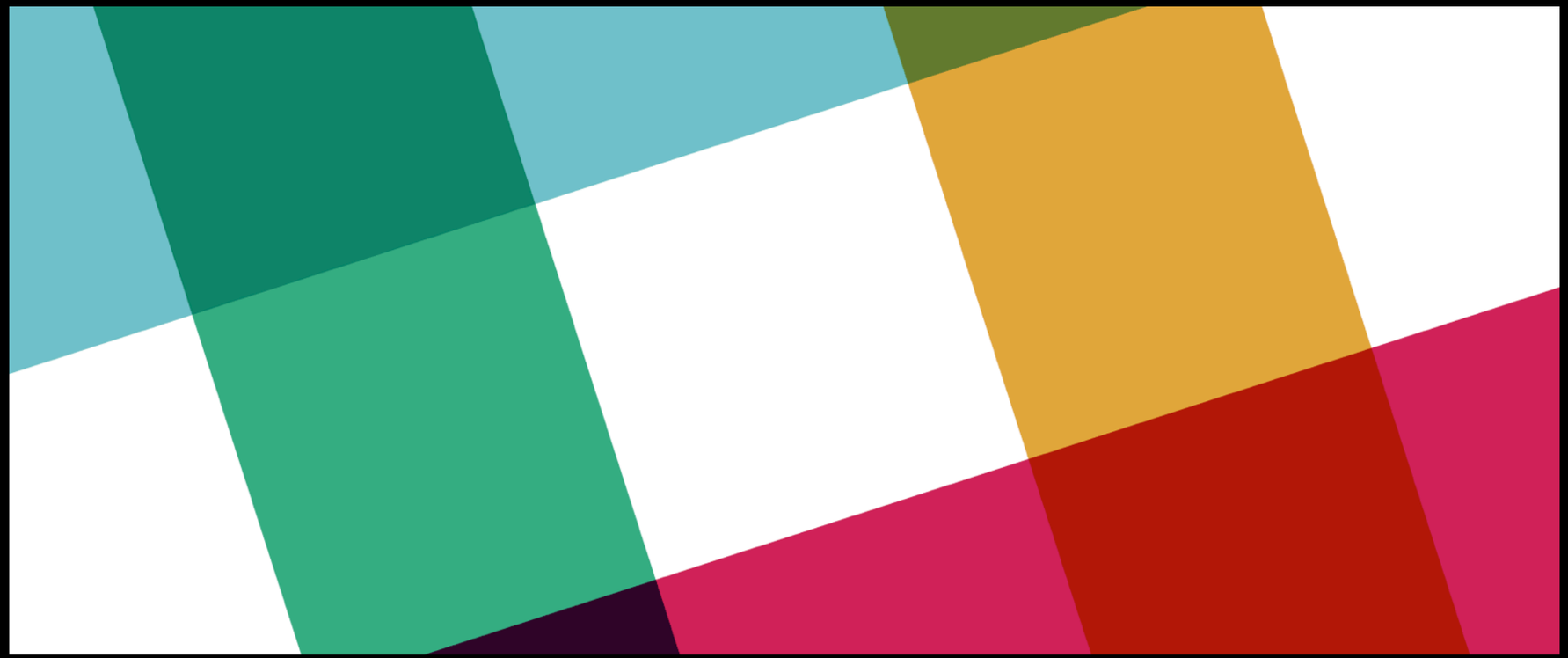
# Swift Package Manager

**@jmsmith, Site Reliability Engineer @SlackHQ**

🐟 **@Yasumoto**



- Engineers should have excellent tools.
- Work is mobile, but DevOps/SRE tools aren't yet.
- Swift libraries straddle both Server and Mobile.
- Pure-Swift libraries to work with infrastructure:
  - 🔐 SSL Certificates
  - 📈 Metrics retrieval
  - 📟 PagerDuty queries
  - 👨‍🍳 Chef
  - ☁️ AWS (thanks @noppoMan and @jonnymacs!)

# Agenda

1. 📦 Creating a Package
   — Example with `vapor/http`
   — Concepts
2. 🆕 New in Swift 4.2
3. 💻 Contributing!
   — Starter Bugs
   — Evolution Ideas

# 📦 Creating a Package

*and then I'll put that box inside of another box...*

# 📦 Creating a Package

- The Swift Package Manager makes it simple to organize your code into reusable modules
- You can point to git repositories of Swift code to pull that into your project
- Automatically installed with the rest of Swift.
  - macOS via Xcode
  - Ubuntu Linux via swift.org

# Example[curl]

```
~ $ mkdir ./curl
~ $ cd ./curl
~/curl $ ls
~/curl $
```

[curl] https://github.com/Yasumoto/curl

# Example

```
~/curl $ swift package init --type executable
Creating executable package: curl
Creating Package.swift
Creating README.md
Creating .gitignore
Creating Sources/
Creating Sources/curl/main.swift
Creating Tests/
Creating Tests/LinuxMain.swift
Creating Tests/curlTests/
Creating Tests/curlTests/curlTests.swift
Creating Tests/curlTests/XCTestManifests.swift
~/curl $
```

# Example

```
— Package.swift
— Sources/curl/main.swift
```

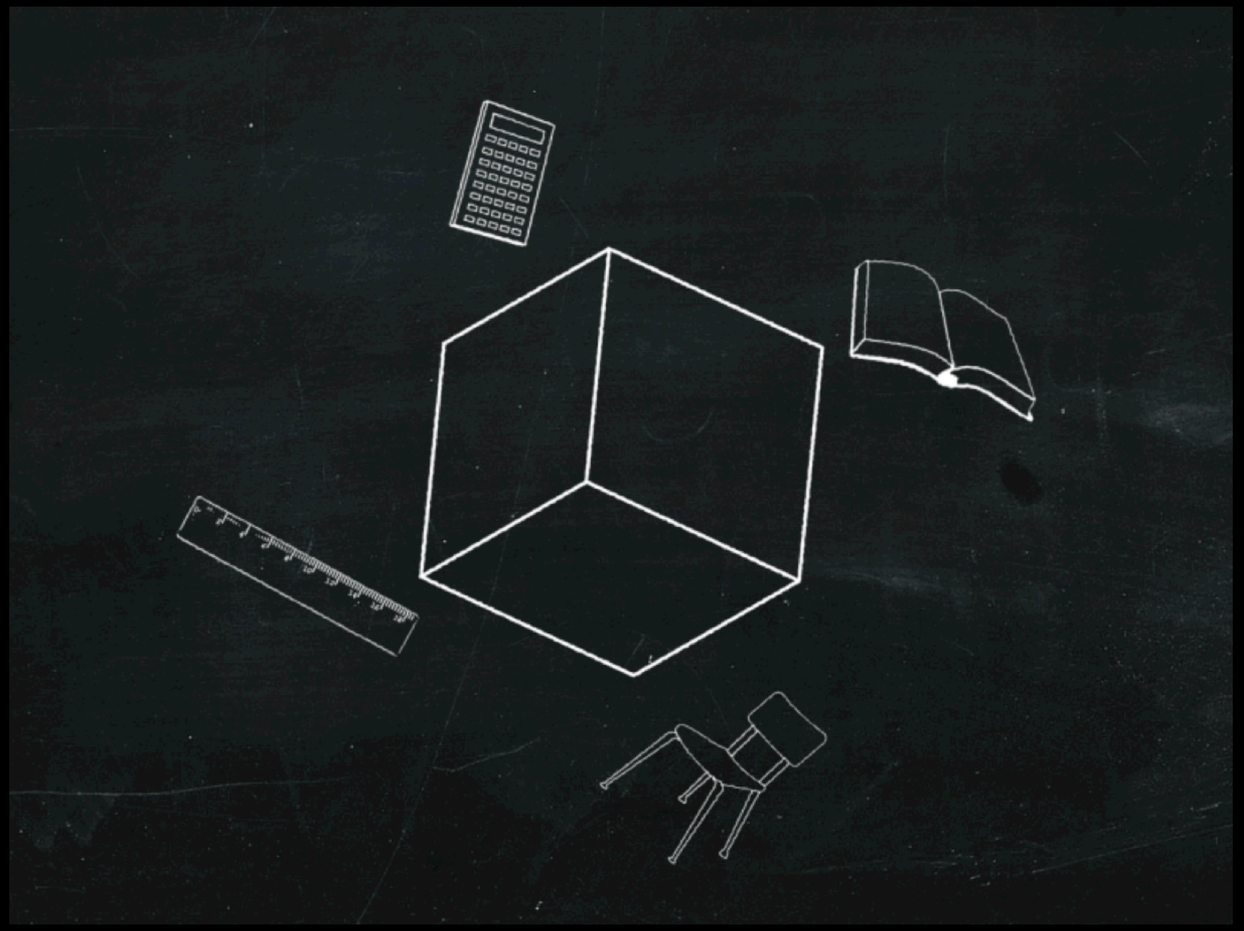
```
// swift-tools-version:4.2

import PackageDescription

let package = Package(
    name: "curl",
    dependencies: [
    ],
    targets: [
        .target(
            name: "curl",
            dependencies: []),
        .testTarget(
            name: "curlTests",
            dependencies: ["curl"]),
    ]
)
```

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

```
~/curl $ swift build
Compile Swift Module 'curl' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/debug/curl
~/curl $
```

```
~/curl $ swift build
Compile Swift Module 'curl' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/debug/curl
~/curl $ ./.build/x86_64-apple-macosx10.10/debug/curl
Hello world!
~/curl $
```

# 🧠 Concepts

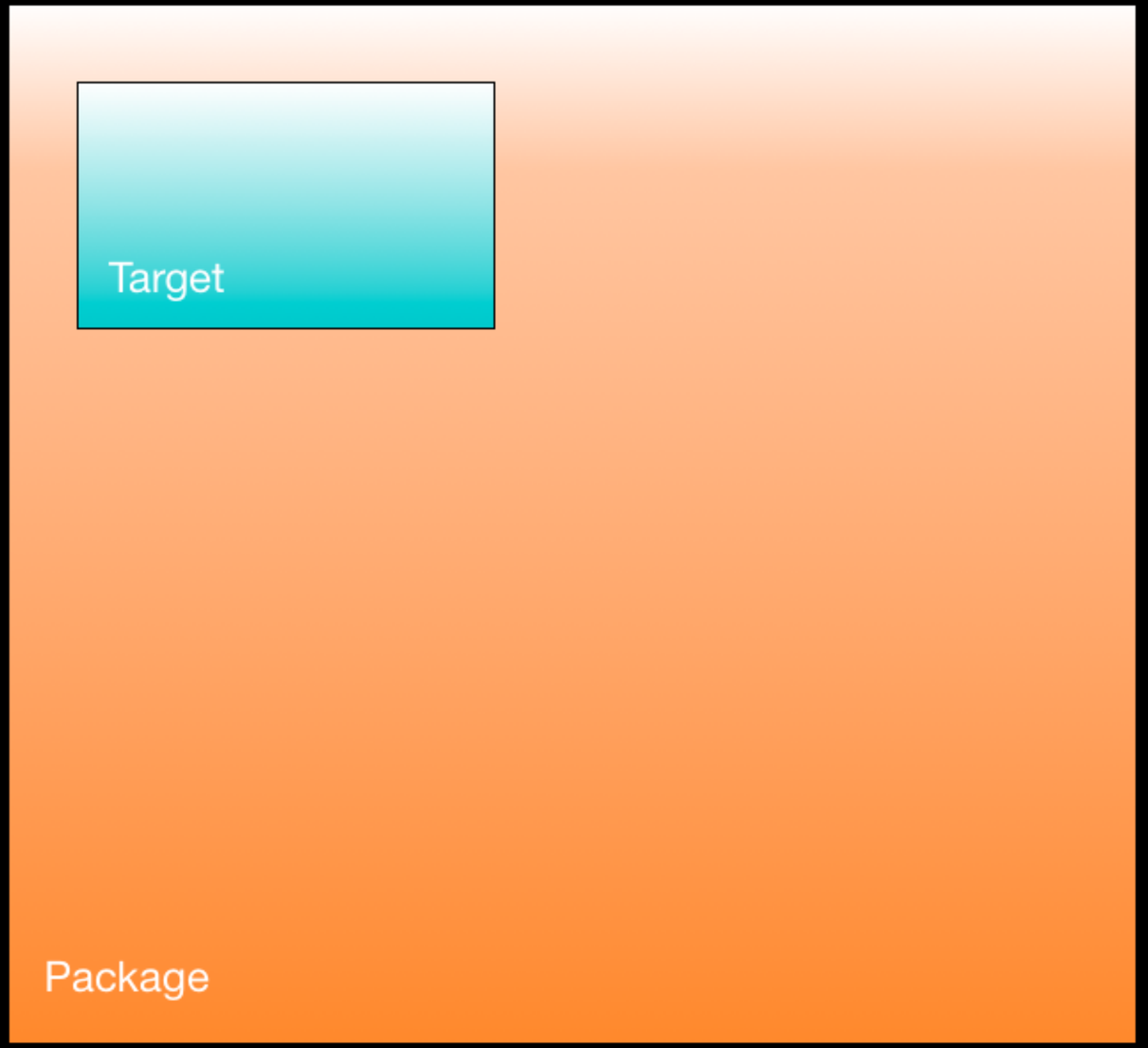



Package

— A Package is the core unit that you will build or operate upon.

— This consists of a collection of .swift source files, as well as a Package Manifest.

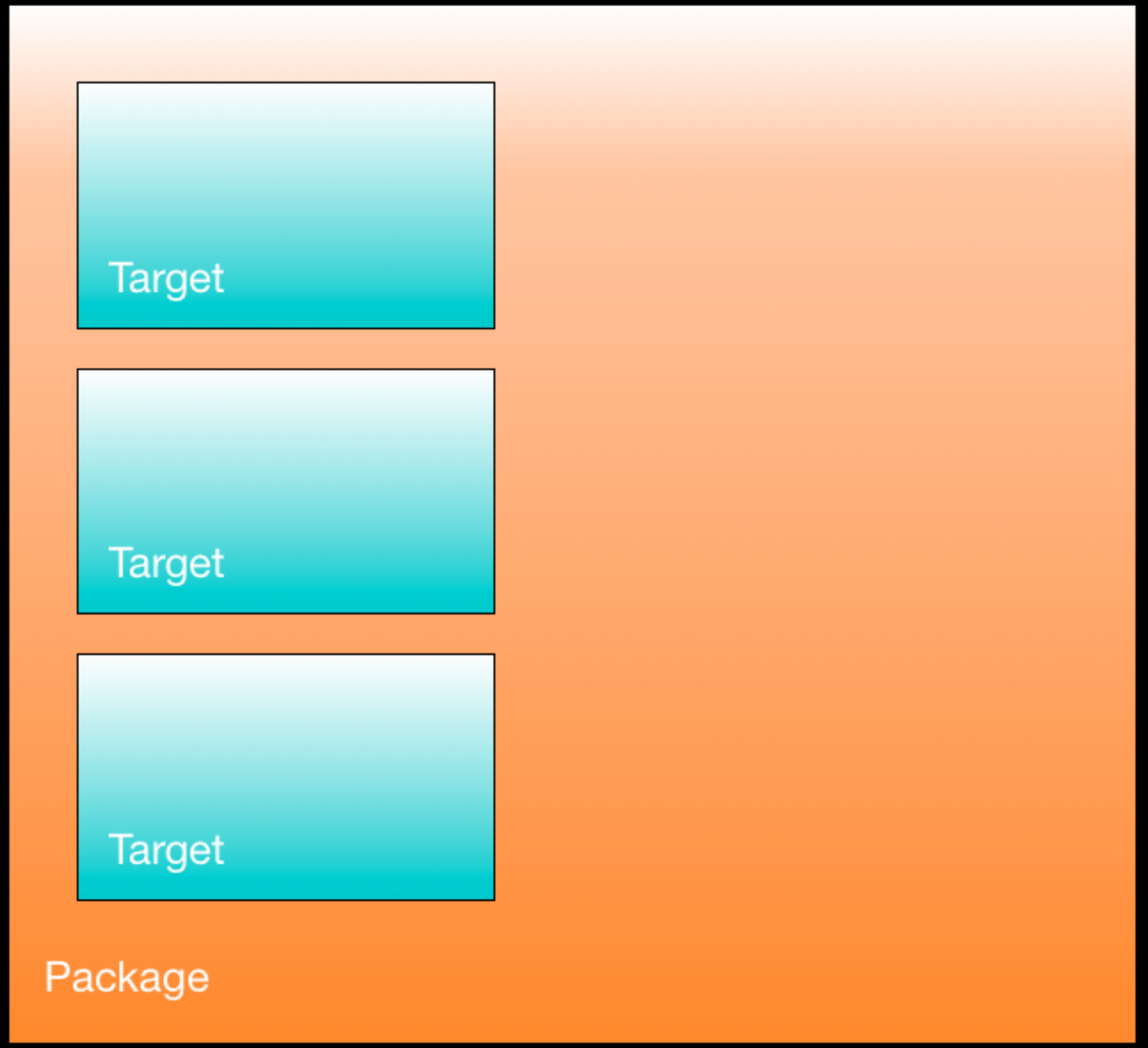
```
Package(
    name: String,
    pkgConfig: String? = nil,
    providers: [SystemPackageProvider]? = nil,
    products: [Product] = [],
    dependencies: [Dependency] = [],
    targets: [Target] = [],
    swiftLanguageVersions: [SwiftVersion]? = nil
)
```


Target
Package

— Code within a package can be broken up into Targets.

— Each Target specifies a directory where its source code resides.

```
target(
    name: String,
    dependencies: [PackageDescription.Target.Dependency] = default,
    path: String? = default,
    exclude: [String] = default,
    sources: [String]? = default,
    publicHeadersPath: String? = default
) -> PackageDescription.Target
```


Target
Target
Target
Package

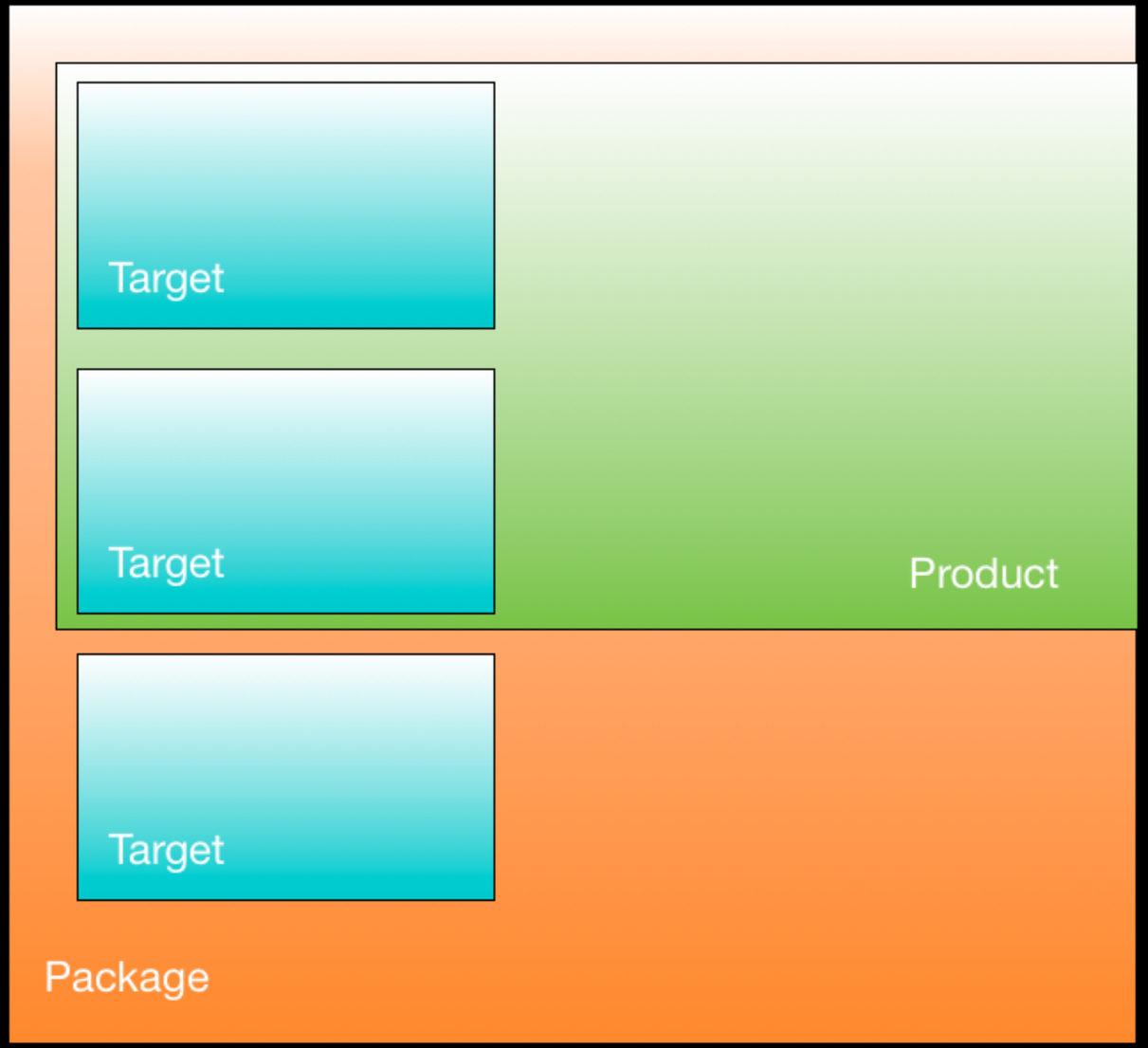
— A package can contain multiple Targets.

— These will contain a disjoint set of Source files from each other.

— Hopefully this will include your testTarget, which will depend on one of your other targets.

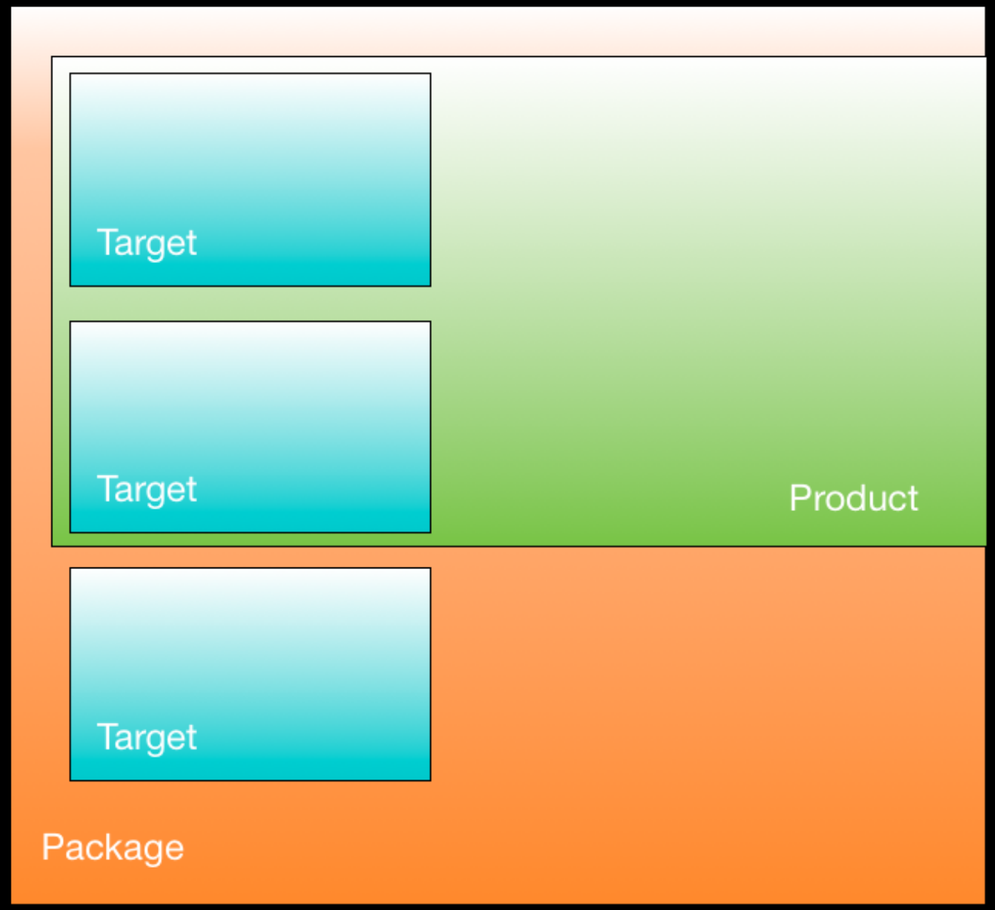
— You can also depend on native code on the system, a systemLibrary.

```
testTarget(
    name: String,
    dependencies: [PackageDescription.Target.Dependency] = default,
    path: String? = default,
    exclude: [String] = default,
    sources: [String]? = default
) -> PackageDescription.Target

systemLibrary(
    name: String,
    path: String? = default,
    pkgConfig: String? = default,
    providers: [PackageDescription.SystemPackageProvider]? = default
) -> PackageDescription.Target
```


Target
Target
Product
Target
Package

- You will group your targets into a final **Product**.
- This will be one of two types:
  - Library: code for other packages to consume
  - Executable: A tool run by end-users

```
library(
    name: String,
    type: PackageDescription.Product.Library.LibraryType? = default,
    targets: [String]
) -> PackageDescription.Product

executable(
    name: String,
    targets: [String]
) -> PackageDescription.Product
```



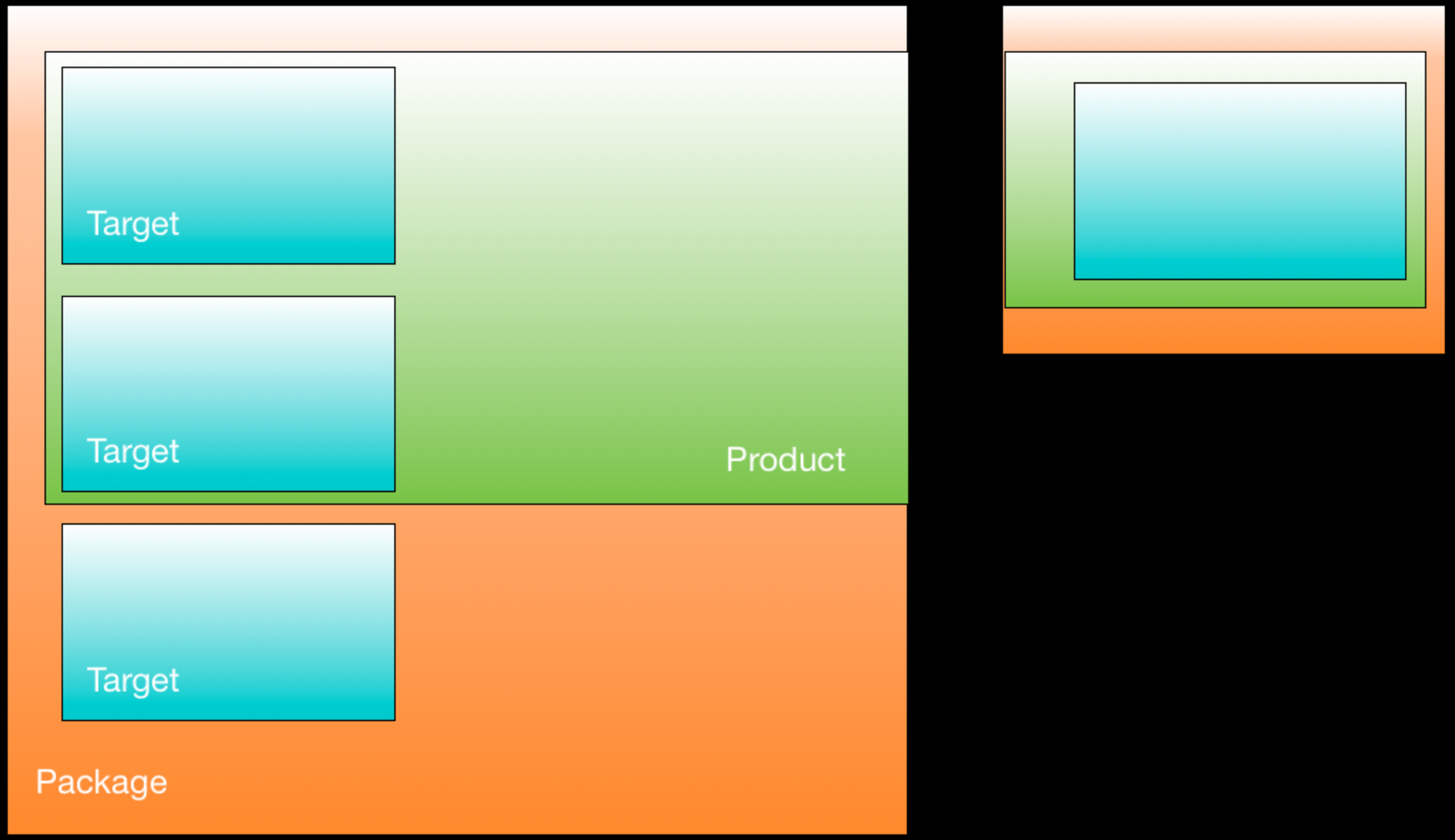

# This is (mostly) a Swift Package! 📦

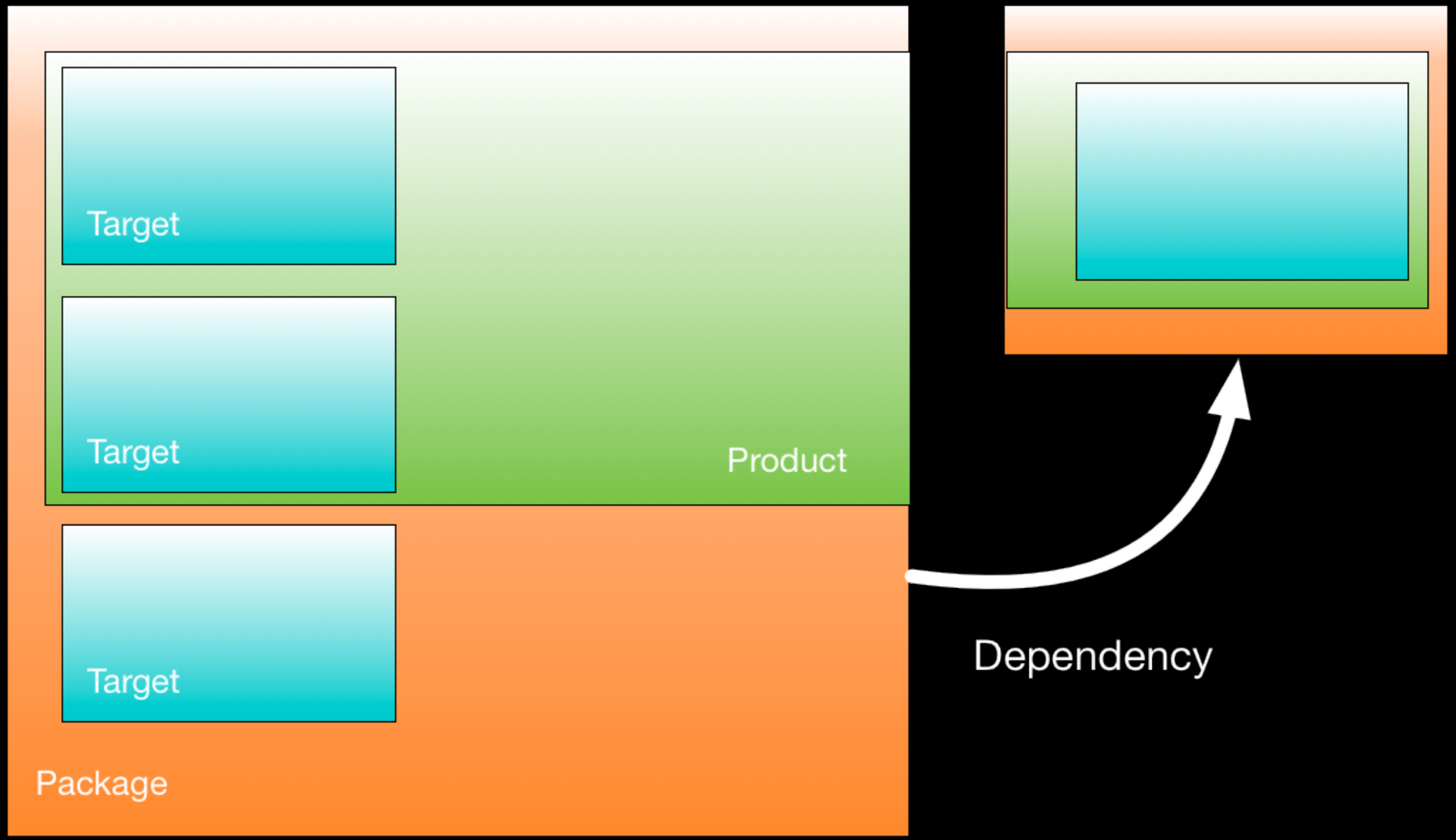
# Dependencies 📦📦📦

— The Swift Package Manager encourages code sharing and reuse.

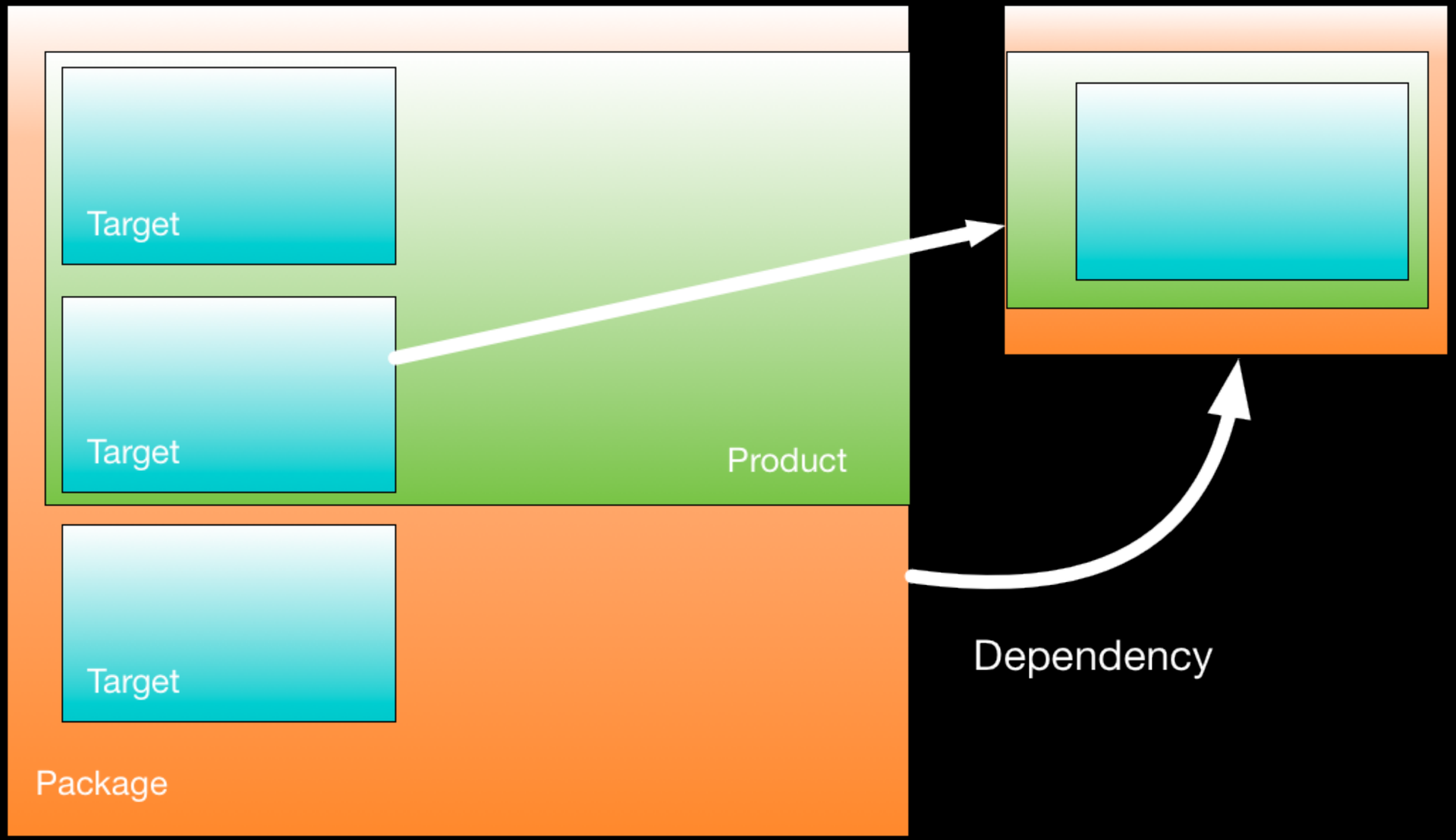
— Your Package needs to define what other packages it depends on.

— Your Target needs to specify what targets it depends on.


Target
Target
Target
Product
Package


Target
Target
Target
Product
Package
Dependency

```
package(
    url: String,
    from version: PackageDescription.Version
) -> PackageDescription.Package.Dependency

package(
    url: String,
    _ requirement: PackageDescription.Package.Dependency.Requirement
) -> PackageDescription.Package.Dependency

package(
    url: String,
    _ range: Range<PackageDescription.Version>
) -> PackageDescription.Package.Dependency

func package(
    url: String,
    _ range: ClosedRange<PackageDescription.Version>
) -> PackageDescription.Package.Dependency

/// Add a dependency to a local package on the filesystem.
package(
    path: String
) -> PackageDescription.Package.Dependency
```


Target
Target
Target
Product
Package
Dependency

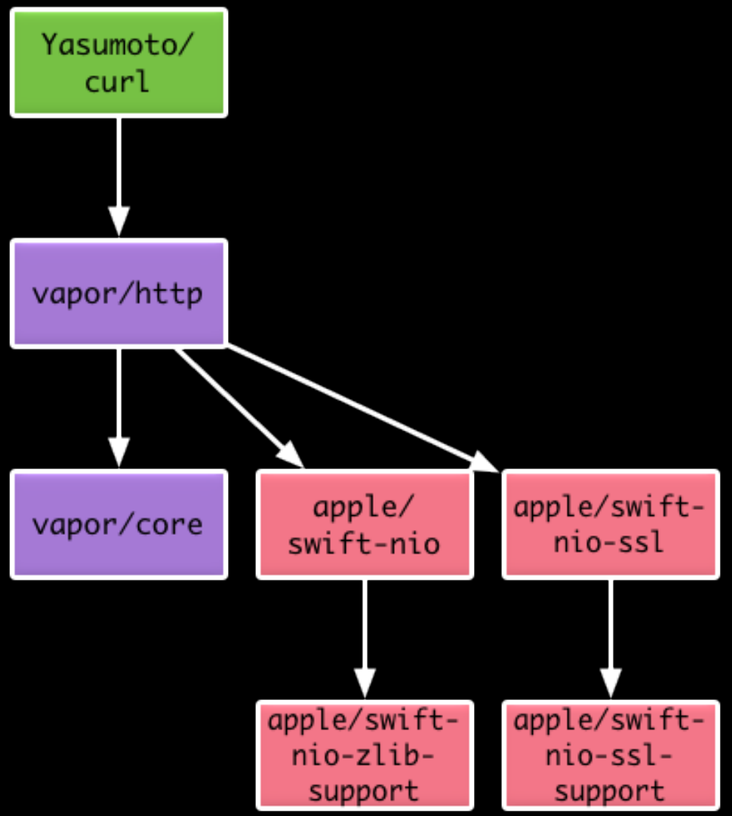
```
/// A dependency on a target in the same package.
target(
    name: String
) -> PackageDescription.Target.Dependency

/// A dependency on a product from a package dependency.
product(
    name: String,
    package: String? = default
) -> PackageDescription.Target.Dependency

byName(
    name: String
) -> PackageDescription.Target.Dependency

Target(stringLiteral value: String)
```

# Example

```
~/curl $ swift package init --type executable
Creating executable package: curl
Creating Package.swift
Creating README.md
Creating .gitignore
Creating Sources/
Creating Sources/curl/main.swift
Creating Tests/
Creating Tests/LinuxMain.swift
Creating Tests/curlTests/
Creating Tests/curlTests/curlTests.swift
Creating Tests/curlTests/XCTestManifests.swift
~/curl $
```

```
// swift-tools-version:4.2

import PackageDescription

let package = Package(
    name: "curl",
    dependencies: [
        .package(url: "https://github.com/vapor/http", from: "3.1.1"),
    ],
    targets: [
        .target(
            name: "curl",
            dependencies: [])]) // Wait for it
```

```
~/curl $ swift package update
Fetching https://github.com/vapor/http
Fetching https://github.com/vapor/core.git
Fetching https://github.com/apple/swift-nio.git
Fetching https://github.com/apple/swift-nio-ssl.git
Fetching https://github.com/apple/swift-nio-zlib-support.git
Fetching https://github.com/apple/swift-nio-ssl-support.git
Completed resolution in 6.91s
Cloning https://github.com/apple/swift-nio-ssl-support.git
Resolving https://github.com/apple/swift-nio-ssl-support.git at 1.0.0
Cloning https://github.com/vapor/core.git
Resolving https://github.com/vapor/core.git at 3.4.1
Cloning https://github.com/apple/swift-nio-ssl.git
Resolving https://github.com/apple/swift-nio-ssl.git at 1.2.0
Cloning https://github.com/vapor/http
Resolving https://github.com/vapor/http at 3.1.1
Cloning https://github.com/apple/swift-nio.git
Resolving https://github.com/apple/swift-nio.git at 1.9.2
Cloning https://github.com/apple/swift-nio-zlib-support.git
Resolving https://github.com/apple/swift-nio-zlib-support.git at 1.0.0
```


Yasumoto/
curl
vapor/http
vapor/core
apple/
swift-nio
apple/swift-
nio-ssl
apple/swift-
nio-zlib-
support
apple/swift-
nio-ssl-
support

— At this point, work with either Xcode or your favorite text editor.

— All of these commands will work on either Linux or Darwin.

```
~/curl $ swift package generate-xcodeproj
warning: dependency 'HTTP' is not used by any target
generated: ./curl.xcodeproj
```

# Example

```
// swift-tools-version:4.2

import PackageDescription

let package = Package(
    name: "curl",
    dependencies: [
        .package(url: "https://github.com/vapor/http", from: "3.1.1"),
    ],
    targets: [
        .target(
            name: "curl",
            dependencies: ["HTTP"])]) // 💥
```

```
~/curl $ swift package generate-xcodeproj
generated: ./curl.xcodeproj
```

```
import HTTP

struct CurlError: Error {}

struct SlackTest: Codable {
    let ok: Bool
}

let slack = "slack.com"
let test = "/api/api.test"
```

```
let worker = MultiThreadedEventLoopGroup(numberOfThreads: 1)

let request = HTTPRequest(
    method: .GET,
    url: test)
let client = try HTTPClient.connect(
    scheme: .https,
    hostname: slack,
    port: 443,
    on: worker).wait()
let response = try client.send(request).wait()
```

```
let decoder = JSONDecoder()
guard let data = response.body.data else {
    throw CurlError()
}
let test = try decoder.decode(SlackTest.self, from: data)
if test.ok {
    print("Slack is up! 🎉")
} else {
    print("Slack is down!!! 📟")
}
```

```
~/curl $ swift build
Compile CNIOSHA1 c_nio_sha1.c
Compile CNIOLinux shim.c
Compile CNIODarwin shim.c
Compile CNIOOpenSSL empty.c
Compile CNIOZlib empty.c
Compile CNIOHTTPParser c_nio_http_parser.c
Compile CNIOAtomics src/c-atomics.c
Compile Swift Module 'NIOPriorityQueue' (2 sources)
Compile Swift Module 'Debugging' (3 sources)
Compile Swift Module 'COperatingSystem' (1 sources)
Compile Swift Module 'NIOConcurrencyHelpers' (2 sources)
Compile Swift Module 'NIO' (52 sources)
Compile Swift Module 'NIOTLS' (3 sources)
Compile Swift Module 'NIOHTTP1' (8 sources)
Compile Swift Module 'Async' (15 sources)
Compile Swift Module 'NIOFoundationCompat' (1 sources)
Compile Swift Module 'Bits' (12 sources)
Compile Swift Module 'NIOOpenSSL' (13 sources)
Compile Swift Module 'Core' (25 sources)
Compile Swift Module 'HTTP' (25 sources)
Compile Swift Module 'curl' (1 sources)
Linking ./.build/x86_64-apple-macosx10.10/debug/curl
```

```
~/curl $ ./.build/x86_64-apple-macosx10.10/debug/curl
Slack is up! 🎉
```

# 🆕 New in Swift 4.2

# "Limited Terminal" Support [1]

— Simple progress bars for terminals without escape sequence support!
— Helpful for folks who use things like `eshell` in emacs.

[1] https://github.com/apple/swift-package-manager/pull/1489

# Package Version Parsing[2]

— Packages which mix versions of the form vX.X.X with Y.Y.Y will now be parsed and ordered numerically.

[2] https://bugs.swift.org/browse/SR-6978

# Much better scheme generation

— One scheme containing all regular and test targets of the root package.

— One scheme per executable target containing the test targets whose dependencies intersect with the dependencies of the exectuable target.

# Automatic Xcode Project Regeneration[3]

— The generate-xcodeproj command has a new --watch option.

— This will automatically regenerate the Xcode project if changes are detected.

— This uses the watchman[4] tool to detect filesystem changes.

[3] https://github.com/apple/swift-package-manager/pull/1604

[4] https://facebook.github.io/watchman/docs/install.html

# Local Dependencies

— SE-201[5]

— Packages can now specify a dependency as package(path: String) to point to a path on the local filesystem which hosts a package.

— This will enable interconnected projects to be edited in parallel.

[5] https://github.com/apple/swift-evolution/blob/master/proposals/0201-package-manager-local-dependencies.md

# System Library Targets

— SE-208[6]

— The Package manifest now accepts a new type of target, systemLibrary.

— This deprecates "system-module packages" which are now to be included in the packages that require system-installed dependencies.

[6] https://github.com/apple/swift-evolution/blob/master/proposals/0208-package-manager-system-library-targets.md

# SwiftVersion Enum

— SE-209[7]

— The swiftLanguageVersions property no longer takes its Swift language versions via a freeform Integer array.

— Instead it should be passed as a new SwiftVersion enum array.

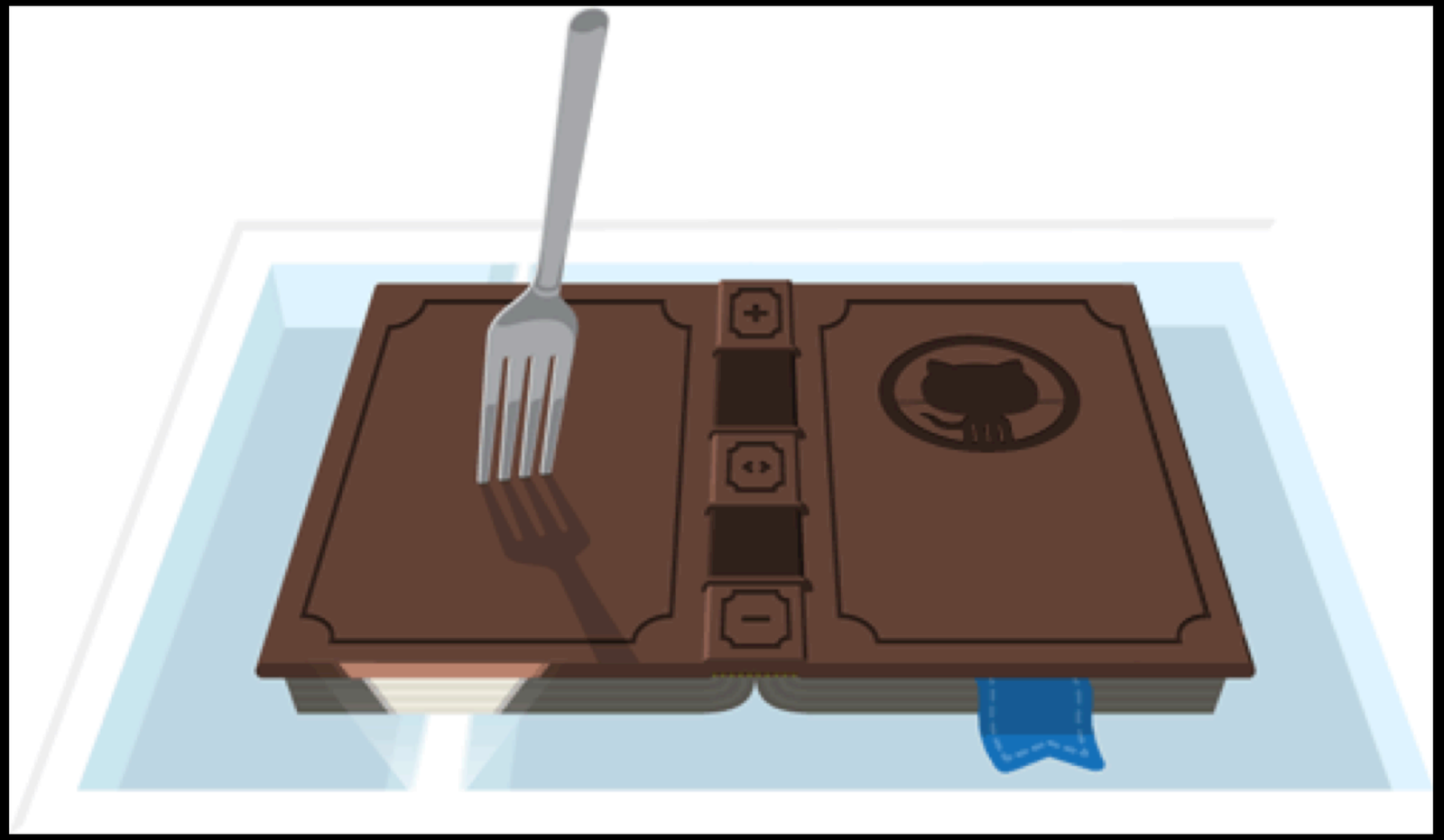
[7] https://github.com/apple/swift-evolution/blob/master/proposals/0209-package-manager-swift-lang-version-update.md

# Lots of Upgrades

— Many bugfixes
— Documentation improvements
— Better diagnostic messages

# Swift 4.2 Demo 🤞

# Contributing!



# Starter Bugs 

— Please feel empowered to pick up any of these!
— Ask questions on the thread or the Swift PM Slack workspace
— Post a PR and for more comments and iteration, then ship!
— StarterBug label on `bugs.swift.org`

# Starter Bugs 🐛

SR-7825

— SwiftPM should consider a target with header files but no sources as a `ClangTarget`.

— https://bugs.swift.org/browse/SR-7825

# Starter Bugs 

SR-7979

— Circular Dependency in SwiftPM Causes Segfault
— Package Foo depends on Bar, yet Bar depends on Foo
— https://bugs.swift.org/browse/SR-7979

# Starter Bugs 

SR-8204

— Sort targets in SwiftPM generated Xcode project
— @hartbit has a suggestion in-thread
— https://bugs.swift.org/browse/SR-8204

# Starter Bugs 🐛

SR-8645 and SR-8646

— Very new!
— BuildPlan Error description improvements
— https://bugs.swift.org/browse/SR-8645
— https://bugs.swift.org/browse/SR-8645

# Evolution Ideas 📜

Thank you Ankit![8]


[8] https://github.com/apple/swift-package-manager/blob/master/Documentation/EvolutionIdeas.md

# Evolution Ideas 📜

SR-3948

— Define specific "build settings" or "language/linker flags" in the Package manifest.

— https://bugs.swift.org/browse/SR-3948

# Evolution Ideas 📜

SR-883

— Conditional Dependencies

— Allow packages to download dependencies only when testing, or only on certain platforms.

— Remove the current "conditional" option in favor of a declarative model.

— https://bugs.swift.org/browse/SR-883

# Evolution Ideas 📜

SR-2866

— Resource Support
— Packages need to be able to include other files with themselves.
— https://bugs.swift.org/browse/SR-2866

# Evolution Ideas 📜

SR-7837

— Custom `swift package init` template support

— https://bugs.swift.org/browse/SR-7837

# Evolution Ideas 📜

— Documentation Generation Support

— Use SourceKit to extract docstrings and pull out documentation for the package.

# Evolution Ideas 📜

— Tagging and Publishing support
— Add guardrails and better workflow for cutting and releasing new Package versions

# Evolution Ideas 📜

SR-3951

— Multi-Package Repository Support
— Good for folks who live in a monorepo, or want to keep packages in-sync across revisions.
— https://bugs.swift.org/browse/SR-3951

# Evolution Ideas 📜

— Cross-platform Sandboxing
— Machine-Editable `Package.swift`
— Automatic Semantic Versioning

# Thank you![9]

— Joe Smith, @Yasumoto
— https://slack.com/jobs
— SwiftPM Slack
    — https://swift-package-manager.herokuapp.com
— Vapor Discord
    — https://discordapp.com/invite/vapor

[9] https://github.com/Yasumoto/talks/tree/master/2018/08/28