGERS

- 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

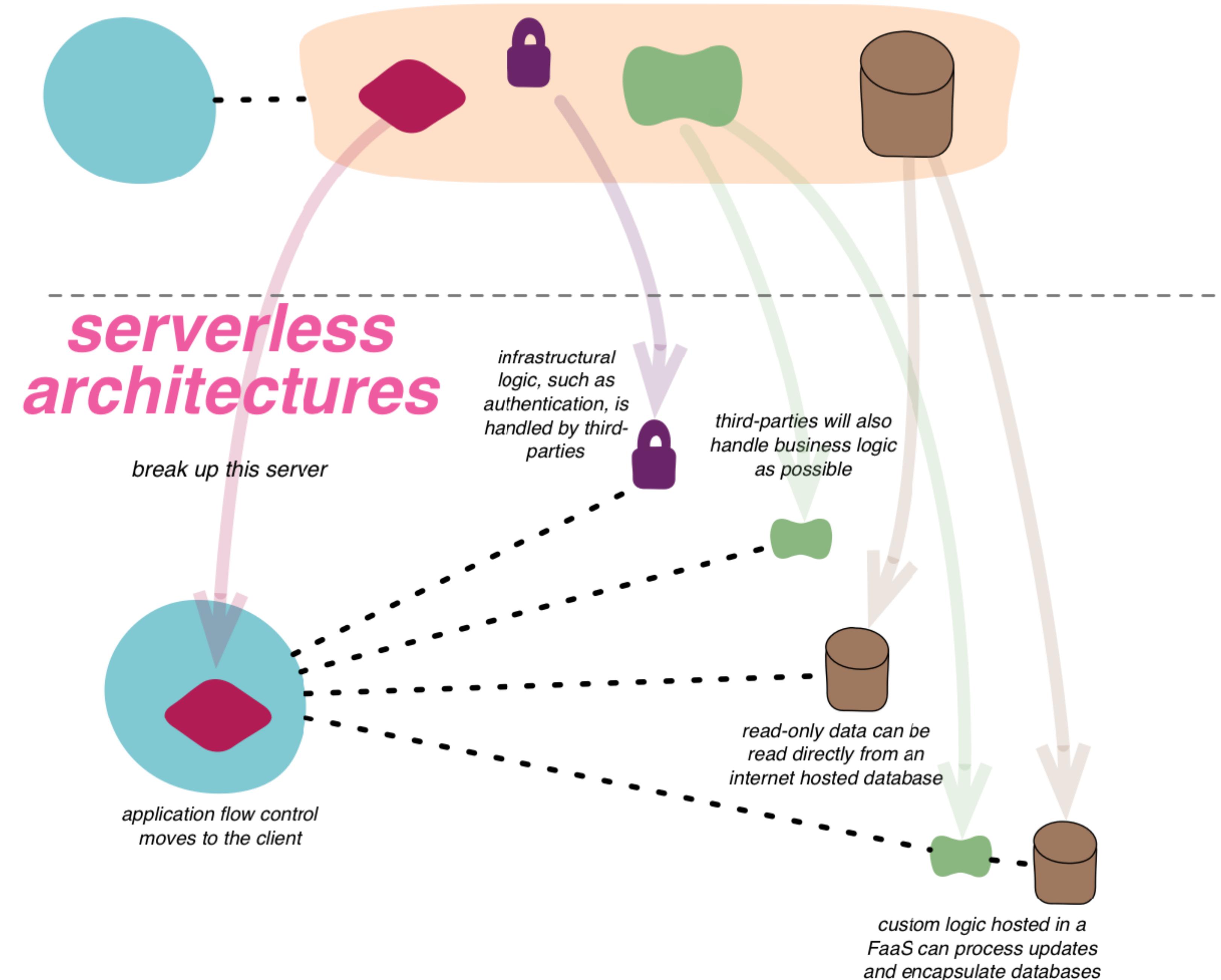
# SERVERLESS ARCHITECTURES

- 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

A traditional internet delivered app has a client communicating with a long-lived server process that handles most aspects of the application's logic

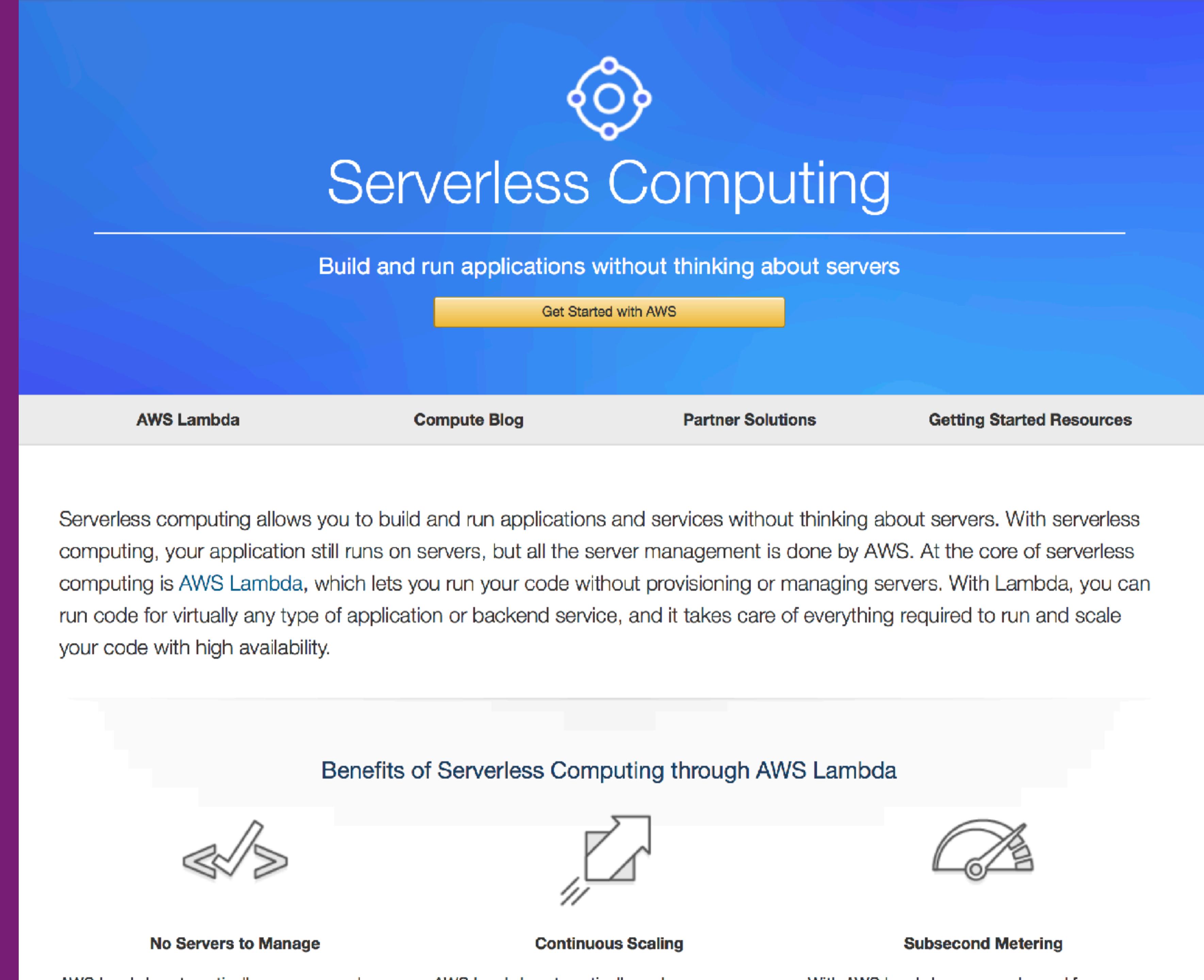


# SERVERLESS ARCHITECTURES

- 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



The screenshot shows the AWS Serverless Computing landing page. At the top center is a blue header with the text "Serverless Computing" and a circular icon. Below the header is a sub-header "Build and run applications without thinking about servers". A yellow "Get Started with AWS" button is positioned below the sub-header. The main content area features a large text block explaining the concept of serverless computing, mentioning AWS Lambda and its benefits. Below this text are three icons representing "No Servers to Manage", "Continuous Scaling", and "Subsecond Metering". The footer contains navigation links for "AWS Lambda", "Compute Blog", "Partner Solutions", and "Getting Started Resources".

Serverless Computing

Build and run applications without thinking about servers

Get Started with AWS

AWS Lambda Compute Blog Partner Solutions Getting Started Resources

Serverless computing allows you to build and run applications and services without thinking about servers. With serverless computing, your application still runs on servers, but all the server management is done by AWS. At the core of serverless computing is [AWS Lambda](#), which lets you run your code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service, and it takes care of everything required to run and scale your code with high availability.

Benefits of Serverless Computing through AWS Lambda

No Servers to Manage Continuous Scaling Subsecond Metering

AWS Lambda Compute Blog Partner Solutions Getting Started Resources

# SERVERLESS ARCHITECTURES

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

Realm Tasks With Wit

Regex

# Serverless Logic with Realm: Introducing Realm Functions

```
var Wit = require("node-wit").Wit;
var WIT_ACCESS_TOKEN = "YBXCCZLQH7V7P7ZKJ7V7P7ZKJ7V7P7ZK";
var witClient = new Wit({accessToken: WIT_ACCESS_TOKEN});

module.exports = function(change_event) {
  var realm = change_event.realm;
  var changes = change_event.changes.Task;
  var taskIndexes = changes.modifications;
  console.log("Changes detected: ", changes);
  taskIndexes.forEach(function(index) {
    realm.write(function() {
      realm.deleteTask(index);
    });
  });
}
```

Realm Team

May 23 2017

< Go to News      Transcript      About the Speaker      Twitter      Facebook      Email

Today we're announcing **Realm Functions**, a new part of Realm that makes building server-side functionality a lot easier for mobile developers. Now, you can make server-side features without enlisting backend developers, plus you get all the benefits of building on top of the Realm Mobile Platform: you don't need to add another endpoint to a server, and then write the serialization and networking code that would let you connect with it. You just connect your app to Realm, write a **Realm Function** in your web dashboard, and watch your code execute reactively as data streams in. Today's release is a **beta**, and it's available today to everyone, whether you're building an app in an enterprise-scale team or for a small side project.

# SERVERLESS ARCHITECTURES

- Frameworks for server less
  - Write once, deploy everywhere

## The way cloud should be.

Serverless is your toolkit for deploying and operating serverless architectures. Focus on your application, not your infrastructure.

[Quick Start Docs](#)

[Sign Up](#)



```
# Install serverless globally  
$ npm install serverless -g  
  
# Login to your Serverless account  
$ serverless login  
  
# Create a serverless function  
$ serverless create --template hello-world  
  
# Deploy to cloud provider  
$ serverless deploy  
  
# Function deployed! Trigger with live url  
$ http://xyz.amazonaws.com/hello-world
```



Serverless Framework



Event Gateway



# 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

## Storage

- S3
- EFS
- Glacier
- Storage Gateway

## Database

- RDS
- DynamoDB
- ElastiCache

## Developer Tools

- CodeStar
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- X-Ray

## Management Tools

- CloudWatch
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Trusted Advisor
- Managed Services

## Analytics

- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- Data Pipeline
- QuickSight 

## Artificial Intelligence

- Lex
- Polly
- Rekognition
- Machine Learning

## Internet Of Things

- AWS IoT

## Application Services

- Step Functions
- SWF
- API Gateway
- Elastic Transcoder

## Messaging

- Simple Queue Service
- Simple Notification Service
- SES

## Business Productivity

- WorkDocs
- WorkMail
- Amazon Chime 

## Desktop & App Streaming

AWS has a lot  
of offerings

# AWS LAMBDA

- AWS Lambda  
serverless  
architecture

The screenshot shows the AWS Lambda landing page. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, 'T. Andrew Binkowski', and other user options. Below the navigation is a sidebar titled 'AWS Lambda' with 'Dashboard' and 'Functions' options. The main content area has a dark background with white text. It features a large heading 'AWS Lambda' followed by the subtext 'lets you run code without thinking about servers.' Below this, a paragraph explains the cost model: 'You pay only for the compute time you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.' To the right, there's a 'Get started' section with a button to 'Create a function'. Further down, a 'How it works' section describes the serverless nature of Lambda and includes a link to 'Read more in FAQs'. On the far right, a sidebar titled 'More resources' lists links to 'Documentation', 'API reference', 'Serverless Application Model', 'SAM Local', and 'Forums'.

**AWS Lambda**

COMPUTE

# AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

## How it works

Upload your code to AWS Lambda, set it up to trigger from other AWS services, HTTP endpoints, or in-app activity, and let Lambda run and scale your code with high availability — all without provisioning or managing any servers.

[Read more in FAQs](#)

**Get started**

Author a Lambda function from scratch, or choose from one of preconfigured examples.

**Create a function**

**More resources**

[Documentation](#)

[API reference](#)

[Serverless Application Model](#)

[SAM Local](#)

[Forums](#)

## Related services

# 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

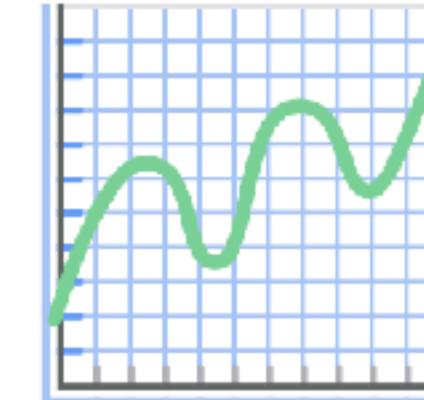


## AWS Lambda

AWS Lambda lets you run code in response to events, without provisioning or managing servers. Just upload your code and Lambda will take care of everything required to run and scale it with high availability.

[Get Started Now](#)

[Learn more about AWS Lambda](#)



### Manage

our code without  
ge servers. Just write

### Continuous Scaling

AWS Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

With AWS Lambda, you a  
code executes and the nu  
triggered. You don't pay a  
running.

[AWS Lambda Documentation and Support](#)

# AWS LAMBDA

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

Lambda > New function

## Select blueprint

Configure triggers

Configure function

Review



## Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your desired scenario and customize as needed, or skip this step if you want to author a Lambda function and configure an event source separately. Except where otherwise noted, blueprints are licensed under [CC0](#).

Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below.



Select runtime

Filter

« < Viewing 1-9 of 94 > »

### Blank Function

Configure your function from scratch.  
Define the trigger and deploy your code  
by stepping through our wizard.

custom

### kinesis-firehose-syslog-to-json

An Amazon Kinesis Firehose stream  
processor that converts input records  
from RFC3164 Syslog format to JSON.

nodejs · kinesis-firehose

### alexa-skill-kit-sdk-factskill

Demonstrate a basic fact skill built with  
the ASK NodeJS SDK

nodejs6.10 · alexa



### batch-get-job-python27

Returns the current status of an AWS  
Batch Job.

python2.7 · batch

### kinesis-firehose-apachelog-to...

An Amazon Kinesis Firehose stream  
processor that converts input records  
from Apache Common Log format to

python2.7 · kinesis-firehose

### cloudfront-modify-response-h...

Blueprint for modifying CloudFront  
response header implemented in  
NodeJS.

nodejs · cloudfront · response header



# AWS LAMBDA

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

## hello-world-python

A starter AWS Lambda function.

---

python2.7



# AWS LAMBDA

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

## Basic information [Info](#)

Name\*

Role\*

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s) ▾

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name\*

Enter a name for your new role.

Policy templates

Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

Basic Edge Lambda permissions X

# 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

Select blueprint

Configure triggers

Configure function

Review

## Configure function

A Lambda function consists of the custom code you want to execute. [Learn more about Lambda functions.](#)

Name\*

Description

Runtime\*

### Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

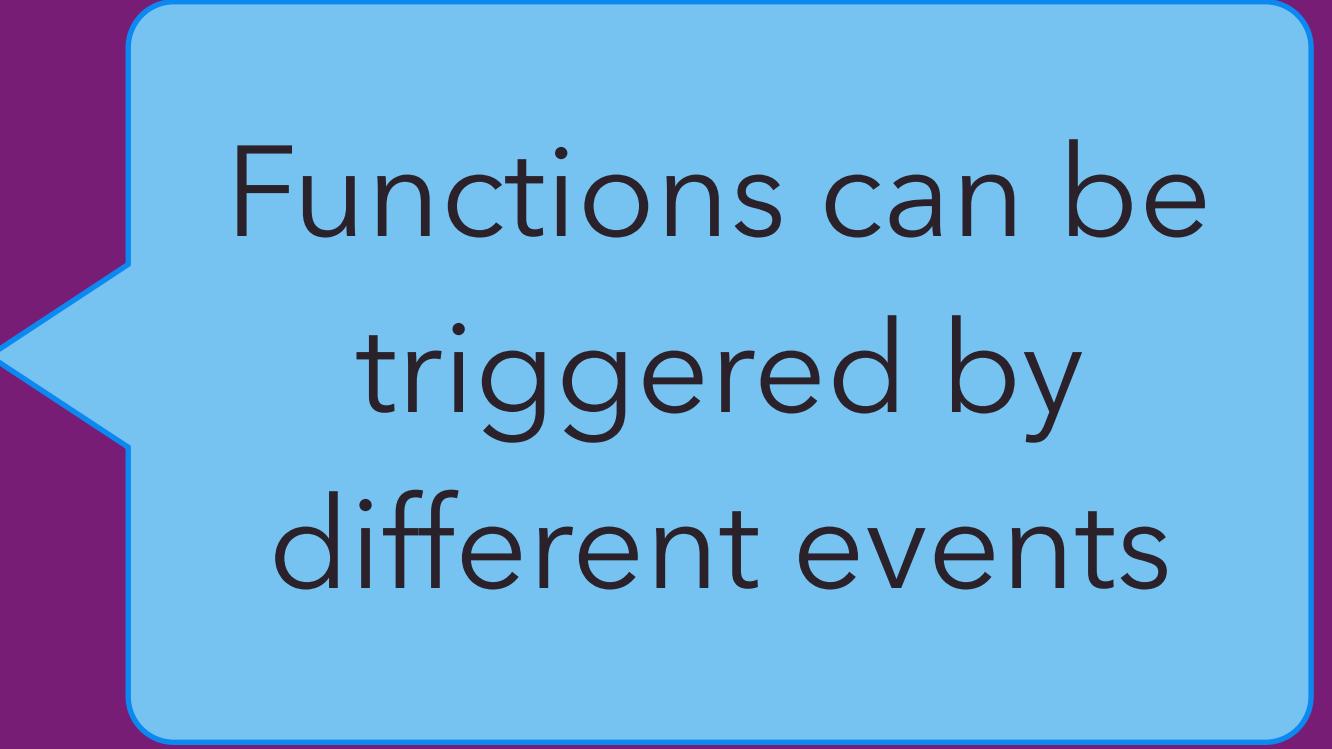
Code entry type

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return event['key1'] # Echo back the first key-value
```

Limited third party modules

# AWS LAMBDA

- Triggers define when your function runs
  - Event based on 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

## Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8- def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13    return event['key1'] # Echo back the first key value
14    #raise Exception('Something went wrong')
15
```

main function

You can define Environment Variables as key-value pairs that are accessible from your function code. To settings without the need to change function code. [Learn more](#). For storing sensitive information, we re and the console's encryption helpers.

# AWS LAMBDA

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

## Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8- def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13    return event['key1'] # Echo back the first key value
14    #raise Exception('Something went wrong')
15
```

You can define Environment Variables as key-value pairs that are accessible from your function code. To set environment variables, go to the Environment Variables tab. You can also define environment variables in the Lambda function configuration. Environment variables are useful for storing configuration settings without the need to change function code. [Learn more](#). For storing sensitive information, we recommend using AWS KMS or AWS Secrets Manager. You can also use the Lambda console's encryption helpers.

# AWS LAMBDA

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

```
function.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



Sufficient for Hello  
World

**Timeout\***

0

min

3

sec

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

Execution result: succeeded ([logs](#))

▼ Details

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

"value1"

Test

### Summary

Code SHA-256

6chaTsf/AFVe6WIJ3HrQFLvl  
t7xggyqTeTa8DvnxQeA=

Request ID

# AWS LAMBDA

Lambda > Functions > hello-world-2017-autym

ARN - arn:aws:lambda:us-east-2:735044932848:f

## hello-world-2017-autym

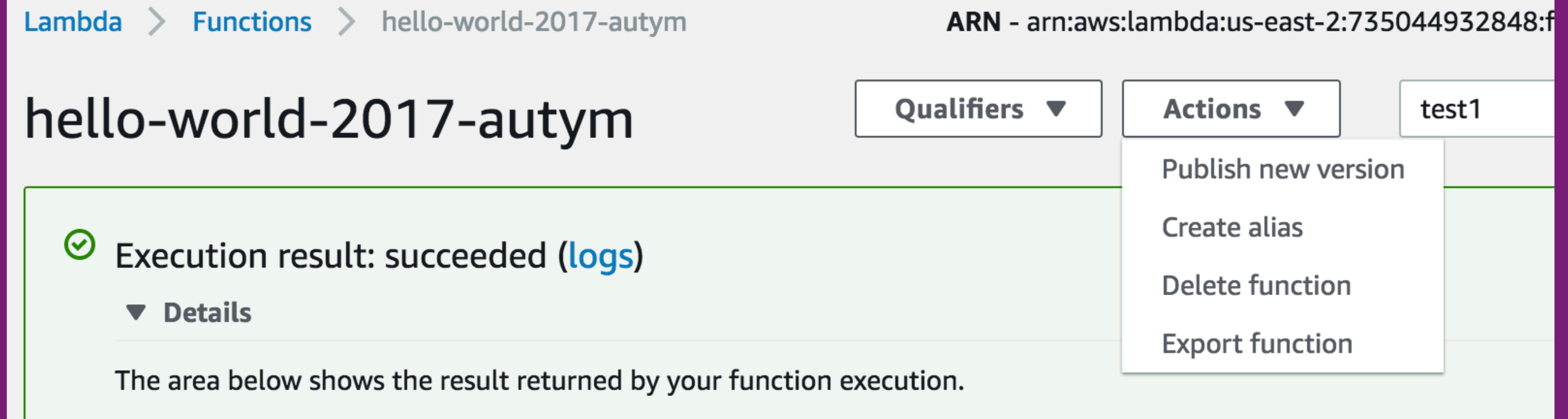
Qualifiers ▾ Actions ▾ test1

Execution result: succeeded ([logs](#))

▼ Details

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

Publish new version  
Create alias  
Delete function  
Export function



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

# AWS LAMBDA

- Test function
  - Response
  - Log output
  - Runtime summary

Execution result: succeeded ([logs](#))

▼ Details

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

"value1"

**Summary**

Code SHA-256  
6chaTsf/AFve6WLJ3HrQFLvl  
t7xggyqTeTa8DvnxQeA=

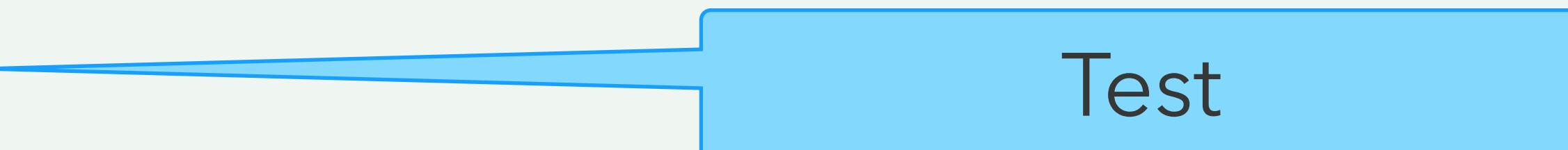
Request ID  
9cd2f27a-d3de-11e7-ad1b-  
97d23359cdfc

Duration  
0.26 ms

Billed duration  
100 ms

Resources configured  
128 MB

Max memory used  
19 MB



# 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
```

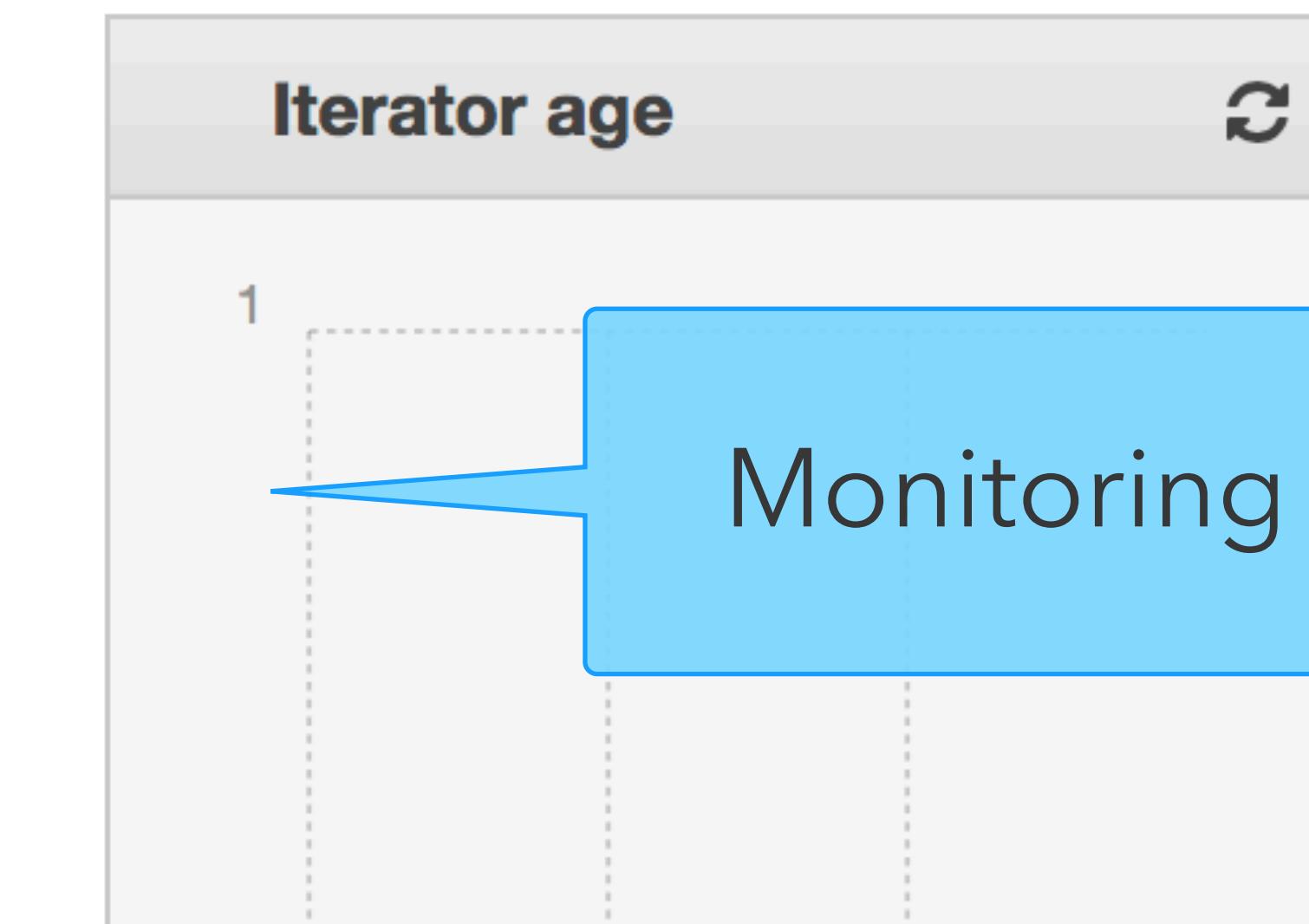
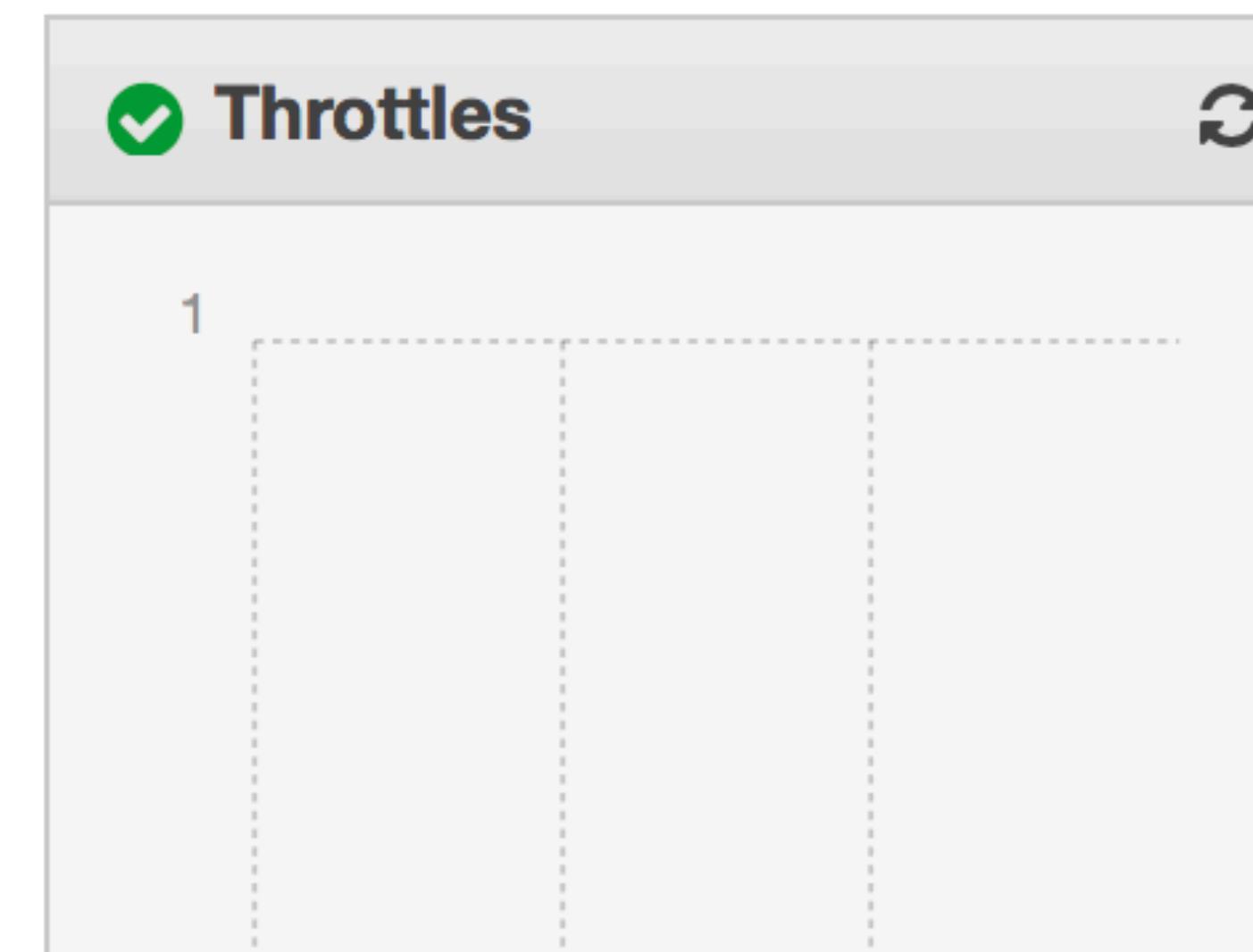
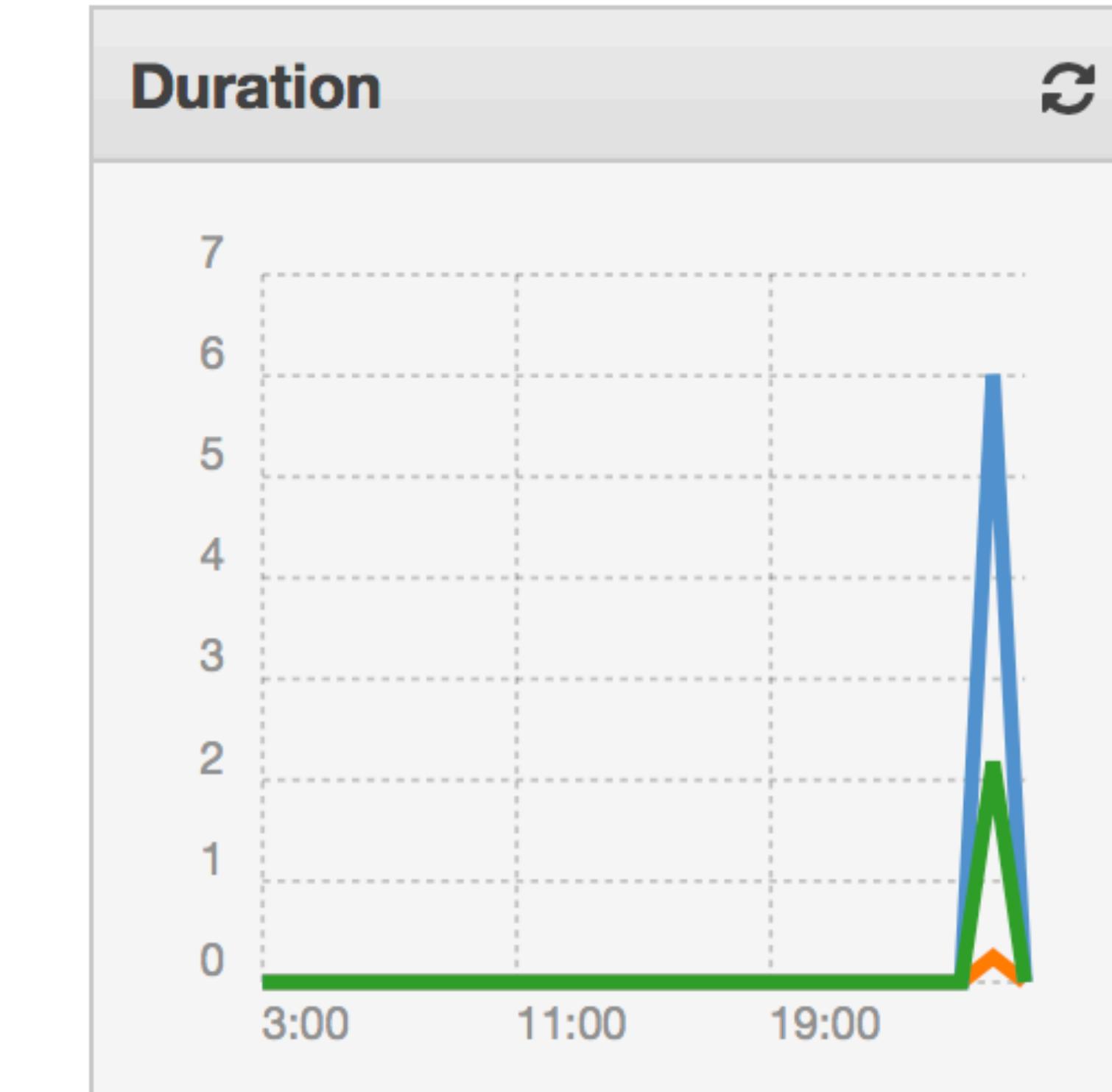
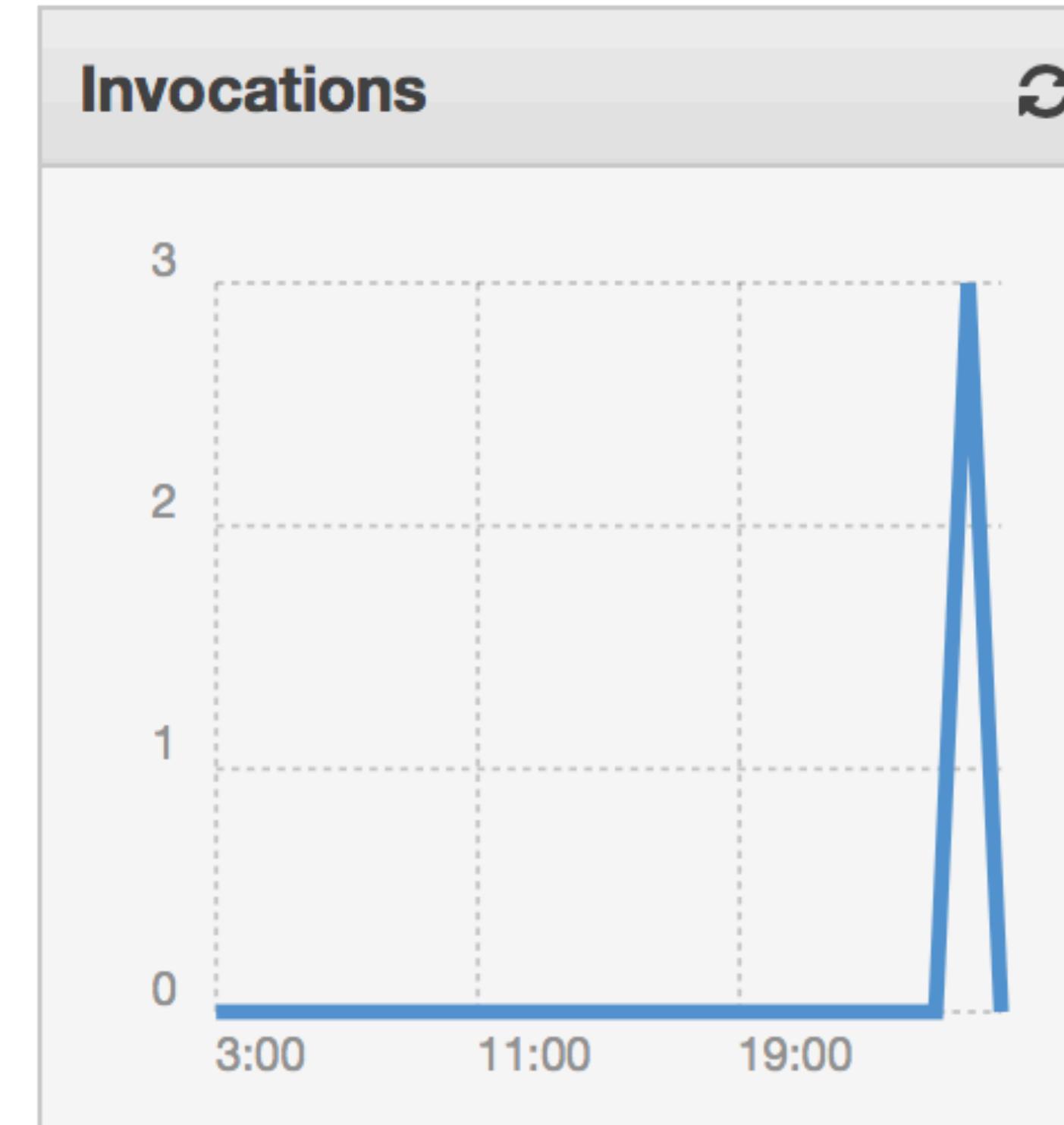
```
Memory Size:
```

```
128 MB Max Memory Used: 19 MB
```

# AWS LAMBDA

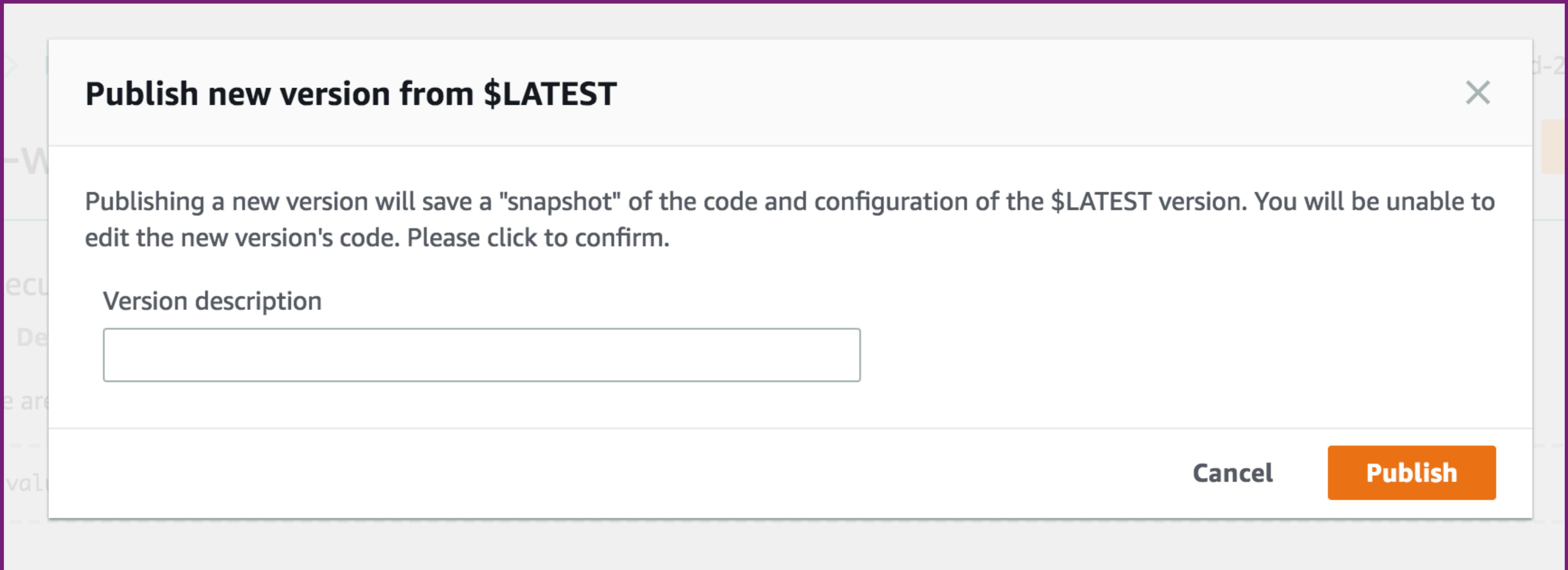
- Monitor your function

## CloudWatch metrics at a glance (last 24 hours)



Monitoring

# AWS LAMBDA



- Publish

# AWS LAMBDA

- Set up a trigger

## Add trigger



Configure your Lambda function **helloWorld** to respond to events from the selected trigger. Click on the box below to select your trigger type.



*Filter integrations*

API Gateway

AWS IoT

CloudFront

CloudWatch Events - Schedule

CloudWatch Logs

CodeCommit

Cognito Sync Trigger

DynamoDB

**Cancel**

**Submit**

# AWS LAMBDA

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

Add trigger X

Configure your Lambda function **helloWorld** to respond to events from the selected trigger. Click on the box below to select your trigger type.

API Gateway  → Lambda 

Please go to the [IAM console](#) to configure the security for your API endpoint. X

We'll set up an API Gateway endpoint with a [proxy integration type](#) (learn more about the [input](#) and [output](#) format for your function). Any method (GET, POST, etc.) will trigger your Lambda function. To set up more advanced method mappings or subpath routes, visit [Amazon API Gateway console](#).

**API name**  i

**Deployment stage**  i

**Security**  i

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function. [Learn more](#) about the Lambda permissions model.

Cancel Submit

# 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.

X

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

The screenshot shows the AWS API Gateway interface for a Lambda function named "LambdaMicroservice". The left sidebar lists various API management features like Resources, Stages, Authorizers, Models, Documentation, Binary Support, and a Dashboard. The main content area is focused on the "Resources" tab, specifically the "/helloWorld" endpoint under the "ANY" method. The "Actions" dropdown is open, showing "Edit" and "Test".

**Query Strings**

**{helloWorld}**

d

**Headers**

**{helloWorld}**

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg.  
Accept:application/json.

**Stage Variables**

No stage variables exist for this method.

**Client Certificate**

No client certificates have been generated.

**Request Body**

1

**Response Headers**

{}

**Logs**

Execution log for request test-request  
Wed May 31 08:20:19 UTC 2017 : Starting execution for request: test-invoke-request  
Wed May 31 08:20:19 UTC 2017 : HTTP Method: POST, Resource Path: /helloWorld  
Wed May 31 08:20:19 UTC 2017 : Method request path: {}  
Wed May 31 08:20:19 UTC 2017 : Method request query string: {d=null}  
Wed May 31 08:20:19 UTC 2017 : Method request headers: {}  
Wed May 31 08:20:19 UTC 2017 : Method request body before transformations:  
Wed May 31 08:20:19 UTC 2017 : Endpoint request URI: https://lambda.us-east-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-east-2:735044932848:function:helloWorld/inversations  
Wed May 31 08:20:19 UTC 2017 : Endpoint request headers: {x-amzn-lambda-integration-type=test-request, Authorization=\*\*\*\*\*4f5dc9, X-Amz-Source-Arn=arn:aws:execute-api:us-east-2:735044932848:c3xcr7zk94, X-Amz-Date=20170531T082019Z, x-amzn-apigateway-api-id=c3xcr7zk94, X-Amz-Security-Token=FQoDYXdzEIj//////////wEaDFN0RkKM/WfU5j0lnSK3A10UKq5jlmCIDc/5t1KIIk3sGvGZHsE0JkKw8NCRY8e0GK4UxFJG0FhfTCmS/NT9B23ftWhb+L9qqW/y50le+jhvQA1EUAluH/Hjt0eeDpYkkmW/7sJ+cXzXthk3d16jZkCemSdSNmqS13aIeSSYu4dWJXjVsm6HULVta+qbIpr5uNgRT4LThFugAvHKtpcw/VWSvW5TBk/ji5tzsb0Sl21KM0YfeBtaF8W7jwyYx4xCy/tC4jQoZIh4YPM0cqEyV4W5D3DuS0HCi/4Z8V2G867eI4f5TM3cURfBiHXExd5l6+gLOOSchNEhtI [TRUNCATED]  
Wed May 31 08:20:19 UTC 2017 : Endpoint request body after transformations: {"resource":"/helloWorld","path":"/helloWorld","httpMethod":"POST","headers":null,"queryStringParameters":{"d":null}, "pathParameters":null,"stageVariables":null,"requestContext":{"path":"/helloWorld","accountId":"735044932848","resourceId":"3svakf","stage":"test-invok

- 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

CloudWatch Events



Lambda

## Rule

Pick an existing rule, or create a new one.

Create a new rule



Select or create a new rule

## Rule name\*

Enter a name to uniquely identify your rule.

## Rule description

Provide an optional description for your rule.

## Rule type

Trigger your target based on an event pattern, or based on an automated schedule.

Event pattern

Schedule expression

## Schedule expression\*

Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

e.g. rate(1 day), cron(0 17 ? \* MON-FRI \*)

# 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
<a href="#">Invoke</a> request body payload size (RequestResponse)	6 MB
<a href="#">Invoke</a> request body payload size (Event)	128 K
<a href="#">Invoke</a> response body payload size (RequestResponse)	6 MB

The following table lists the Lambda account limits per region.

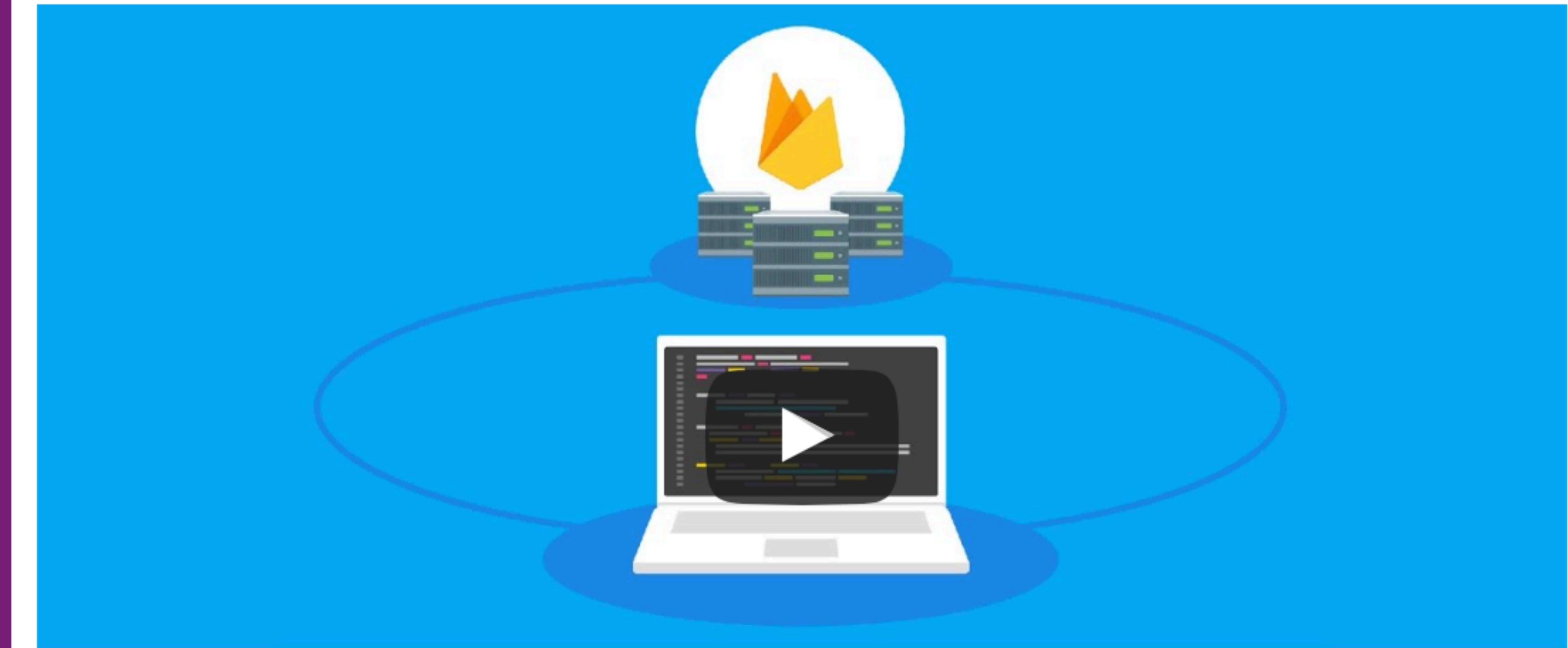
## AWS Lambda Account Limits Per Region

Resource	Default Limit
Concurrent executions (see <a href="#">Lambda Function Concurrent Executions</a> )	1000

# FIREBASE CLOUD FUNCTIONS VS LAMBDA

# FIREBASE CLOUD FUNCTIONS

- <https://youtu.be/vr0Gfvp5v1A>



**Cloud Functions**  
for Firebase

# 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



# AWS Lambda

Run code without thinking about servers  
Pay for only the compute time you consume

[Get started with AWS Lambda](#)

# 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-browser development and deployment

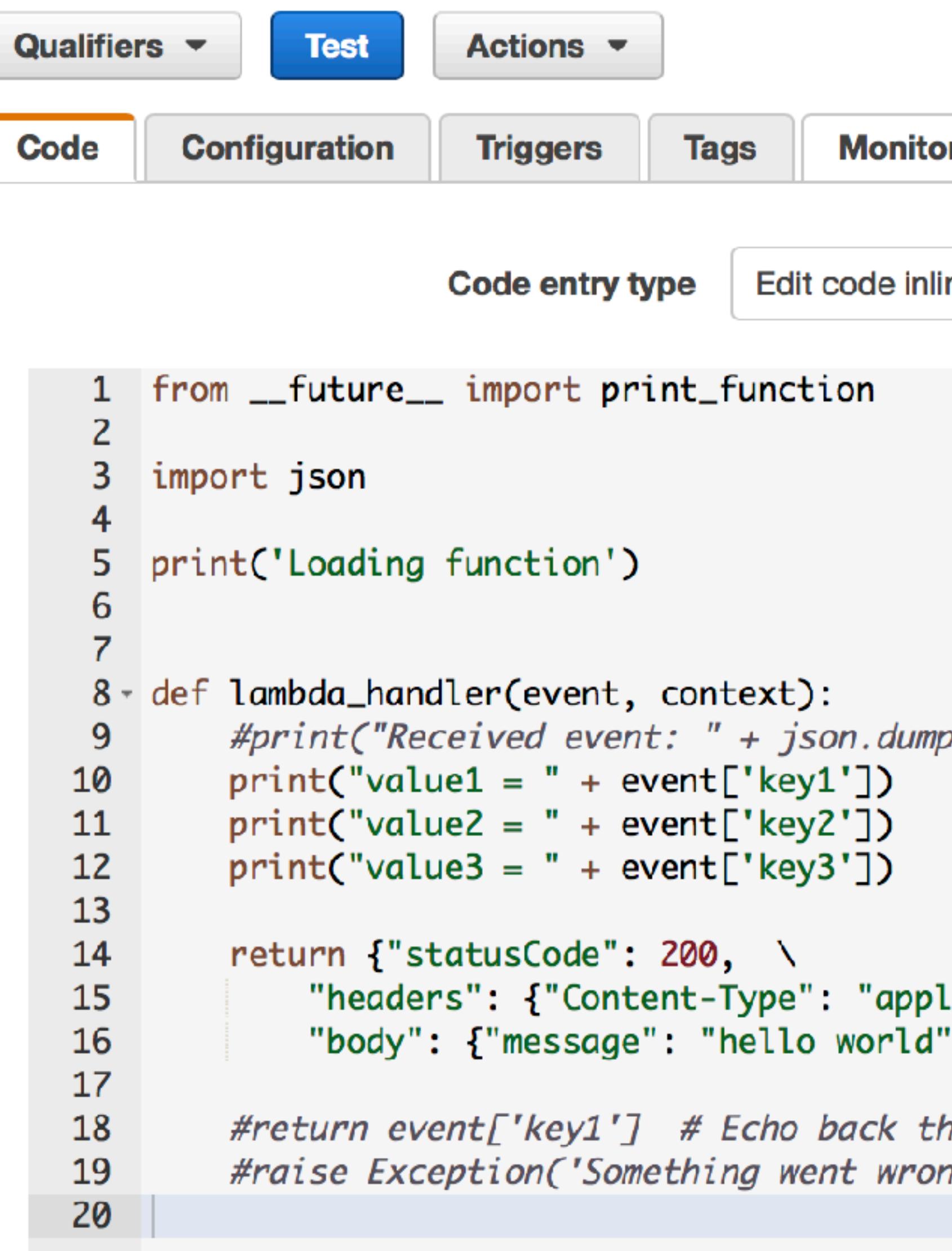
`aws deploy`

`firebase deploy`

# TESTING

# TESTING

- Lambda
  - Great testing environment
  - Test functions and API



The screenshot shows the AWS Lambda console interface for a function named "helloWorld". At the top, there are tabs for "Qualifiers", "Test" (which is highlighted in blue), and "Actions". Below these are tabs for "Code", "Configuration", "Triggers", "Tags", and "Monitor". A sub-header "Code entry type" is visible above the code editor, with an option to "Edit code inline". The main area contains the following Python code:

```

1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 - def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the first key value
19    #raise Exception('Something went wrong')
20

```

# TESTING

- Lambda
  - Debug console and logging provides useful information

Qualifiers ▾ Test Actions ▾

Execution result: succeeded ([logs](#))

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{  
  "statusCode": 400,  
  "headers": {  
    "Content-Type": "application/json"  
  },  
  "body": "{\"error\":\"Email or phone in use\""}  
}
```

**Summary**

**Code SHA-256** xLN61YszDgIYBIRyEfH4dNqvZE3ZKC18XC58P5BPXul=

**Request ID** 13cbc61c-23af-11e7-8598-0552d0ec5bf0

**Duration** 237.84 ms

**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: 13cbc61c-23af-11e7-8598-0552d0ec5bf0 Version: $LATEST  
| END RequestId: 13cbc61c-23af-11e7-8598-0552d0ec5bf0  
| REPORT RequestId: 13cbc61c-23af-11e7-8598-0552d0ec5bf0 Duration: 237.84 ms Billed Duration:
```

# TESTING

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



The screenshot shows the AWS Lambda console interface for a function named 'helloWorld'. At the top, there are tabs for 'Qualifiers', 'Test' (which is highlighted in blue), and 'Actions'. Below these are tabs for 'Code', 'Configuration', 'Triggers', 'Tags', and 'Monitor'. A sub-header 'Code entry type' is visible, along with a button 'Edit code inline'. The main area contains the following Python code:

```

1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 - def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the first key-value pair
19    #raise Exception('Something went wrong')
20

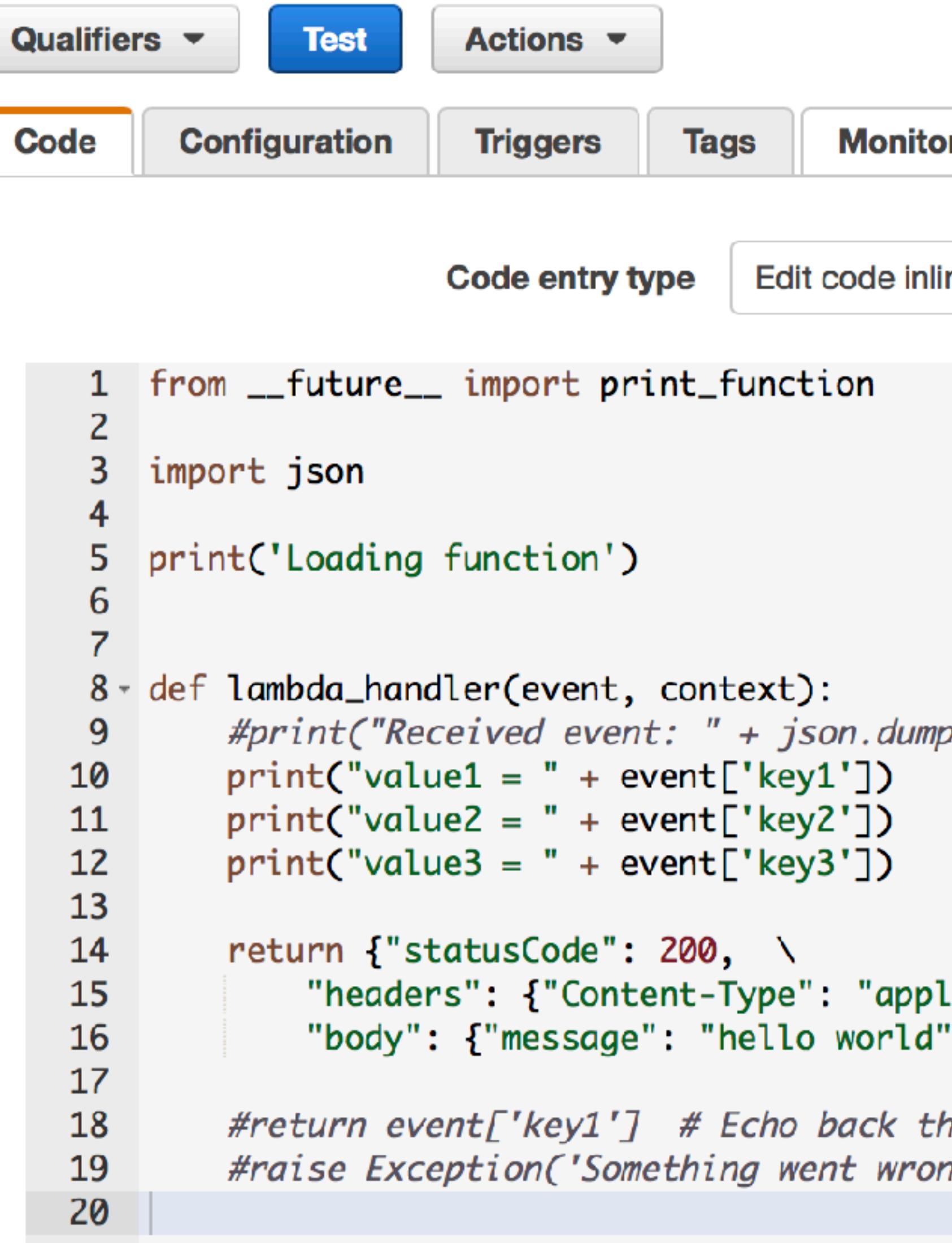
```

# PRICING

# PRICING

- Pricing metrics

- Number of invocations
- Duration of invocation
- Hardware



The screenshot shows the AWS Lambda console interface for a function named 'helloWorld'. At the top, there are tabs for 'Qualifiers', 'Test' (which is highlighted in blue), and 'Actions'. Below these are tabs for 'Code', 'Configuration', 'Triggers', 'Tags', and 'Monitor'. The 'Code' tab is currently active, showing a code entry type of 'Edit code inline'. The code itself is a Python script:

```

1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 - def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the first key value
19    #raise Exception('Something went wrong')
20

```

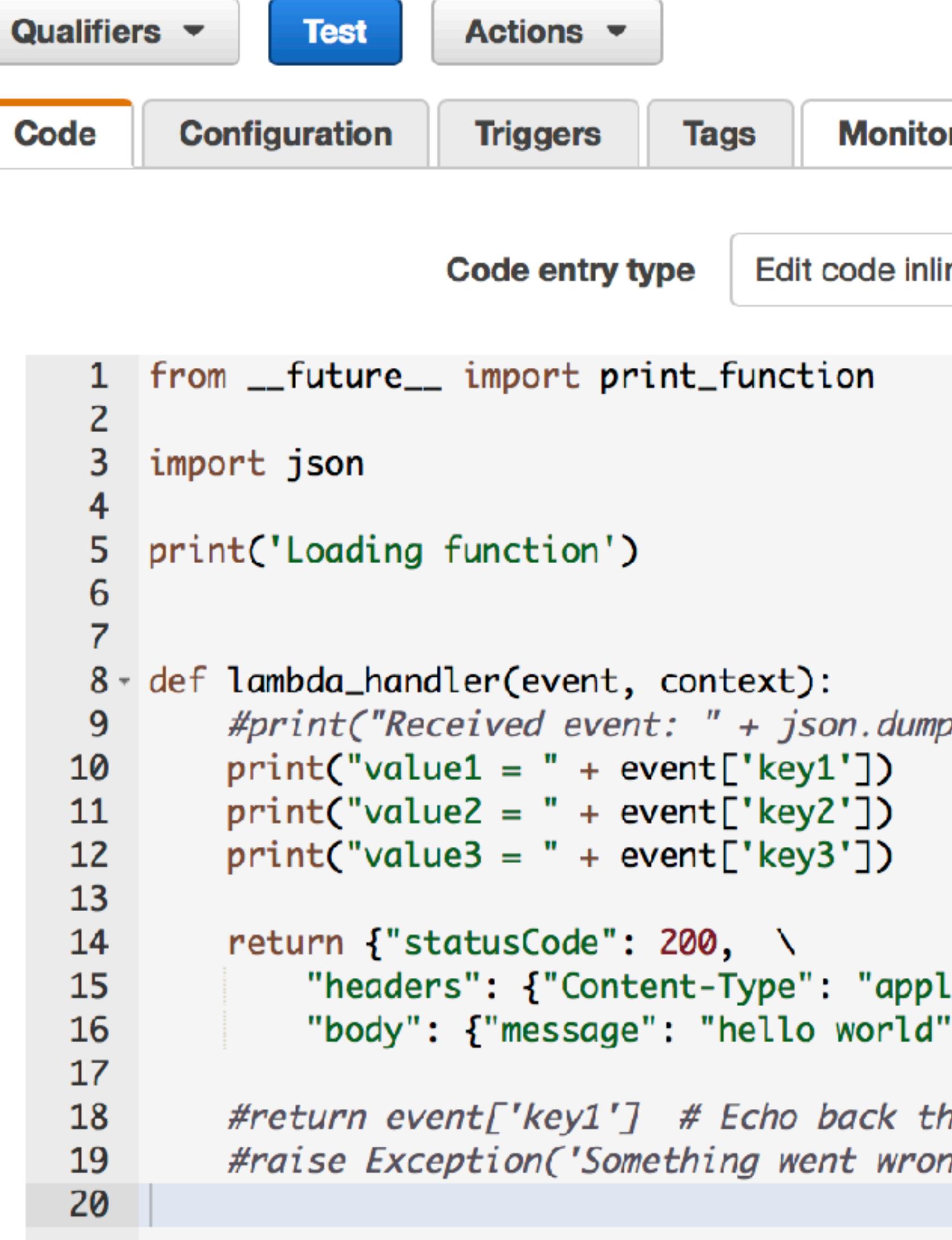
# PRICING

- Cloud Functions

- \$.40 per million (2M free)

- Lambda

- \$.20 per millions (1M free)



The screenshot shows the AWS Lambda console interface for a function named 'helloWorld'. At the top, there are tabs for 'Qualifiers', 'Test' (which is selected), and 'Actions'. Below these are tabs for 'Code', 'Configuration', 'Triggers', 'Tags', and 'Monitor'. The 'Code' tab is active, displaying a Python script. The script starts with importing \_\_future\_\_ and json, then prints a loading message. It defines a lambda\_handler function that prints received event details and returns a success response with headers and body. The code ends with comments about returning the event or raising an exception.

```

1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 - def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the first key-value pair
19    #raise Exception('Something went wrong')
20

```

ODDS AND ENDS

# STATUS AND DOCUMENTATION

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

## CLOUD FUNCTIONS BETA

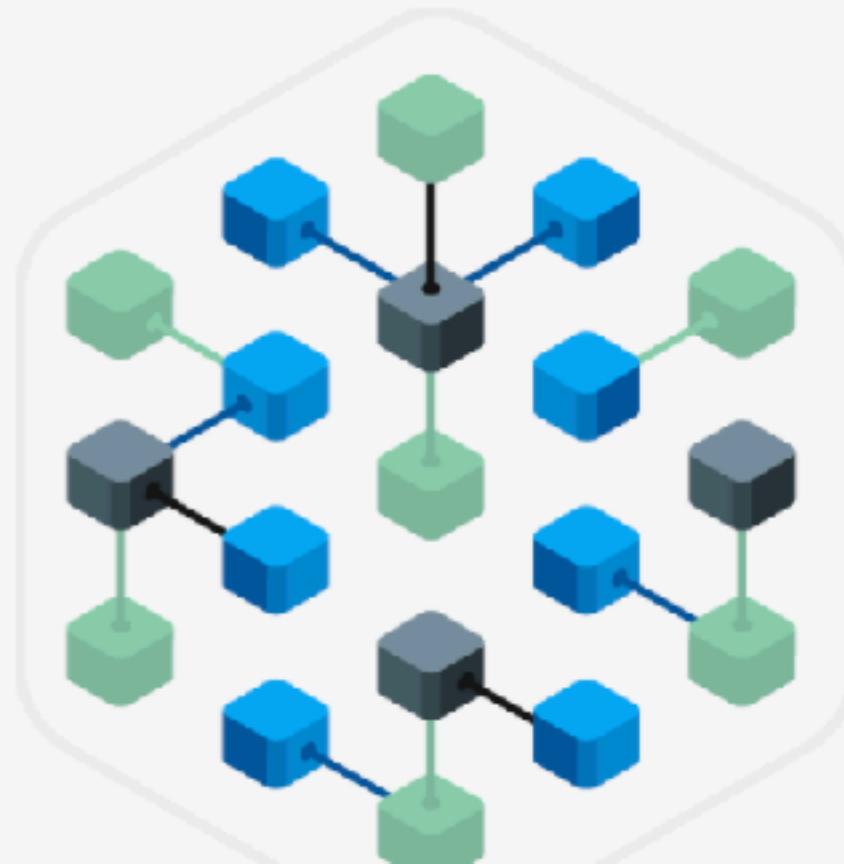
A serverless environment to build and connect cloud services

[...] [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-demand in response to events originating from anywhere. Construct applications from bite-sized business logic billed in the nearest 100 milliseconds, only while your code is running. Serve users from zero to planet-scale, all without managing any infrastructure.



#### Microservices Over Monolith

Developer agility comes from building systems with functionality focused on doing one thing well. Break down monolithic services at the level of a single function, not VMs.

# STATUS AND DOCUMENTATION

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



## AWS Lambda

Run code without thinking about servers.  
Pay for only the compute time you consume.

[Get started with AWS Lambda](#)

# STATUS AND DOCUMENTATION

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

## GOOGLE APP ENGINE

Build scalable web and mobile backends in any language on Google's infrastructure



VIEW DOCUMENTATION

### App Engine for All

Build modern web and mobile applications on an open cloud platform: bring your own language runtimes, frameworks, and third party libraries. Google App Engine is a fully managed platform that completely abstracts away infrastructure so you focus only on code. Go from zero to planet-scale in minutes. See why some of today's most successful companies power their applications on App Engine.

#### For All Language Communities

Out of the box, App Engine supports Node.js, Java, Ruby, C#, Go, Python, and PHP. Developers from these language communities can be productive immediately in a familiar environment without adding code.



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

## Solve the toughest challenges in mobile app dev

- Powerful offline-first features
- Mobile-savvy REST API integrations
- Seamless mobilization of legacy data
- Realtime performance and live collaboration
- Advanced caching and edge compute

[Learn More](#)

IN THE BEGINNING...

# REALM

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

## Realm Mobile Database

Loved by developers and more than a billion users, Realm Mobile Database is fast, easy to use, open source, and totally free.



# REALM

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

## Realm Mobile Database

Loved by developers and more than a billion users, Realm Mobile Database is fast, easy to use, open source, and totally free.



# 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  Linux 

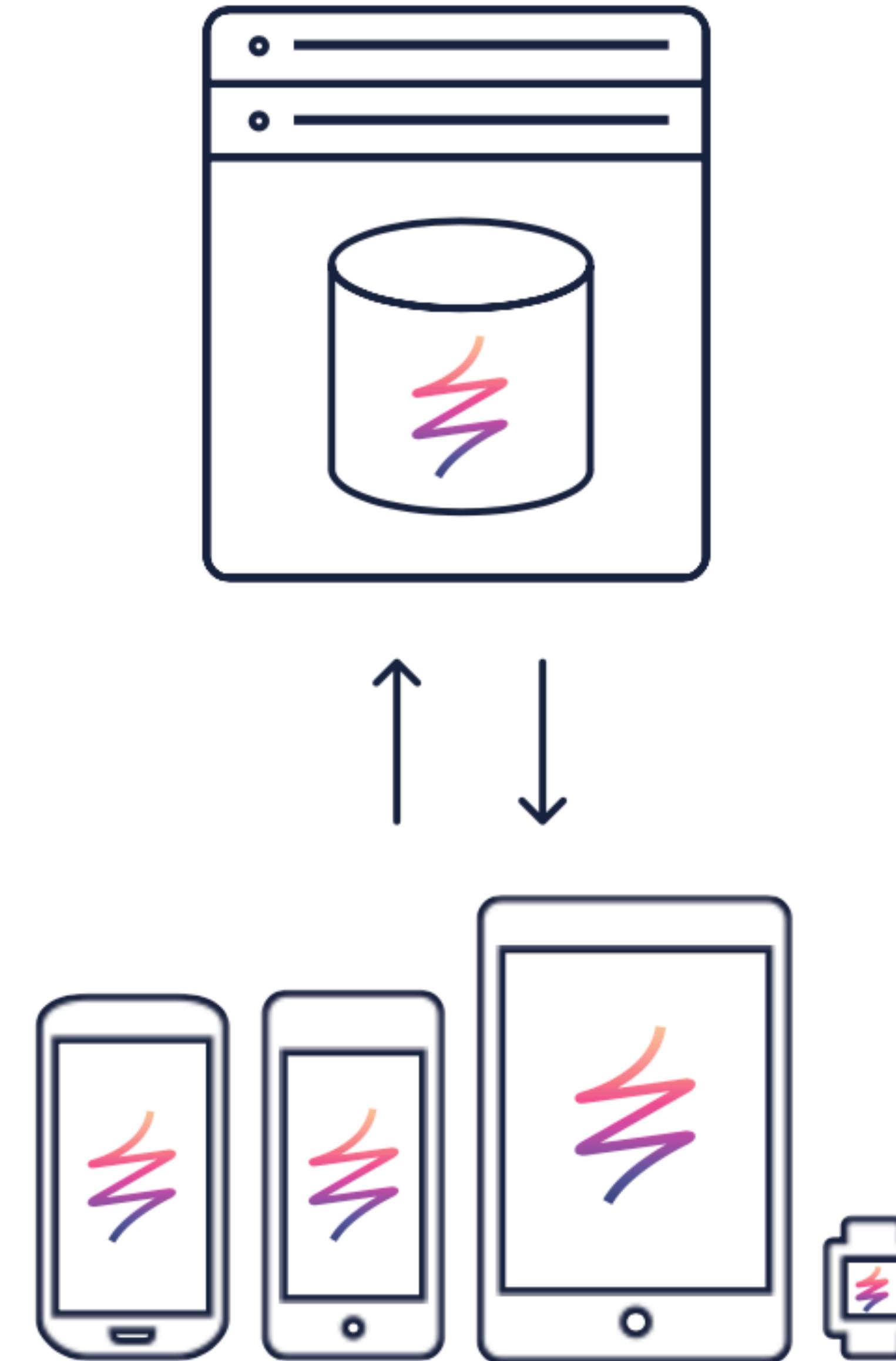
The perfect backend for the next generation of reactive mobile apps

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



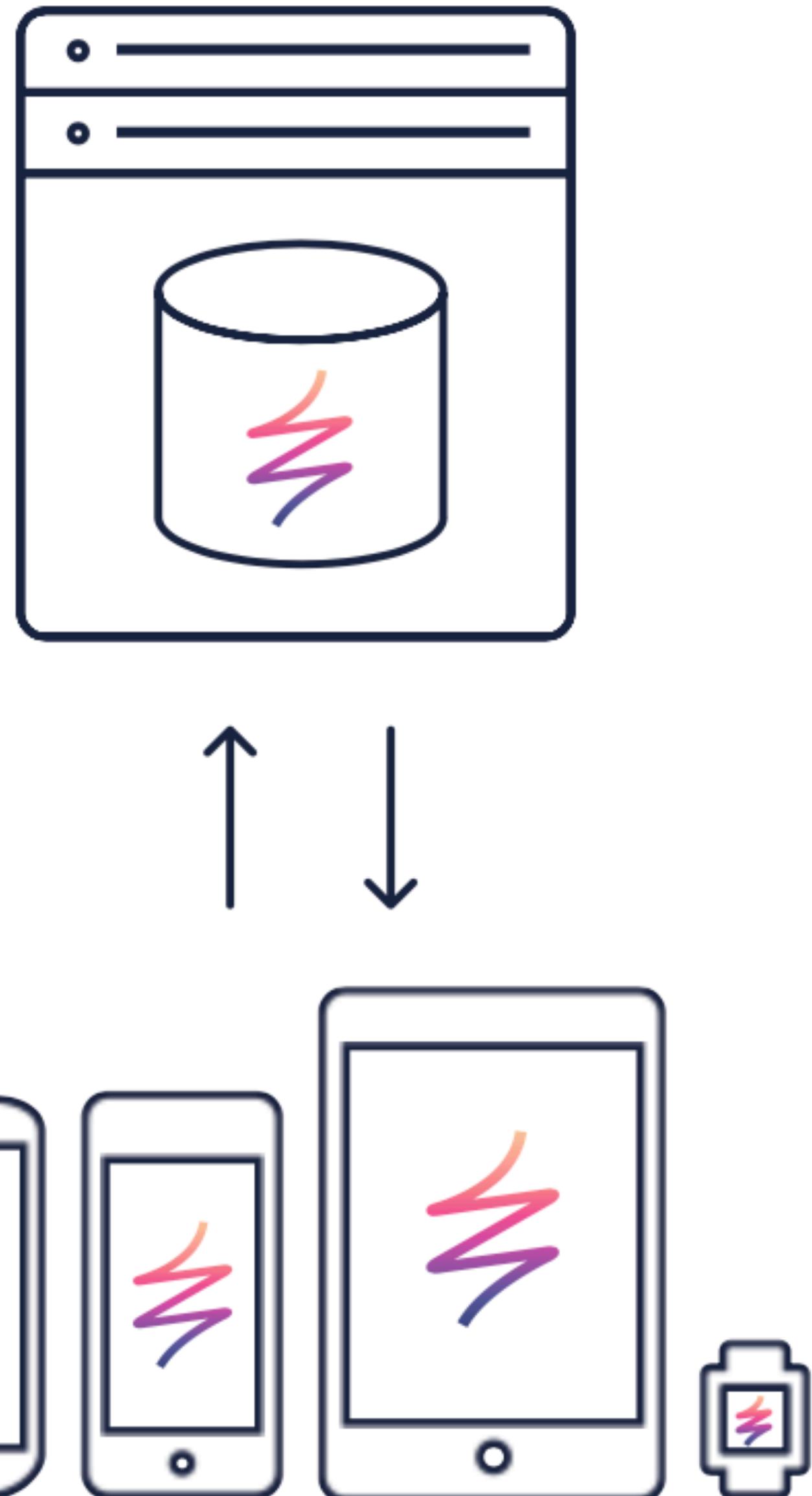
# REALM

- 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)



# Serverless Logic with Realm: Introducing Realm Functions

Realm Team

May 23 2017

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

 Transcript



About the Speaker



Today we're announcing Realm Functions, a new part of Realm that makes building server-side functionality a lot easier for mobile developers. Now, you can make server-side features without listing backend developers, plus you get all the benefits of building on top of the Realm Mobile Platform: you don't need to add another endpoint to a server, and then write the serialization and networking code that would let you connect with it. You just connect your app to Realm, write a Realm Function in your web dashboard, and watch your code execute reactively as data streams in. Today's release is a beta, and it's available today to everyone, whether you're building an app in an enterprise-scale team or for a small side project.

# REALM

The image shows two screenshots demonstrating the use of Realm Functions. On the left, a screenshot of the Realm Object Server interface at `localhost` displays a code editor with a snippet of JavaScript. The code interacts with a `witClient.message` function to receive a task from Wit.ai and then writes it to a realm database. On the right, a screenshot of an iPhone 7 Plus running iOS 10.3 shows a list of tasks under the heading "My Tasks". The tasks listed are:

- Test Realm Functions At 5pm
- Today, 5:00 PM
- Listen To Music
- Clean The Office
- Learn How To Code
- Finish Reading Book
- Pick Up Groceries
- Go To The Gym

The "Clean The Office" task has a cursor icon over it, indicating it is selected.

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

# REALM PLATFORM

# Realm Platform 2.0

**Everything you love about the Realm Platform just got better:**

- Database
- Platform
- Studio
- **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.

## Realm Connect

Connect to Realm Object Server, and Realm Connect converts existing REST APIs and data sources to live objects, freeing you to focus on features rather

## Event handling

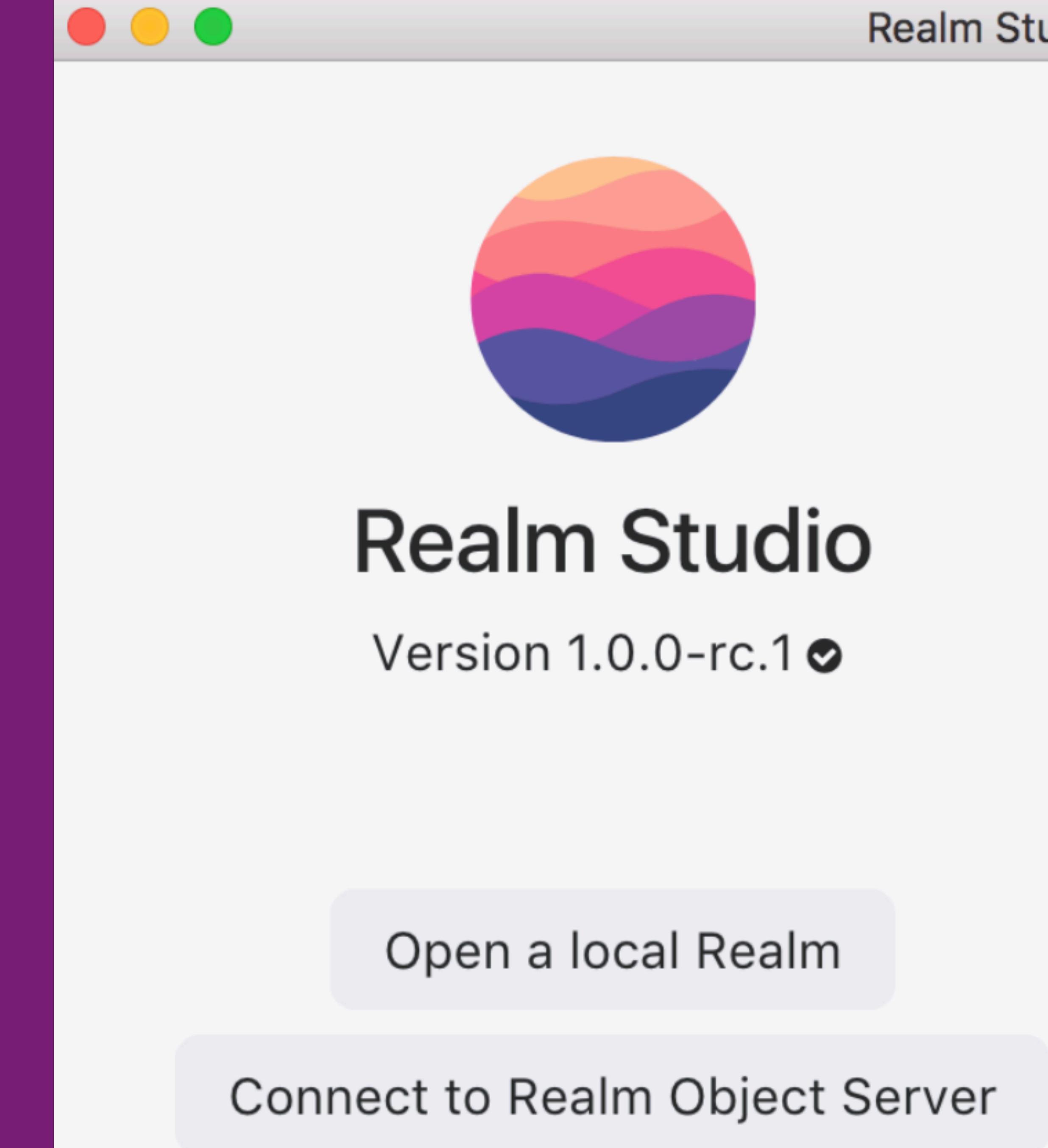
Realm Object Server's event handling functionality means you can easily build server-side features with simple JavaScript. When synced data

## 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

[Realms](#)[Users](#)[Logs](#)

Connected to http://localhost:9080

ID	ROLE	# REALMS
8cebf082313c701a821c0d1b56ecf14	Administrator	0

# REALM PLATFORM

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

Realm Browser

CLASSES	Query..	placeName
Place	2790	string?
	90001	Los Angeles
	90002	Los Angeles
	90003	Los Angeles
	90004	Los Angeles
	90005	Los Angeles
	90006	Los Angeles
	90007	Los Angeles
	90008	Los Angeles
	90009	Los Angeles
	90010	Los Angeles
	90011	Los Angeles
	90012	Los Angeles
	90013	Los Angeles
	90014	Los Angeles
	90015	Los Angeles
	90016	Los Angeles
	90017	Los Angeles
	90018	Los Angeles
	90019	Los Angeles

# 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 advantage 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 capabilities
class Santa: NSObject {

    // The dynamic declaration modifier is required whenever you want
    // 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 Swift
let observation = santa.observe(\.string) { (santa, change) in
    print("We observed a change in a property. The new values is: \(change.newValue)")
}

//: # Make an observable change #
// We are chaing this...watch what happens
print("Getting ready 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 introspection
class Santa: NSObject {

    // The dynamic declaration modifier is required
    // 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 #
// ".string" is shorthand for Santa (in this case)
let observation = santa.observe(\.string) {
    print("We observed a change in a property")
}

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

# REALM PLATFORM

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

```
subclass of NSObject #  
ivate objective-c's introspection capability  
ct {  
  
claration modifier is required whenever you use  
jective-C's dynamism  
g: String  
  
L Santa #  
  
santa.string)")  
  
erver #  
I for Santa (in this case); new Key Path in Santa  
anta.observe(\.string) { (santa, change) in  
I a change in a property. The new values is  
  
variable change #  
Is...watch what happens  
to change....")  
ta"  
anta
```

# REALM PLATFORM

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

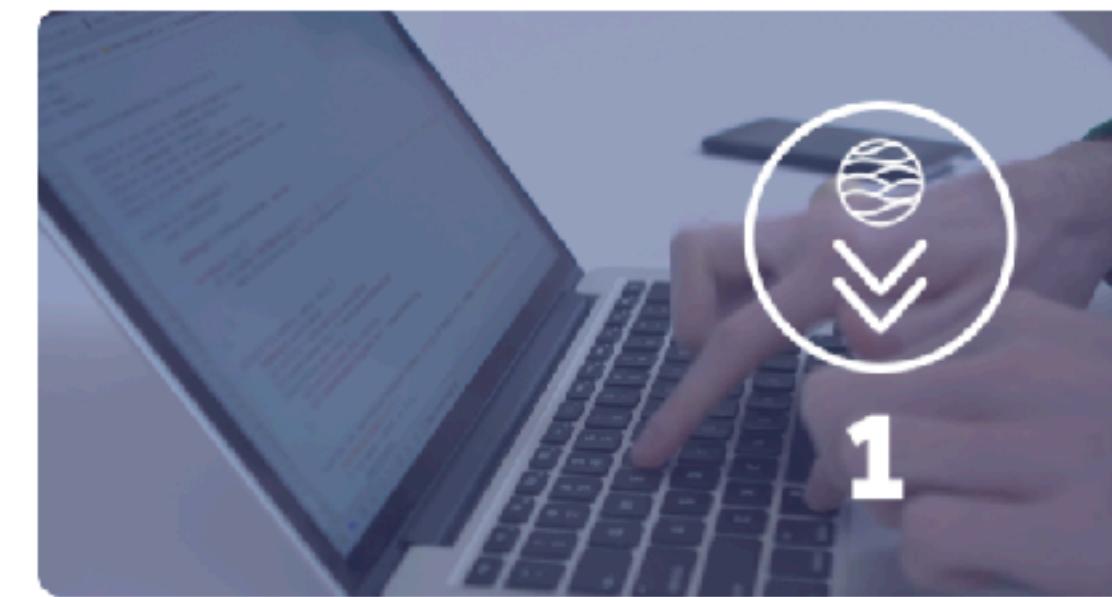


## Build a Realtime Swift App with Realm

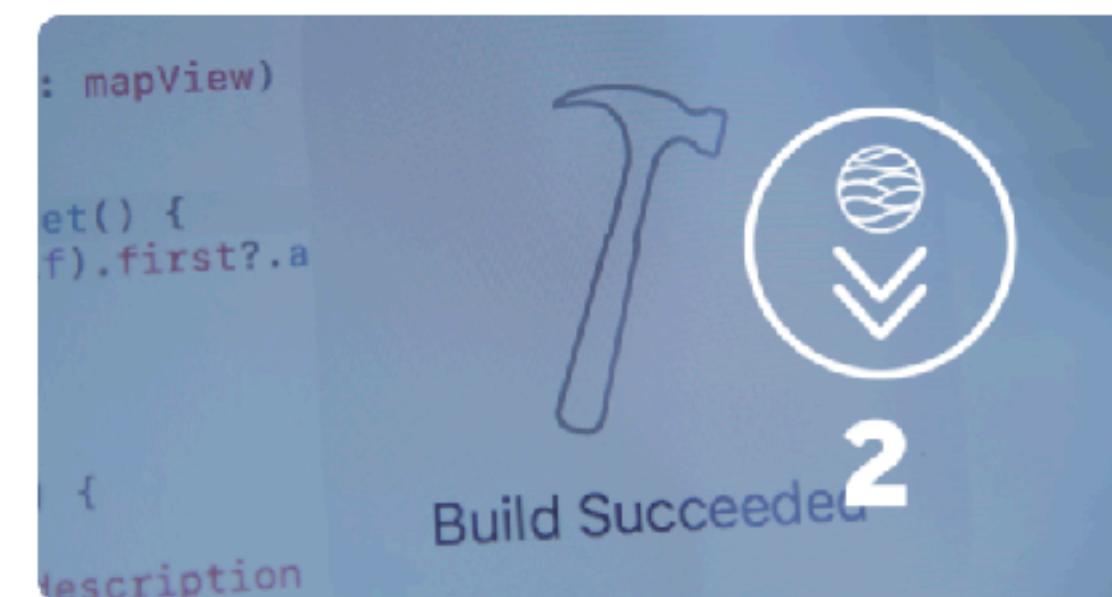
Get hands-on with the Realm Platform, build a realtime Santa-tracking app, and learn ne...

# REALM PLATFORM

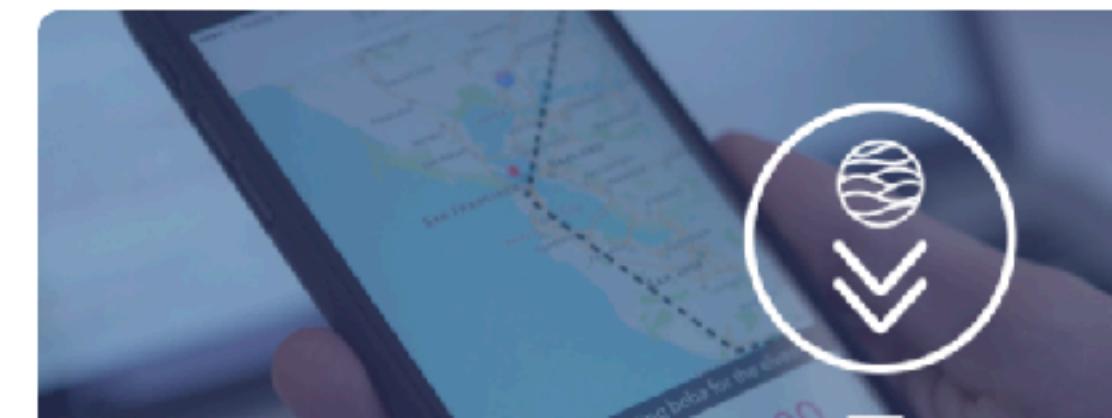
- [https://academy.realm.io/  
posts/learning-path-build-a-  
realtime-swift-app-with-realm/](https://academy.realm.io/posts/learning-path-build-a-realtime-swift-app-with-realm/)



global tour!



data changes.



## Track Santa with Realm: Part 1

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

## Track Santa with Realm: Part 2

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

## 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

The screenshot shows the Realm Browser interface with a list of objects from a 'Place' class. The columns are labeled 'CLASSES', 'Place', '2790', 'postalCode', 'placeName', and a search bar labeled 'Query...'. The 'Place' column lists 2790 entries. The 'postalCode' column contains values like 90001 through 90019. The 'placeName' column contains 'Los Angeles' repeated 19 times. A search bar at the top right is empty.

CLASSES	Place	2790	postalCode	placeName	Query...
			string?	string?	
			90001	Los Angeles	
			90002	Los Angeles	
			90003	Los Angeles	
			90004	Los Angeles	
			90005	Los Angeles	
			90006	Los Angeles	
			90007	Los Angeles	
			90008	Los Angeles	
			90009	Los Angeles	
			90010	Los Angeles	
			90011	Los Angeles	
			90012	Los Angeles	
			90013	Los Angeles	
			90014	Los Angeles	
			90015	Los Angeles	
			90016	Los Angeles	
			90017	Los Angeles	
			90018	Los Angeles	
			90019	Los Angeles	

## 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](#).

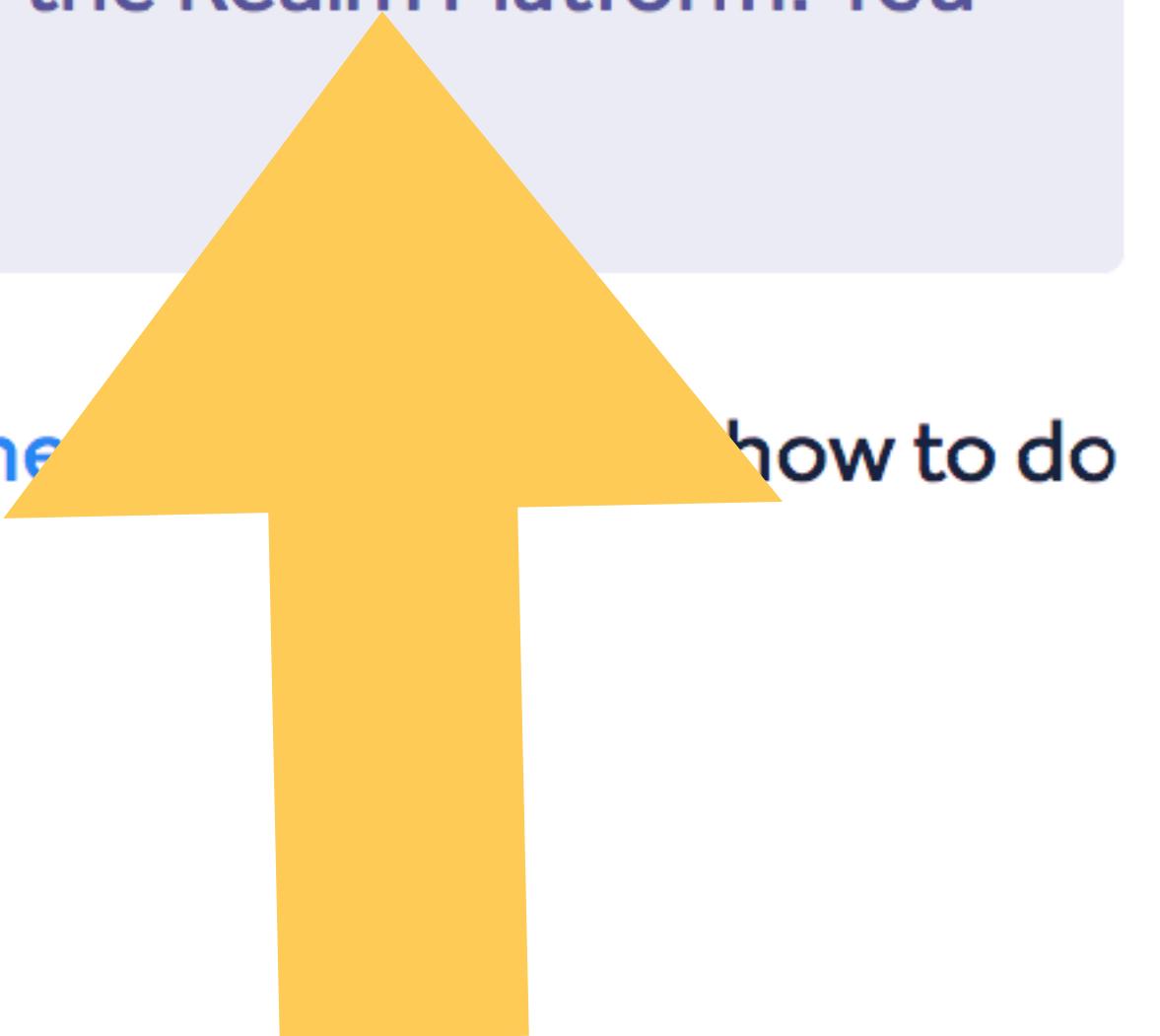
So I'm going to gloss right over setting up your Realm Object Server. If you're interested in how to do that. For this tutorial, I'm assuming you have it set up and running.

We'll open our `index.js` file:

```
'use strict';
```

Syncing My First Data With Realm Object Server

Since there is no data yet in the server, let's use `realm-js` to create a sample data set. To complete this step, you will need to sign up for our Professional Edition trial [here](#).



# TO REALM

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

The screenshot shows the Realm Browser interface with a list of objects from a 'Place' class. The columns are labeled 'Place', '2790', 'postalCode', and 'placeName'. The 'placeName' column is currently sorted by ascending postal code.

Place	2790	postalCode	placeName
		90001	Los Angeles
		90002	Los Angeles
		90003	Los Angeles
		90004	Los Angeles
		90005	Los Angeles
		90006	Los Angeles
		90007	Los Angeles
		90008	Los Angeles
		90009	Los Angeles
		90010	Los Angeles
		90011	Los Angeles
		90012	Los Angeles
		90013	Los Angeles
		90014	Los Angeles
		90015	Los Angeles
		90016	Los Angeles
		90017	Los Angeles
		90018	Los Angeles
		90019	Los Angeles

# 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 trial](#)

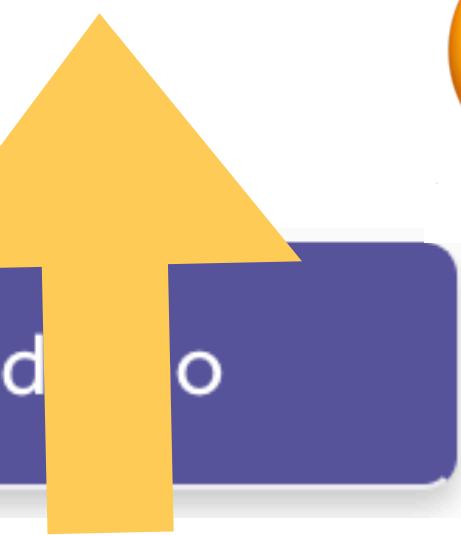
Enterprise Edition

## Free Demo

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

Pricing scales with use case.

[Contact us for demo](#)



# TO REALM

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

## Realm Platform

Developer Edition	Professional Edition
<b>Free</b>	<b>Free Trial</b>
Enable realtime two-way data sync for mobile devices and servers.	Powerful features like simplified integrations, event handling, server side data access.
No cost to use, even in production for commercial use cases. <a href="#">Read the terms.</a>	14-day free trial; pricing starts at \$1,750 per month.
<a href="#">Get the download</a>	<a href="#">Download free trial</a>
<b>Object Database</b> ✓	<b>Object Database</b> ✓
<b>On-premises or Public Cloud</b> ✓	<b>On-premises or Public Cloud</b> ✓
<b>Cross Platform</b> Android, iOS, Xamarin, and JavaScript	<b>Cross Platform</b> Android, iOS, Xamarin, and JavaScript
<b>Data Encryption</b> AES-256 at rest; SSL/TLS in flight	<b>Data Encryption</b> AES-256 at rest; SSL/TLS in flight
<b>Offline-first Functionality</b> ✓	<b>Offline-first Functionality</b> ✓