

#WWDC19

# Creating Swift Packages

Boris Buegling, Developer Tools Engineer  
Ankit Aggarwal, Developer Tools Engineer

Creating local packages

Publishing packages

Package manifest API

Editing packages

Open source project

Creating local packages

Publishing packages

Package manifest API

Editing packages

Open source project

# Share Code

Within your workspace

With your team

With the open source community

# Creating Local Packages

# Local Swift Packages

Can be thought of as sub-projects in a Xcode workspace

# Local Swift Packages

Can be thought of as sub-projects in a Xcode workspace

Platform independent

# Local Swift Packages

Can be thought of as sub-projects in a Xcode workspace

Platform independent

Great for refactoring out reusable code



# Local Swift Packages

Can be thought of as sub-projects in a Xcode workspace

Platform independent

Great for refactoring out reusable code

Not versioned

# Local Swift Packages

Can be thought of as sub-projects in a Xcode workspace

Platform independent

Great for refactoring out reusable code

Not versioned

Can publish and share with others later

***Demo***

Creating local packages

# Publishing Packages

# Semantic Versioning

[semver.org](https://semver.org)

1.2.4

Major Version  
Breaking changes

1.2.4

Minor Version  
Compatible additions

1.2.4

Patch Version  
Bugfixes



0.x.y

Initial Development

**2.0.0-alpha.1**

Prerelease Version

BestProject

- BestProject
  - AppDelegate.swift
  - SceneDelegate.swift
  - ViewController.swift
  - Main.storyboard
  - Assets.xcassets
  - LaunchScreen.storyboard
  - Info.plist
- Products
- Swift Package Dependencies
  - BestPackage 5.0.0-beta.6

BestProject

Info Build Settings Swift Packages

PROJECT

- BestProject

TARGETS

- BestProject

▼ Packages (1 item)

Name	Version Rules	Location
BestPackage	5.0.0-beta.1 – Next Major	<a href="https://github.com/WWDC19/BestPackage.git">https://github.com/WWDC19/BestPackage.git</a>

+ -

+ - Filter

▼ Packages (1 item)

Name	Version Rules	Location
BestPackage	5.0.0-beta.1 – Next Major	<a href="https://github.com/WWDC19/BestPackage.git">https://github.com/WWDC19/BestPackage.git</a>

+ -

BestProject

- BestProject
  - AppDelegate.swift
  - SceneDelegate.swift
  - ViewController.swift
  - Main.storyboard
  - Assets.xcassets
  - LaunchScreen.storyboard
  - Info.plist
- Products
- Swift Package Dependencies
  - BestPackage 5.0.0-beta.6

BestProject

Info Build Settings Swift Packages

PROJECT

- BestProject

TARGETS

- BestProject

▼ Packages (1 item)

Name	Version Rules	Location
BestPackage	5.0.0-beta.1 – Next Major	<a href="https://github.com/WWDC19/BestPackage.git">https://github.com/WWDC19/BestPackage.git</a>

+ -

- BestProject
  - BestProject
    - AppDelegate.swift
    - SceneDelegate.swift
    - ViewController.swift
    - Main.storyboard
    - Assets.xcassets
    - LaunchScreen.storyboard
    - Info.plist

- PROJECT**
- BestProject
- TARGETS**
- BestProject

**Packages (1 item)**

Name	Version Rules	Location
BestPackage	5.0.0-beta.1 – Next Major	<a href="https://github.com/WWDC19/BestPackage.git">https://github.com/WWDC19/BestPackage.git</a>

**Swift Package Dependencies**

- ▶ BestPackage 5.0.0-beta.6

***Demo***

Publishing packages

# Package Manifest API

Ankit Aggarwal, Developer Tools Engineer



# Package Manifest File

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MyPackage"
)
```

 MyPackage

 Package.swift

# Package Manifest File

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MyPackage"
)
```

 MyPackage

 Package.swift

# Package Manifest File

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MyPackage"
)
```

 MyPackage

 Package.swift

# Package Manifest File

```
// swift-tools-version:5.1
import PackageDescription
```

```
let package = Package(
    name: "MyPackage"
)
```

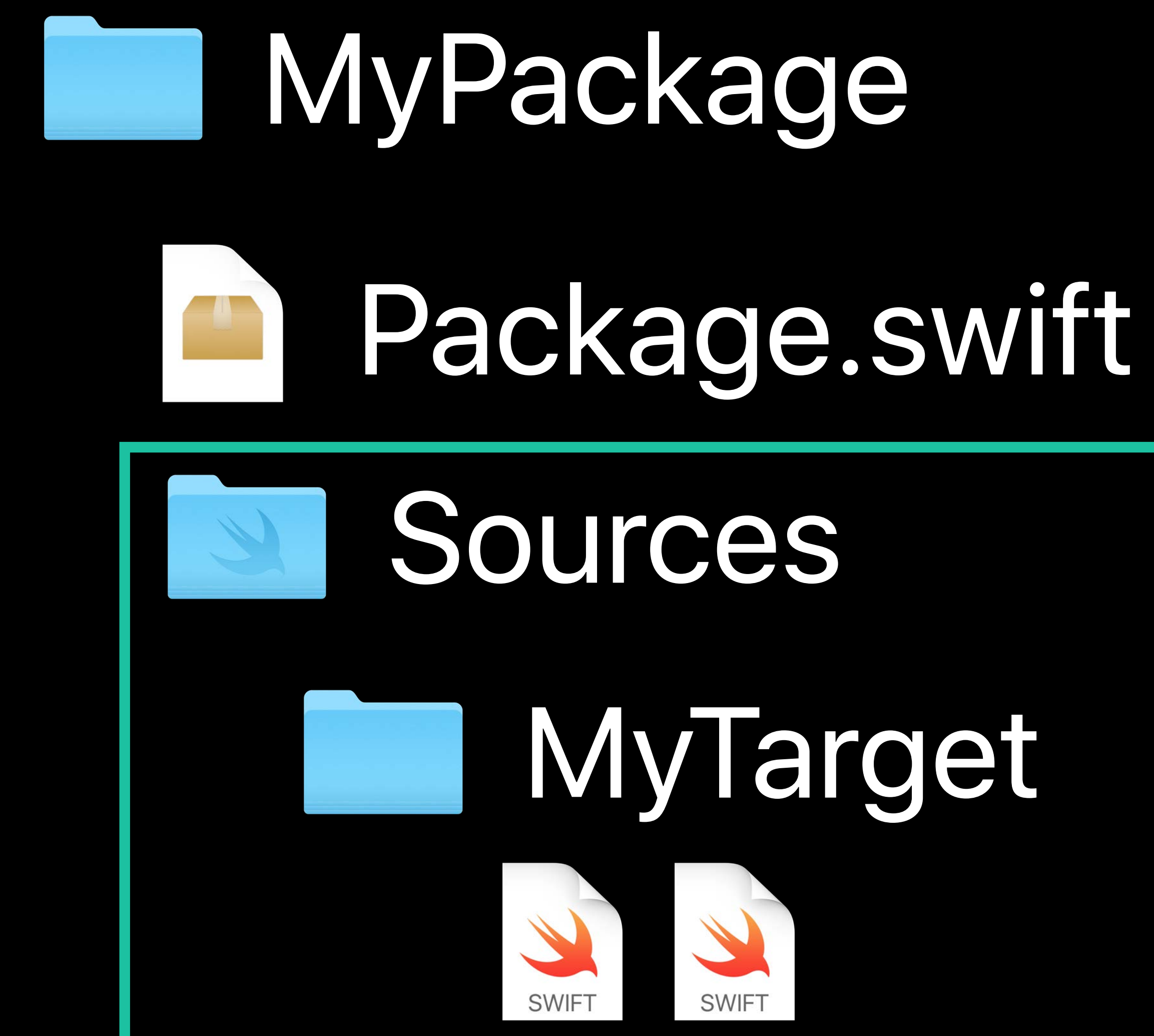
 MyPackage

 Package.swift

# Configuring Targets

```
// swift-tools-version:5.1
import PackageDescription

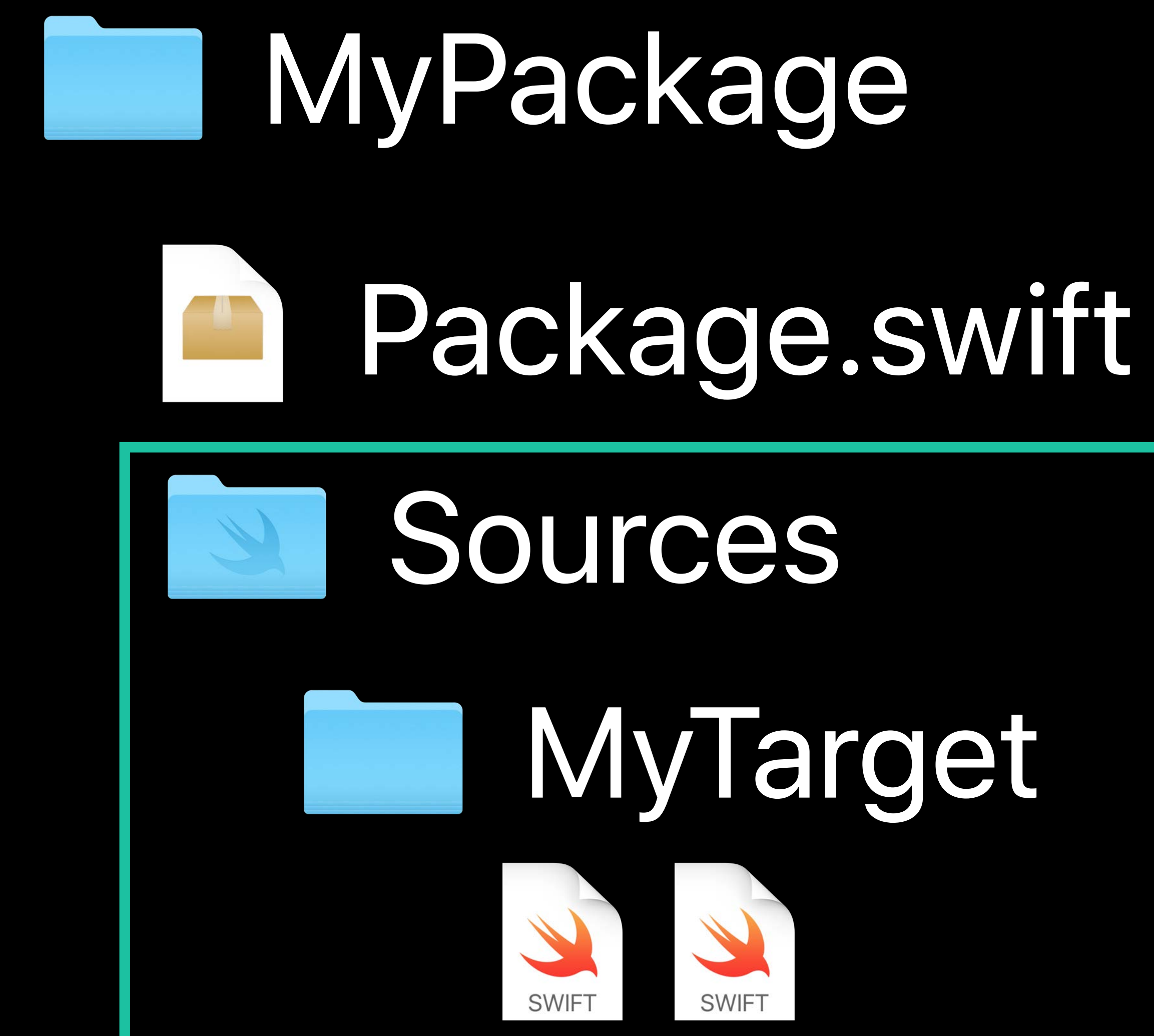
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
    ]
)
```



# Configuring Targets

```
// swift-tools-version:5.1
import PackageDescription

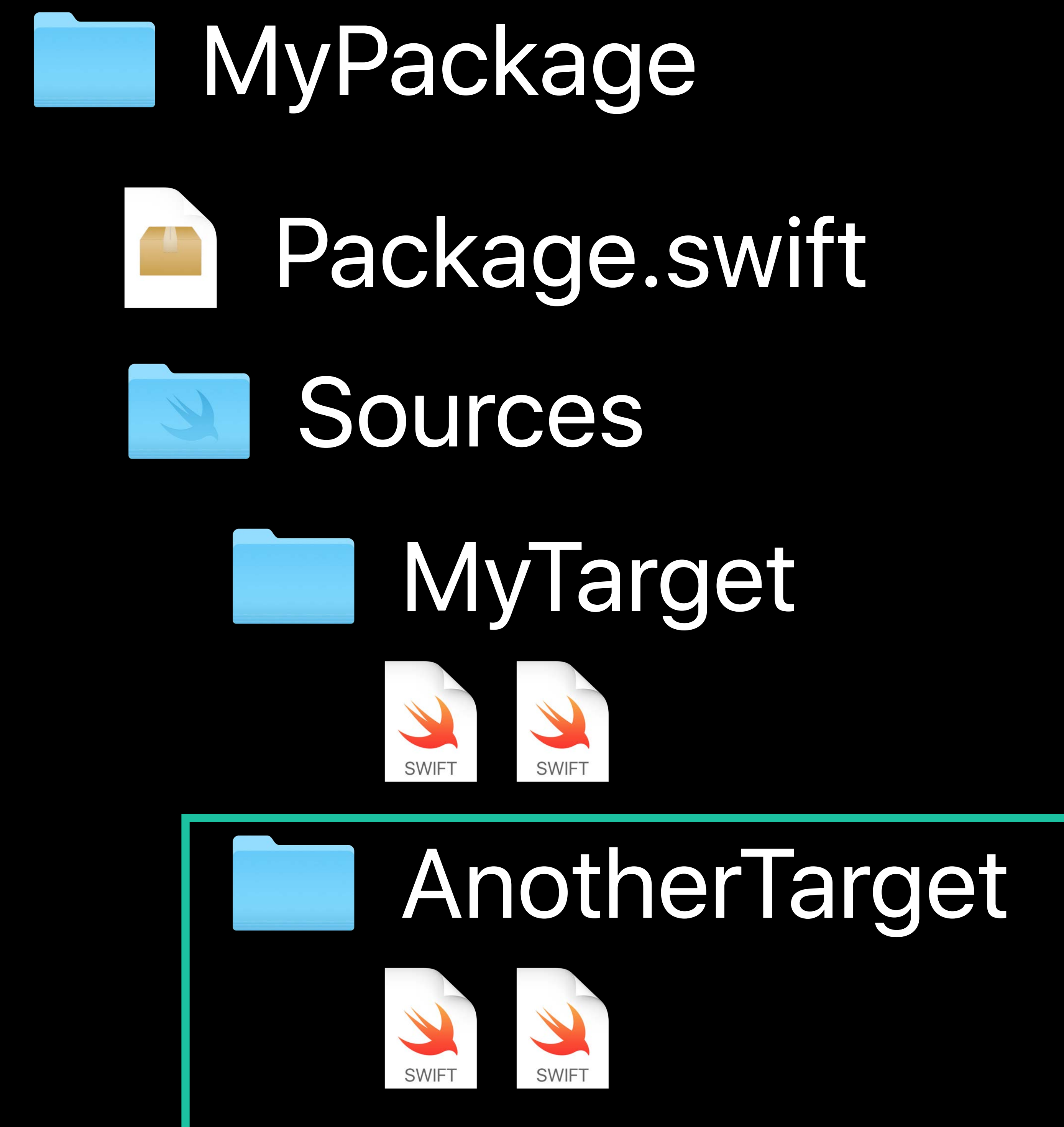
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
    ]
)
```



# Configuring Targets

```
// swift-tools-version:5.1
import PackageDescription

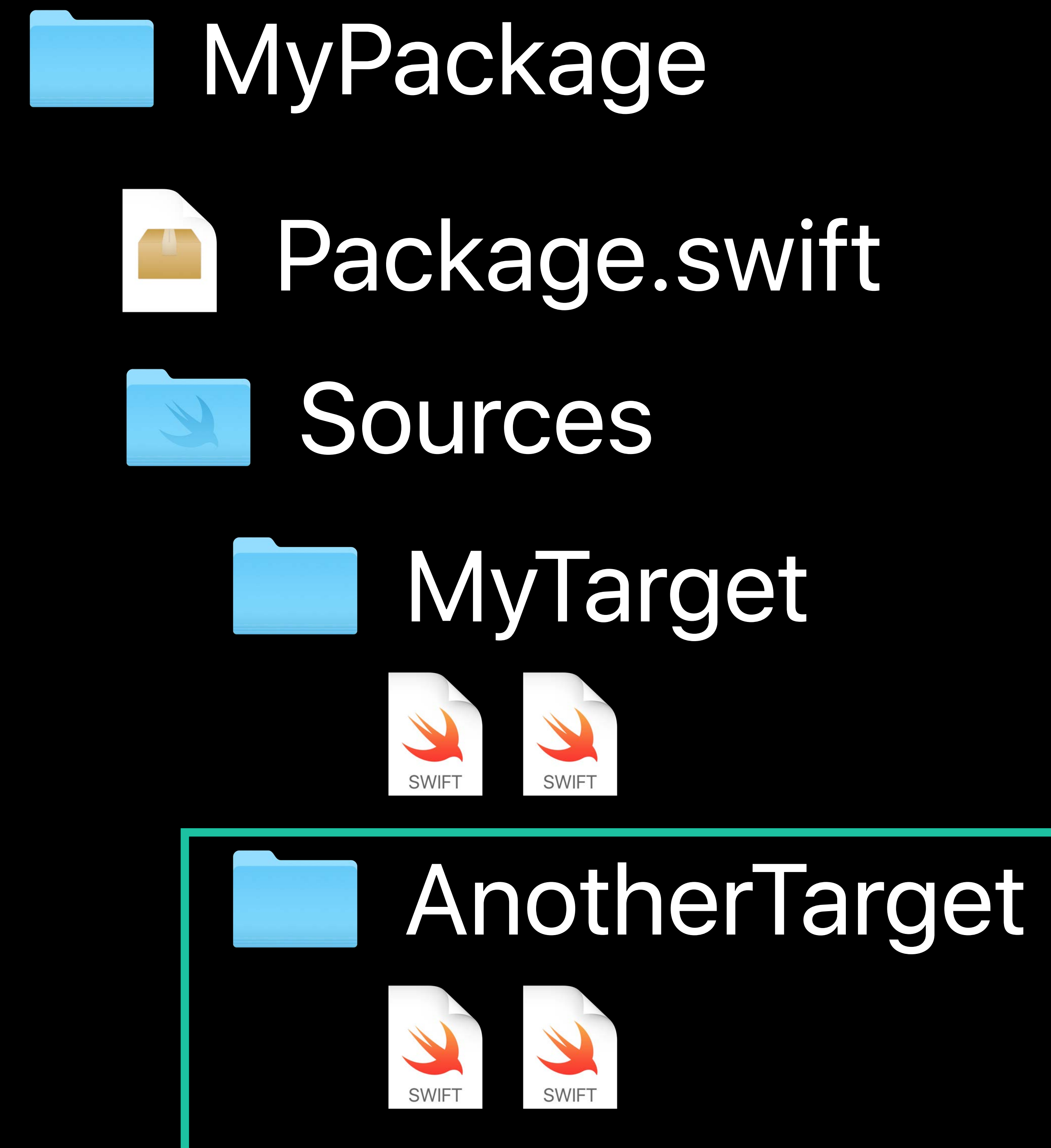
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
        .target(name: "AnotherTarget"),
    ]
)
```



# Configuring Targets

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
        .target(name: "AnotherTarget"),
    ]
)
```

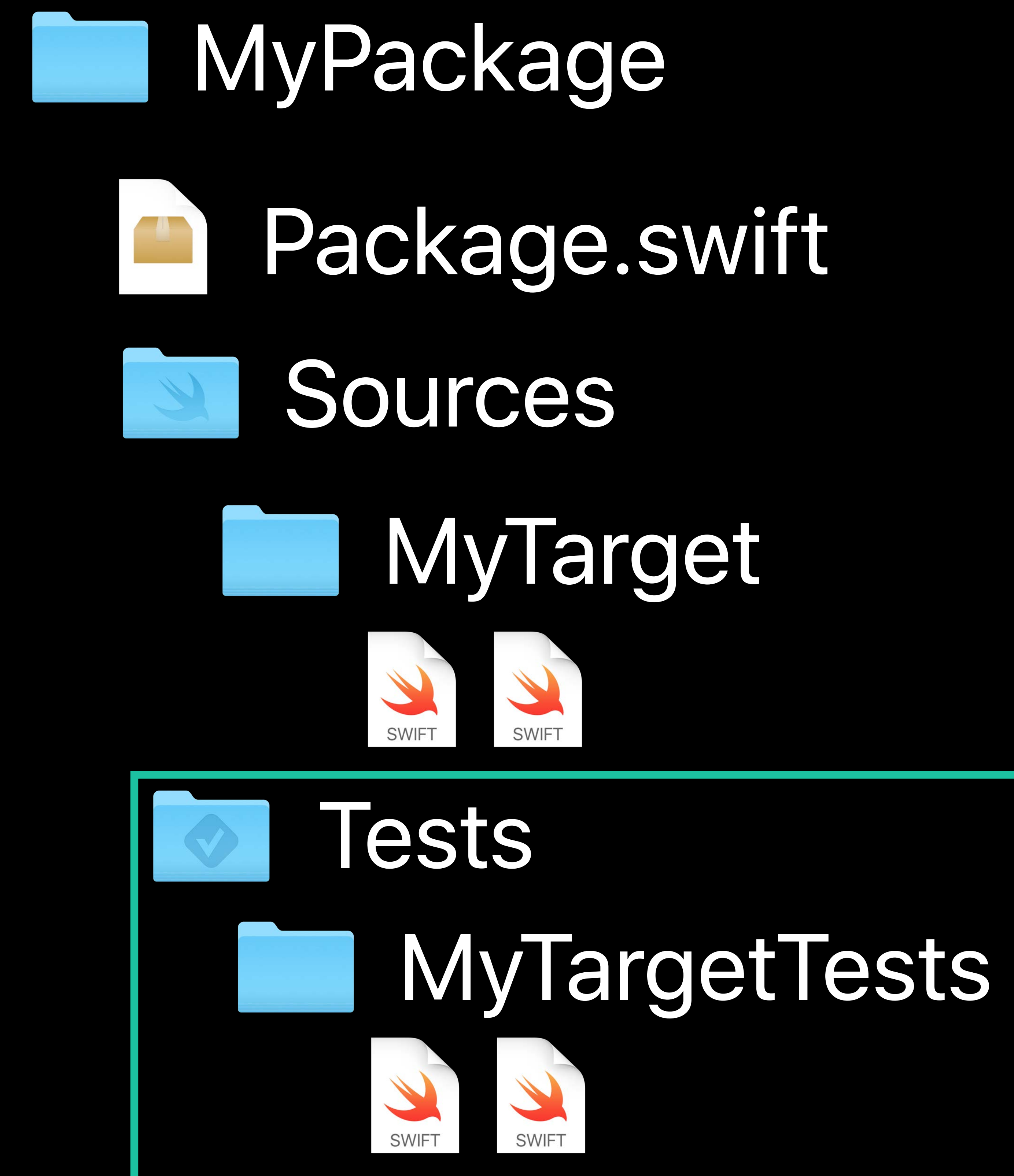




# Configuring Test Targets

```
// swift-tools-version:5.1
import PackageDescription

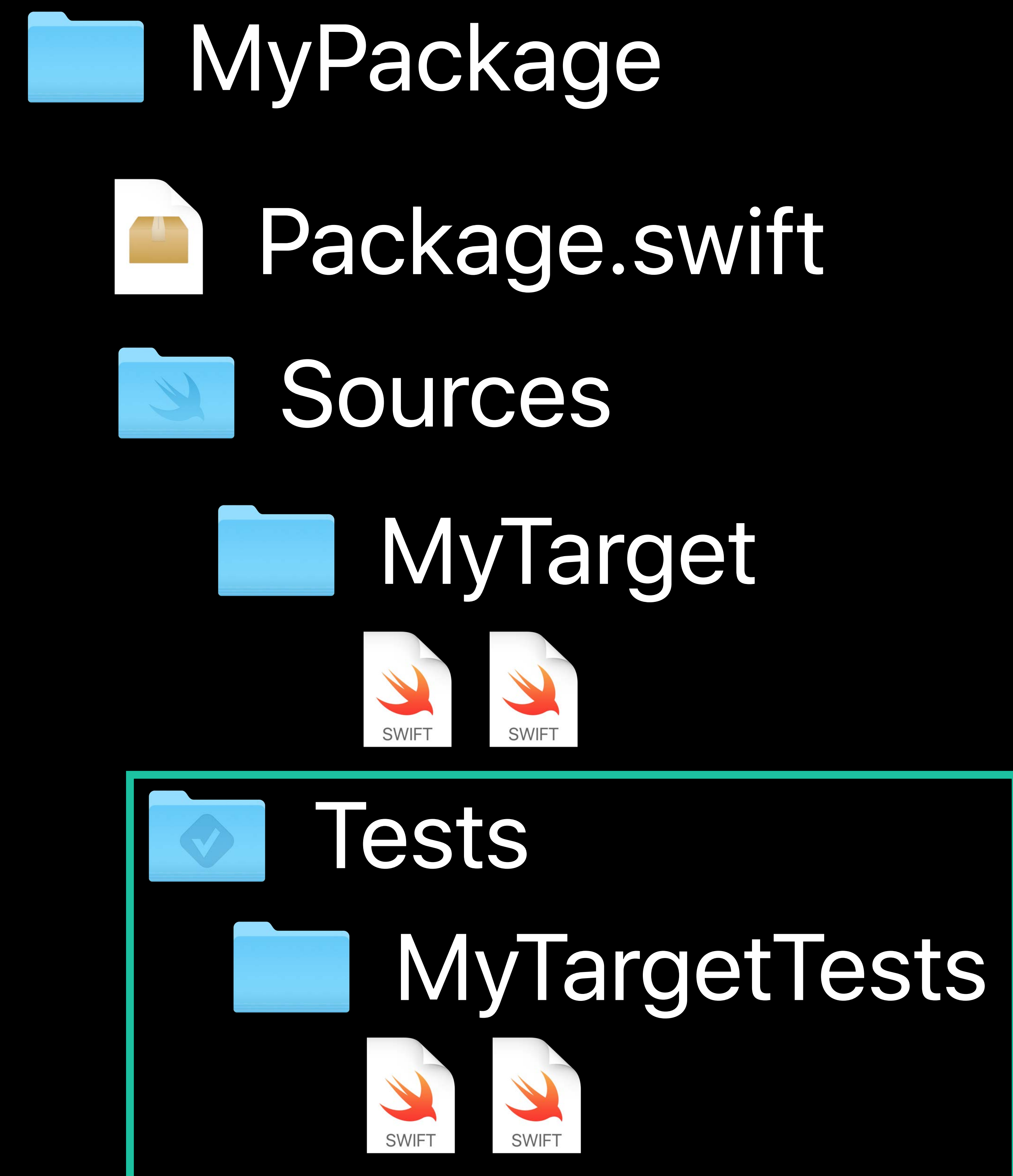
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
        .testTarget(name: "MyTargetTests"),
    ]
)
```



# Configuring Test Targets

```
// swift-tools-version:5.1
import PackageDescription

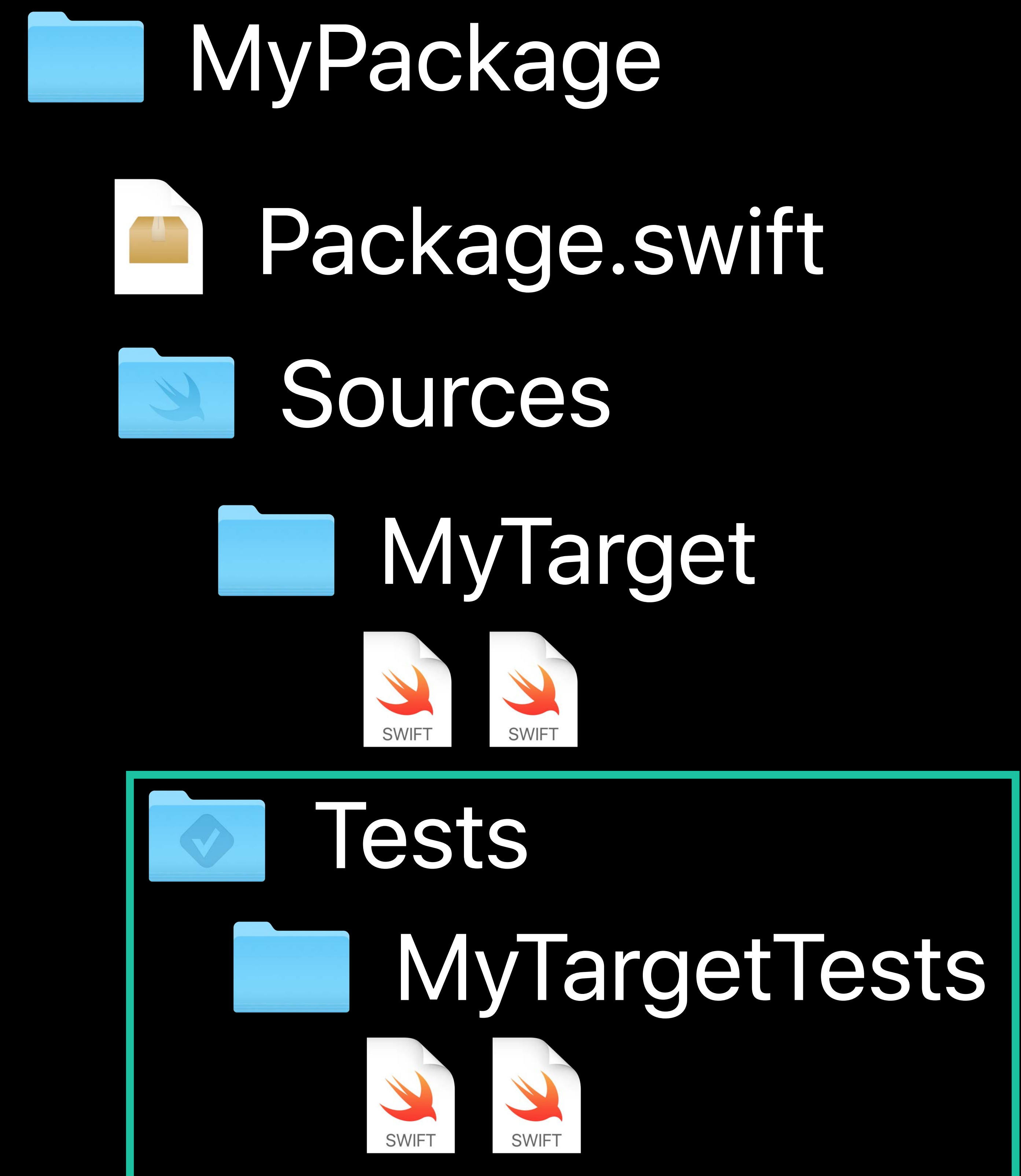
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
        .testTarget(name: "MyTargetTests"),
    ]
)
```



# Configuring Test Targets

```
// swift-tools-version:5.1
import PackageDescription

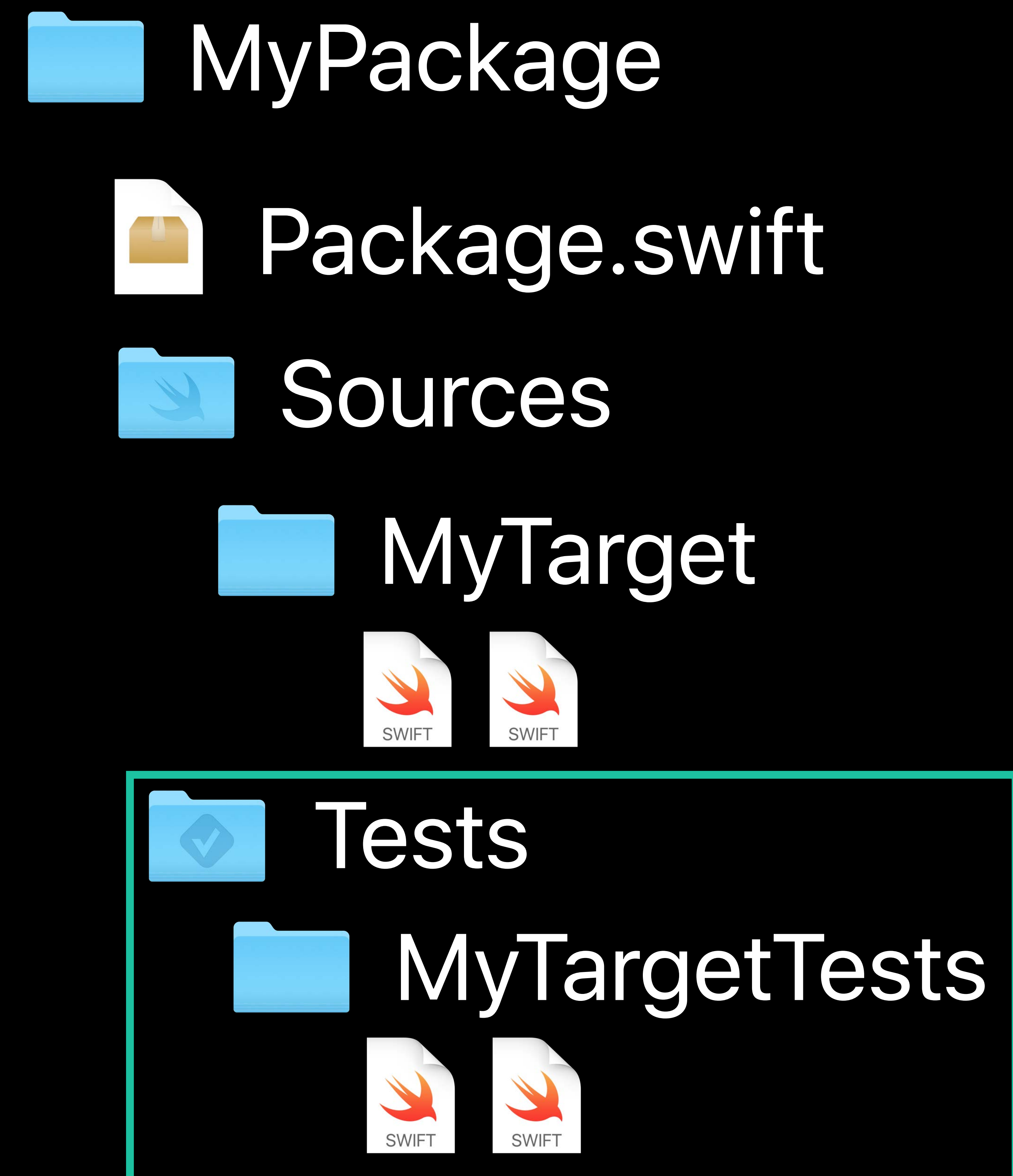
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
        .testTarget(
            name: "MyTargetTests",
            dependencies: ["MyTarget"]
        ),
    ]
)
```



# Configuring Test Targets

```
// swift-tools-version:5.1
import PackageDescription

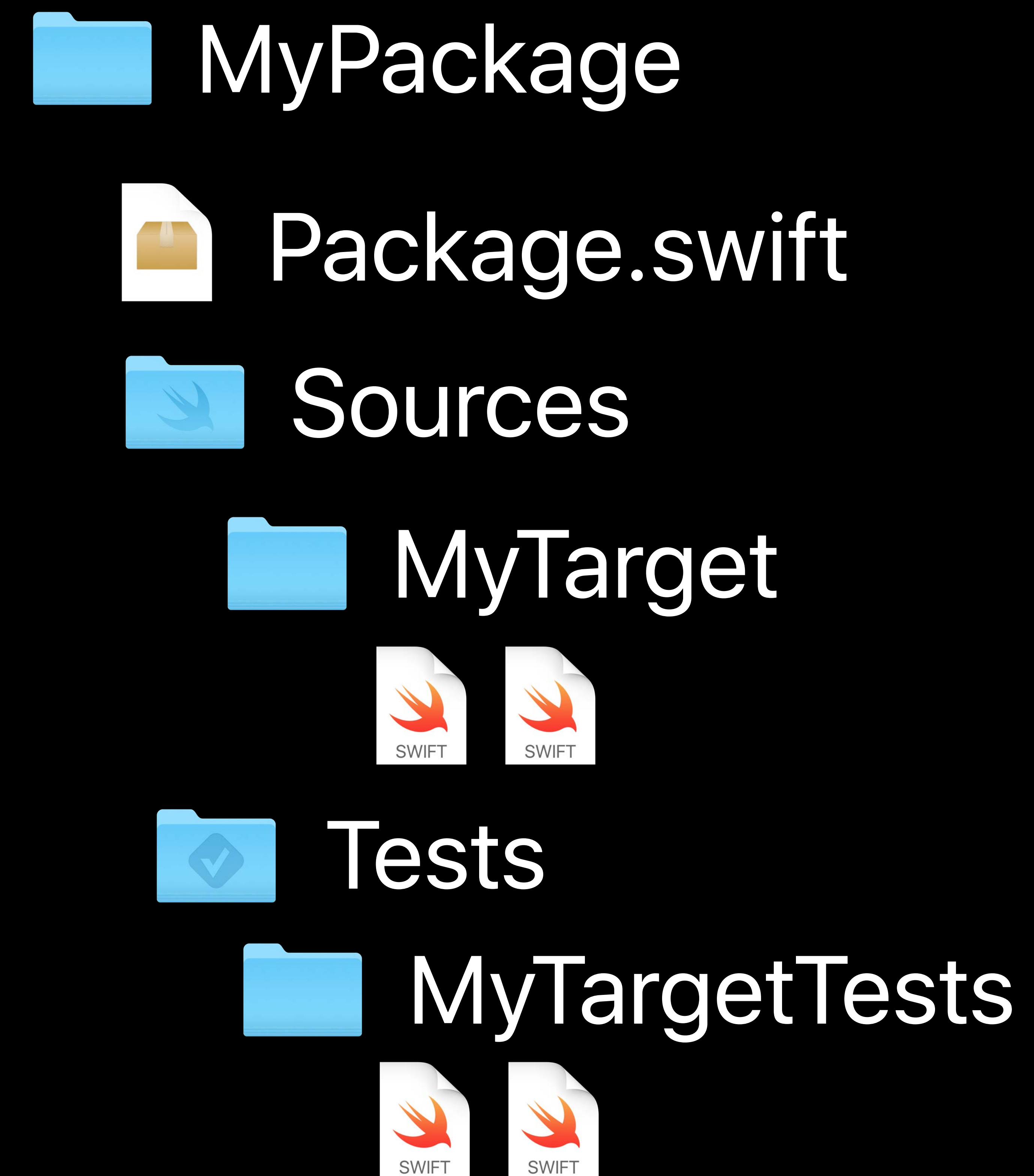
let package = Package(
    name: "MyPackage",
    targets: [
        .target(name: "MyTarget"),
        .testTarget(
            name: "MyTargetTests",
            dependencies: ["MyTarget"]
        ),
    ],
)
```



# Configuring Products

```
// swift-tools-version:5.1
import PackageDescription

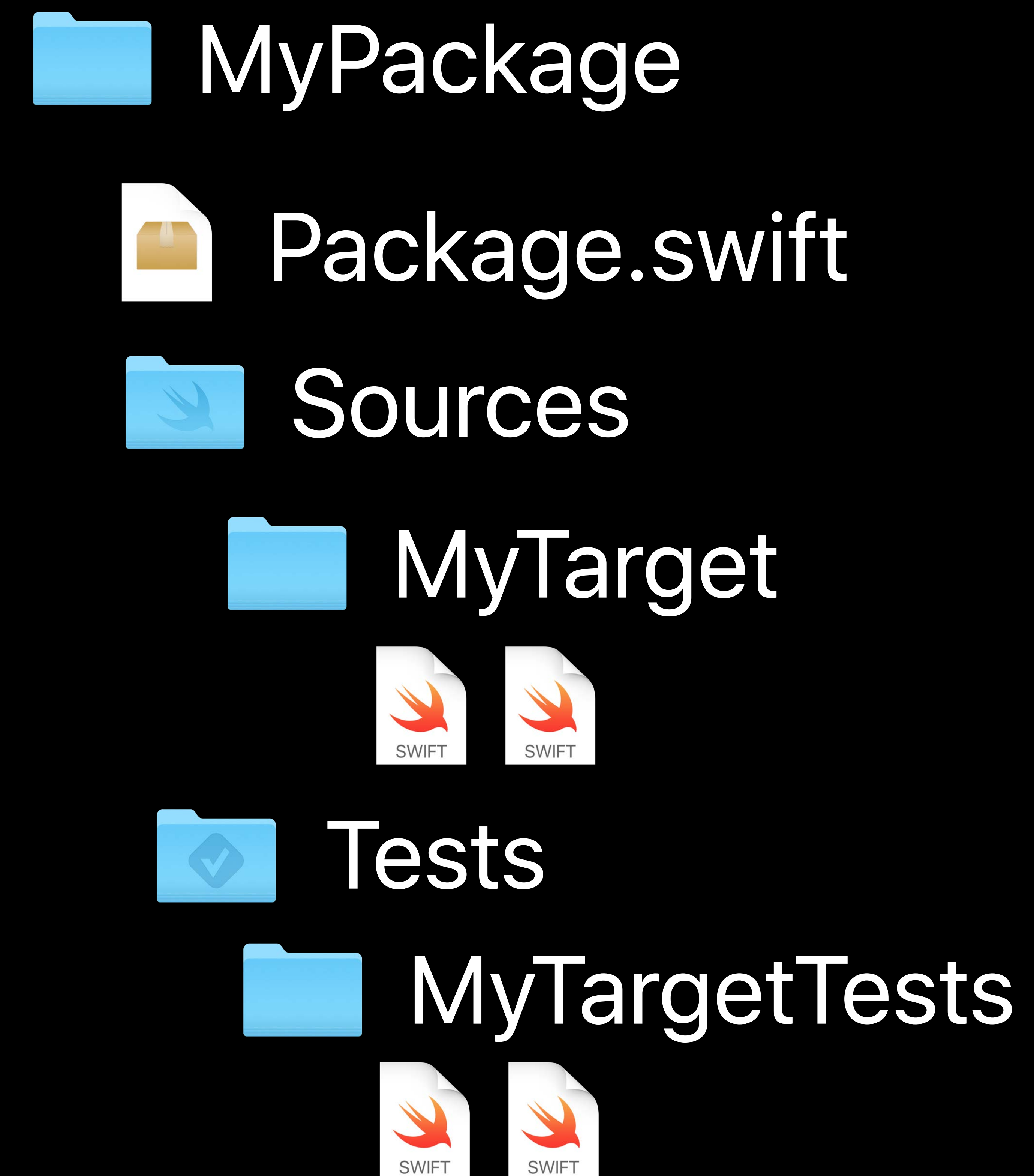
let package = Package(
    name: "MyPackage",
    products: [
        .library(
            name: "MyProduct",
            targets: ["MyTarget"]
        ),
    ],
    targets: [
        .target(name: "MyTarget"),
        ...
    ]
)
```



# Configuring Products

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MyPackage",
    products: [
        .library(
            name: "MyProduct",
            targets: ["MyTarget"]
        ),
    ],
    targets: [
        .target(name: "MyTarget"),
        ...
    ]
)
```



**Adding support in existing projects**

# Example — MenuDownloader

 MenuDownloader

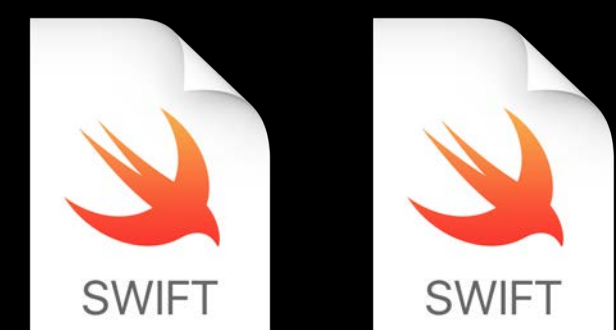
 MenuDownloader.xcodeproj

 MenuDownloader.podspec

 LegacyCode



 SwiftyMenuDownloader





# Example — MenuDownloader

 MenuDownloader

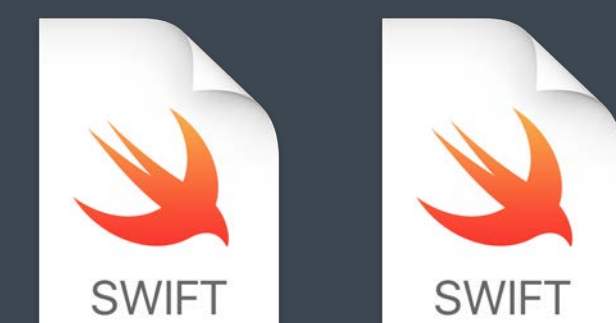
 MenuDownloader.xcodeproj

 MenuDownloader.podspec

 LegacyCode



 SwiftyMenuDownloader



# Example — MenuDownloader

 MenuDownloader

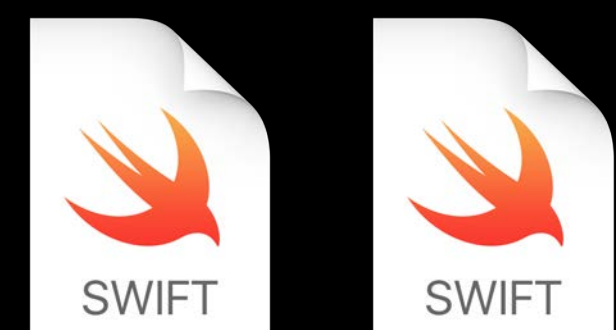
 MenuDownloader.xcodeproj

 MenuDownloader.podspec

 LegacyCode



 SwiftyMenuDownloader

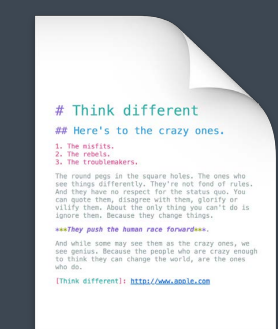


# Example — MenuDownloader

 MenuDownloader



MenuDownloader.xcodeproj



MenuDownloader.podspec

 LegacyCode



 SwiftyMenuDownloader



# Example — MenuDownloader

 MenuDownloader

 Package.swift

 MenuDownloader.xcodeproj

 MenuDownloader.podspec

 LegacyCode

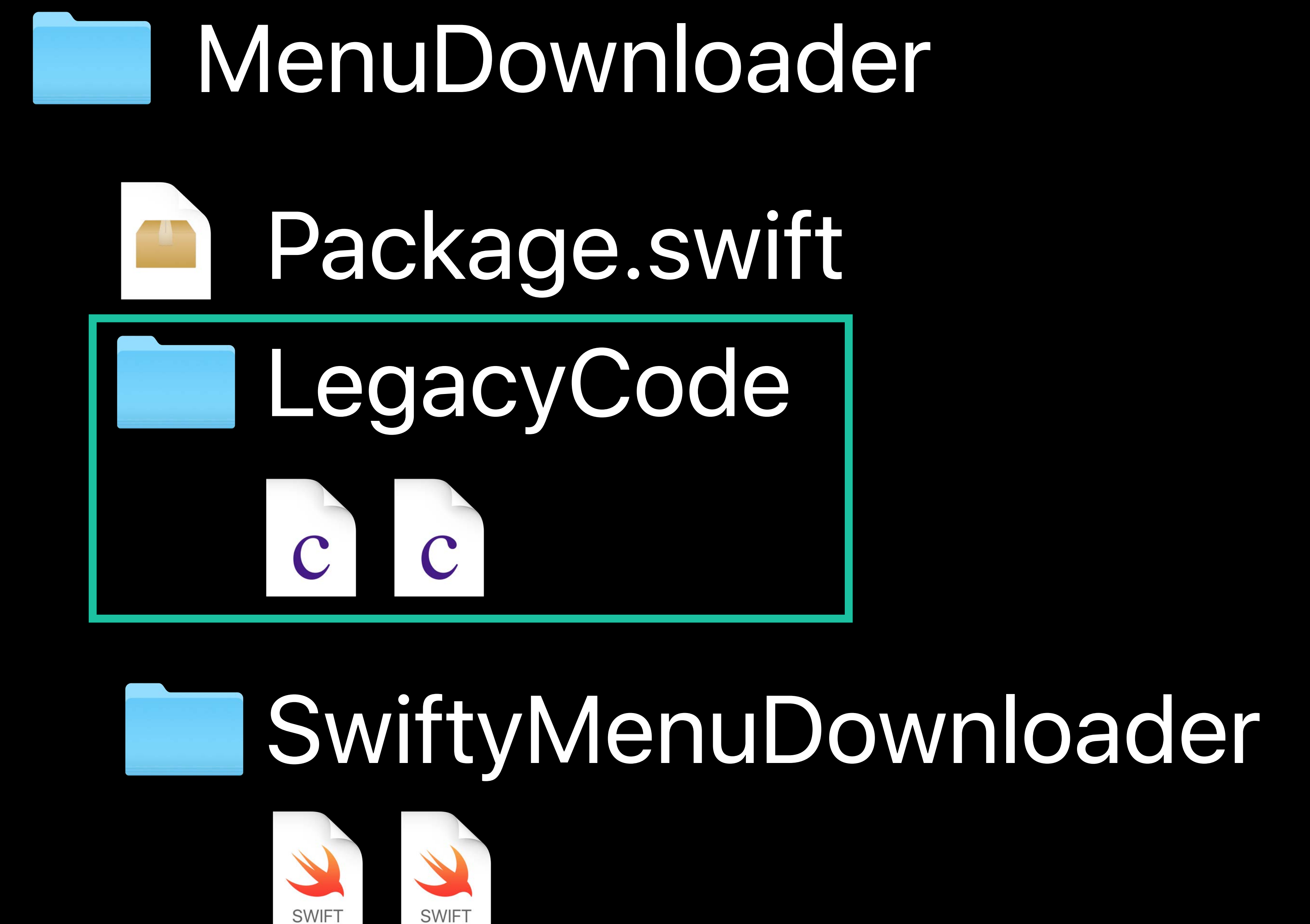


 SwiftyMenuDownloader



# Configuring Targets

```
targets: [  
  .target(  
    name: "LegacyMenuDownloader",  
    path: "LegacyCode",  
    cSettings: [  
      .define("SHOW_SECRET_MENU", to: "1")  
    ]  
  ),  
]
```



# Configuring Targets

```
targets: [  
  .target(  
    name: "LegacyMenuDownloader",  
    path: "LegacyCode",  
    cSettings: [  
      .define("SHOW_SECRET_MENU", to: "1")  
    ]  
  ),  
]
```

 MenuDownloader

 Package.swift

 LegacyCode

 c

 c

 SwiftyMenuDownloader

 SWIFT

 SWIFT

# Configuring Targets

```
targets: [  
  .target(  
    name: "LegacyMenuDownloader",  
    path: "LegacyCode",  
    cSettings: [  
      .define("SHOW_SECRET_MENU", to: "1")  
    ]  
  ),  
]
```


 MenuDownloader

 Package.swift

 LegacyCode

 SwiftyMenuDownloader

# Configuring Targets

```
targets: [  
  .target(  
    name: "LegacyMenuDownloader",  
    path: "LegacyCode",  
    cSettings: [  
      .define("SHOW_SECRET_MENU", to: "1")  
    ]  
  ),  
]
```



 MenuDownloader

 Package.swift

 LegacyCode

 c  c

 SwiftyMenuDownloader

 SWIFT  SWIFT



# Configuring Targets

```
targets: [  
    ...  
    .target(  
        name: "SwiftyMenuDownloader",  
        path: "SwiftyMenuDownloader",  
        dependencies: ["LegacyMenuDownloader"]  
    ),  
]
```

 MenuDownloader

 Package.swift

 LegacyCode

 c  c

 SwiftyMenuDownloader

 SWIFT  SWIFT

# Configuring Targets

```
targets: [  
    ...  
    .target(  
        name: "SwiftyMenuDownloader",  
        path: "SwiftyMenuDownloader",  
        dependencies: ["LegacyMenuDownloader"]  
    ),  
]
```



 MenuDownloader

 Package.swift

 LegacyCode

 SwiftyMenuDownloader

# Configuring Targets

```
targets: [  
    ...  
    .target(  
        name: "SwiftyMenuDownloader",  
        path: "SwiftyMenuDownloader",  
        dependencies: ["LegacyMenuDownloader"]  
    ),  
]
```



 MenuDownloader

 Package.swift

 LegacyCode

 SwiftyMenuDownloader

```
// Configuring Library Products
    ...
products: [
    .library(
        name: "SwiftMenuDownloader",
        targets: ["SwiftMenuDownloader"]
    ),
    .library(
        name: "LegacyMenuDownloader",
        type: .dynamic,
        targets: ["LegacyMenuDownloader"]
    ),
]
```

```
// Configuring Library Products
    ...
products: [
    .library(
        name: "SwiftyMenuDownloader",
        targets: ["SwiftyMenuDownloader"]
    ),
    .library(
        name: "LegacyMenuDownloader",
        type: .dynamic,
        targets: ["LegacyMenuDownloader"]
    ),
]
```

```
// Configuring Library Products
    ...
products: [
    .library(
        name: "SwiftMenuDownloader",
        targets: ["SwiftMenuDownloader"]
    ),
    .library(
        name: "LegacyMenuDownloader",
        type: .dynamic,
        targets: ["LegacyMenuDownloader"]
    ),
]
```

```
// Configuring Library Products
    ...
products: [
    .library(
        name: "SwiftMenuDownloader",
        targets: ["SwiftMenuDownloader"]
    ),
    .library(
        name: "LegacyMenuDownloader",
        type: .dynamic,
        targets: ["LegacyMenuDownloader"]
    ),
]
```

# Package Dependencies



```
// Configuring Package Dependencies
```

```
let package = Package(  
  name: "MenuDownloader",  
  products: [  
    ...  
  ],  
  dependencies: [  
    .package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0")),  
  ],  
  targets: [  
    ...  
  ]  
)
```

```
// Configuring Package Dependencies
```

```
let package = Package(  
  name: "MenuDownloader",  
  products: [  
    ...  
  ],  
  dependencies: [  
    .package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0")),  
  ],  
  targets: [  
    ...  
  ]  
)
```

```
// Configuring Package Dependencies
```

```
let package = Package(  
  name: "MenuDownloader",  
  products: [  
    ...  
  ],  
  dependencies: [  
    .package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0")),  
  ],  
  targets: [  
    ...  
  ]  
)
```

```
// Configuring Package Dependencies
// A version starting from 2.0.0 up to but not including the next major version (3.0.0)

let package = Package(
  name: "MenuDownloader",
  products: [
    ...
  ],
  dependencies: [
    .package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0")),
  ],
  targets: [
    ...
  ]
)
```

```
// Configuring Package Dependencies
// A version starting from 2.0.0 up to but not including the next major version (3.0.0)

let package = Package(
  name: "MenuDownloader",
  products: [
    ...
  ],
  dependencies: [
    .package(url: "https://github.com/jpsim/Yams", from: "2.0.0"),
  ],
  targets: [
    ...
  ]
)
```

```
// Configuring Package Dependencies
// A version starting from 2.0.0 up to but not including the next major version (3.0.0)

let package = Package(
  name: "MenuDownloader",
  products: [
    ...
  ],
  dependencies: [
    .package(url: "https://github.com/jpsim/Yams", from: "2.0.0"),
  ],
  targets: [
    ...
  ]
)
```

```
// Configuring Package Dependencies
```

```
// Version-based requirements.
```

```
.package(url: "https://github.com/jpsim/Yams", from: "2.0.0")
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMinor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .exact("2.0.0"))
```

```
// Configuring Package Dependencies
```

```
// Version-based requirements.
```

```
.package(url: "https://github.com/jpsim/Yams", from: "2.0.0")
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMinor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .exact("2.0.0"))
```



```
// Configuring Package Dependencies
```

```
// Version-based requirements.
```

```
.package(url: "https://github.com/jpsim/Yams", from: "2.0.0")
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMinor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .exact("2.0.0"))
```

```
// Configuring Package Dependencies
```

```
// Version-based requirements.
```

```
.package(url: "https://github.com/jpsim/Yams", from: "2.0.0")
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMajor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .upToNextMinor(from: "2.0.0"))
```

```
.package(url: "https://github.com/jpsim/Yams", .exact("2.0.0"))
```

```
// Configuring Package Dependencies
```

```
// Branch-based requirement.
```

```
.package(url: "https://github.com/jpsim/Yams", .branch("master"))
```

```
// Revision-based requirement.
```

```
.package(url: "https://github.com/jpsim/Yams", .revision("85cfe06"))
```

```
// Configuring Package Dependencies
```

```
// Branch-based requirement.
```

```
.package(url: "https://github.com/jpsim/Yams", .branch("master"))
```

```
// Revision-based requirement.
```

```
.package(url: "https://github.com/jpsim/Yams", .revision("85cfe06"))
```

```
// Configuring Package Dependencies
```

```
// Branch-based requirement.
```

```
.package(url: "https://github.com/jpsim/Yams", .branch("master"))
```

```
// Revision-based requirement.
```

```
.package(url: "https://github.com/jpsim/Yams", .revision("85cfe06"))
```

```
// Configuring Package Dependencies
    ...
targets: [
    ...
    .target(
        name: "SwiftyMenuDownloader",
        dependencies: ["LegacyMenuDownloader", "Yams"]
    ),
    ...
]
```

# Swift Tools Version

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MenuDownloader",
    ...
)
```

# Swift Tools Version

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MenuDownloader",
    ...
)
```



Manifest API version



# Swift Tools Version

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MenuDownloader",
    ...
)
```



Manifest API version



Dependency resolution

# Swift Tools Version

```
// swift-tools-version:5.1
import PackageDescription

let package = Package(
    name: "MenuDownloader",
    ...
)
```



Manifest API version



Dependency resolution



Minimum compiler version

# Platform Availability

macOS

iOS

tvOS

watchOS

Linux

```
#if os(Linux)
// Code specific to Linux
#endif

#if canImport(Network)
// Code specific to platforms where Network framework is available
#endif
```

# Platform Availability

**macOS**

10.10

**iOS**

8.0

**tvOS**

9.0

**watchOS**

2.0

**Linux**

```
#if os(Linux)
// Code specific to Linux
#endif

#if canImport(Network)
// Code specific to platforms where Network framework is available
#endif
```

```
// Configuring Minimum Deployment Target Version
```

```
let package = Package(  
  name: "MyPackage",  
  platforms: [  
    .macOS(.v10_15), .iOS(.v13),  
  ],  
  products: [  
    ...  
  ],  
  dependencies: [  
    ...  
  ],  
  targets: [  
    ...  
  ]  
)
```

```
// Configuring Minimum Deployment Target Version
```

```
let package = Package(  
  name: "MyPackage",  
  platforms: [  
    .macOS(.v10_15), .iOS(.v13),  
  ],  
  products: [  
    ...  
  ],  
  dependencies: [  
    ...  
  ],  
  targets: [  
    ...  
  ]  
)
```

```
// Configuring Minimum Deployment Target Version
```

```
let package = Package(  
  name: "MyPackage",  
  platforms: [  
    .macOS(.v10_15), .iOS(.v13),  
  ],  
  products: [  
    ...  
  ],  
  dependencies: [  
    ...  
  ],  
  targets: [  
    ...  
  ]  
)
```

```
// Configuring Minimum Deployment Target Version
```

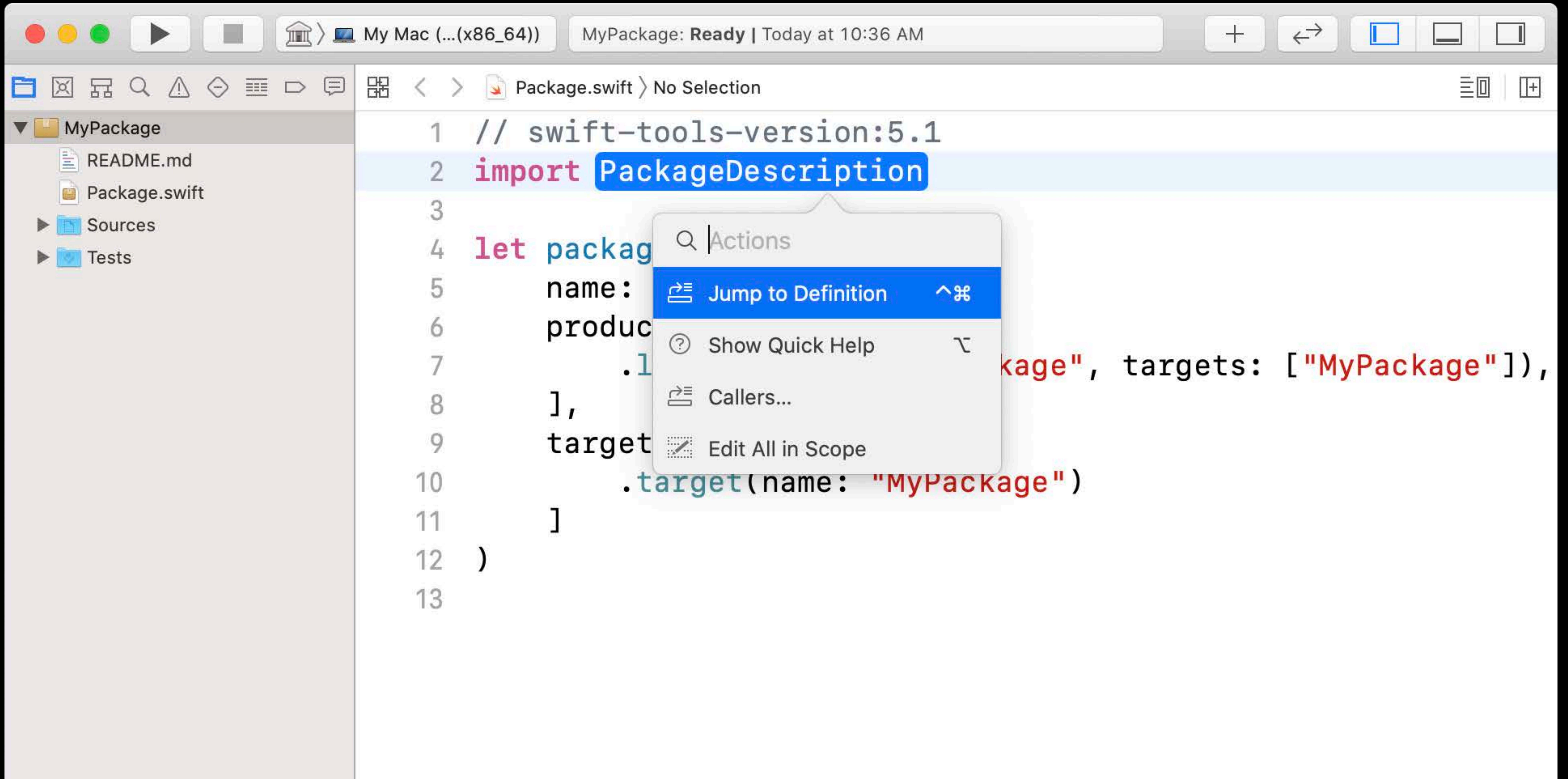
```
let package = Package(  
  name: "MyPackage",  
  platforms: [  
    .macOS("10.15"), .iOS("13"),  
  ],  
  products: [  
    ...  
  ],  
  dependencies: [  
    ...  
  ],  
  targets: [  
    ...  
  ]  
)
```



# Documentation - Generated Module Interface

```
MyPackage > My Mac (64-bit (x86_64)) MyPackage: Ready | Today at 10:24 AM
PackageDescription > Package.Dependency
410 extension Package.Dependency {
411
412     /// Add a package dependency that is required from the given minimum version,
413     /// going up to the next major version.
414     ///
415     /// This is the recommend way to specify a remote package dependency because
416     /// it allows you to specify the minimum version you require and gives
417     /// explicit opt-in for new major versions, but otherwise provides maximal
418     /// flexibility on which version is used. This helps to prevent conflicts in
419     /// your package dependency graph.
420     ///
421     /// For example, specifying
422     ///
423     ///     .package(url: "https://example.com/example-package.git", from: "1.2.3"),
424     ///
425     /// will allow the Swift package manager to select a version like a "1.2.3",
426     /// "1.2.4" or "1.3.0" but not "2.0.0".
427     ///
428     /// - Parameters:
429     ///     - url: The valid Git URL of the package.
430     ///     - version: The minimum version requirement.
431     public static func package(url: String, from version: PackageDescription.Version) ->
432         PackageDescription.Package.Dependency
433
434     /// Add a remote package dependency given a version requirement.
435     ///
```

# Documentation - Generated Module Interface



The screenshot shows the Xcode IDE interface. The top status bar indicates the project is 'MyPackage: Ready' and the time is 'Today at 10:36 AM'. The left sidebar shows the project structure with 'MyPackage' expanded, containing 'README.md', 'Package.swift', 'Sources', and 'Tests'. The main editor displays the 'Package.swift' file with the following code:

```
1 // swift-tools-version:5.1
2 import PackageDescription
3
4 let package = Package(
5     name: "MyPackage",
6     products: [
7         .library(name: "MyPackage", targets: ["MyPackage"]),
8     ],
9     targets: [
10        .target(name: "MyPackage")
11    ]
12 )
13
```

A context menu is open over the 'PackageDescription' import on line 2. The menu is titled 'Actions' and contains the following items:

- Jump to Definition (^⌘)
- Show Quick Help (⌘?)
- Callers...
- Edit All in Scope

# Editing Packages

Boris Buegling, Developer Tools Engineer

# Editing Packages

Local packages can be edited

Same is true for standalone packages

Package dependencies are locked

# Overriding Dependencies

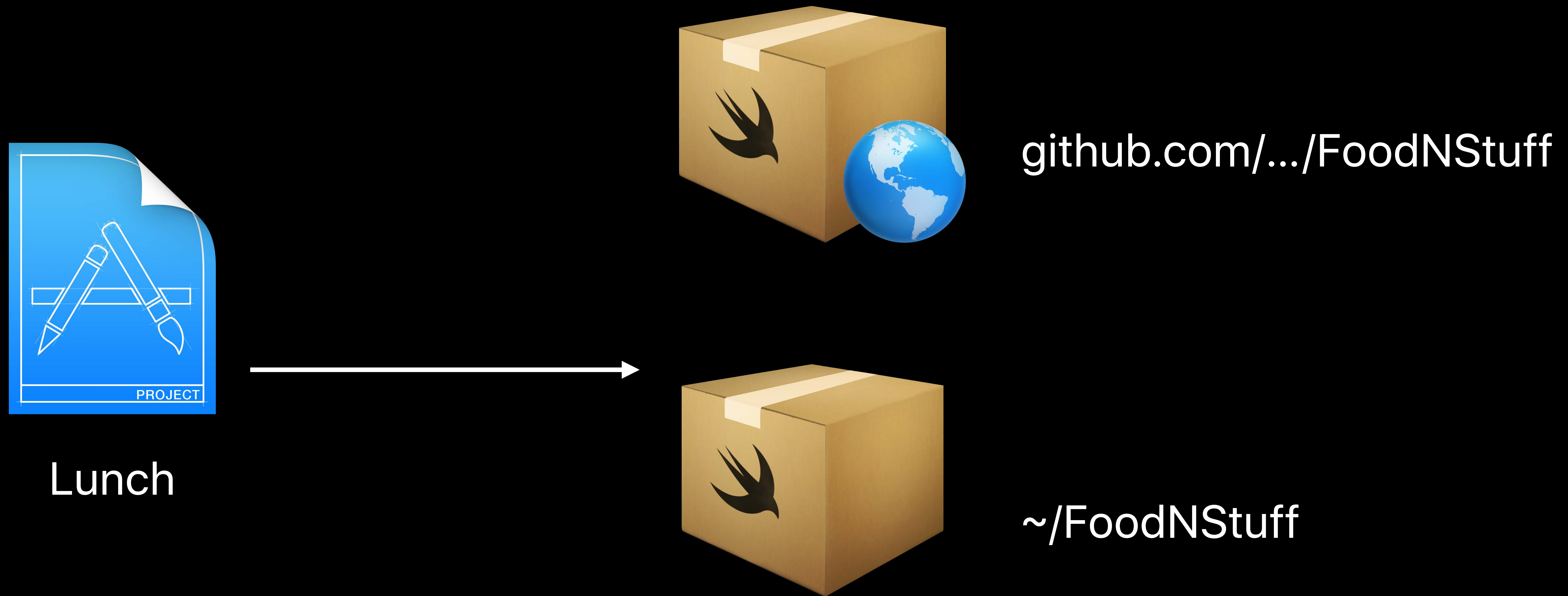


Lunch



[github.com/.../FoodNStuff](https://github.com/.../FoodNStuff)

# Overriding Dependencies



***Demo***

Editing packages

Overriding can be used to  
edit packages you do not own.



**Open Source Project**

Swift Package Manager

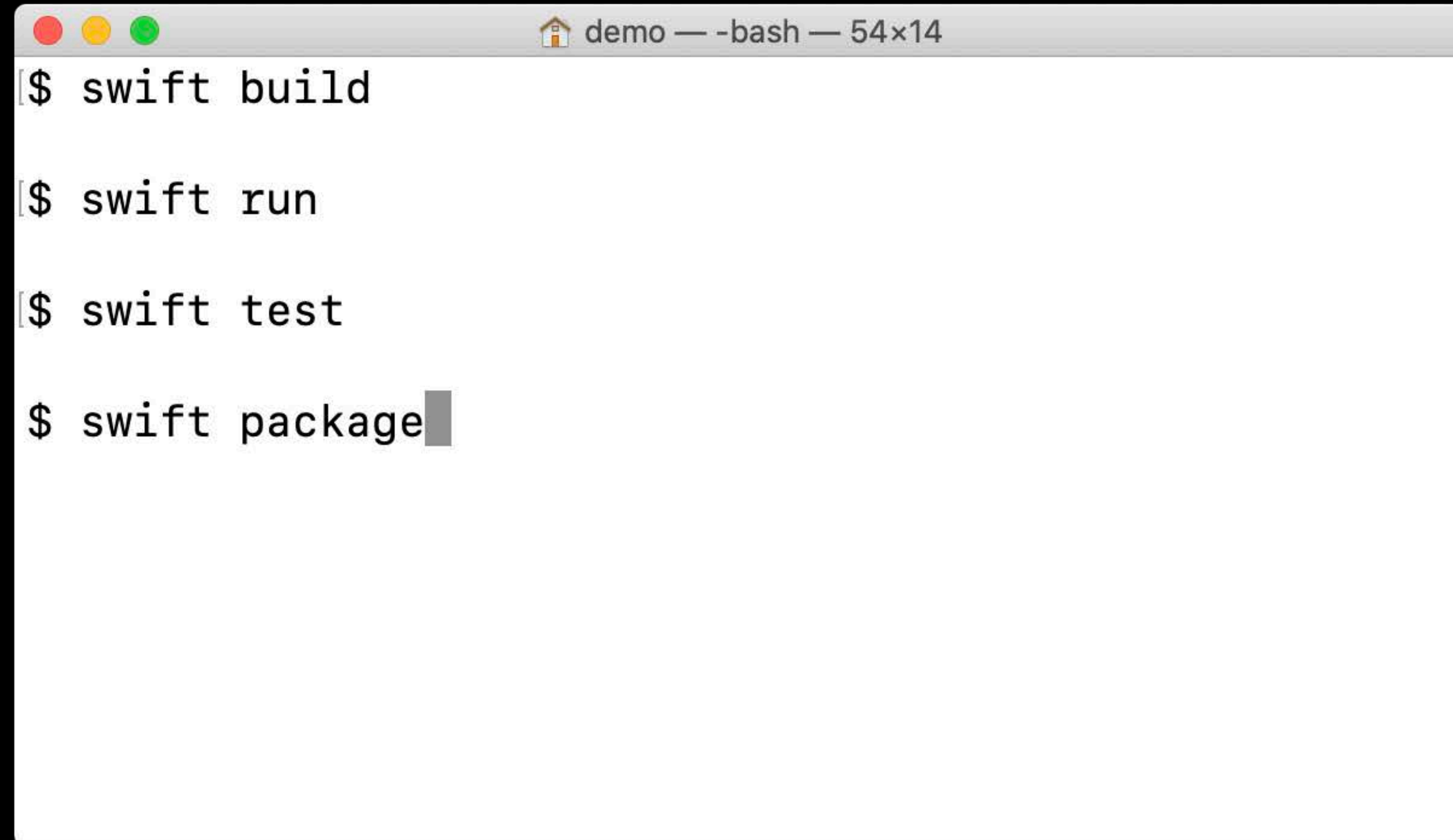
# A Cross-Platform Build System for Swift

**macOS**

**Linux**

**Future  
Platforms**

# SwiftPM Commands

A terminal window with a title bar that reads "demo — -bash — 54x14". The window contains four lines of text, each representing a command entered in a shell. The first three lines are enclosed in square brackets, indicating they were entered from a prompt. The fourth line is not enclosed in brackets and has a cursor at the end of the text.

```
[demo — -bash — 54x14]
[$ swift build ]
[$ swift run ]
[$ swift test ]
$ swift package
```

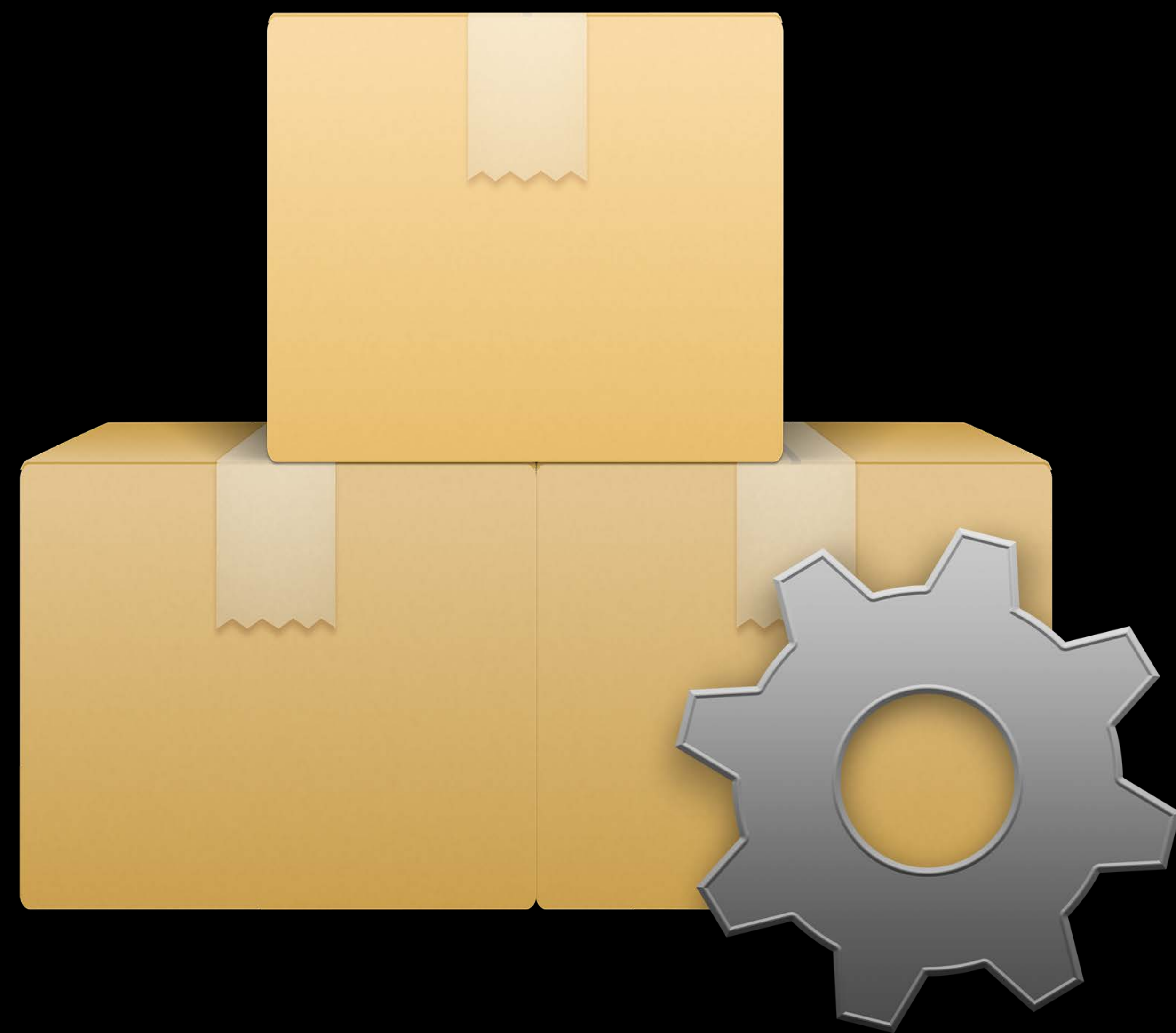
# SwiftPM Commands

A terminal window with a title bar that reads "demo — -bash — 54x14". The terminal contains four lines of text, each starting with a prompt character: "[ \$ swift build", "[ \$ swift run", "[ \$ swift test", and "\$ swift package" followed by a cursor. The first three lines are enclosed in square brackets on the right side, suggesting they are part of a list or a specific context.

```
demo — -bash — 54x14
[ $ swift build
[ $ swift run
[ $ swift test
$ swift package
```

xcodesbuild

# Swift Package Support in Xcode

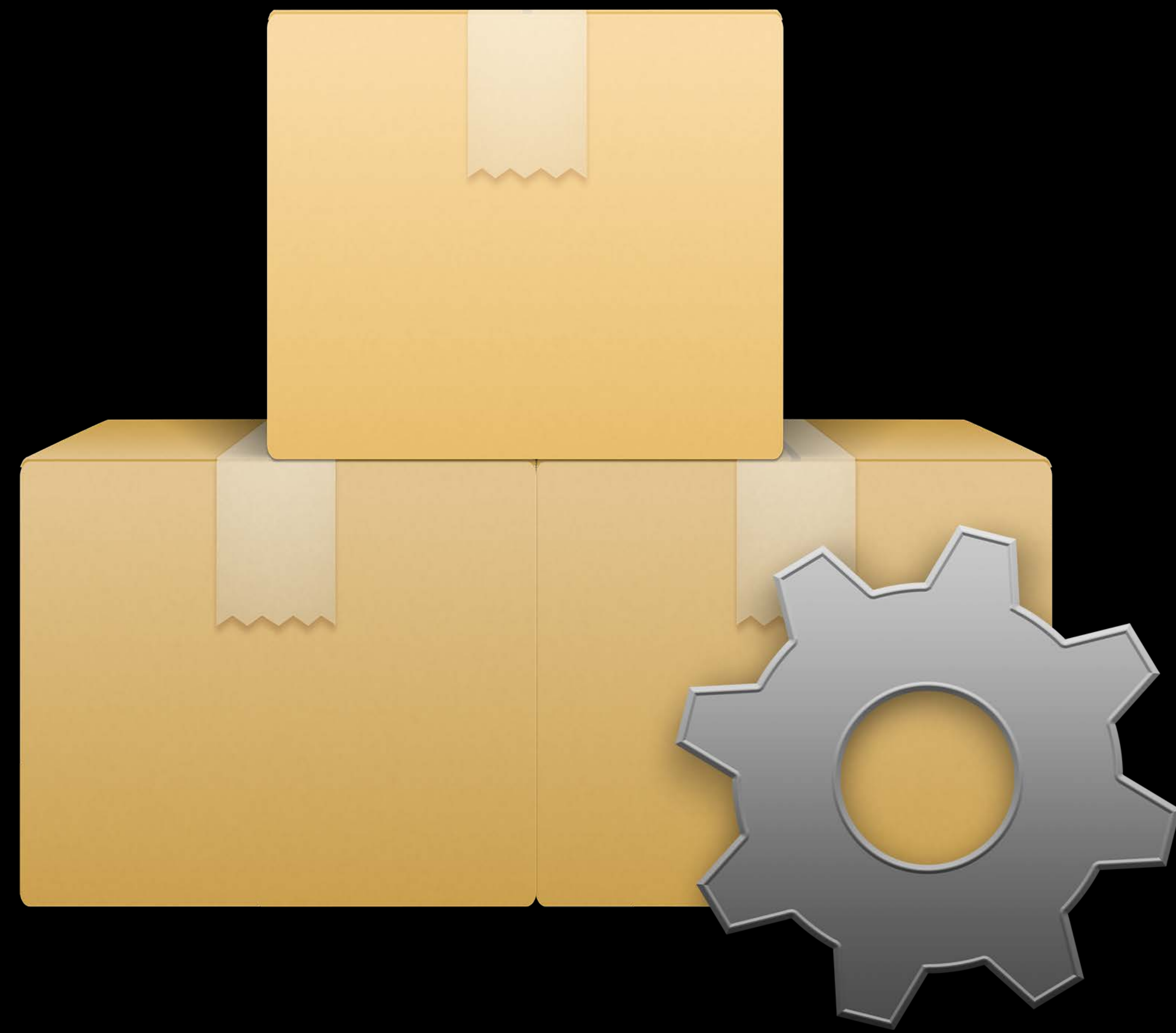


libSwiftPM

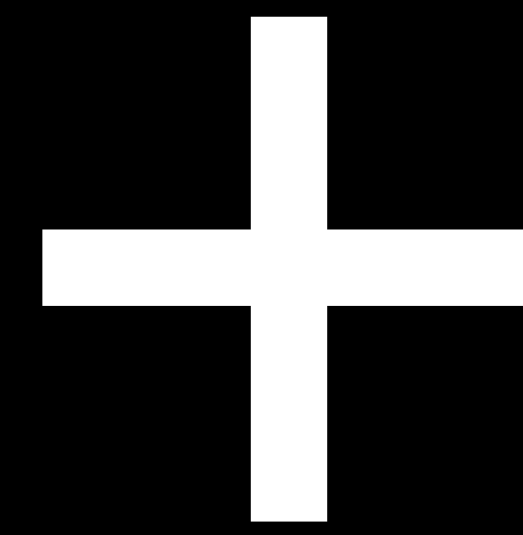


Xcode

# Swift Package Support in Xcode

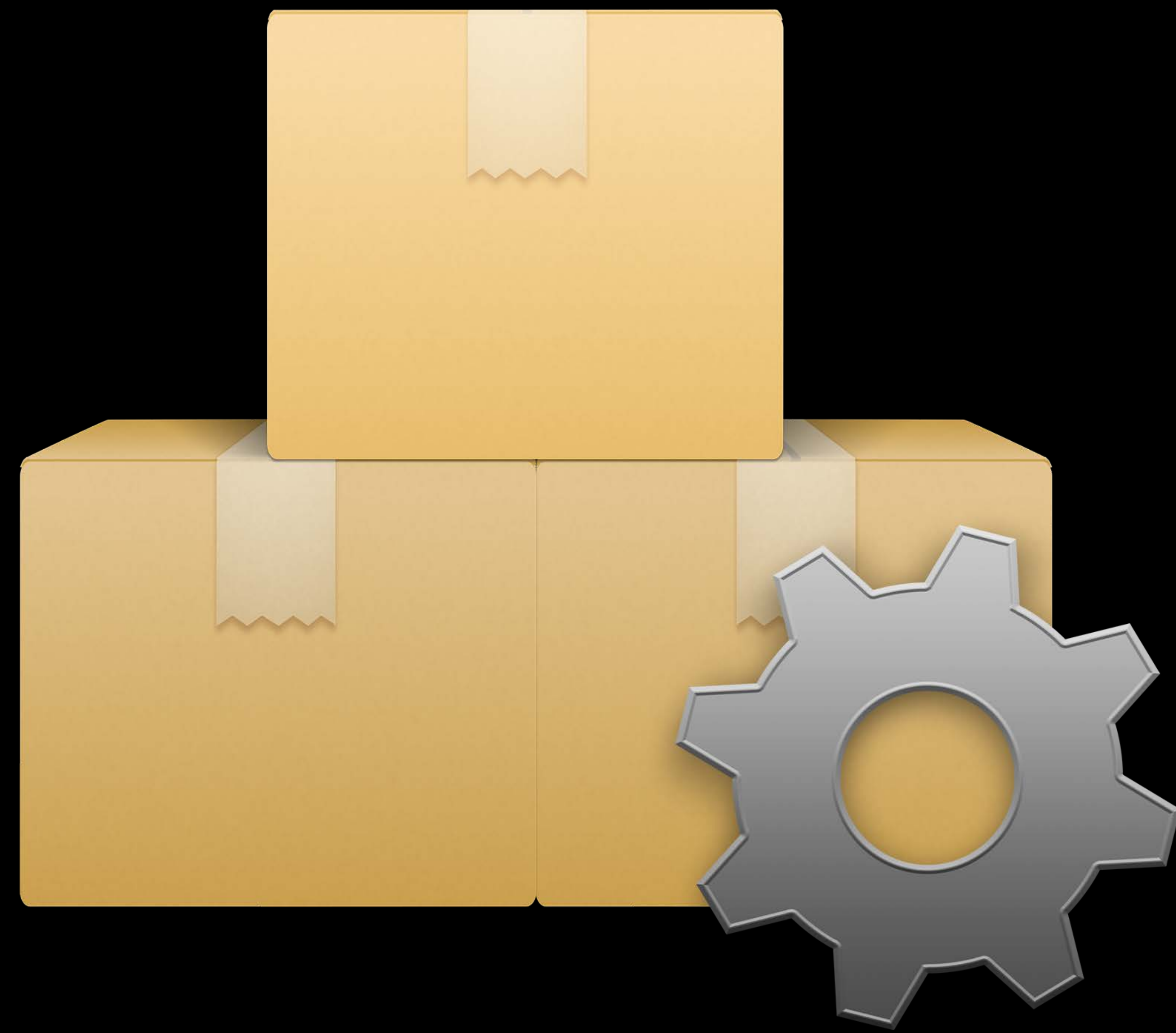


libSwiftPM



Xcode

# SourceKit-LSP



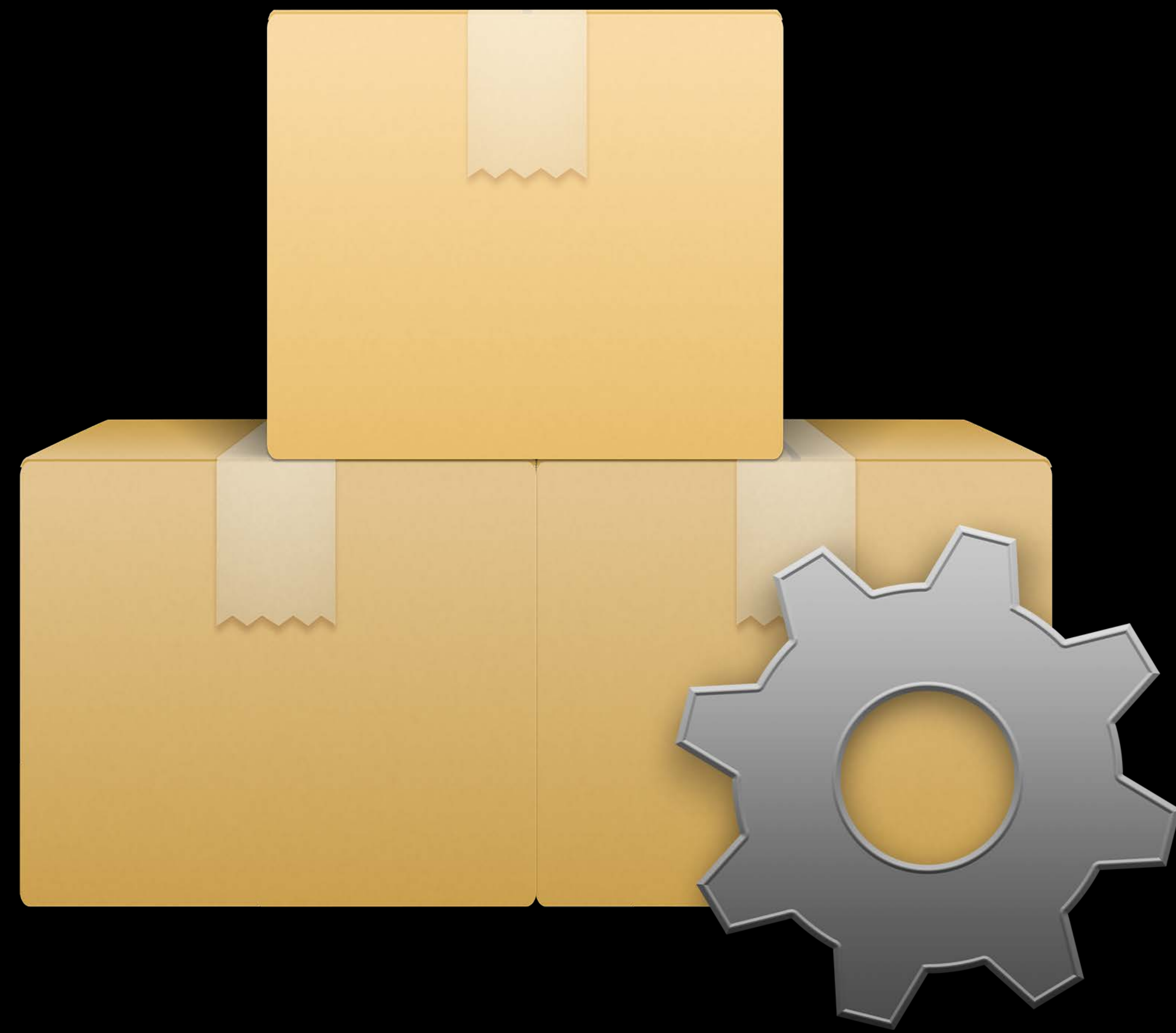
libSwiftPM



SourceKit-LSP



# SourceKit-LSP

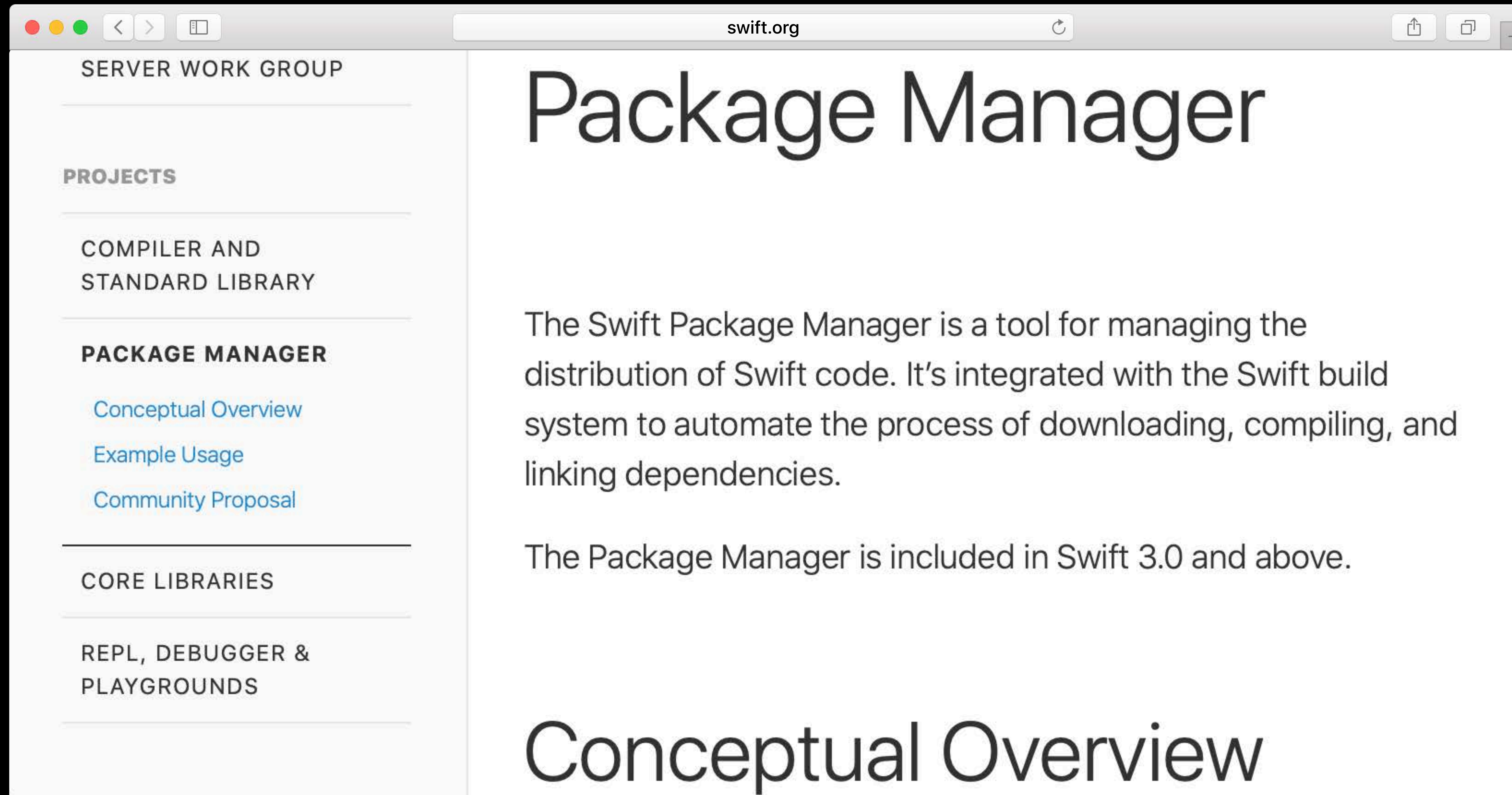


libSwiftPM

+

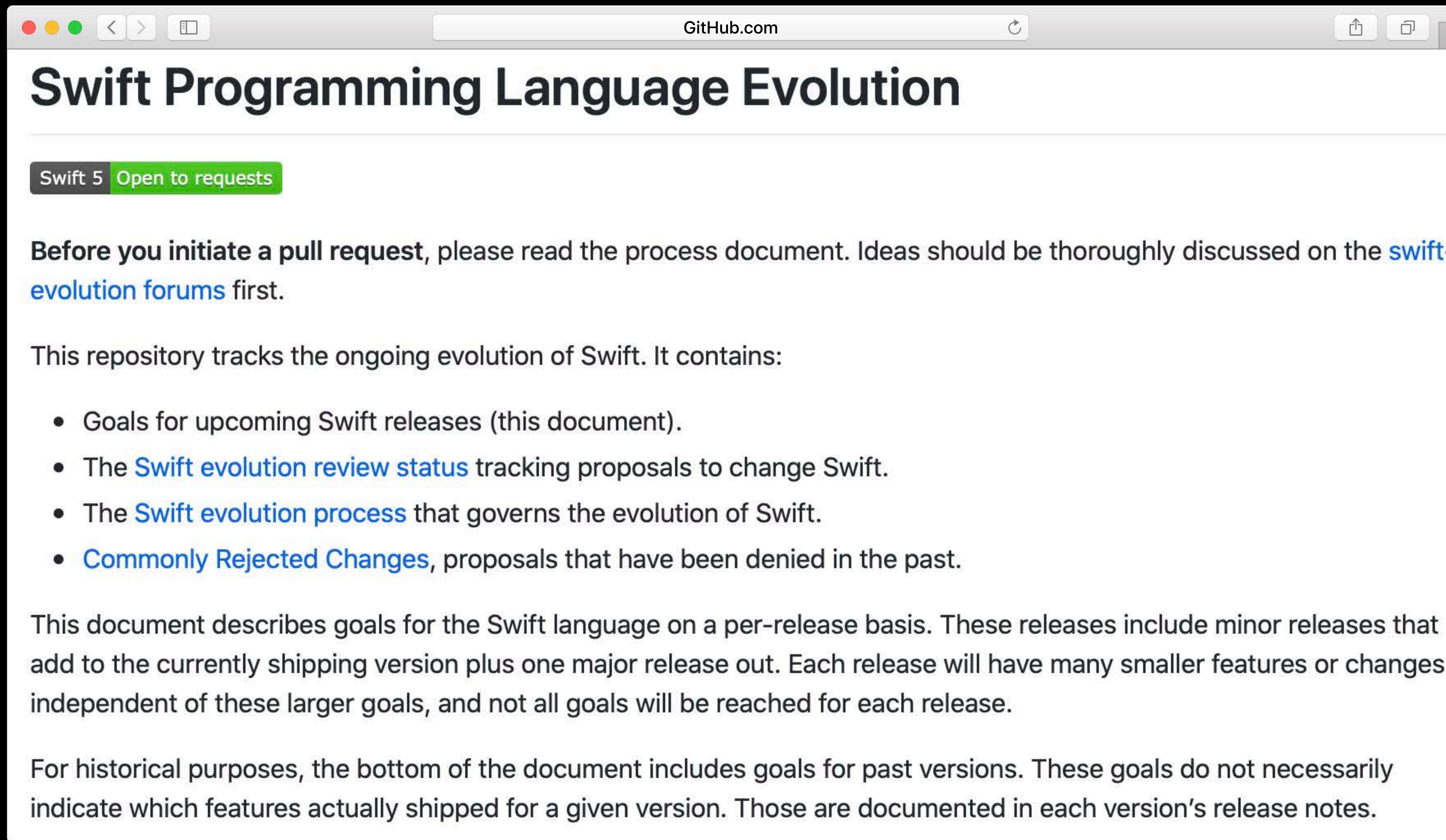


SourceKit-LSP



[swift.org/package-manager/](https://swift.org/package-manager/)

# Swift Evolution



The image shows a screenshot of a web browser displaying the GitHub repository page for "Swift Evolution". The browser's address bar shows "GitHub.com". The page title is "Swift Programming Language Evolution". Below the title, there is a navigation bar with "Swift 5" and "Open to requests". The main content area contains a paragraph of text, a list of bullet points, and two more paragraphs of text. The text is in a clean, sans-serif font, and the layout is simple and professional.

## Swift Programming Language Evolution

Swift 5 [Open to requests](#)

Before you initiate a pull request, please read the process document. Ideas should be thoroughly discussed on the [swift-evolution forums](#) first.

This repository tracks the ongoing evolution of Swift. It contains:

- Goals for upcoming Swift releases (this document).
- The [Swift evolution review status](#) tracking proposals to change Swift.
- The [Swift evolution process](#) that governs the evolution of Swift.
- [Commonly Rejected Changes](#), proposals that have been denied in the past.

This document describes goals for the Swift language on a per-release basis. These releases include minor releases that add to the currently shipping version plus one major release out. Each release will have many smaller features or changes independent of these larger goals, and not all goals will be reached for each release.

For historical purposes, the bottom of the document includes goals for past versions. These goals do not necessarily indicate which features actually shipped for a given version. Those are documented in each version's release notes.

[github.com/apple/swift-evolution](https://github.com/apple/swift-evolution)

# Swift Forums

The screenshot shows the Swift Forums website interface. At the top, there is a navigation bar with the Swift logo, a search icon, and buttons for 'Sign Up' and 'Log In'. Below the navigation bar, there are filters for 'Development', 'Package Manager', and 'all tags', along with sorting options for 'Latest' and 'Top'. The main content area displays a list of forum topics with columns for 'Topic', 'Replies', 'Views', and 'Activity'.

Topic	Replies	Views	Activity
Swift Package Manager Evolution Ideas	2	1.2k	Jun '18
About the Package Manager category For developers to discuss the implementation of the Swift package manager. This category will accept email sent to: <a href="mailto:swift+swiftpm@forums.swift.org">swift+swiftpm@forums.swift.org</a>	0	316	Jan '18
SPM Windows: Link issues during build windows	18	273	3h
Generated Xcode project is not testable	2	95	9h

[forums.swift.org/c/development/SwiftPM](https://forums.swift.org/c/development/SwiftPM)

# Package Resources Are Not Supported



# Package Resources

180 lines (112 sloc) | 14.4 KB

Raw Blame History

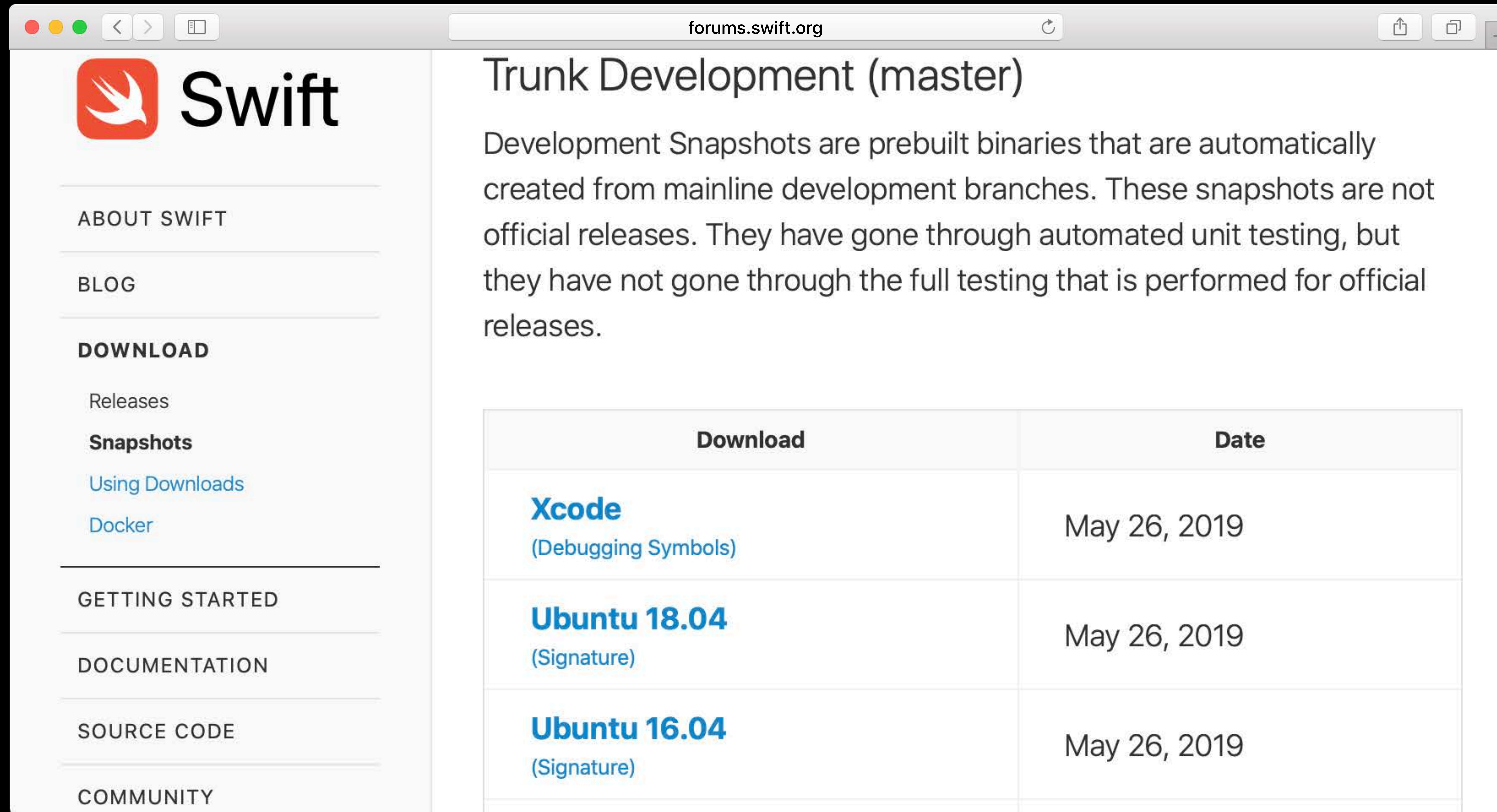
## Package Resources

- Proposal: [SE-NNNN](#)
- Authors: [Anders Bertelrud](#)
- Review Manager: [TBD](#)
- Status: **WIP**

### Introduction

Packages should be able to contain images, data files, or other resources that are needed at runtime. This draft proposal describes SwiftPM support for specifying package resources, and introduces a consistent way of referring to them from the source code in the package.

# Trunk Snapshots



The screenshot shows a web browser window with the URL `forums.swift.org`. The page title is "Trunk Development (master)". The main content area contains a paragraph explaining that development snapshots are prebuilt binaries created from mainline development branches, which are not official releases and have only undergone automated unit testing. Below the text is a table with two columns: "Download" and "Date". The table lists three download links, all dated May 26, 2019.

Download	Date
<a href="#">Xcode</a> (Debugging Symbols)	May 26, 2019
<a href="#">Ubuntu 18.04</a> (Signature)	May 26, 2019
<a href="#">Ubuntu 16.04</a> (Signature)	May 26, 2019

[swift.org/download/#snapshots](https://swift.org/download/#snapshots)

# Final Takeaway





# More Information

[developer.apple.com/wwdc19/410](https://developer.apple.com/wwdc19/410)

---

Swift Packages Lab

Thursday, 12:00

---

Swift Packages Lab

Friday, 12:00

---

 WWDC19