#WWDC18

Getting to Know Swift Package Manager

Session 411

Rick Ballard, SwiftPM Release Manager
Boris Buegling, Developer Tools Engineer

Why a package manager for Swift?

How to use it

The design of SwiftPM

Evolution ideas

Open source process

Why a package manager for Swift?

How to use it

The design of SwiftPM

Evolution ideas

Open source process

A Cross-Platform Build System for Swift

A Cross-Platform Build System for Swift

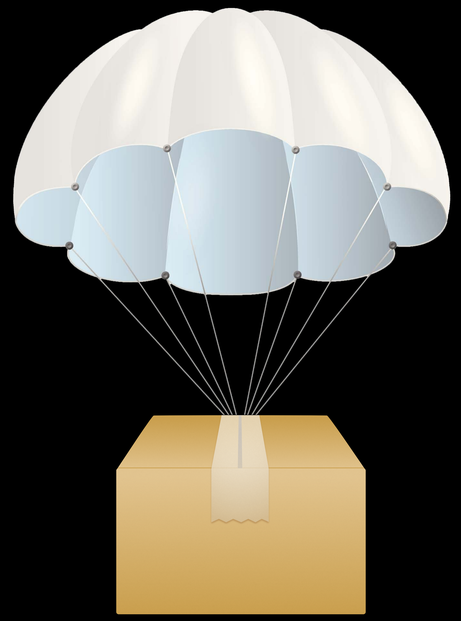
mac OS

Ubuntu

Future Platforms

Canonical Package Management Tool

Canonical Package Management Tool



mac OS

Ubuntu

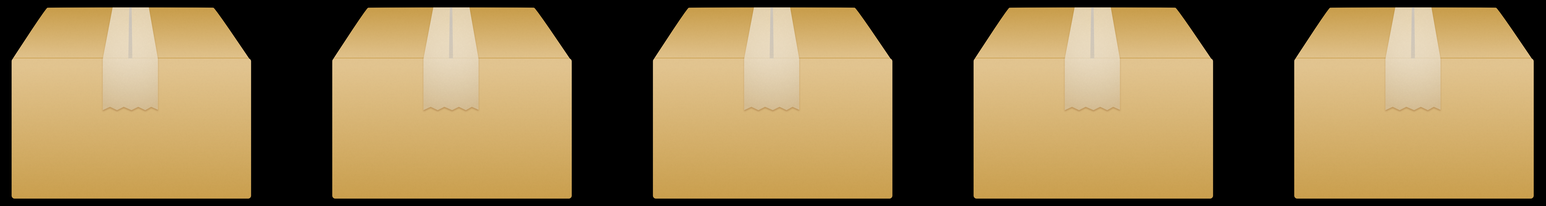
Future Platforms

Code Reuse Beyond the Core Libraries

 Foundation  Dispatch  XCTest

 stdlib  system

Code Reuse Beyond the Core Libraries



Foundation

Dispatch

XCTest

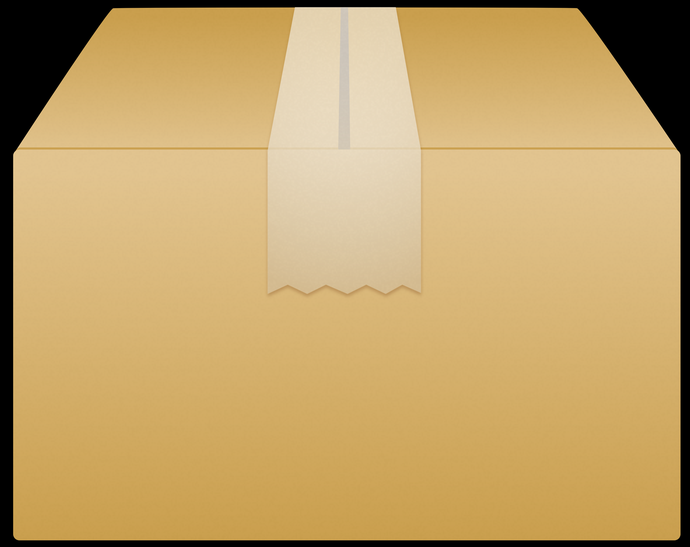
stdlib

system

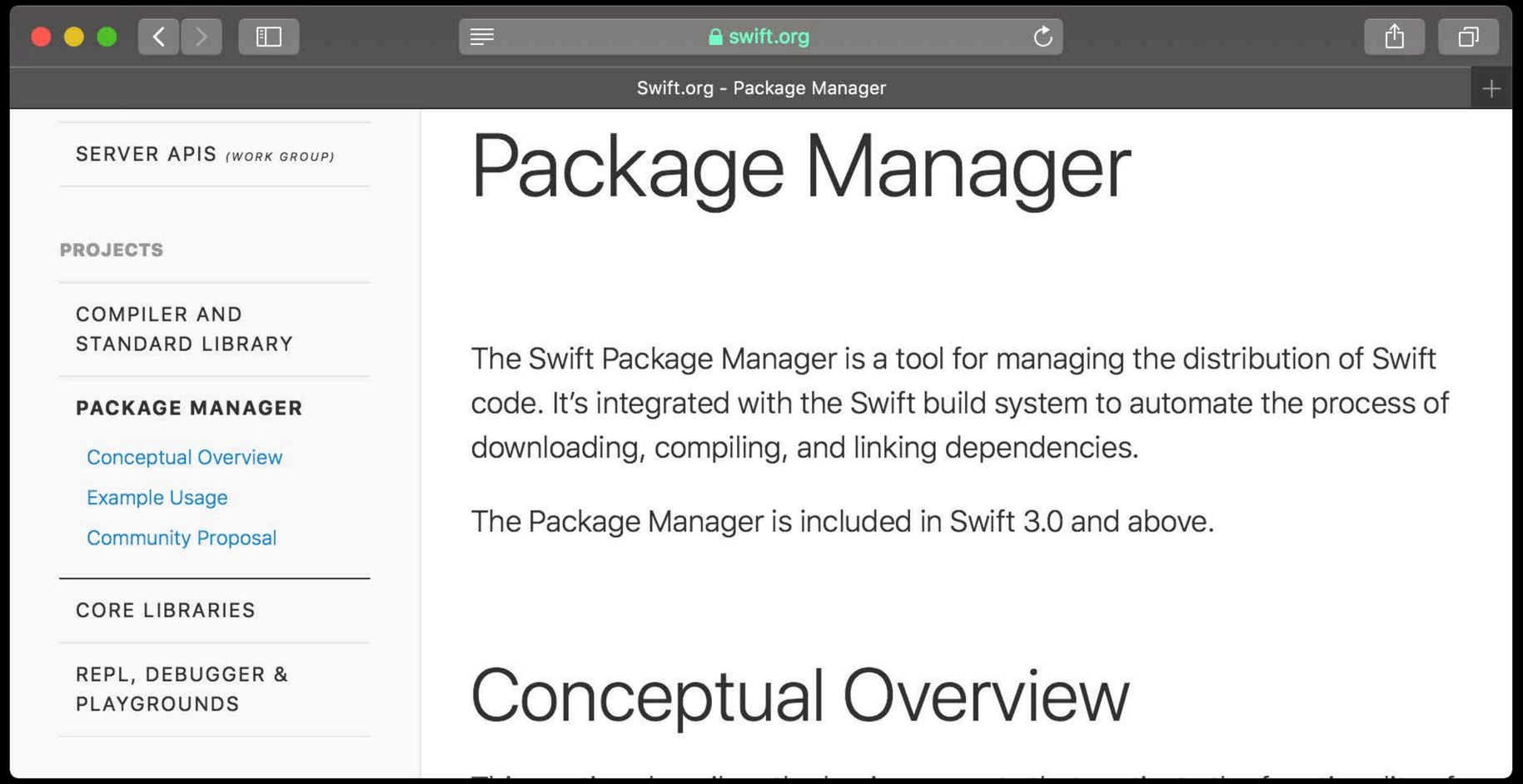
Take Full Advantage of the Power of Swift

Take Full Advantage of the Power of Swift

+

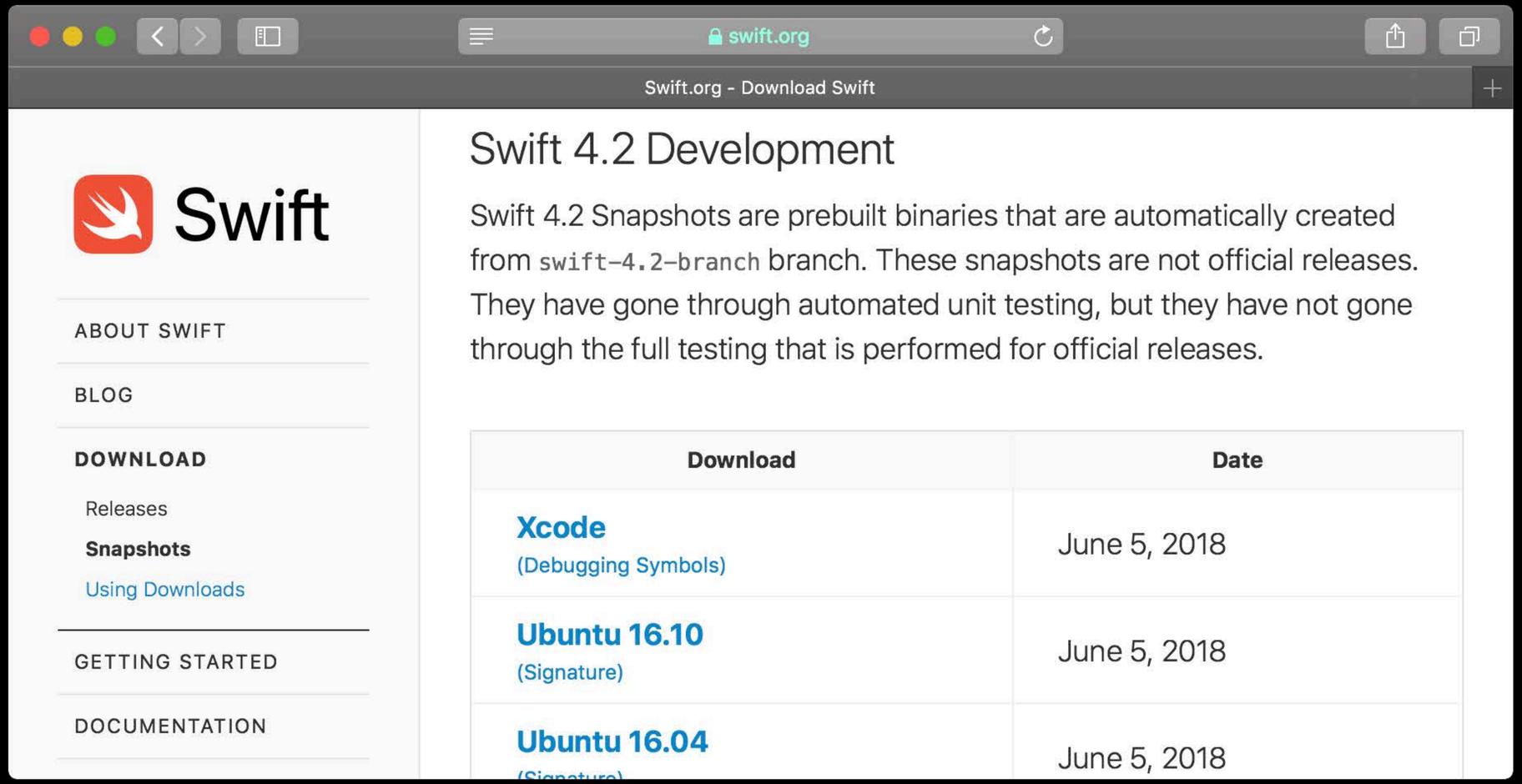


Part of Swift Open Source Project



https://swift.org/package-manager/

Included in Swift Toolchains



https://swift.org/download/

Included with Xcode

Why a package manager for Swift?

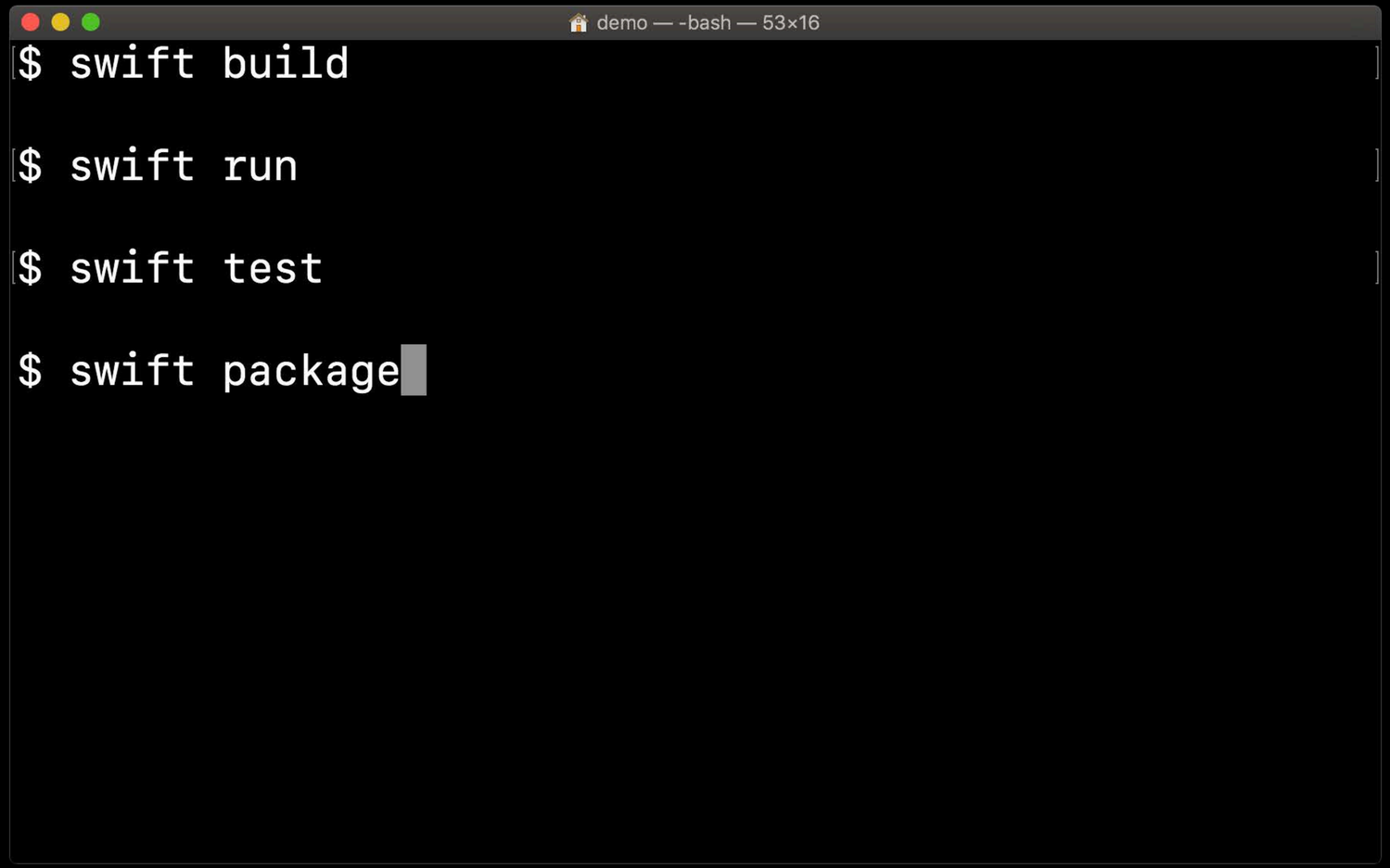
How to use it

The design of SwiftPM

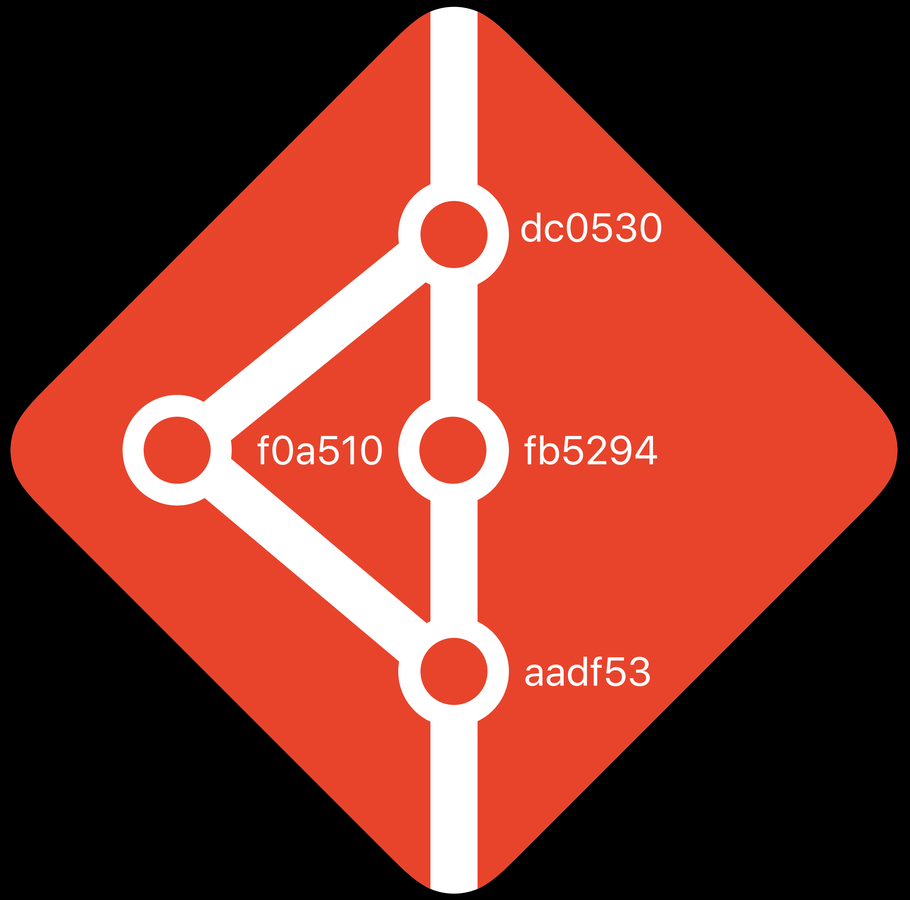
Evolution ideas

Open source process

SwiftPM Commands



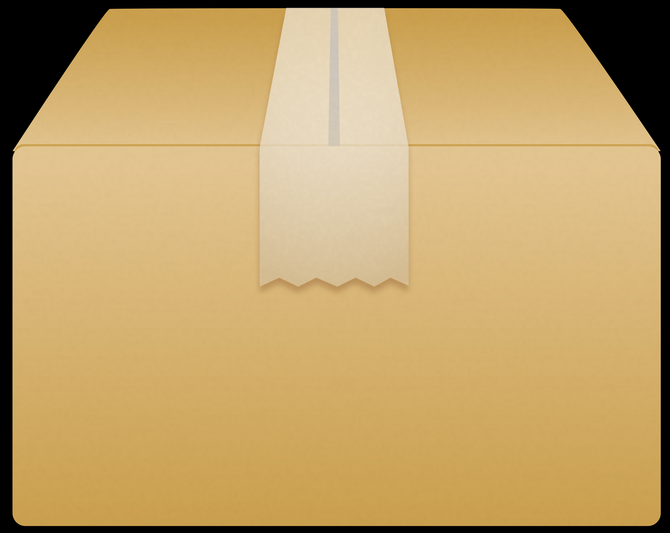
Packages Use Git



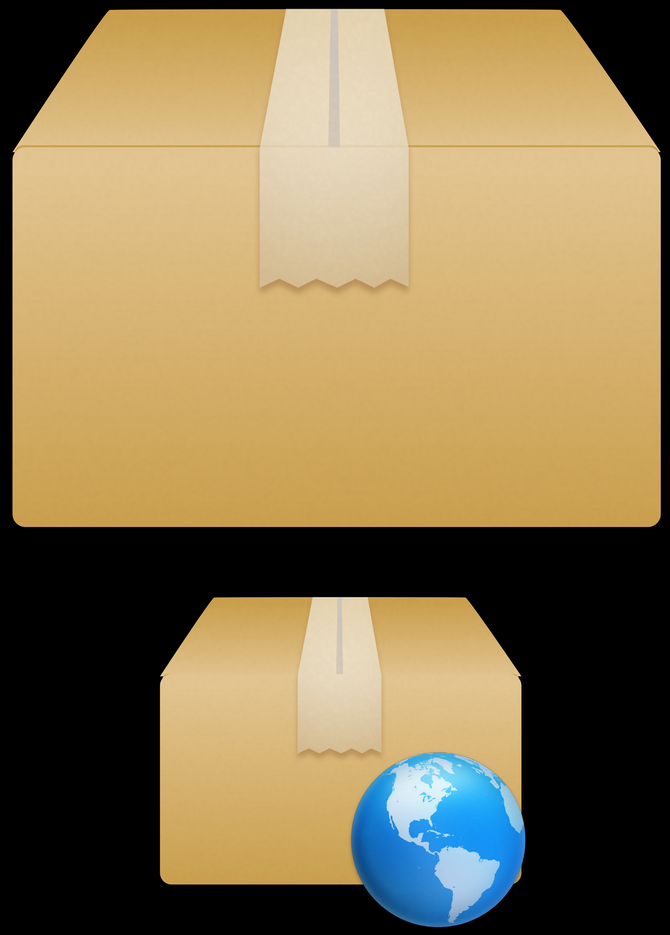
Demo

Creating your first package

Anatomy of a Package



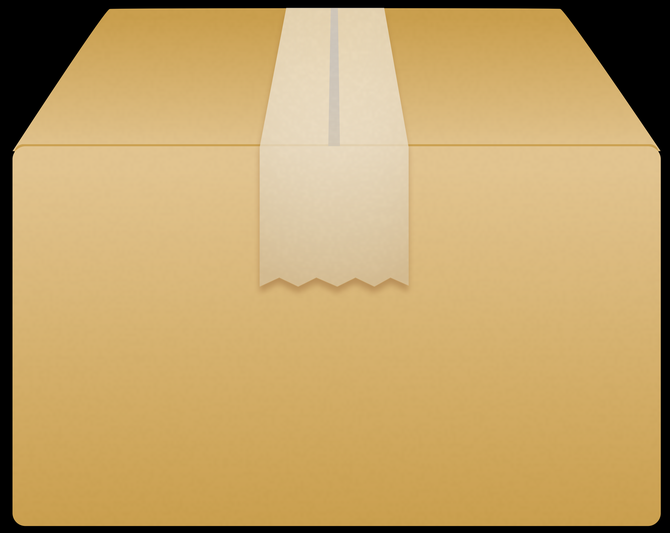
Anatomy of a Package

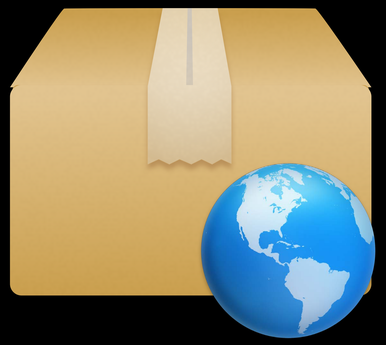


Dependencies

Anatomy of a Package

Targets

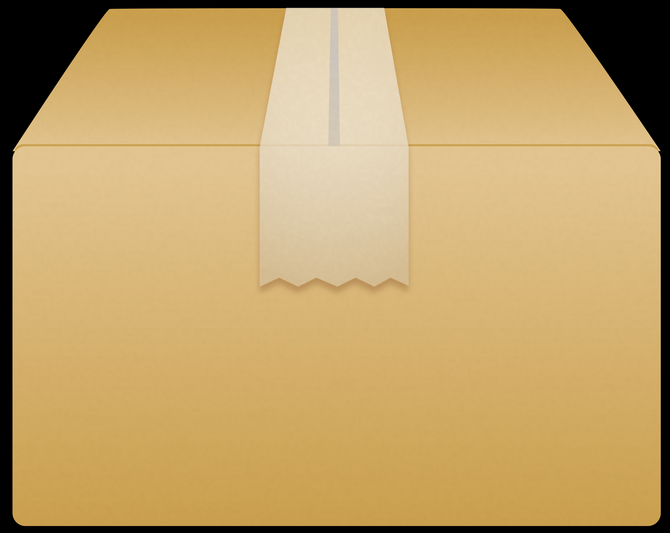


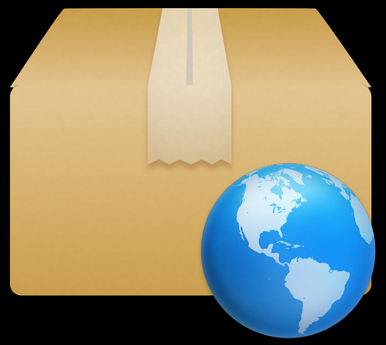


Dependencies

Anatomy of a Package

Targets

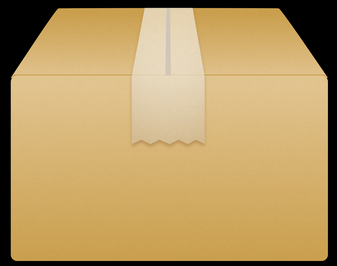




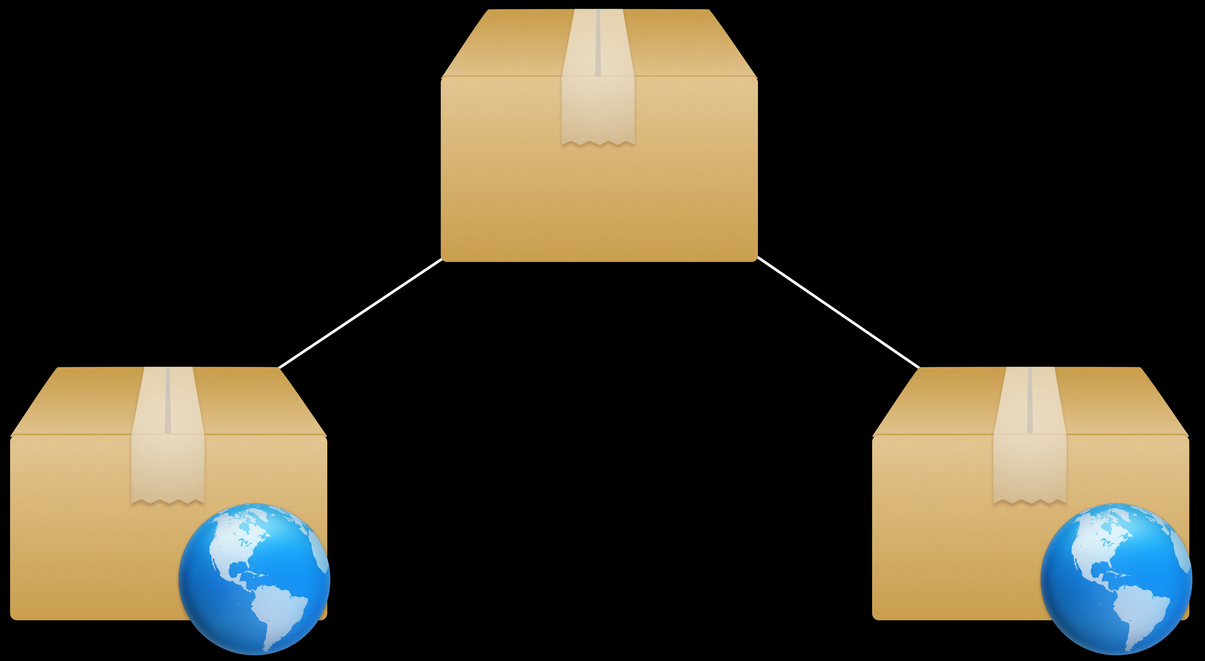
Dependencies

Products

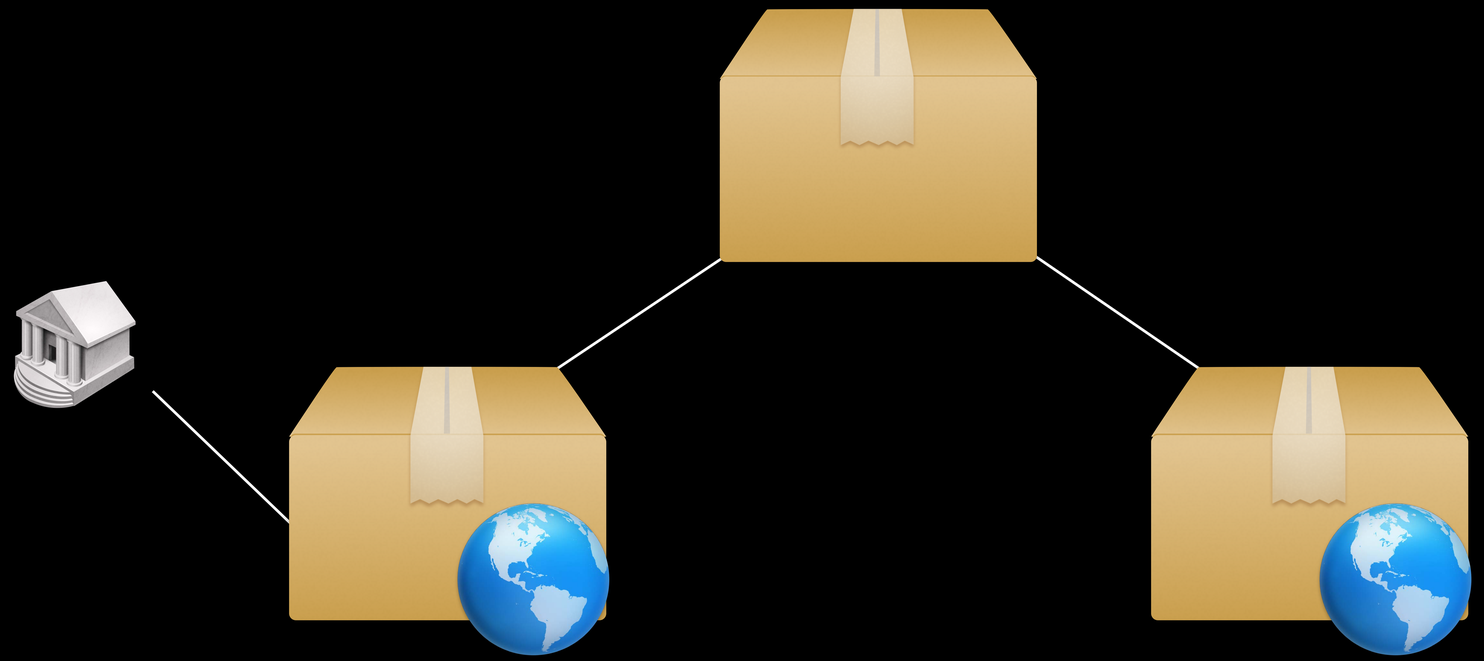
Dependencies



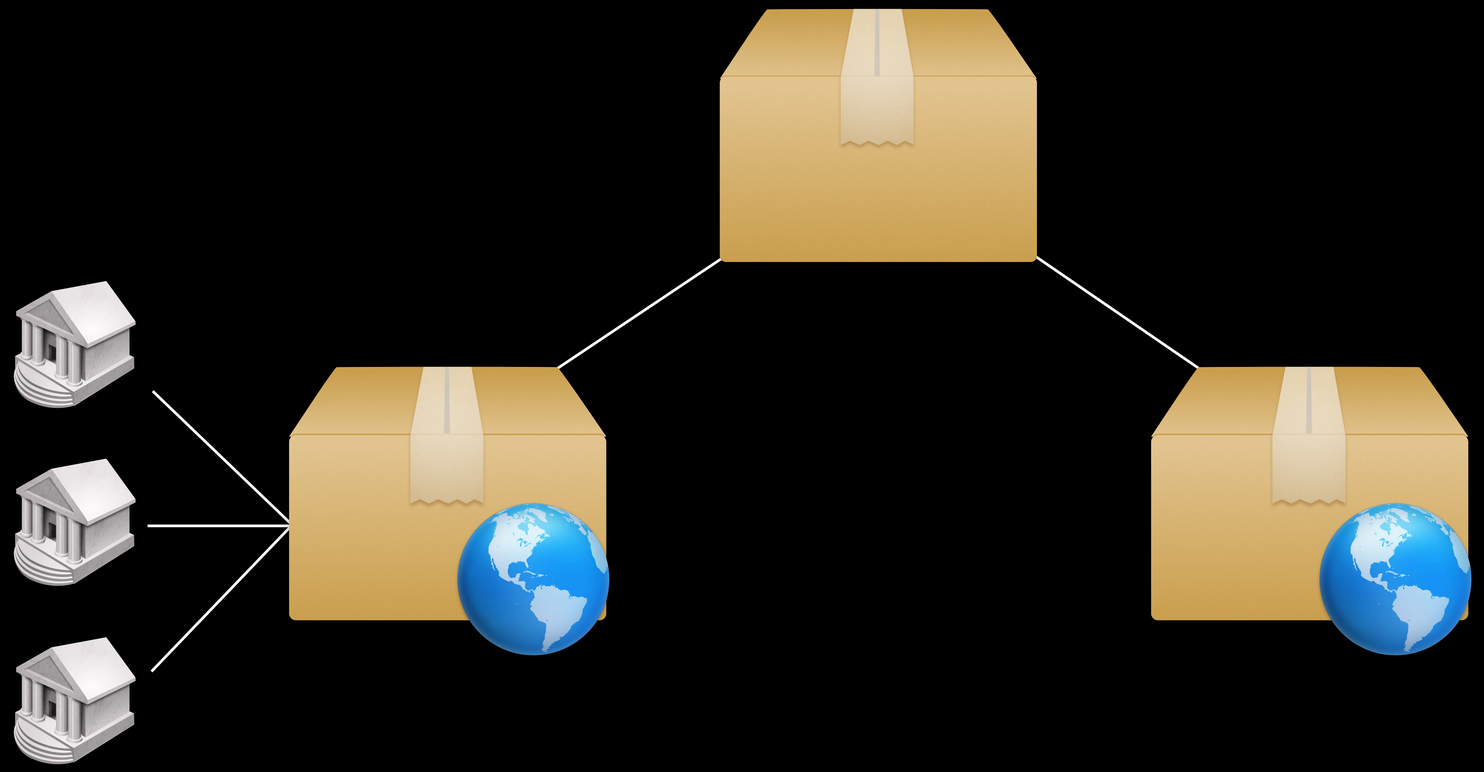
Dependencies



Dependencies



Dependencies



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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

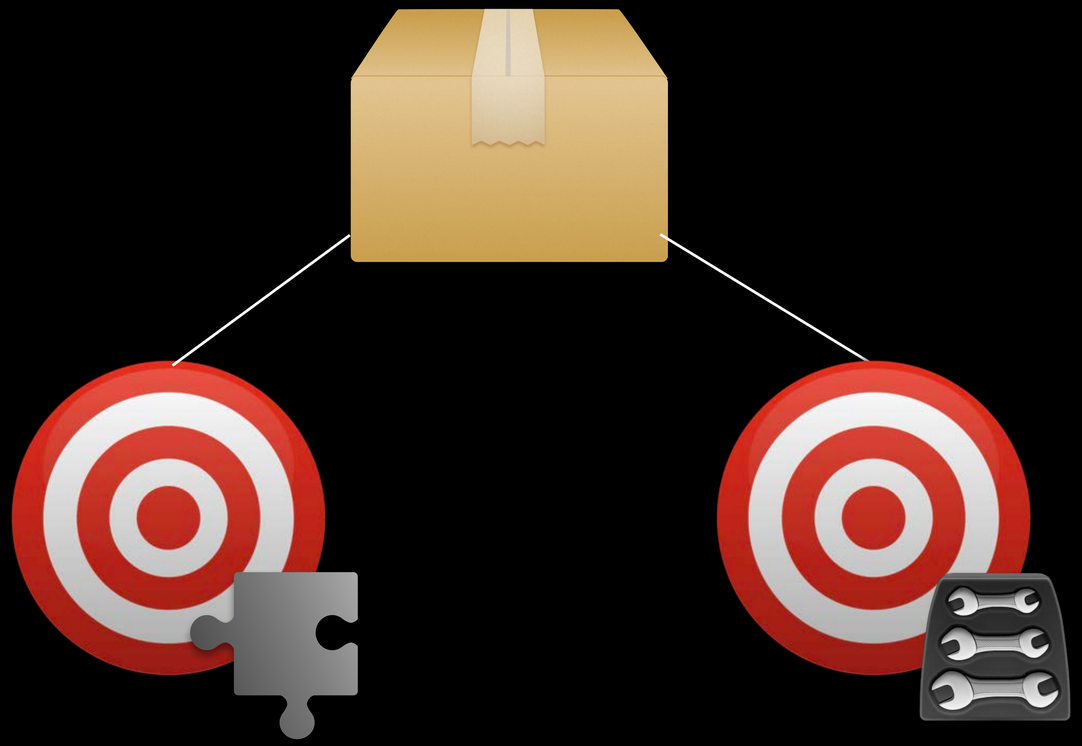
```
// swift-tools-version:4.2
import PackageDescription

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

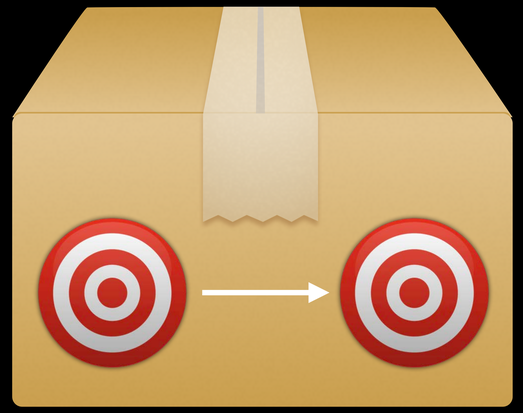
```
// swift-tools-version:4.2
import PackageDescription

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

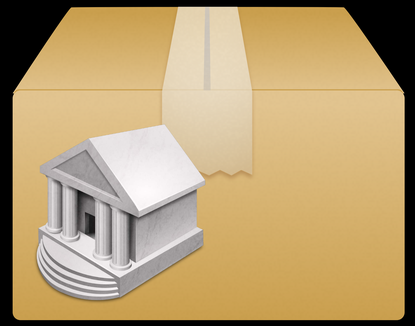
```
// swift-tools-version:4.2
import PackageDescription

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

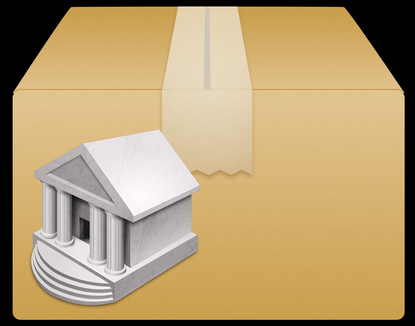
Targets



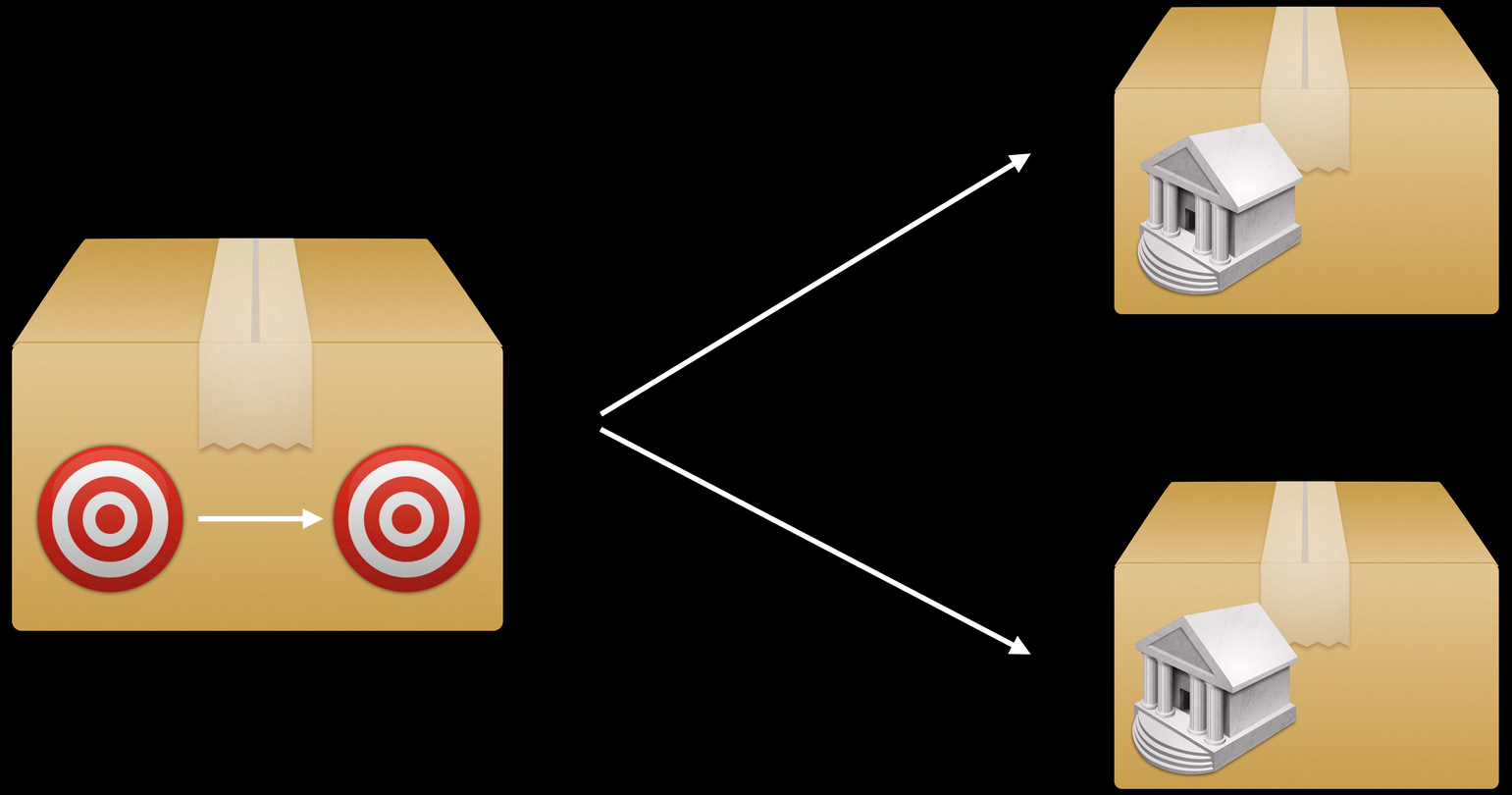
Targets Depend on Other Targets and Products





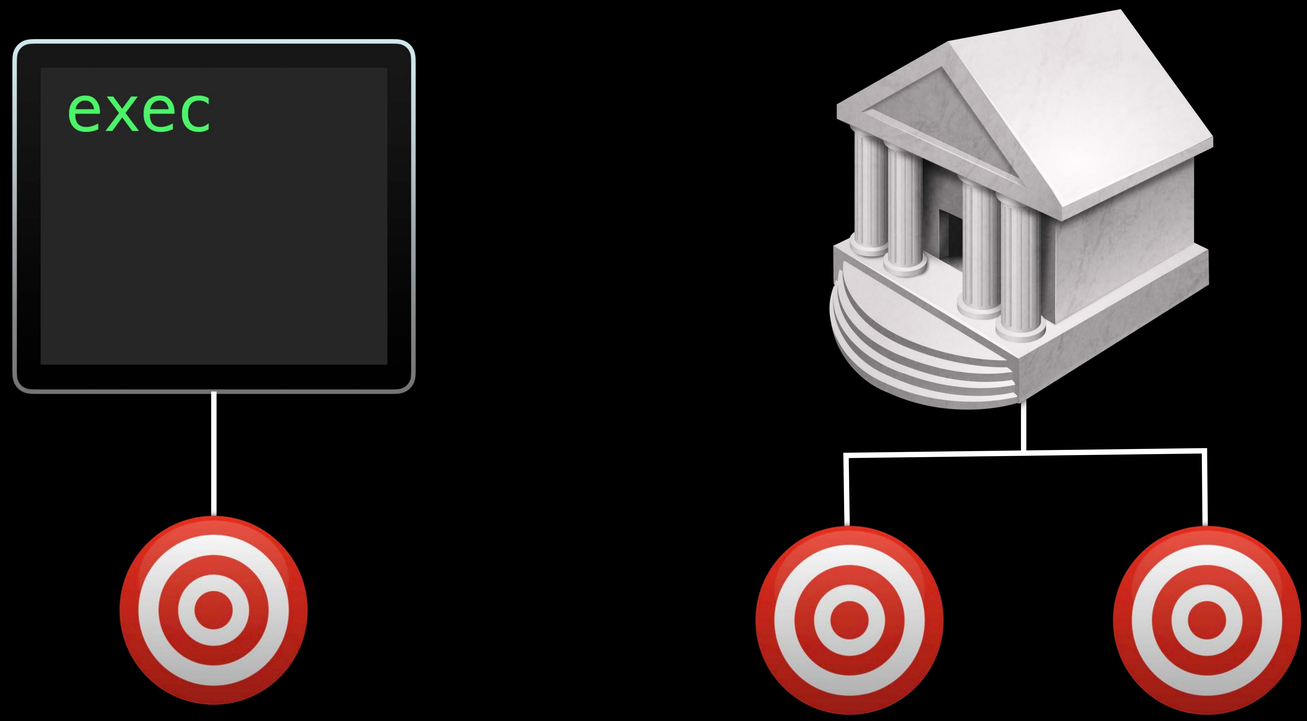


Targets Depend on Other Targets and Products

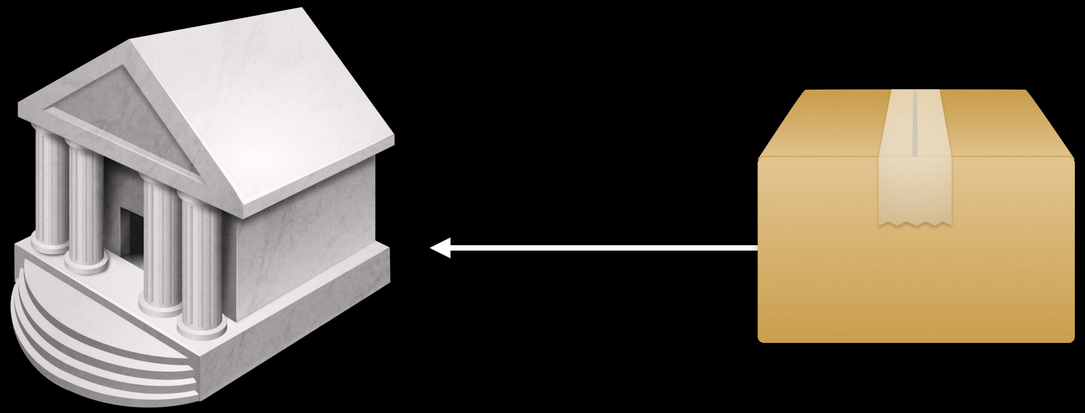


Products

Products



Packages Provide Products



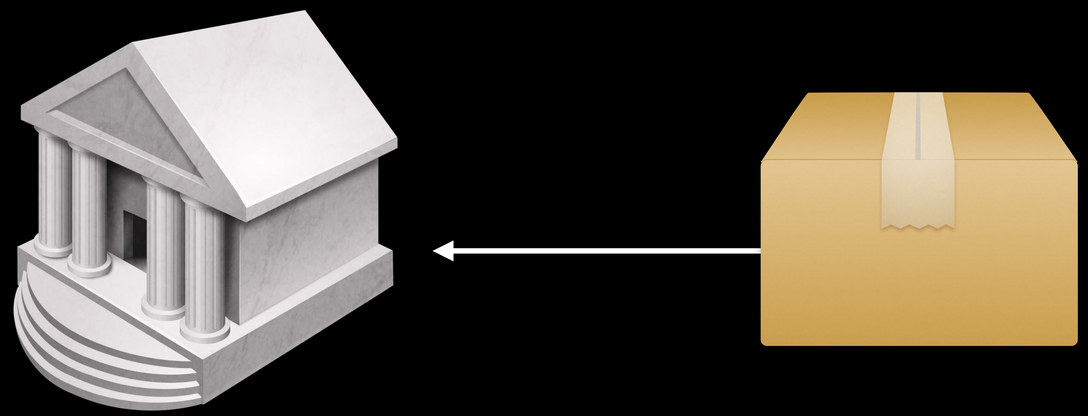
Packages Provide Products

Linkage:

Automatic

Static

Dynamic



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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

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

let package = Package(
    name: "dealer",
    products: [
      .library(name: "libdealer", targets: ["libdealer"]),
      .executable(name: "dealer", targets: ["dealer"]),
    ],
    dependencies: [
      .package(url: "git@github.com:apple/example-package-deckofplayingcards", from: "3.0.0")
    ],
    targets: [
      .target(name: "libdealer", dependencies: ["DeckOfPlayingCards"]),
      .target(name: "dealer", dependencies: ["libdealer"]),
      .testTarget(name: "dealerTests", dependencies: ["libdealer", "dealer"]),
    ]
)
```

Demo

Adding dependencies

Why a package manager for Swift?

How to use It

The design of SwiftPM

Evolution ideas

Open source process

Following Swift's Philosophy

Safe

Fast

Expressive

Following Swift's Philosophy

Safe: isolated build environment

Fast

Expressive

Following Swift's Philosophy

Safe: isolated build environment

Fast: scalable to large dependency graphs

Expressive

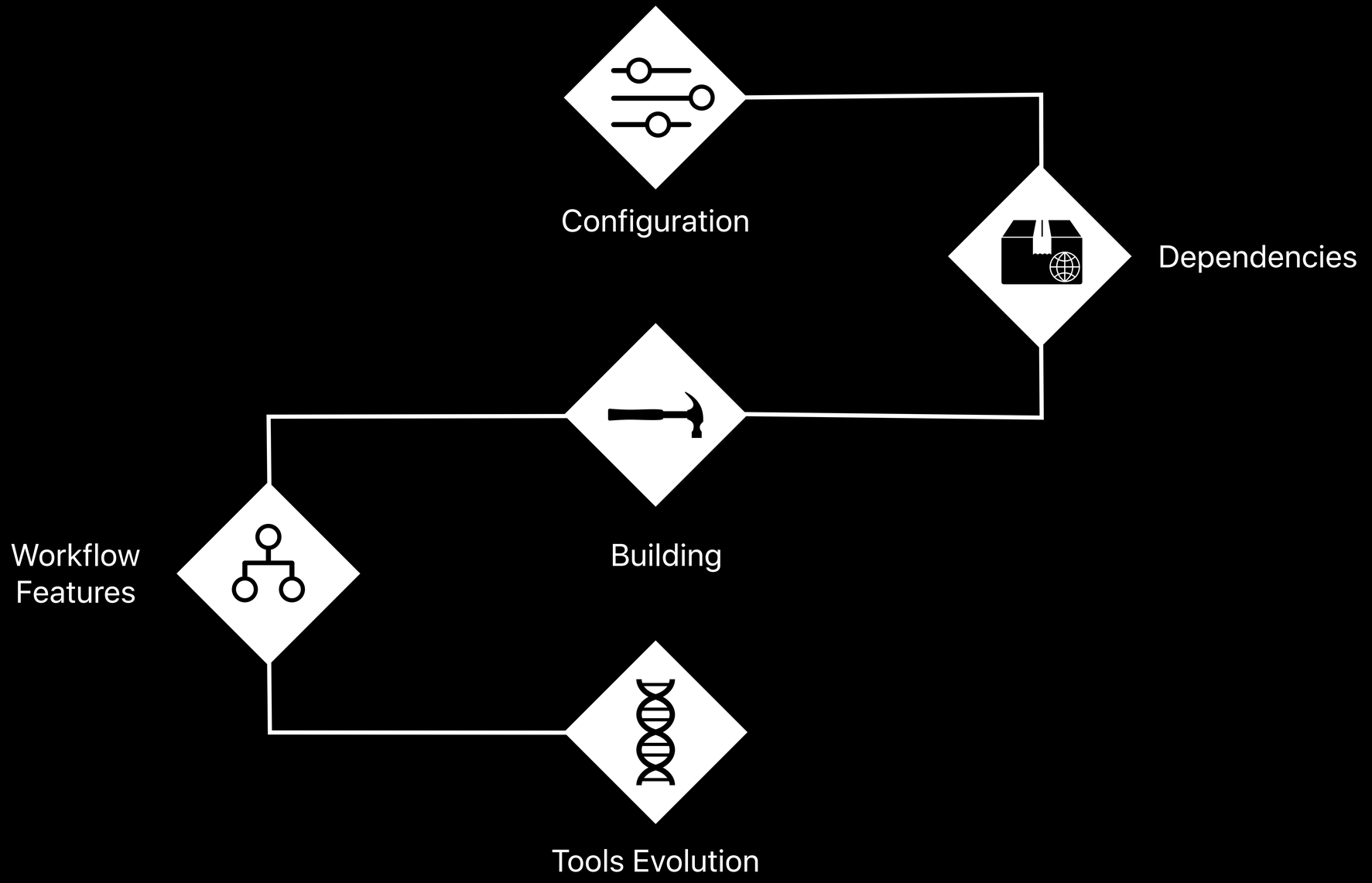
Following Swift's Philosophy

Safe: isolated build environment

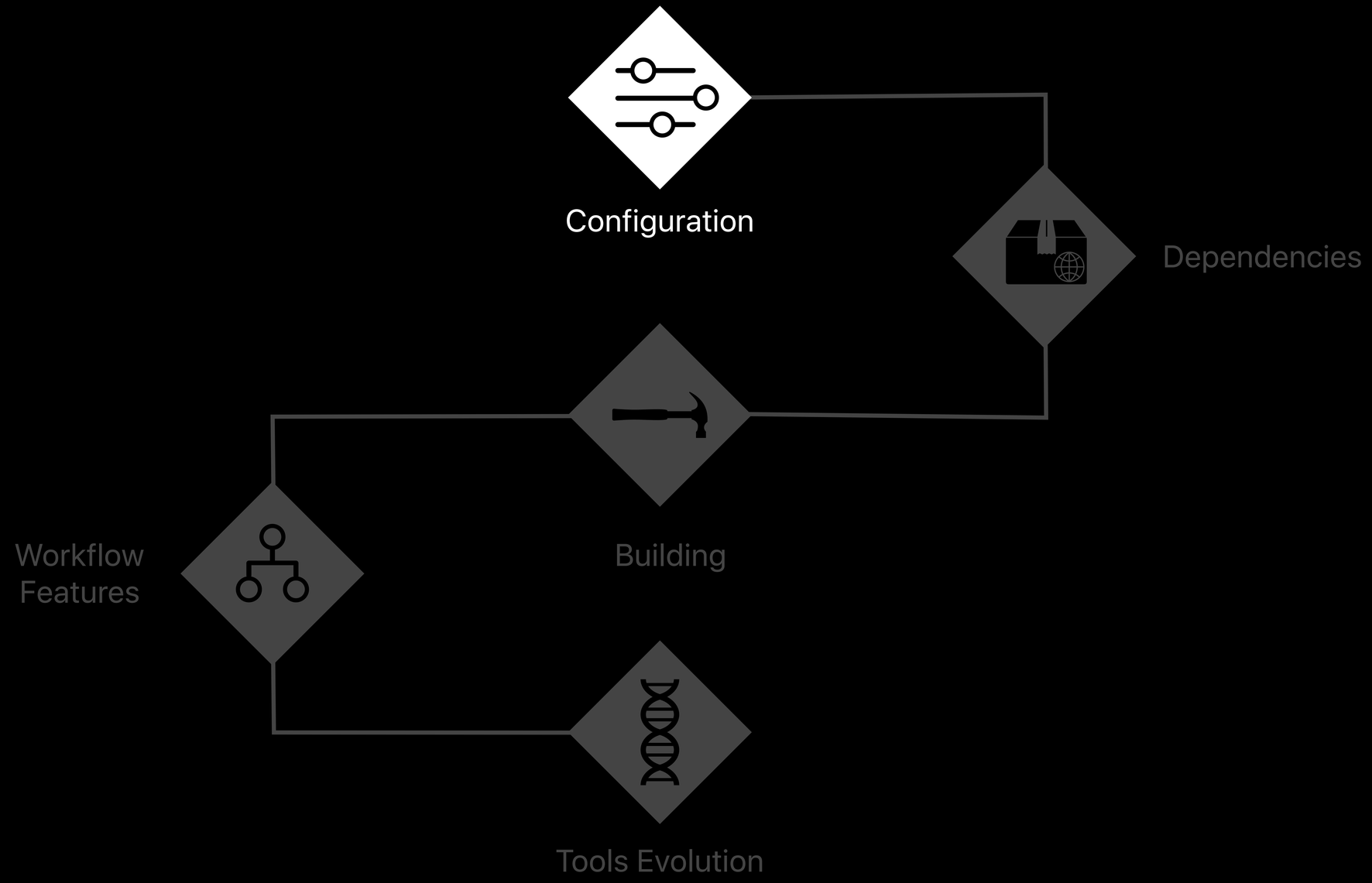
Fast: scalable to large dependency graphs

Expressive: Swift language manifest format

The Design of SwiftPM



The Design of SwiftPM



Swift Language Manifest Format

Easy to understand

Follows Swift API design guidelines

Supported by existing Swift tools

Prefer Declarative Syntax

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

let name = generateName()
let package = Package(
  name: name,
  targets: [
    .target(
      name: name,
      dependencies: []),
    .testTarget(
      name: "\(name)Tests",
      dependencies: [.target(name: name)])
])
```

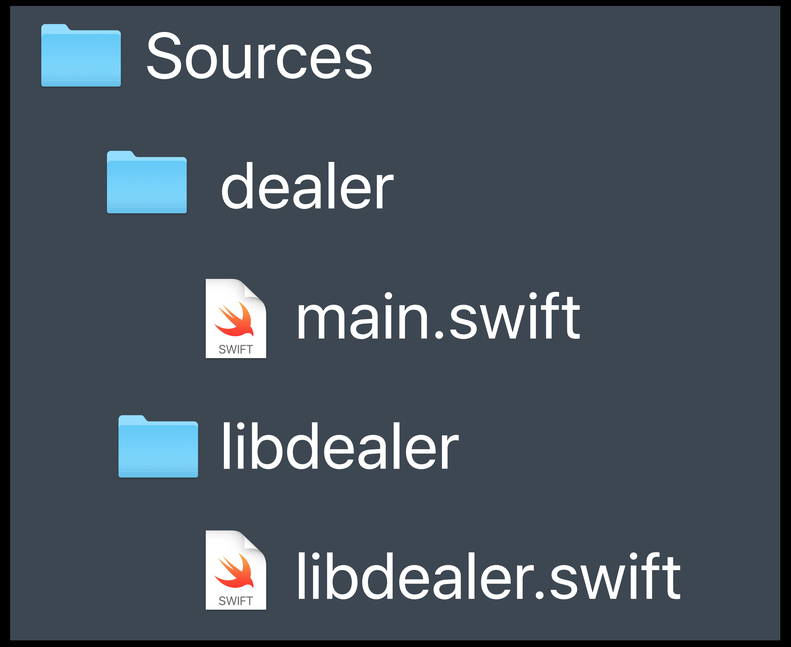
```
// swift-tools-version:4.2
import PackageDescription

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

Specifying Package Source Files

Convention versus Configuration


```
…
products: [
  .library(name: "libdealer",
           targets: ["libdealer"]),
…
targets: [
  .target(name: "libdealer",
          dependencies: ["DeckOfPlayingCards"]),
…
```

Convention versus Configuration

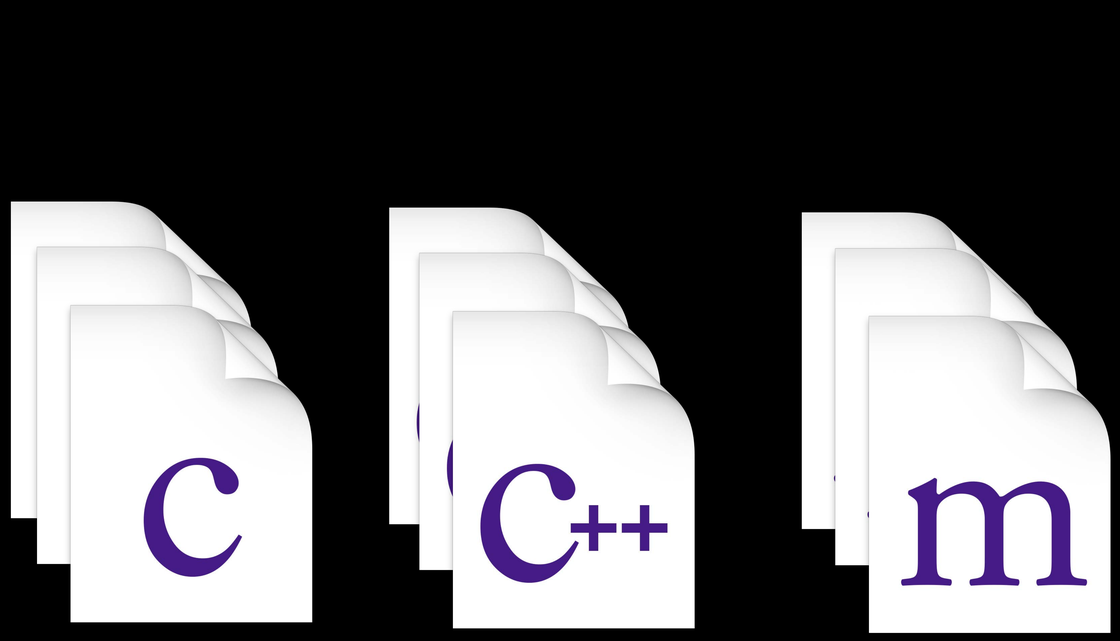



```
…
products: [
  .library(name: "libdealer",
           targets: ["libdealer"]),
…
targets: [
  .target(name: "libdealer",
          dependencies: ["DeckOfPlayingCards"]),
…
```

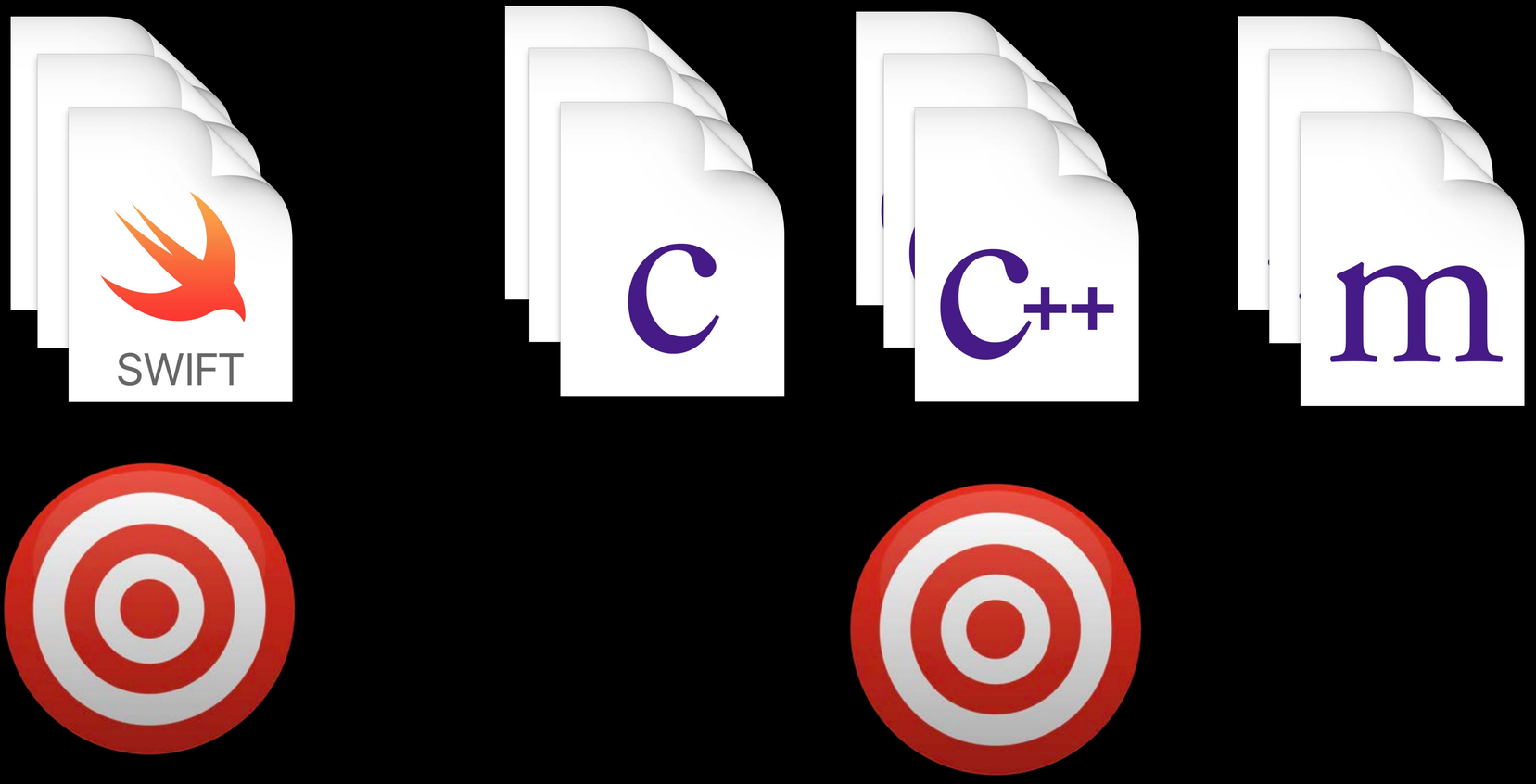
Convention versus Configuration


```
…
products: [
  .library(name: "libdealer",
           targets: ["libdealer"]),
…
targets: [
  .target(name: "libdealer",
          dependencies: ["DeckOfPlayingCards"]),
…
```

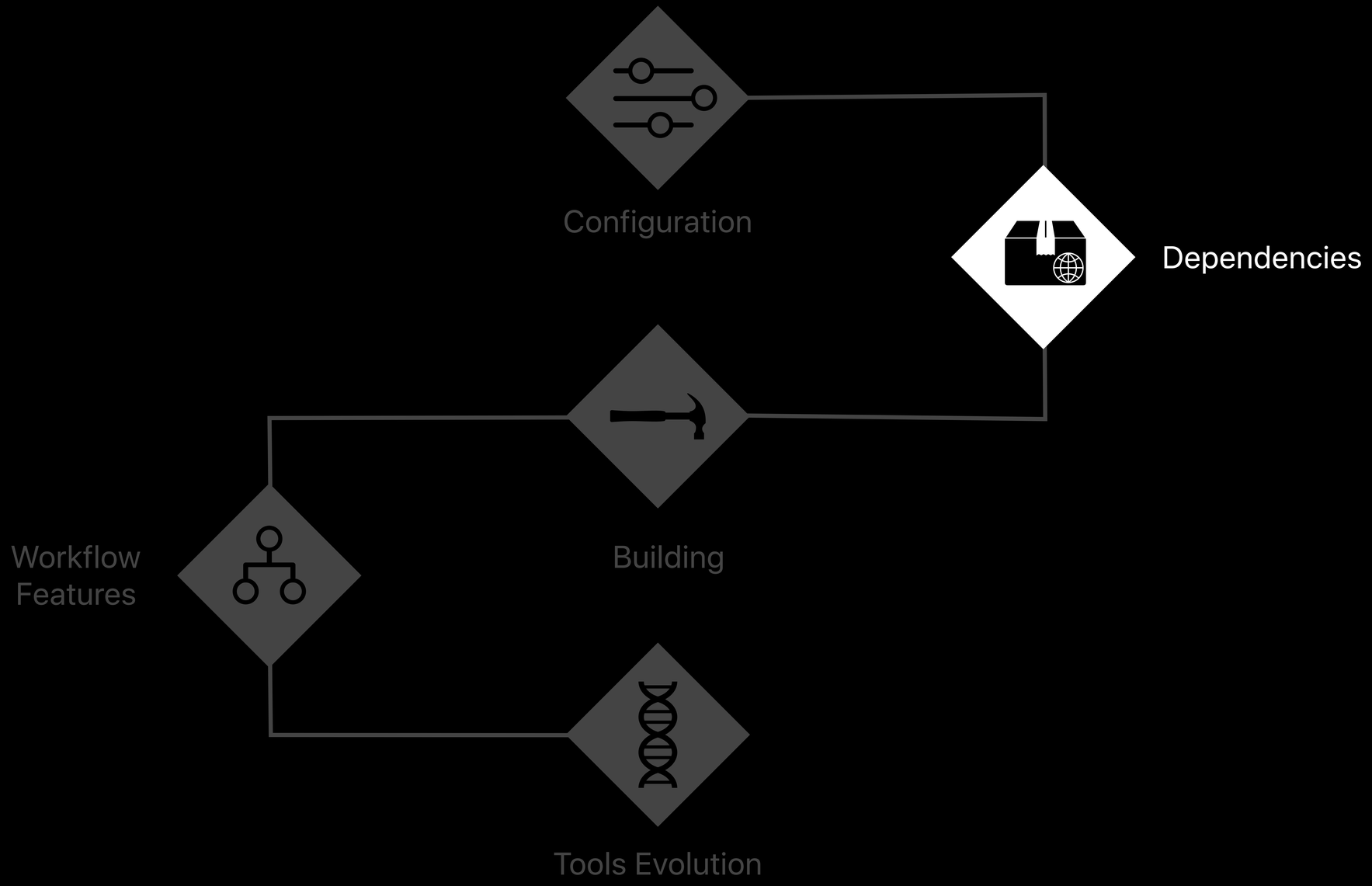
Support for Building Other Languages



Support for Building Other Languages



The Design of SwiftPM



Semantic Versioning

semver.org

1.2.4

Major Version

Breaking changes

1.2.4

Minor Version

Compatible additions

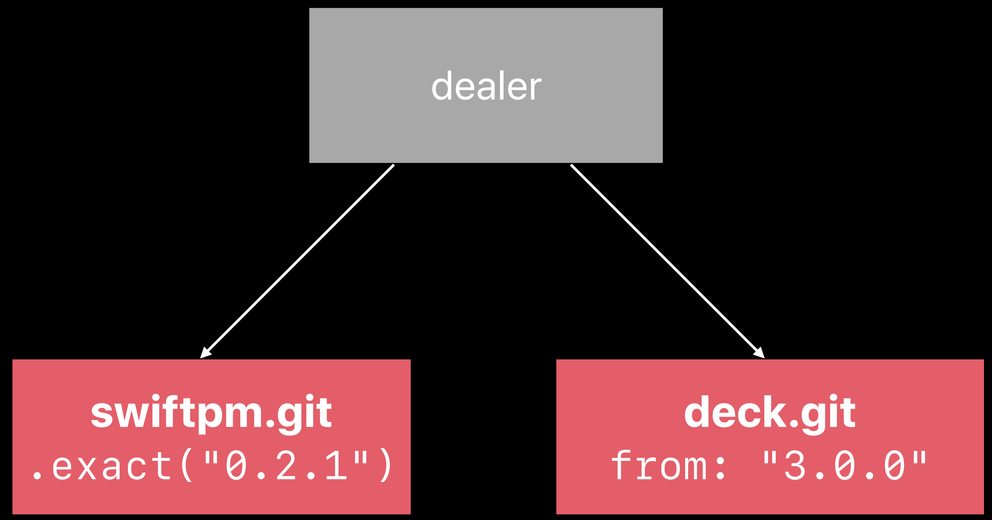
1.2.4

Patch Version

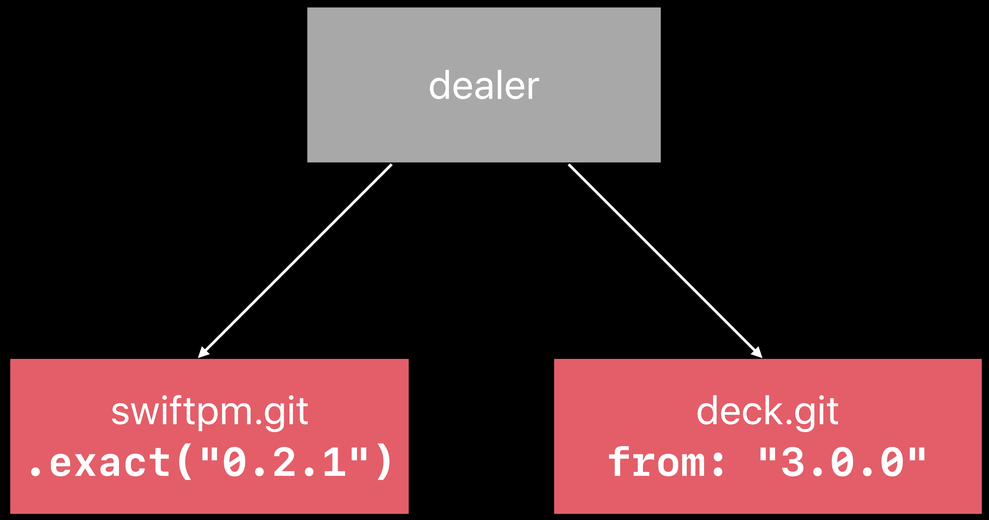
Bug fixes

Dependency Resolution

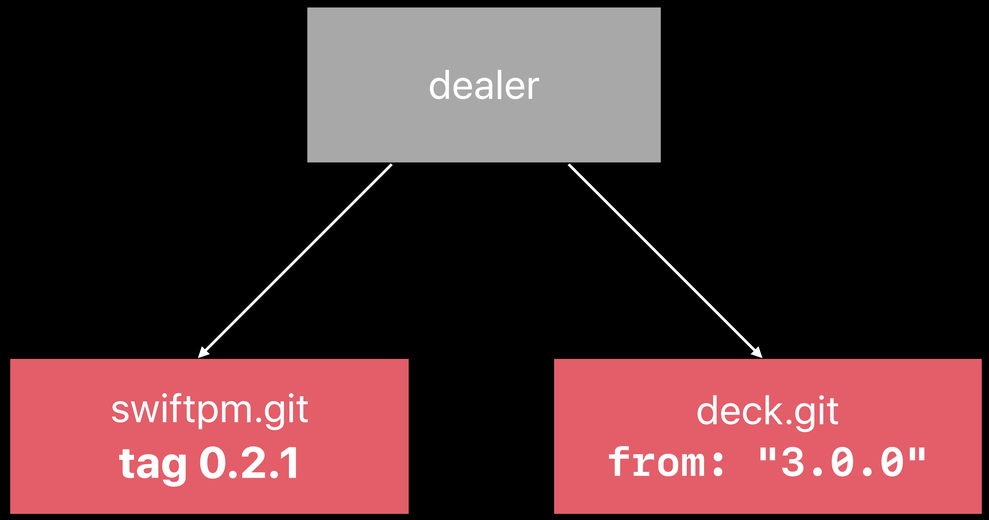
Direct Dependencies



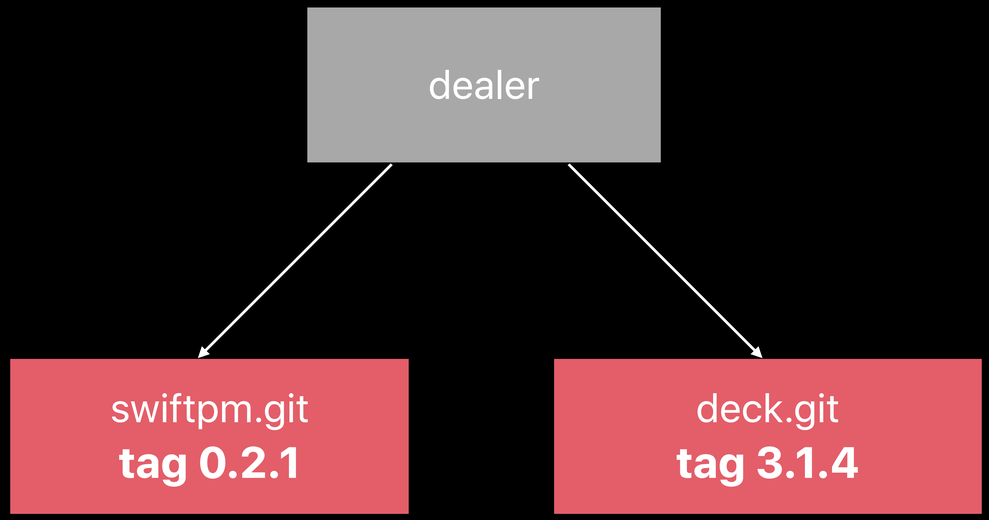
Resolving Versions



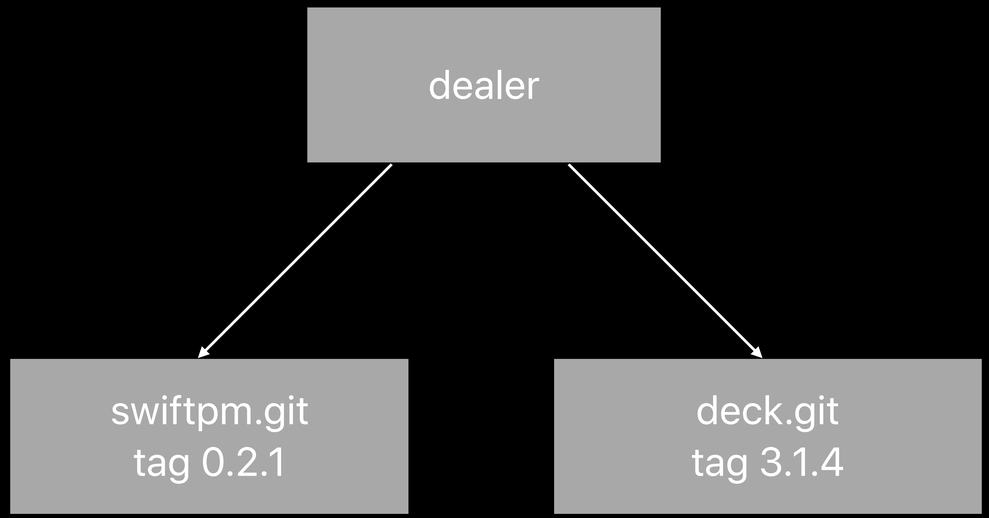
Resolving Versions



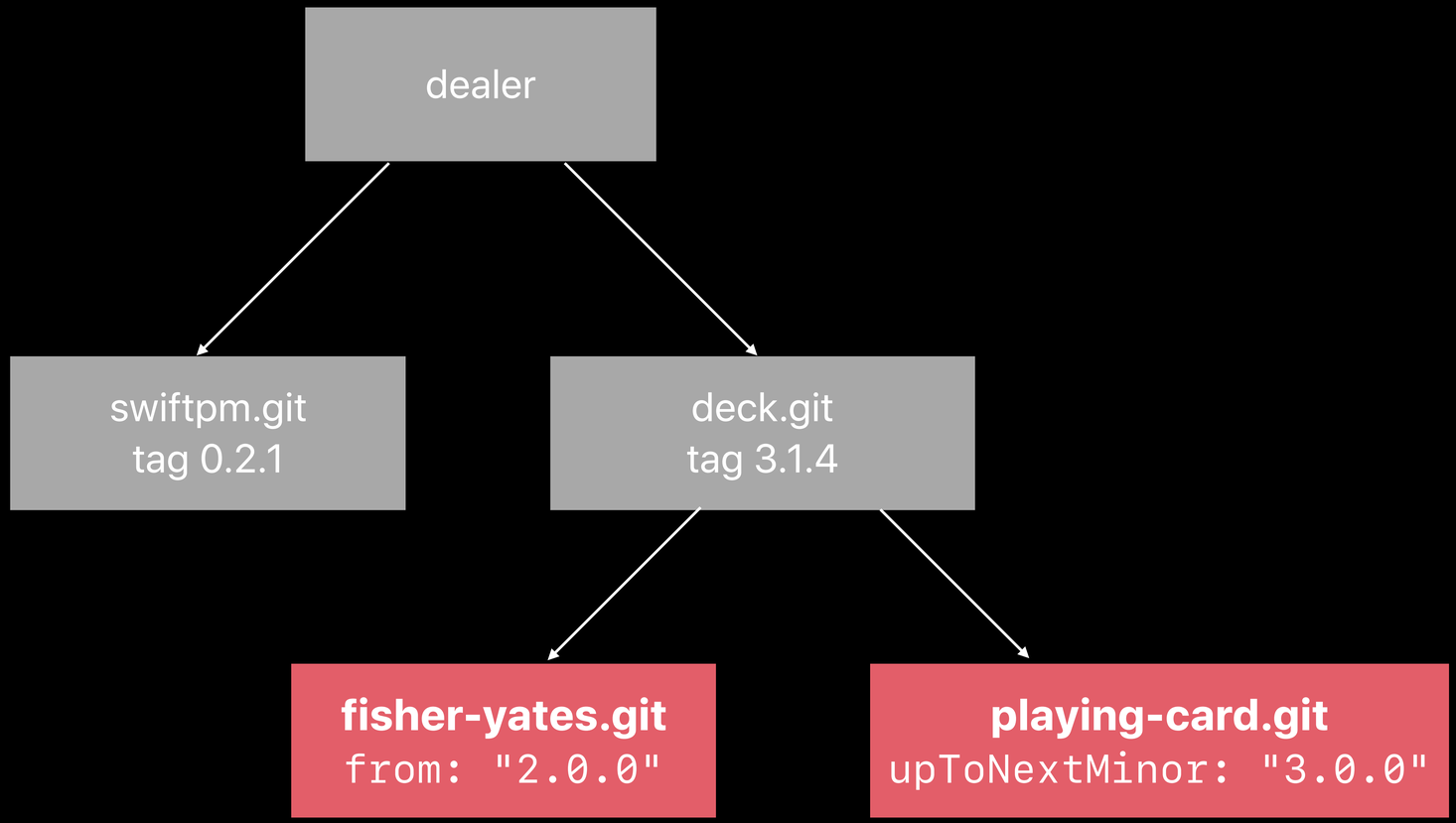
Resolving Versions



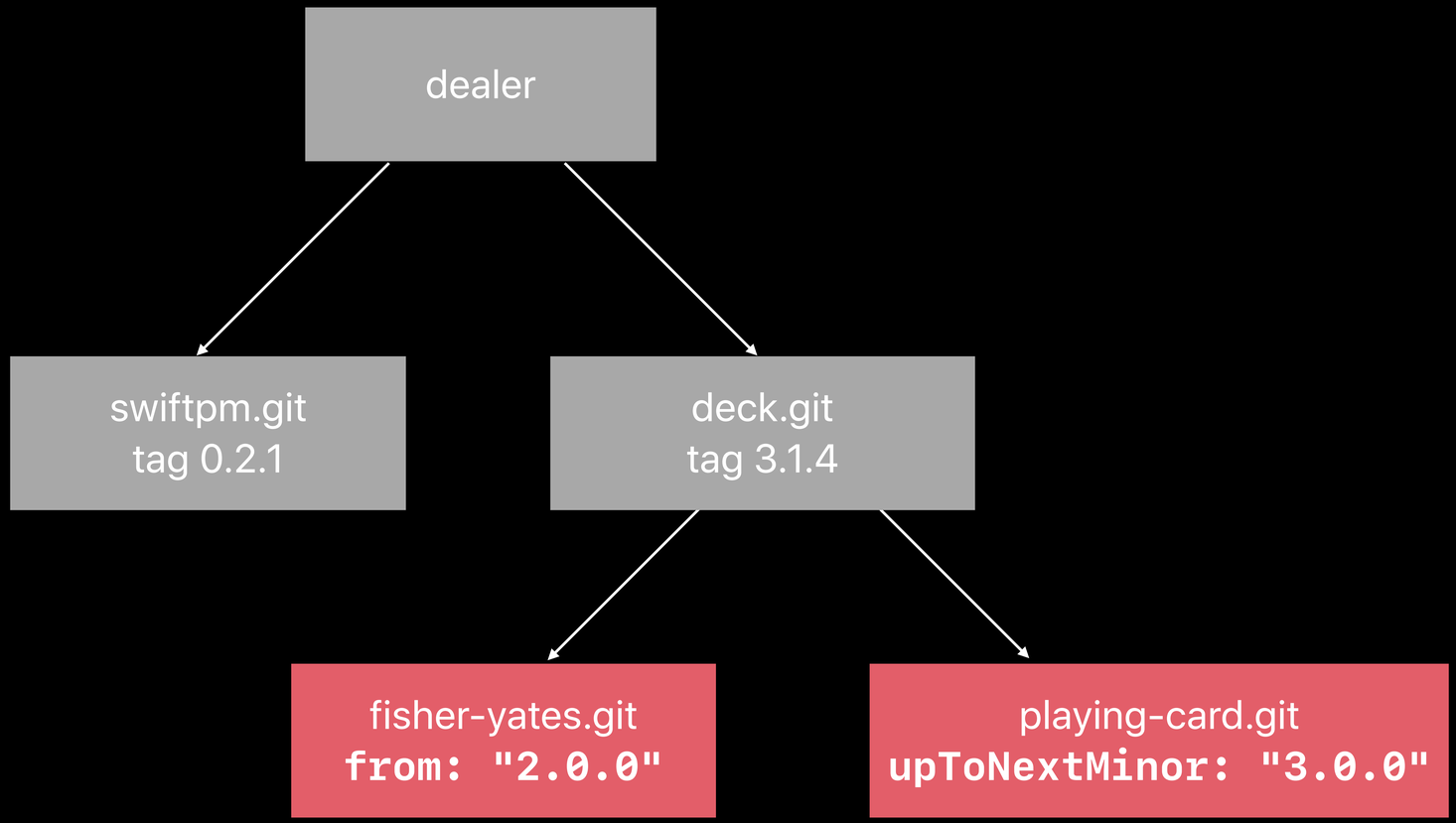
Transitive Dependencies



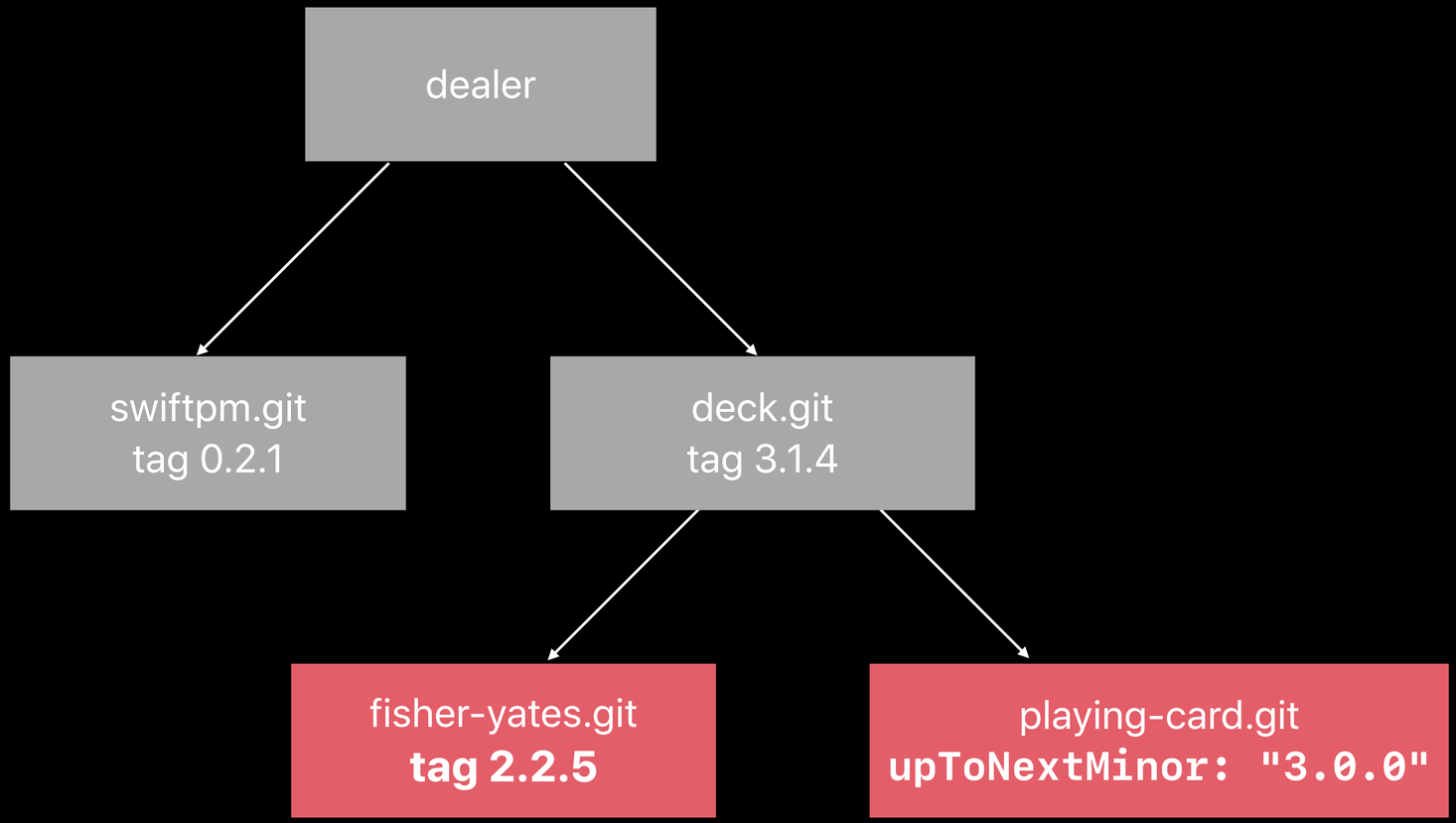
Transitive Dependencies



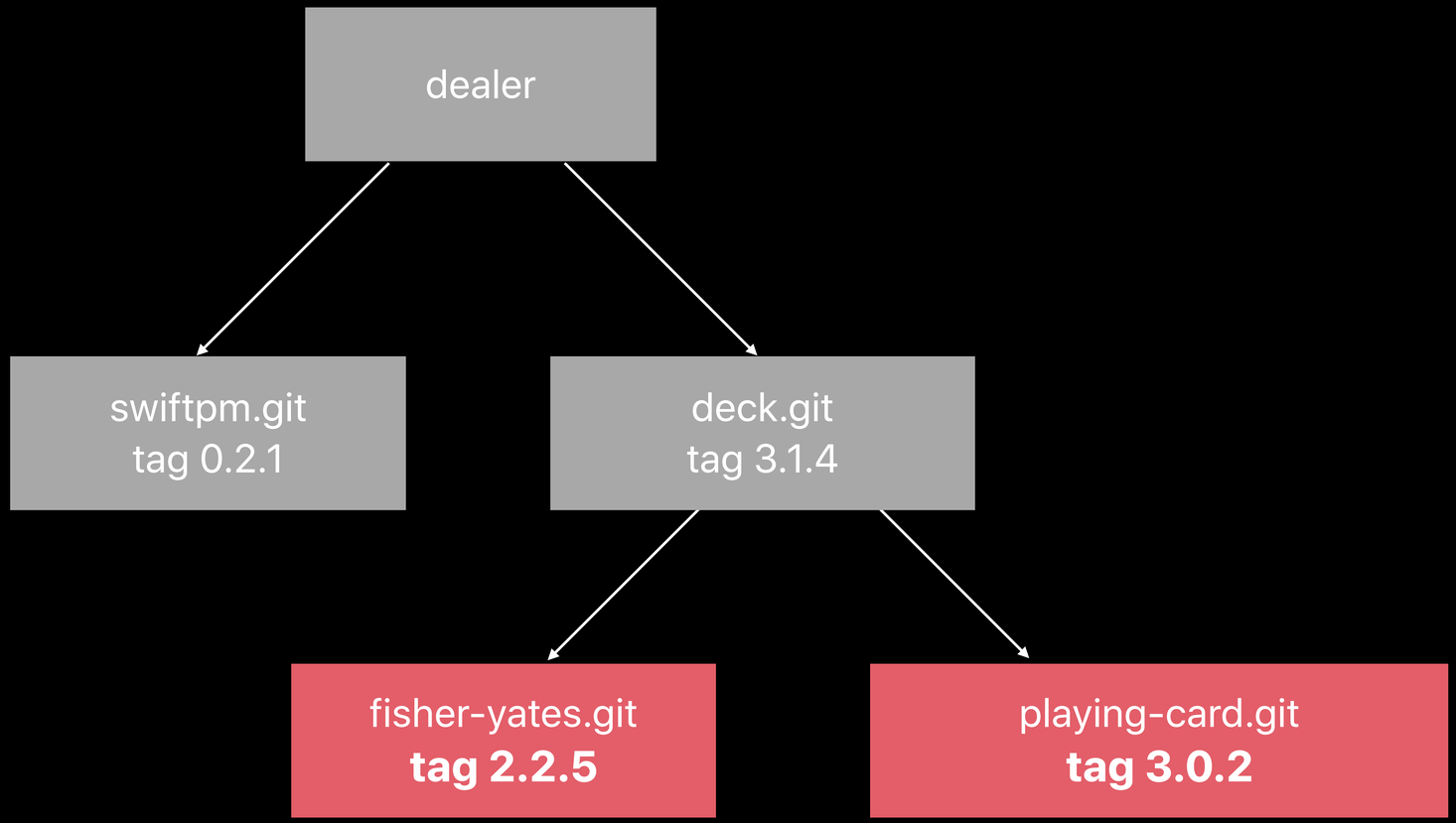
Transitive Dependencies



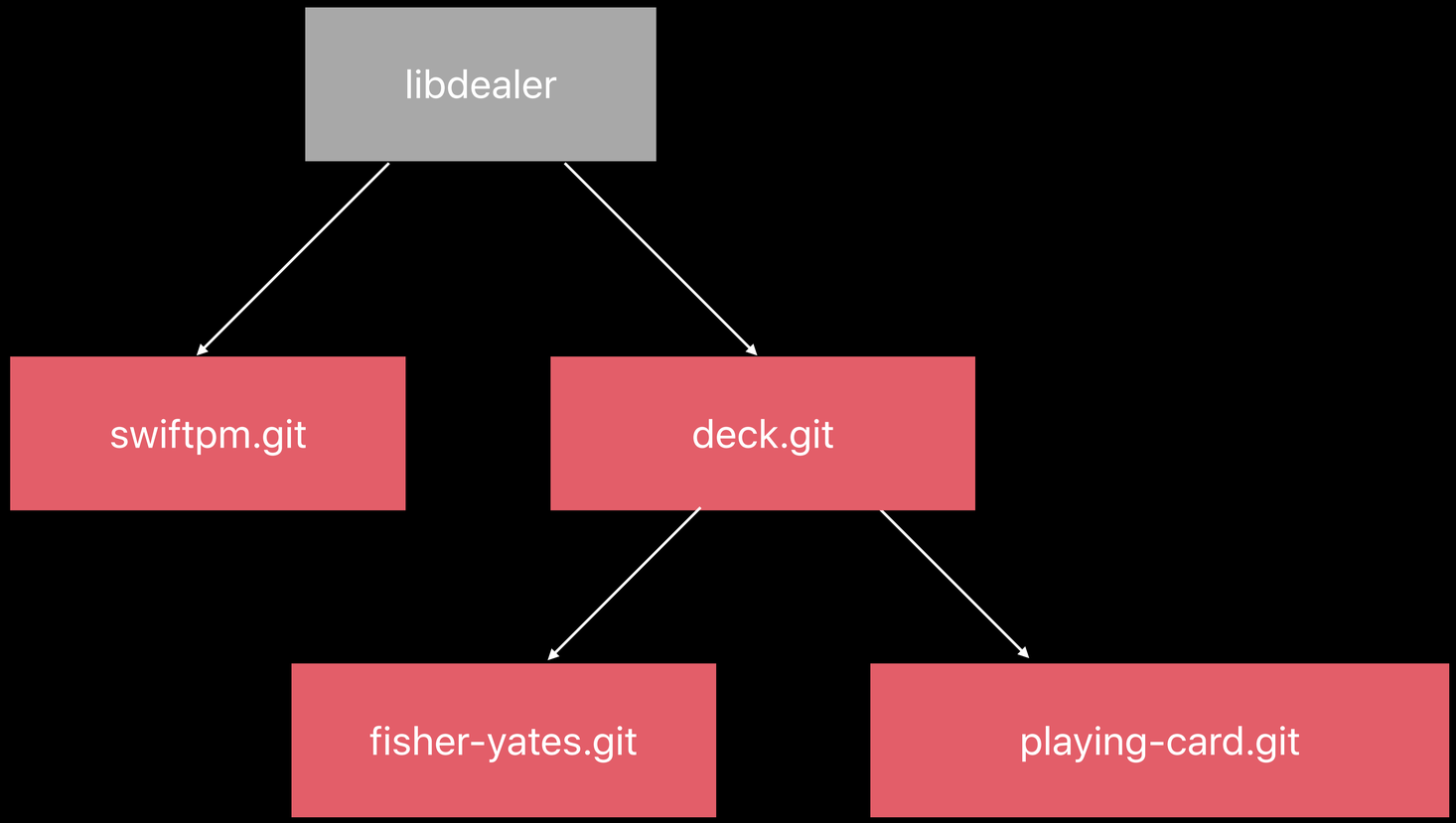
Transitive Dependencies



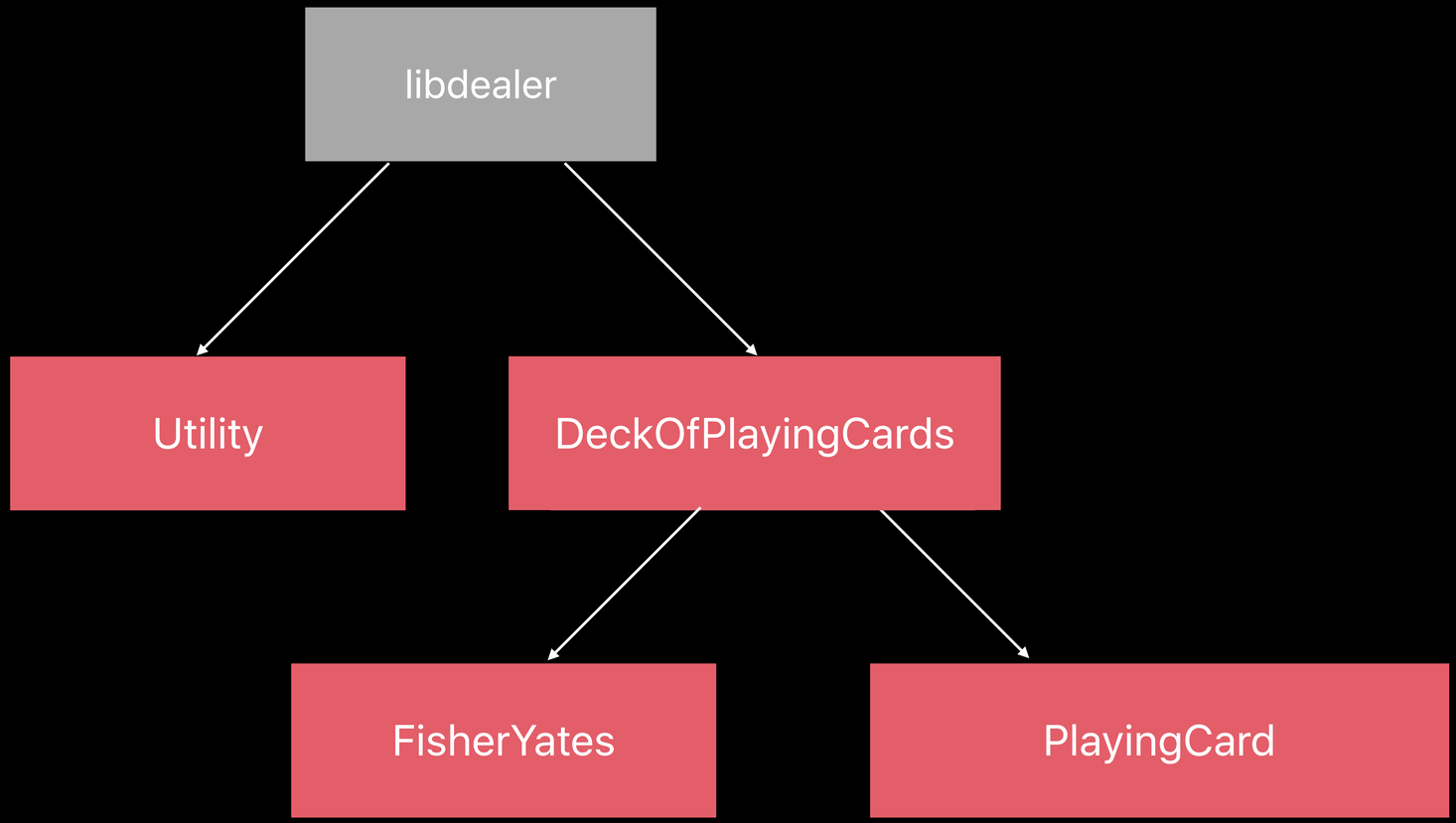
Transitive Dependencies



Resolving Products



Resolving Products



Package.resolved Resolved Versions File

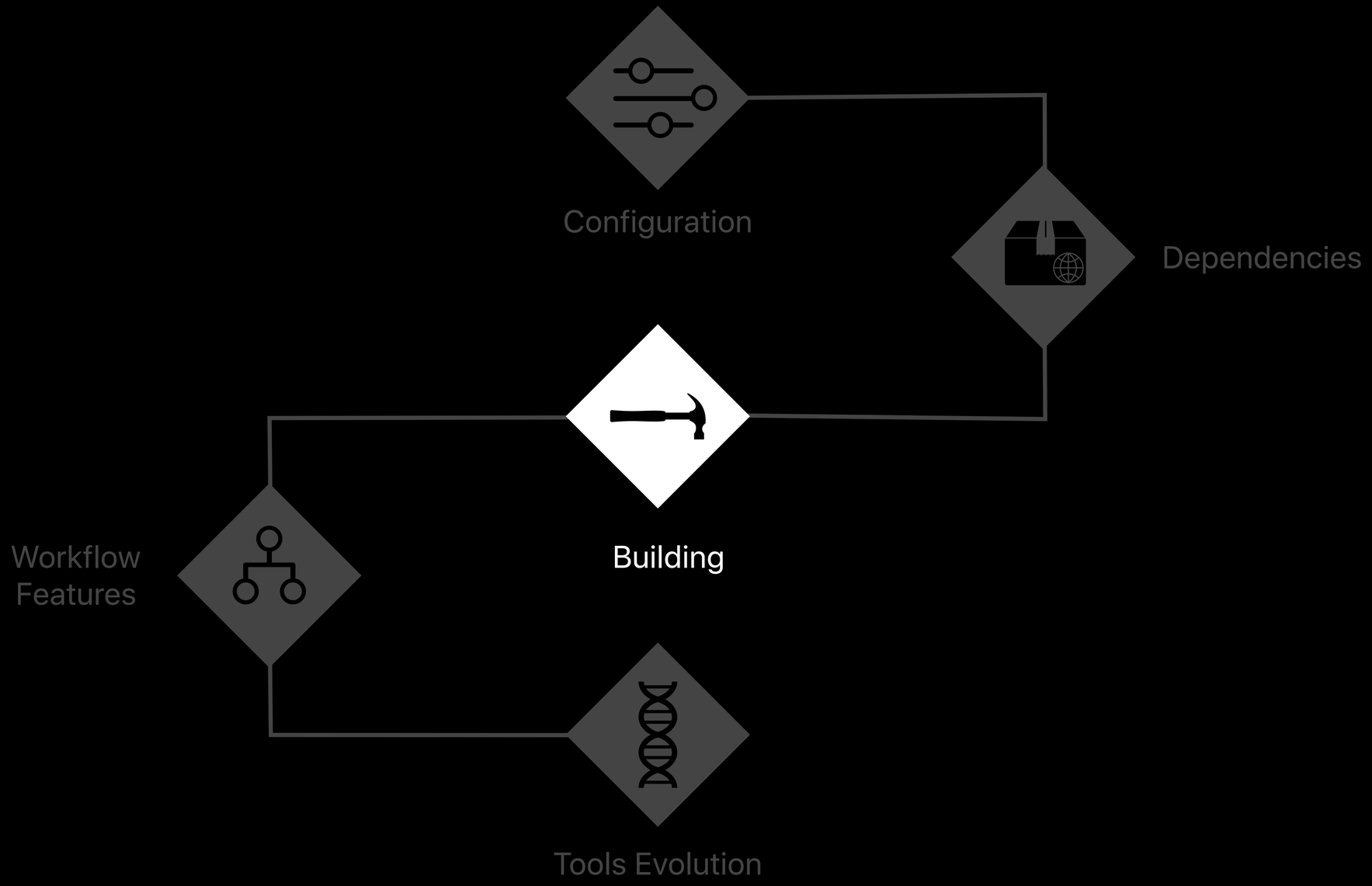
Records resolved package versions

Can be shared for dependable build results

Easy to update with new version resolution

Only used from the top-level package

The Design of SwiftPM



llbuild

SwiftPM’s build execution engine

Provides fast and correct incremental builds

Also used by Xcode’s new build system

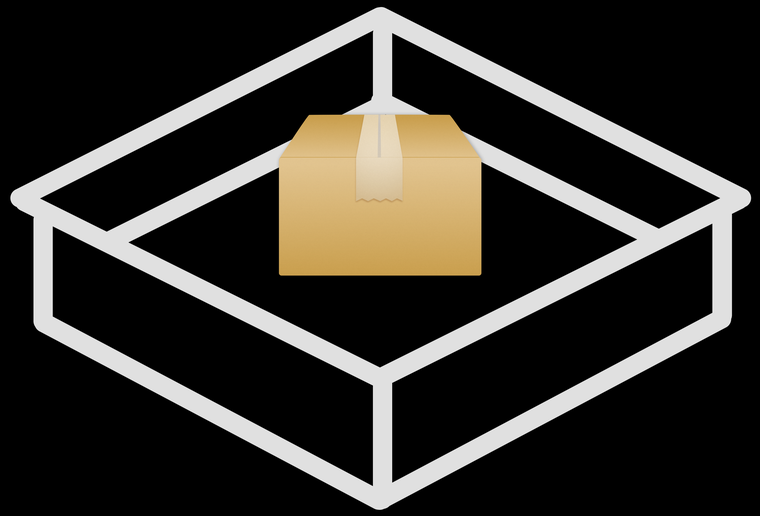
Part of the Swift open source project

Build Environment Isolation

SwiftPM builds packages in isolation

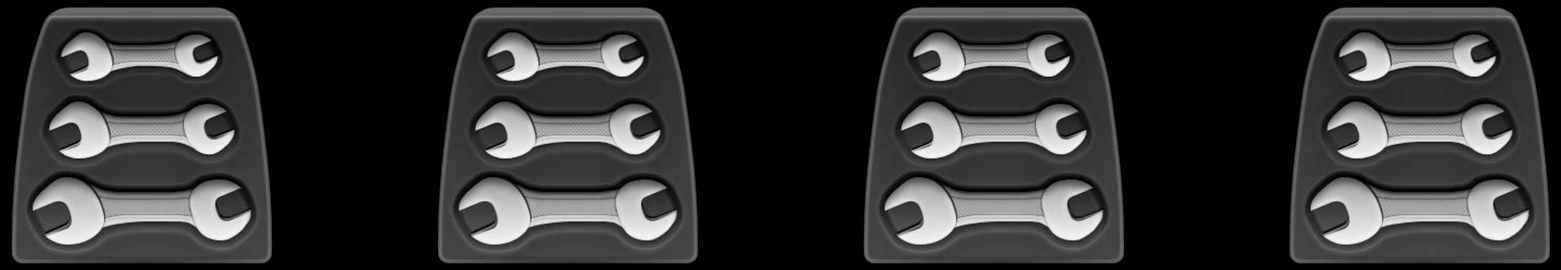
Builds are sandboxed

No arbitrary commands or shell scripts



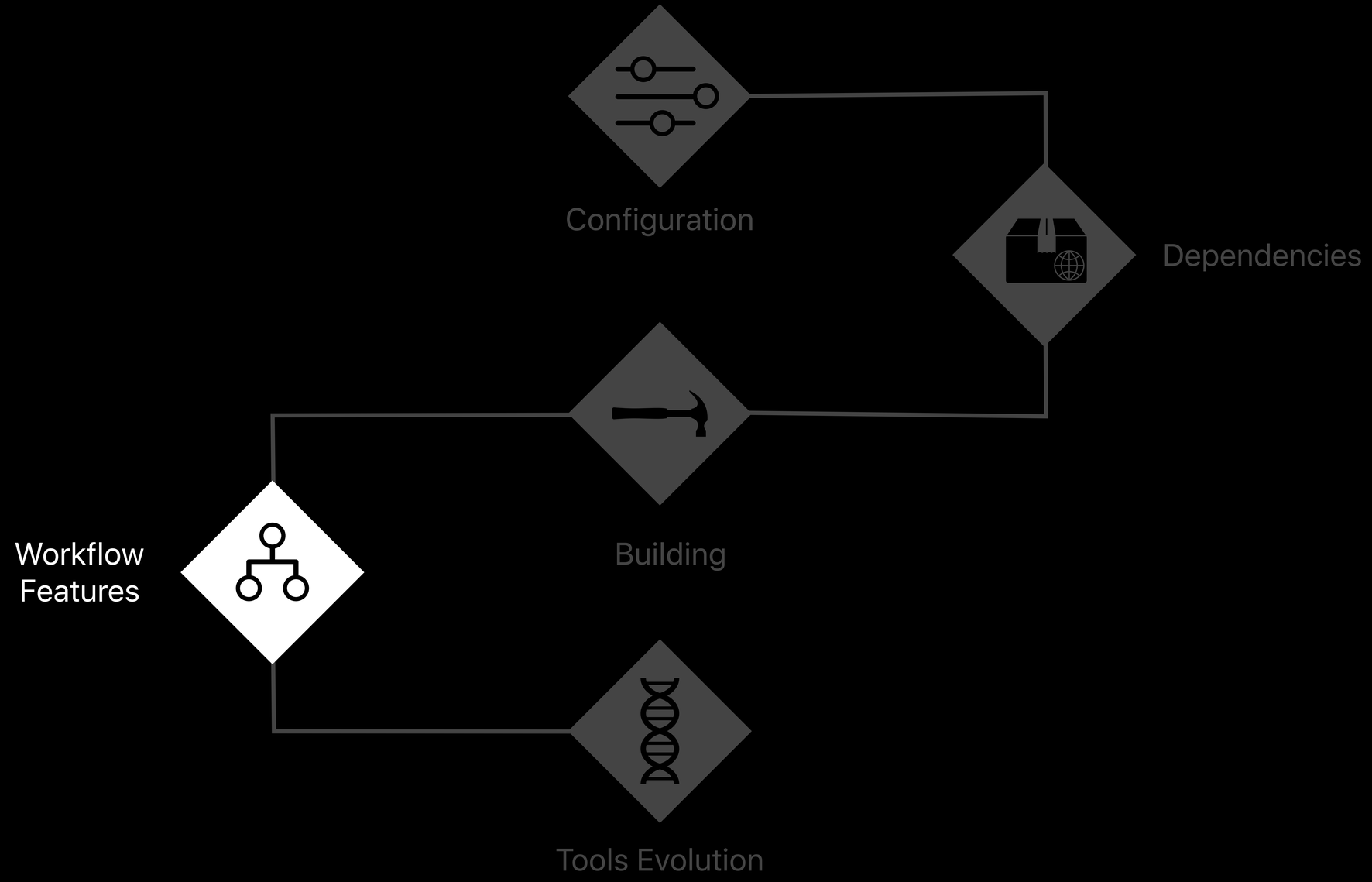
Testing

Parallel Testing

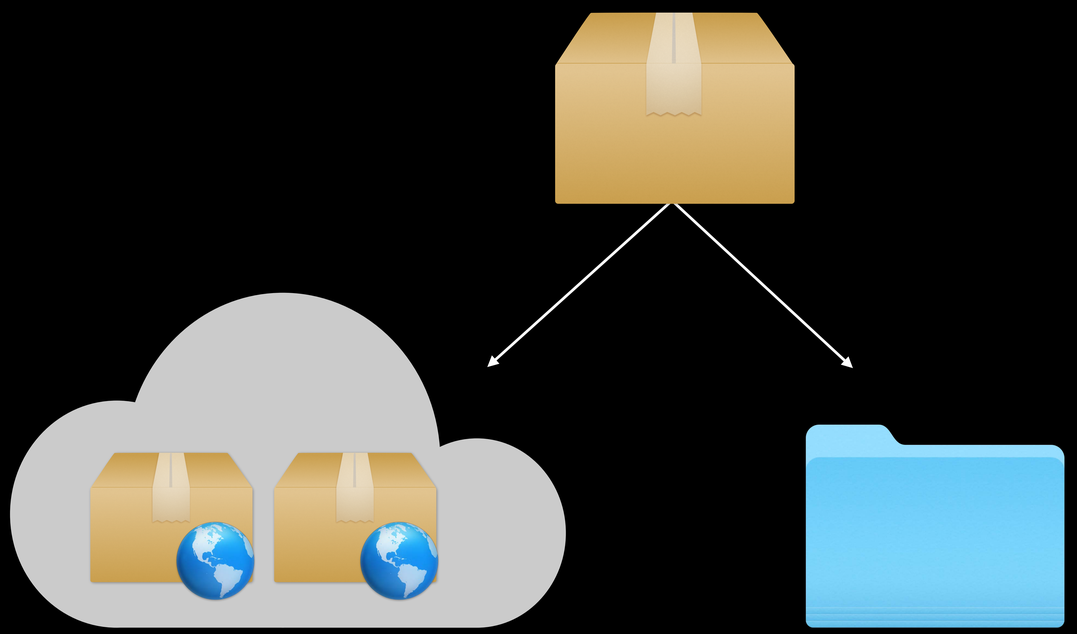


Test Filtering

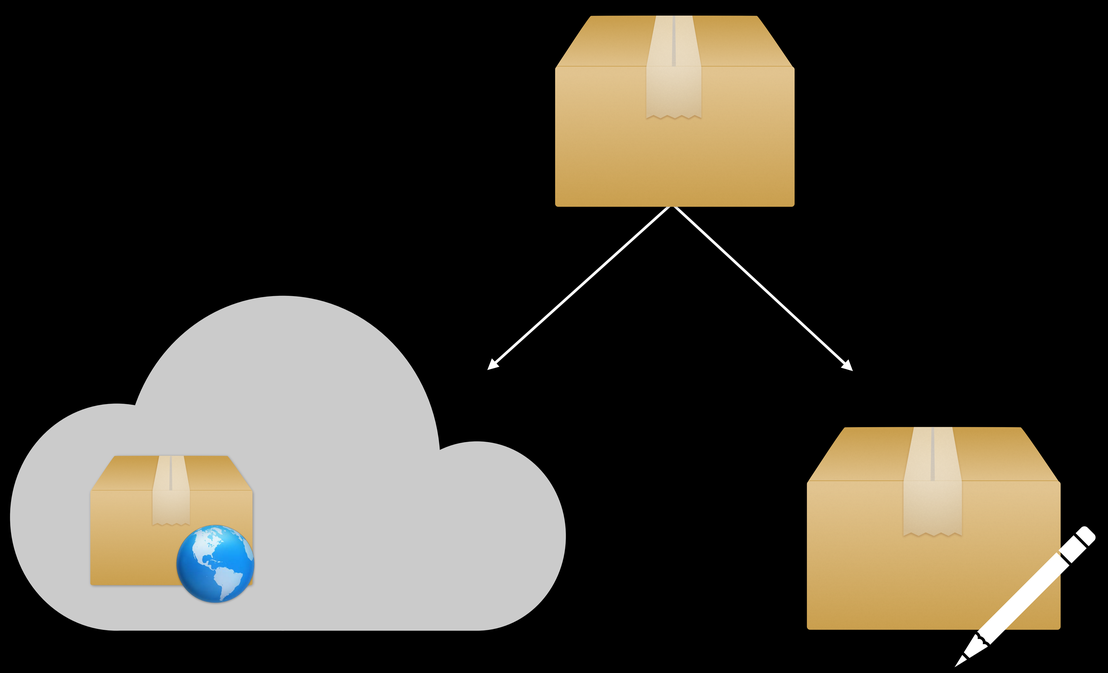
The Design of SwiftPM



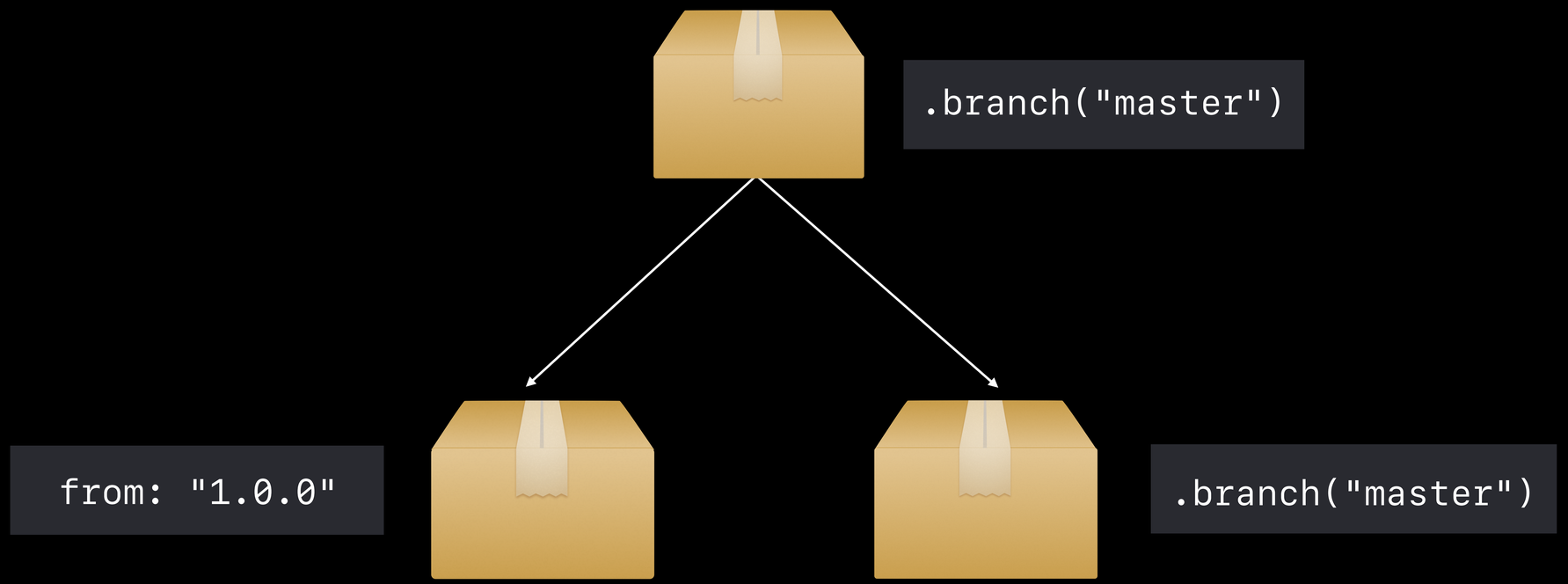
Edit Mode



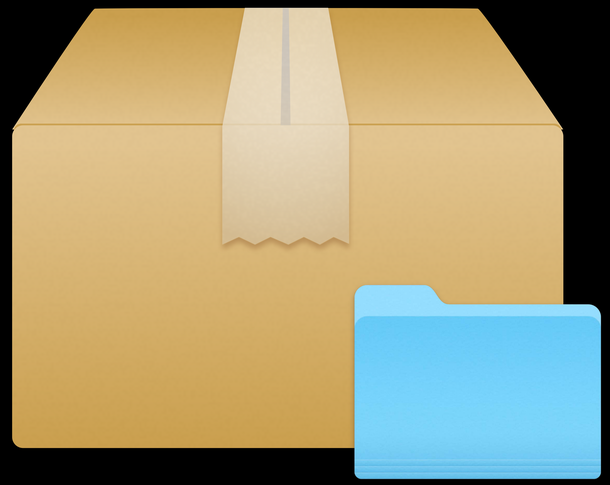
Edit Mode



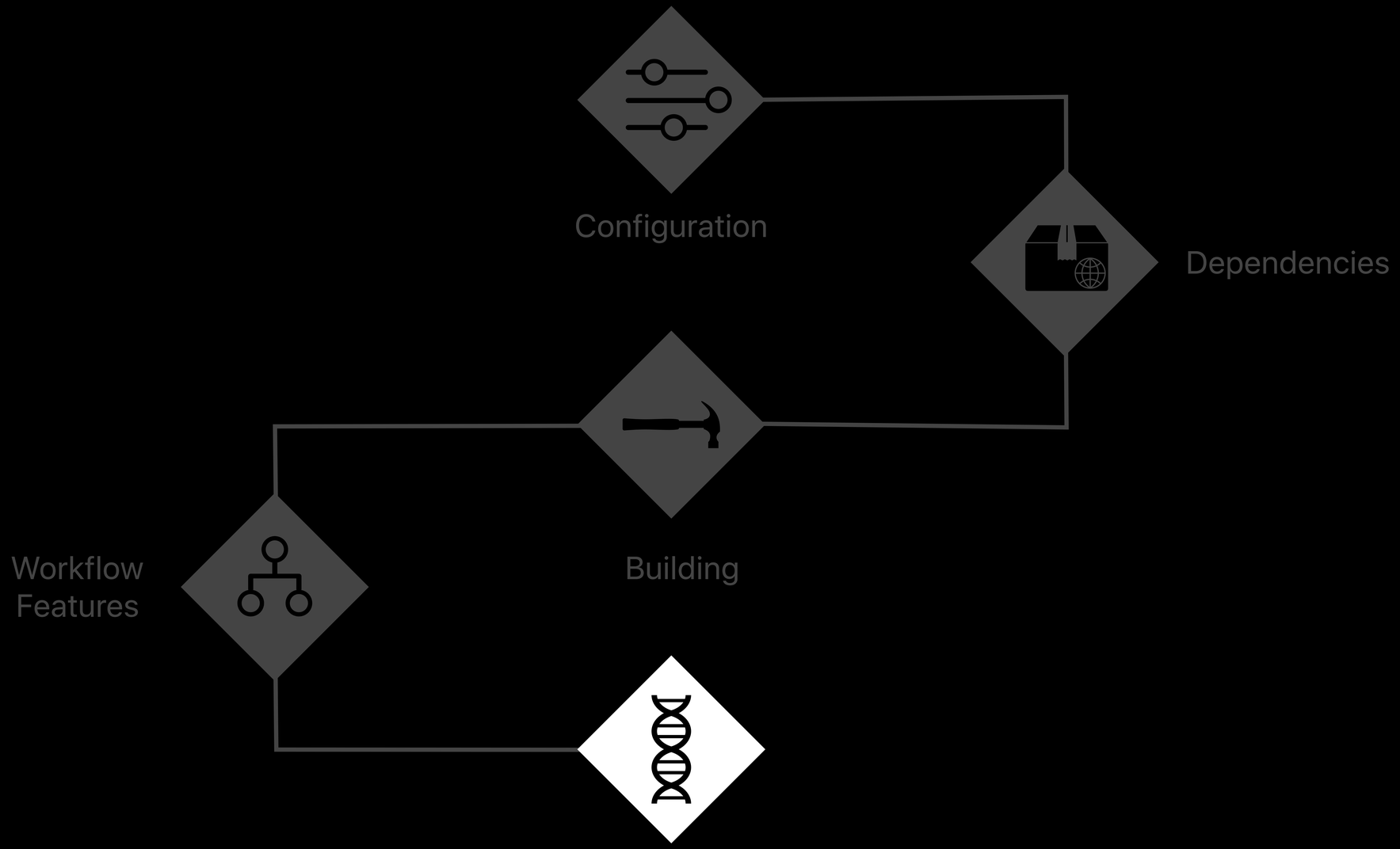
Branch Dependencies



Local Package Dependencies



The Design of SwiftPM



Package.swift Manifest API Evolution

Package API can be updated with each new Swift version

Previous API is still available

Allows using new Swift tools without updating the manifest

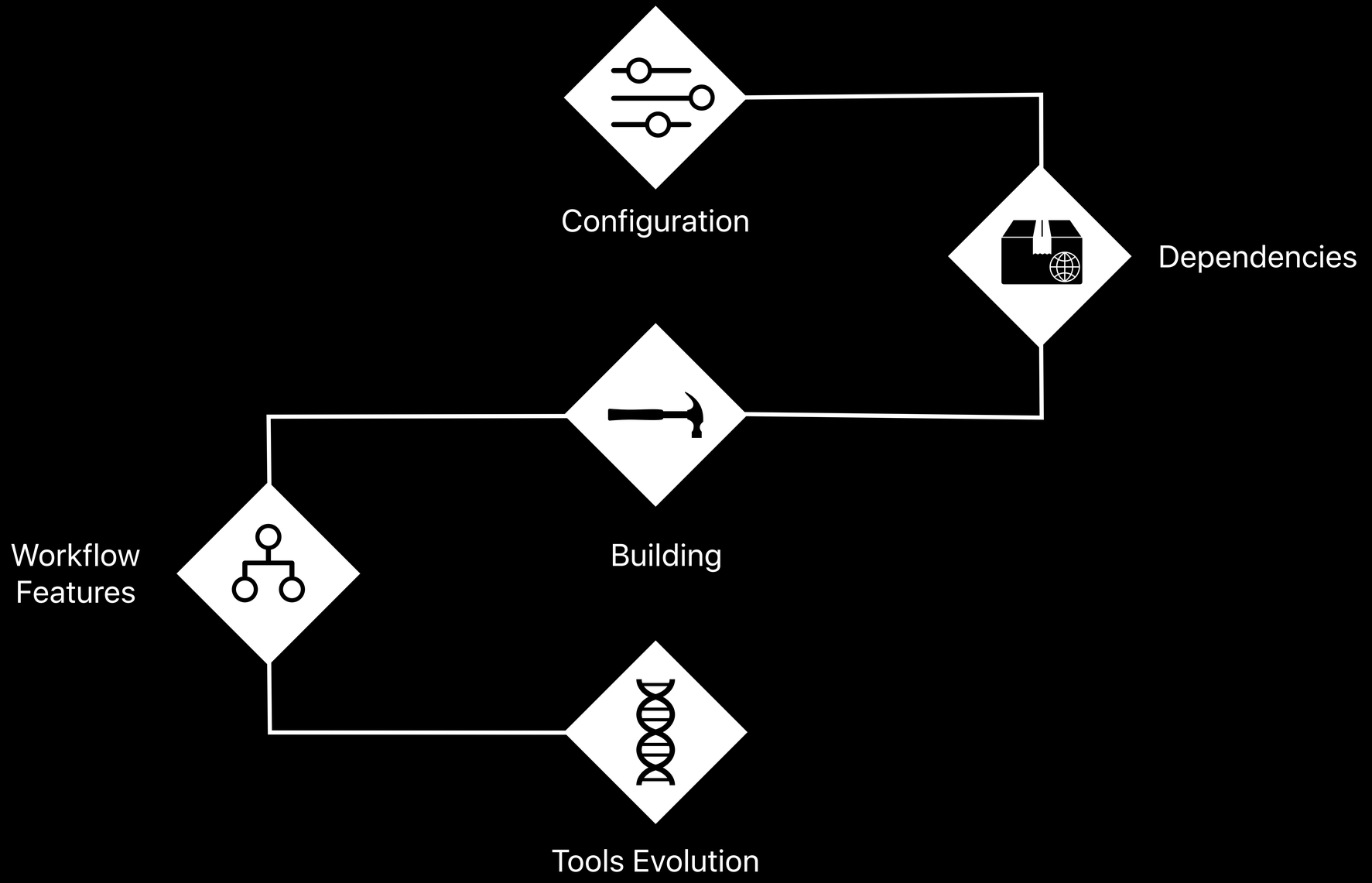
Swift Tools Version

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

let package = Package(…, swiftLanguageVersions: [.v4_2, .v4])
```

Swift Language Version

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

let package = Package(…, swiftLanguageVersions: [.v4_2, .v4])
```

The Design of SwiftPM



Why a package manager for Swift?

How to use it

The design of SwiftPM

Evolution ideas

Open source process

Open Evolution Process

Themes

Great integration with other tools

Publish and deploy

Support complex packages

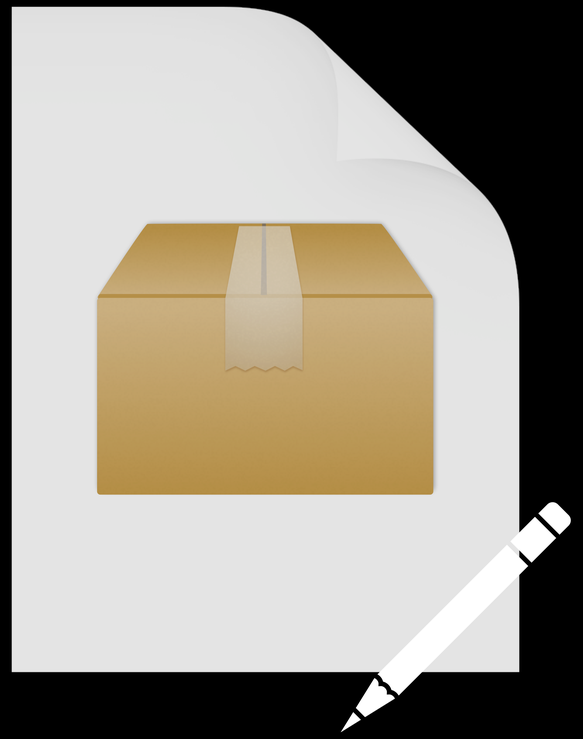
Find, manage, and trust packages

Great Integration with Other Tools

libSwiftPM Available

SwiftPM support in
developer tools is encouraged!

Idea: Machine-Editable Package.swift



Idea: Machine-Editable Package.swift

+

libSyntax

Idea: Machine-Editable Package.swift

```
let package = Package(
    name: "Networking",
    products: [
      .library(name: "Networking", targets: ["Networking"])
    ],
    targets: [
      .target(
          name: "Networking",
          dependencies: []),
   ]
)
```

Idea: Machine-Editable Package.swift

```
let package = Package(
    name: "Networking",
    products: [
      .library(name: "Networking", targets: ["Networking"])
    ],
    targets: [
      .target(
          name: "Networking",
          dependencies: ["NetworkCore"]),
      .target(
          name: "NetworkCore",
          dependencies: []),
    ]
)
```

Publish and Deploy

Idea: Tagging and Publication Support

Idea: Automatic Semantic Versioning

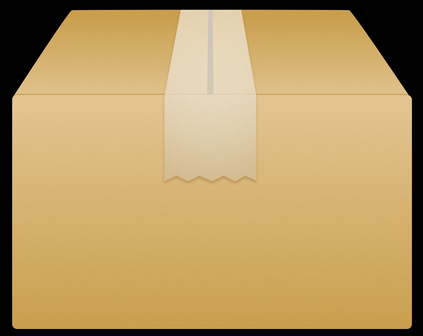
```
public func findPerson(
    firstName: String)
     -> Contact
```

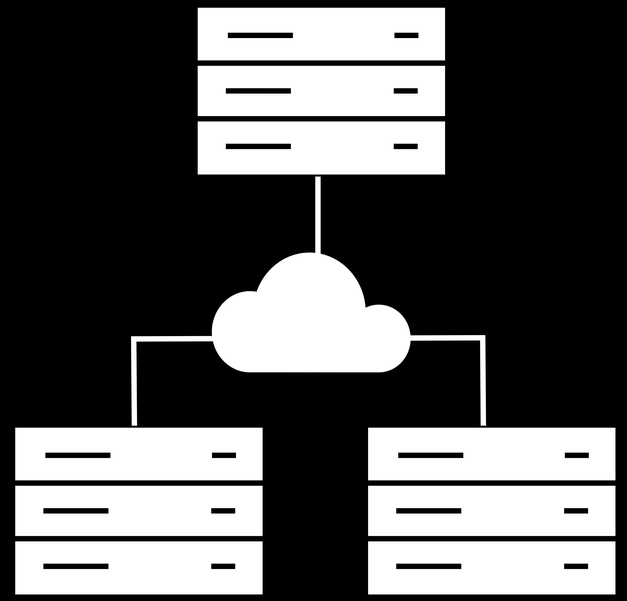
1.6.0

```
public func findPerson(
    firstName: String,
    lastName: String)
     -> Contact
```

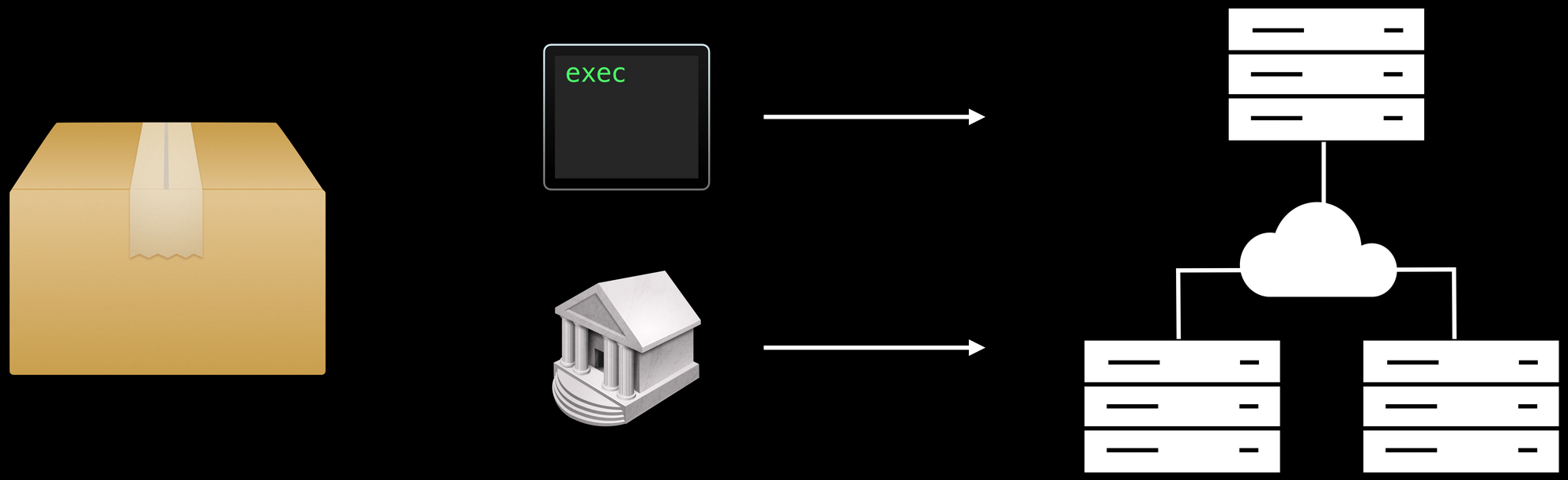
2.0.0

Idea: Deployment Automation



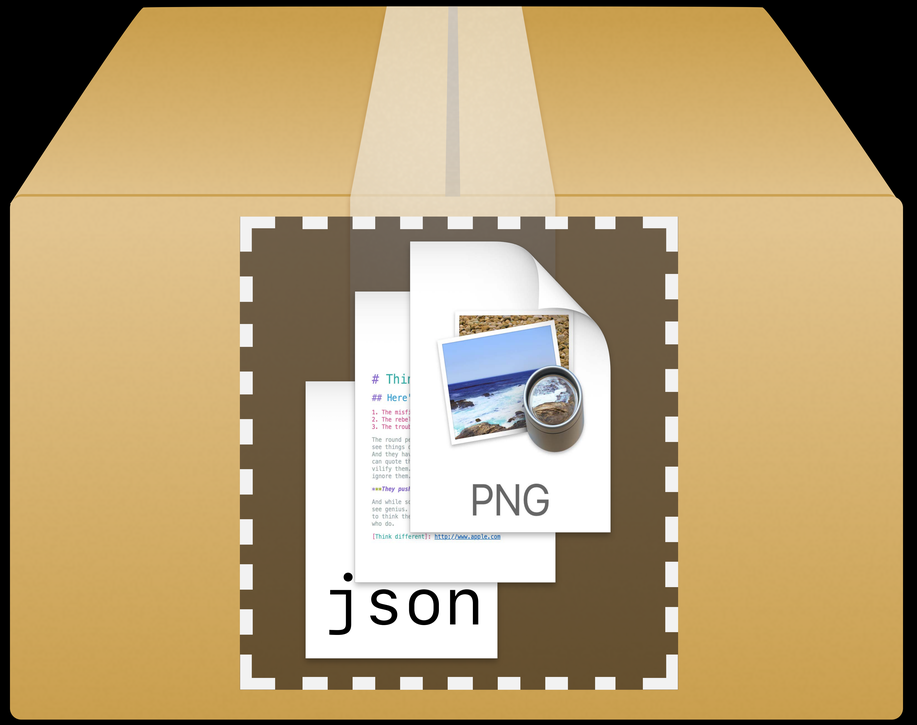


Idea: Deployment Automation



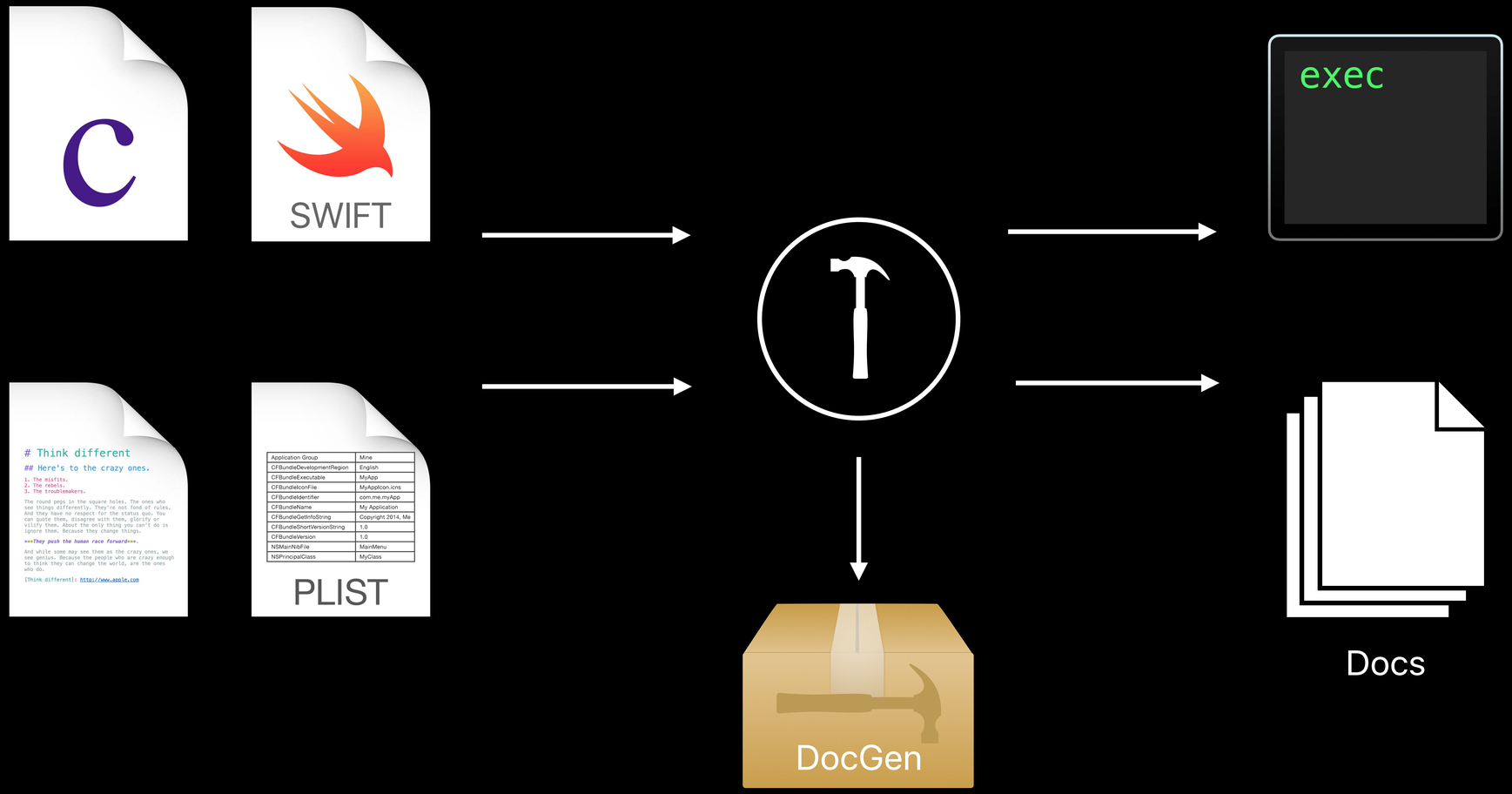
Support Complex Packages

Idea: Resources



Idea: Build Settings

Idea: Extensible Build Tools

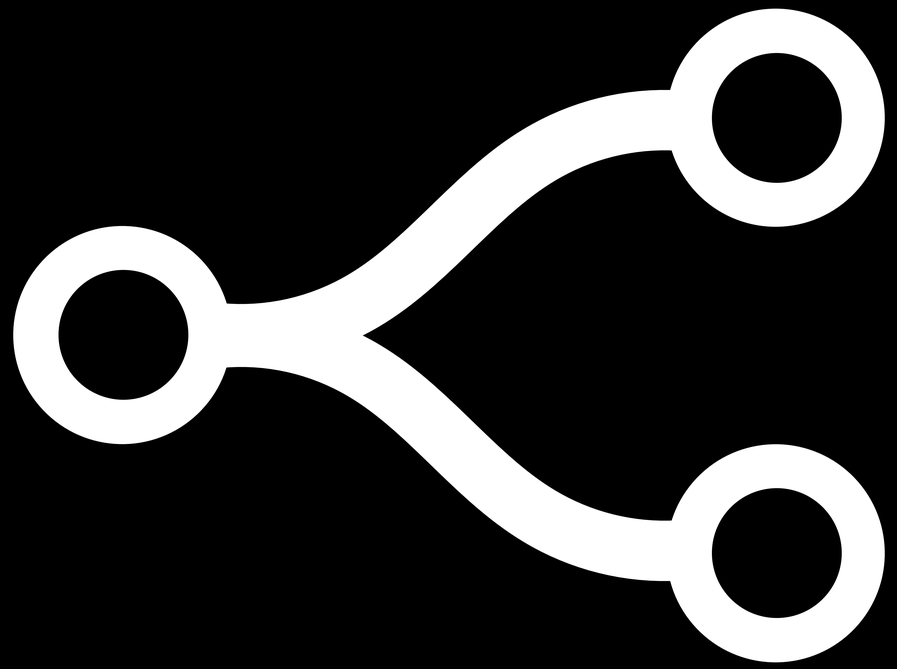


Trust, Manage, and Find Packages

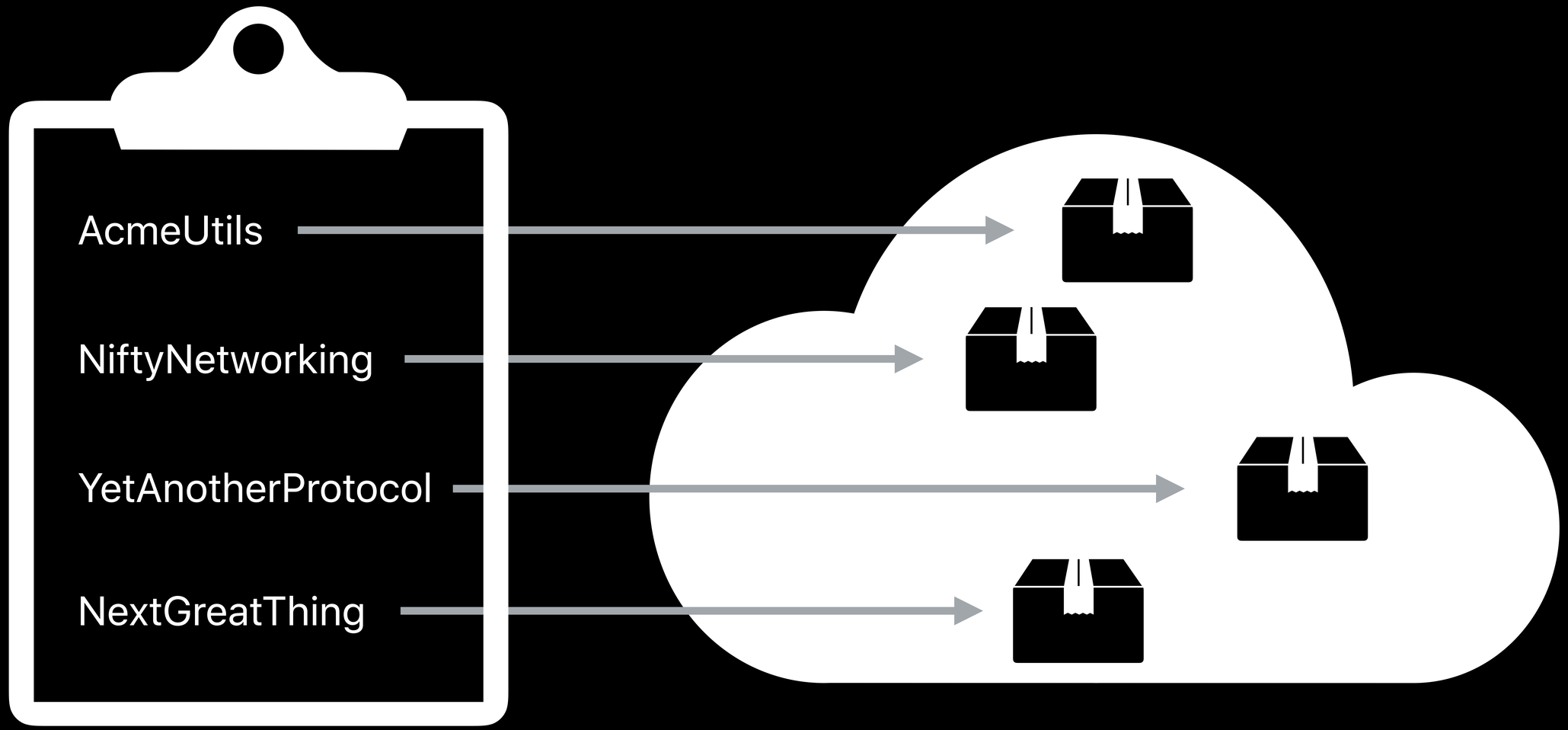
Idea: Package Content Verification

Idea: Cross-Platform Sandboxing

Idea: Fork Support



Idea: Package Index



Deployment Automation

Verify Expected Package Content

Build Settings

Extensible Build Tools

Cross-Platform Sandboxing

Your Contributions

Resources

Fork Support

Automatic SemVer

Machine-Editable Package.swift

Tag & Publish

Package Index

Leverage libSwiftPM

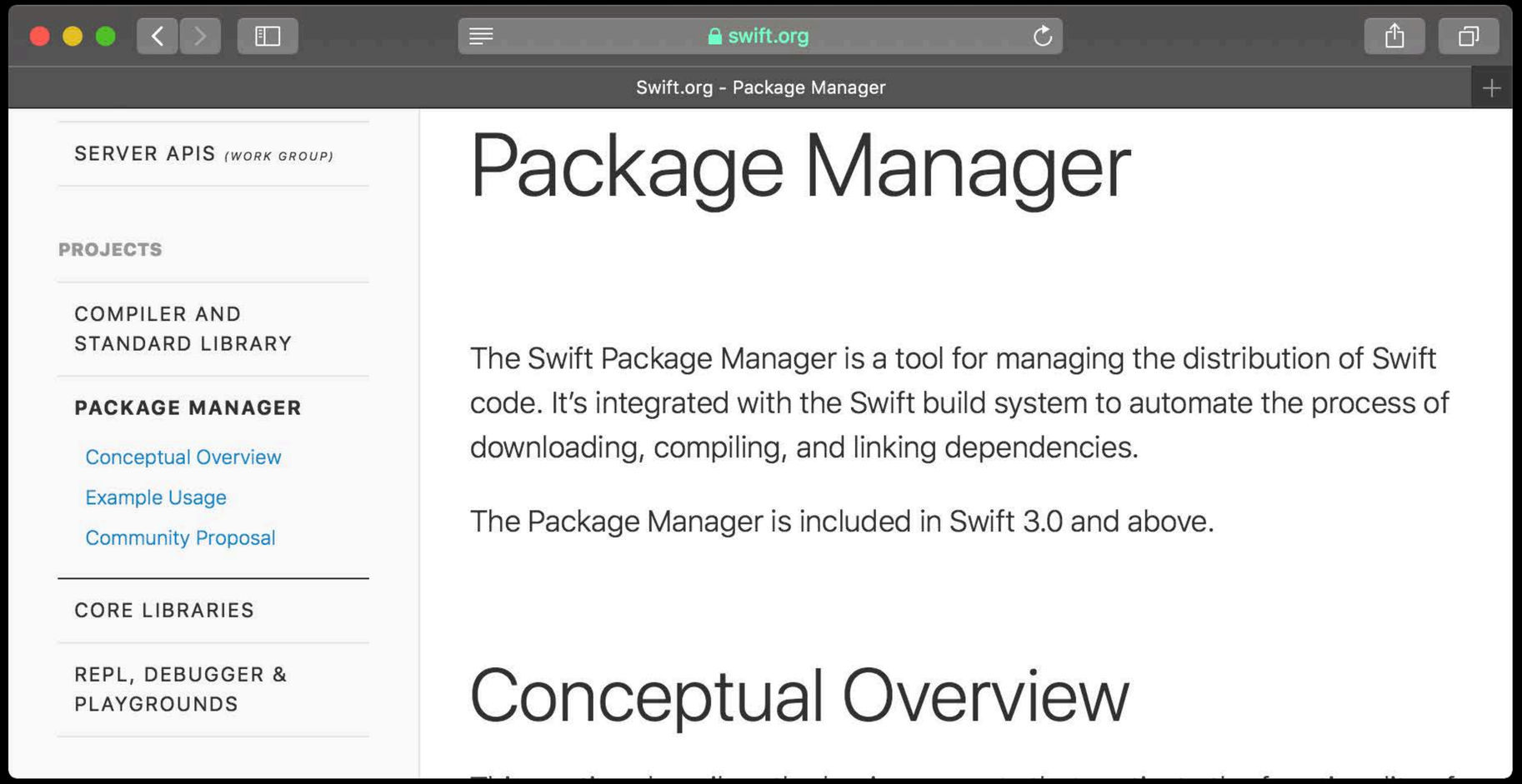
Why a package manager for Swift?

How to use it

The design of SwiftPM

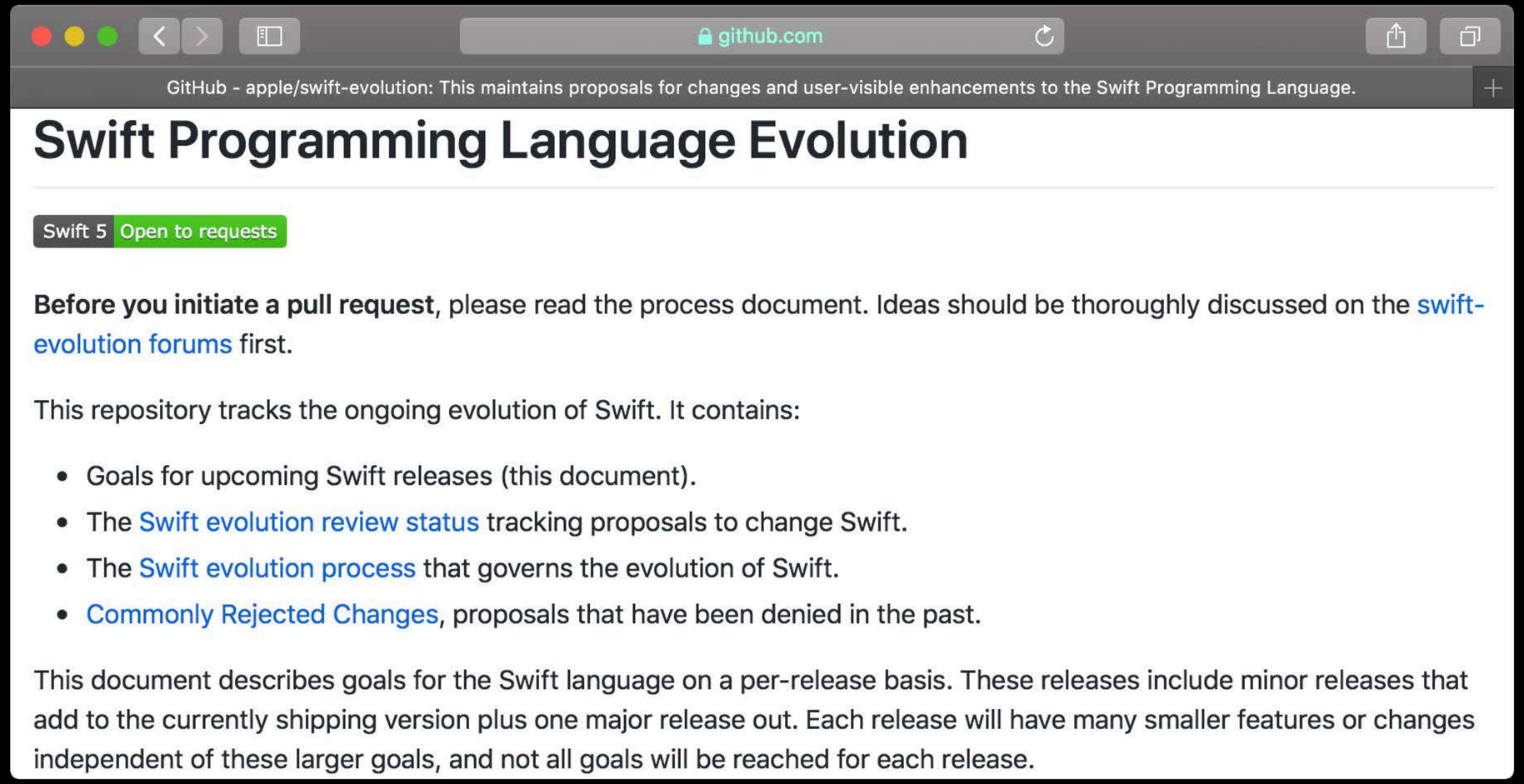
Evolution ideas

Open source process



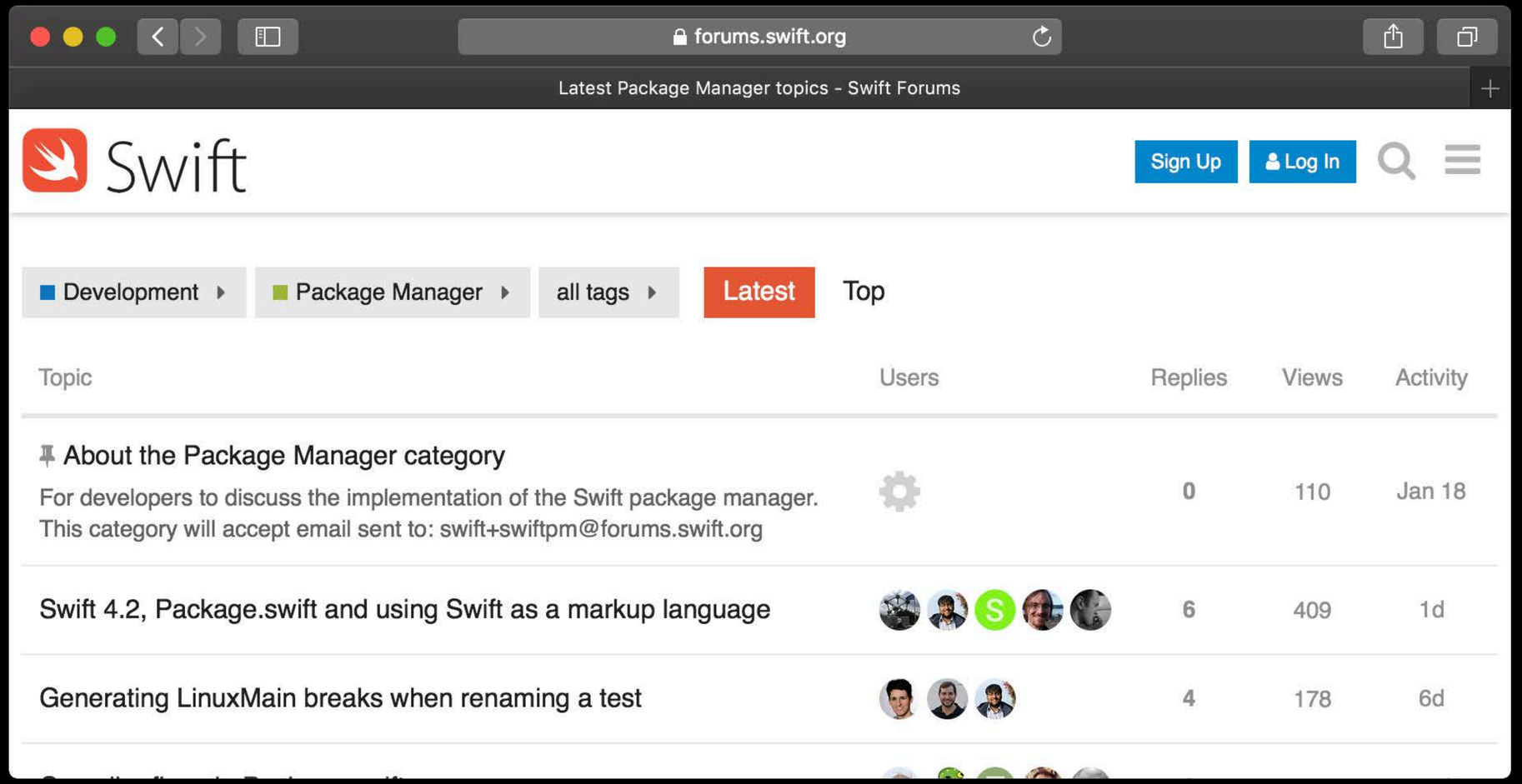
https://swift.org/package-manager/

Swift Evolution



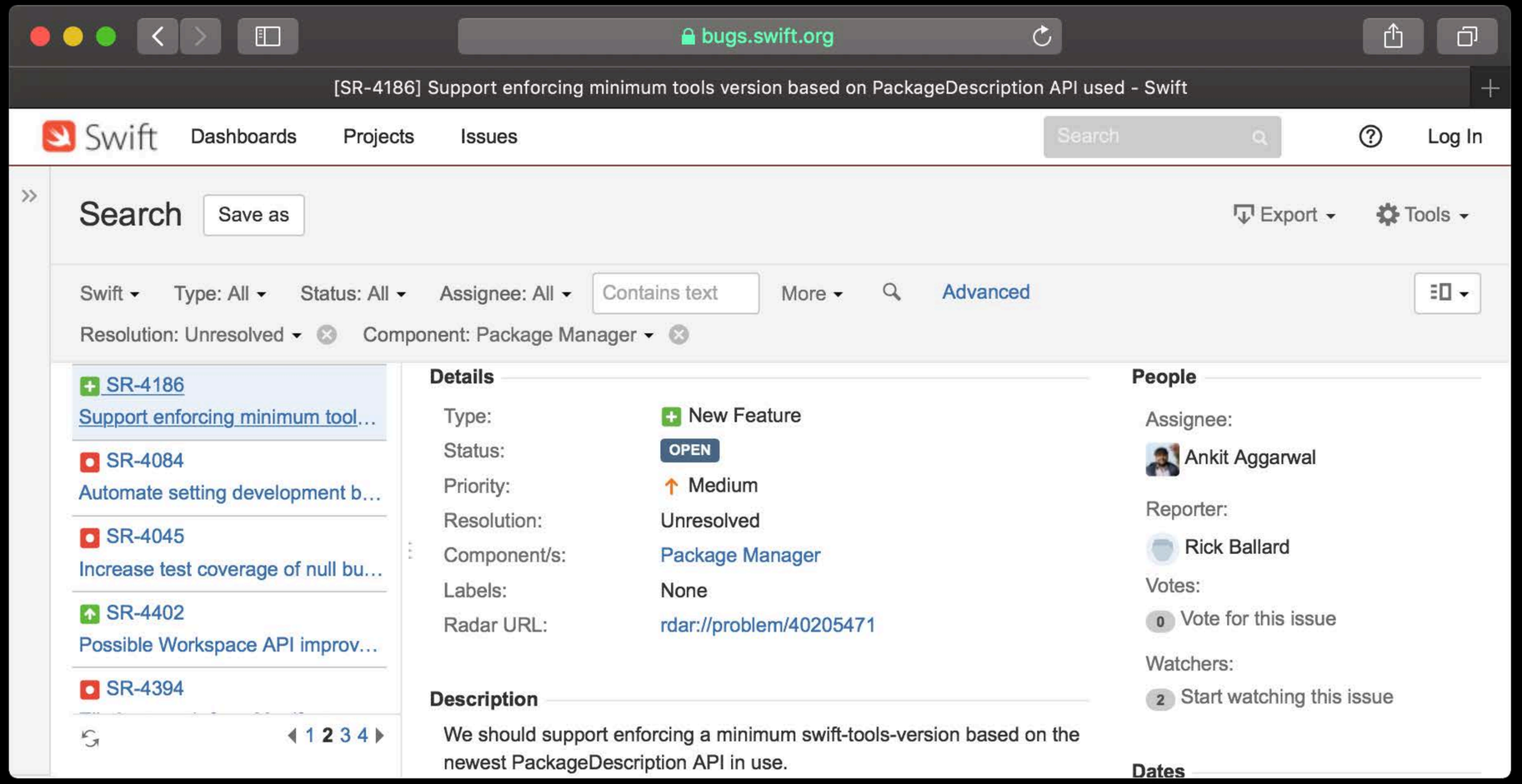
https://github.com/apple/swift-evolution

Swift Forums



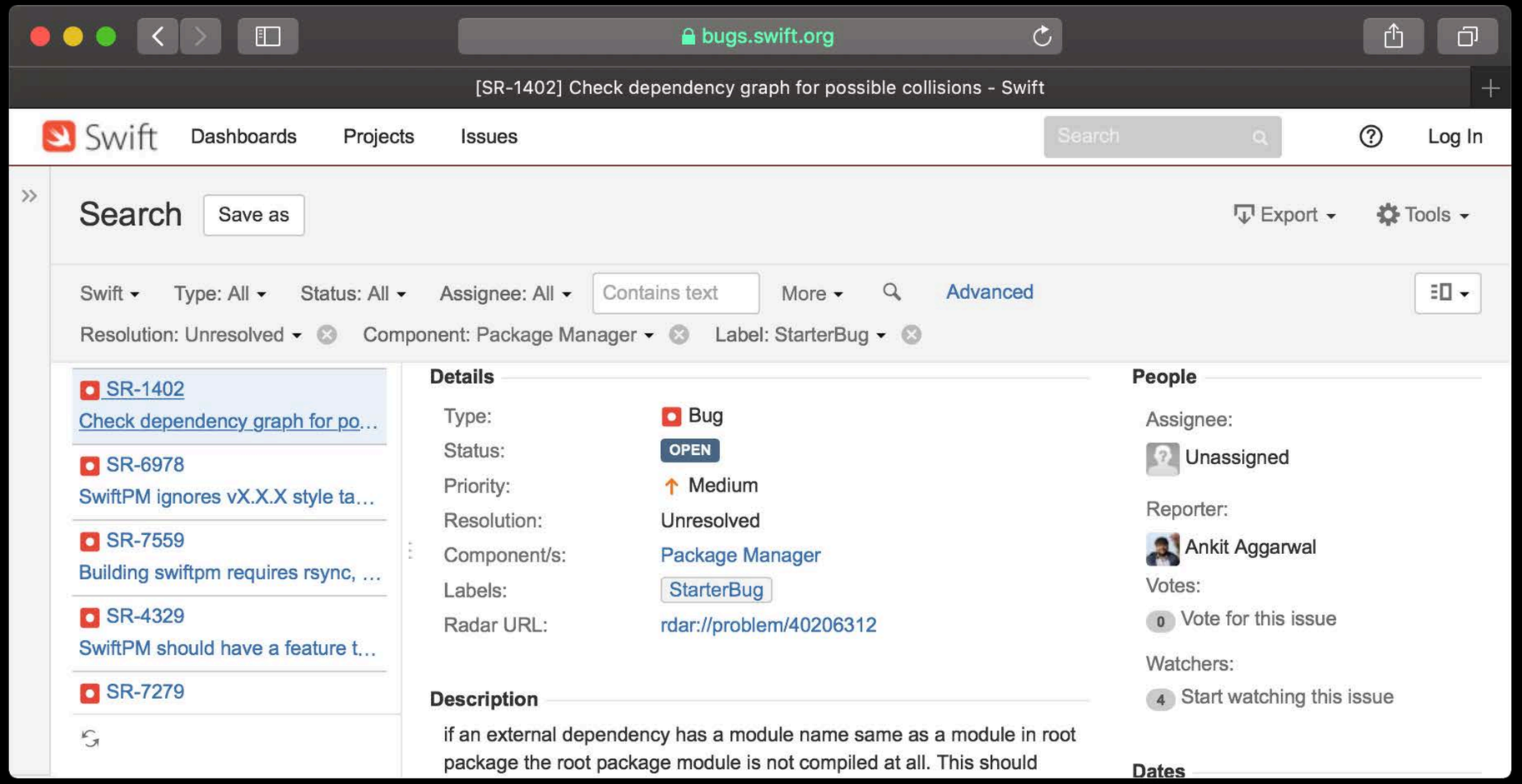
https://forums.swift.org/c/development/SwiftPM

Swift Bug Tracker



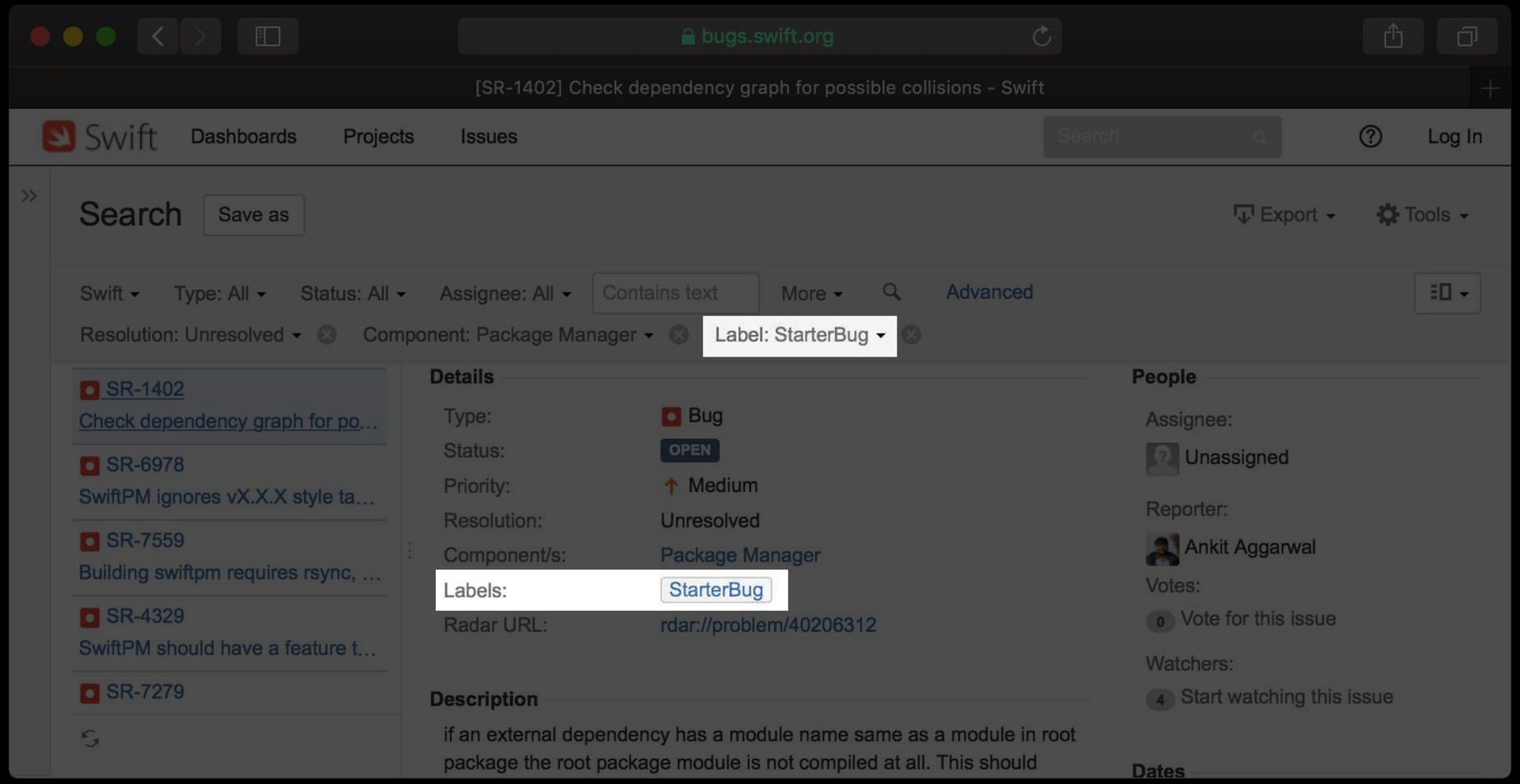
https://bugs.swift.org

Swift Bug Tracker—Starter Bugs



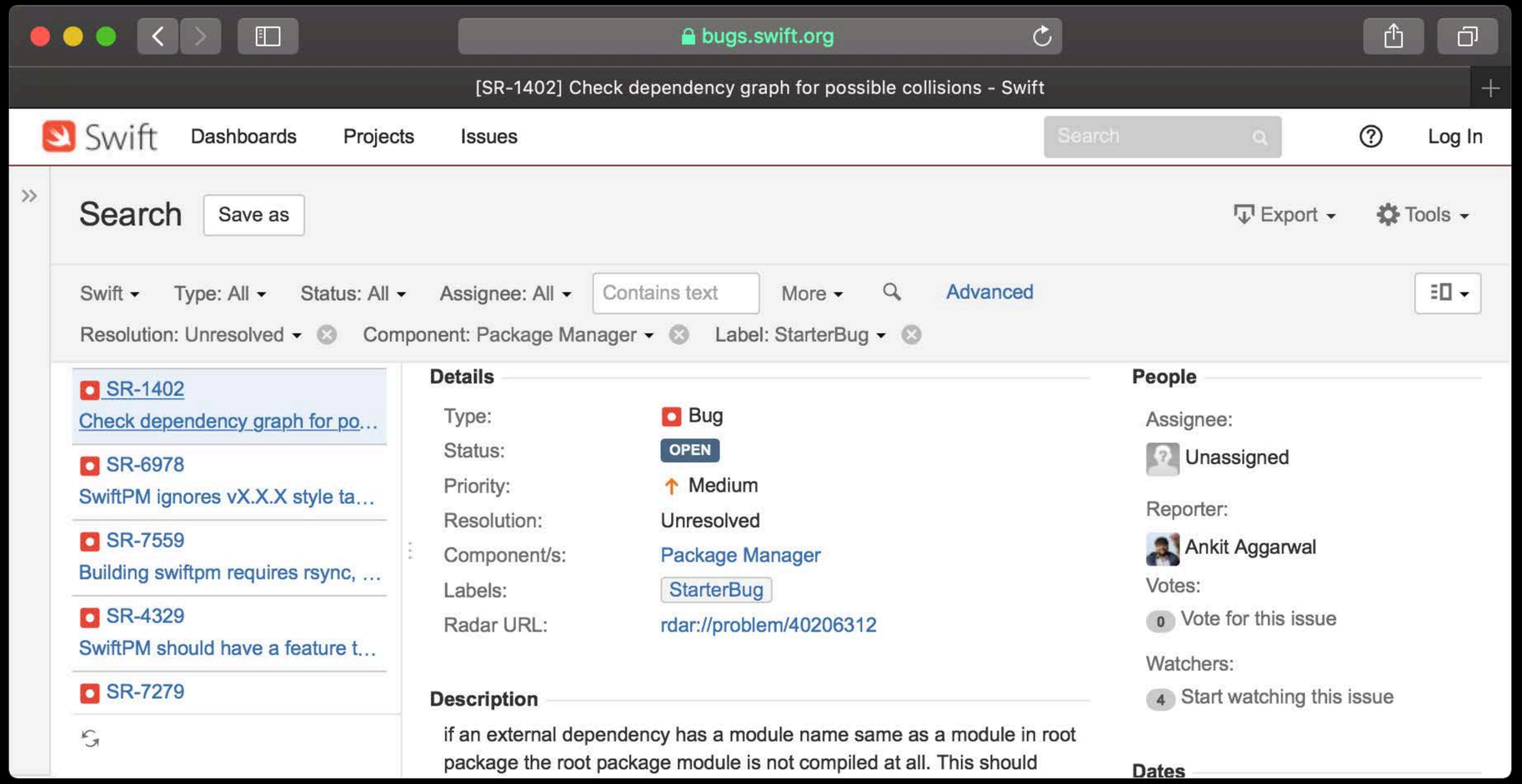
https://bugs.swift.org

Swift Bug Tracker—Starter Bugs



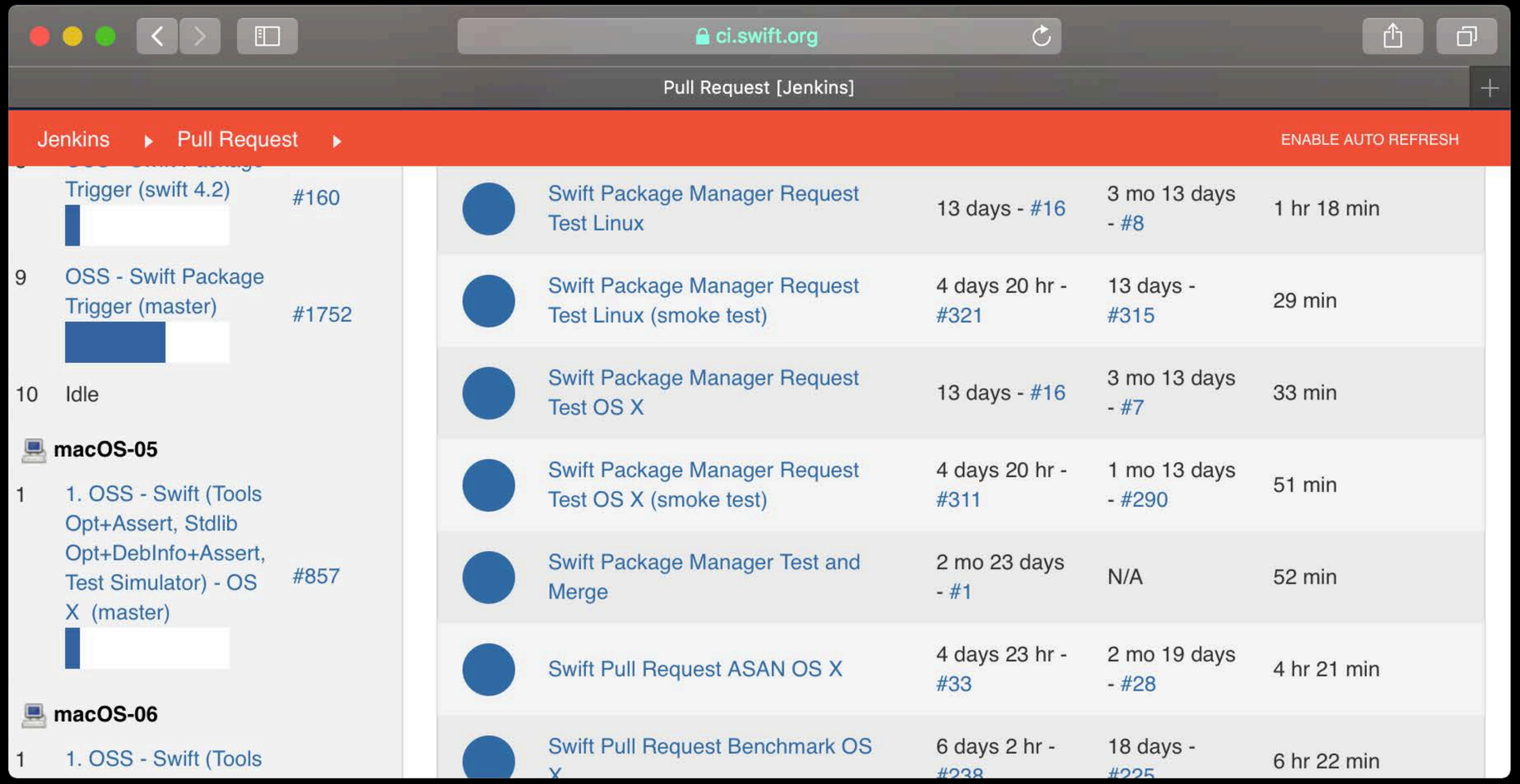
https://bugs.swift.org

Swift Bug Tracker—Starter Bugs



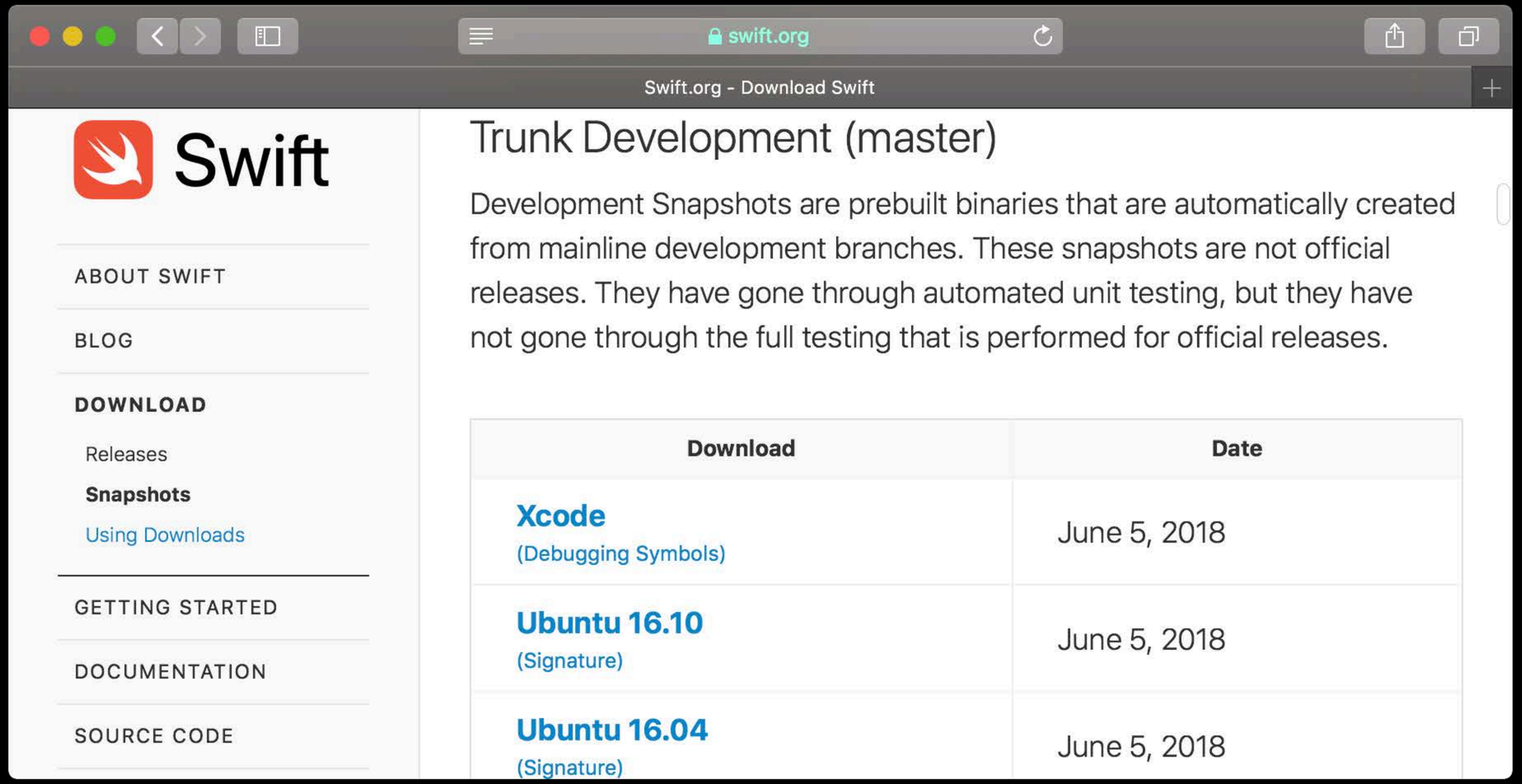
https://bugs.swift.org

Swift Continuous Integration



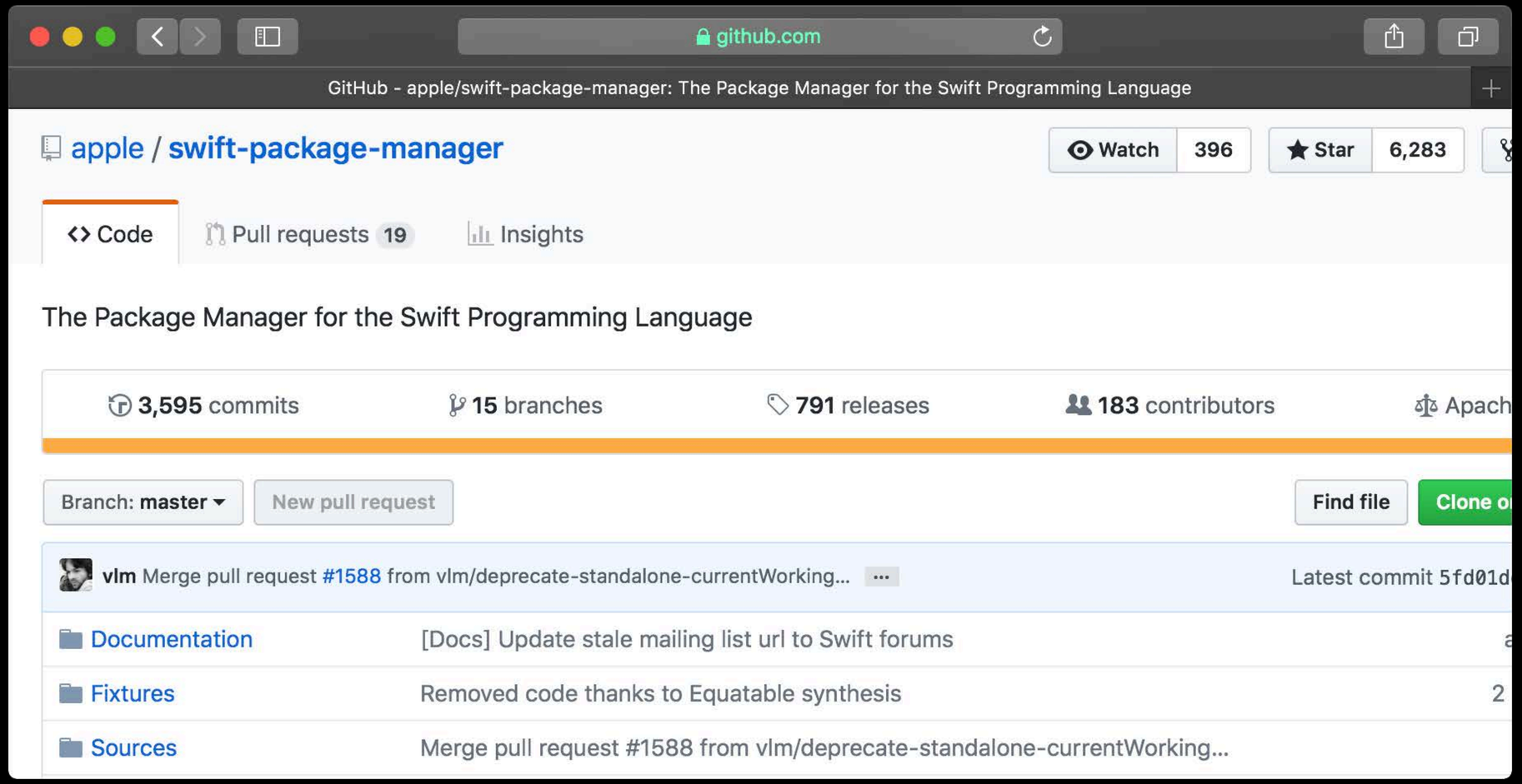
@swift-ci please test

Trunk Snapshots

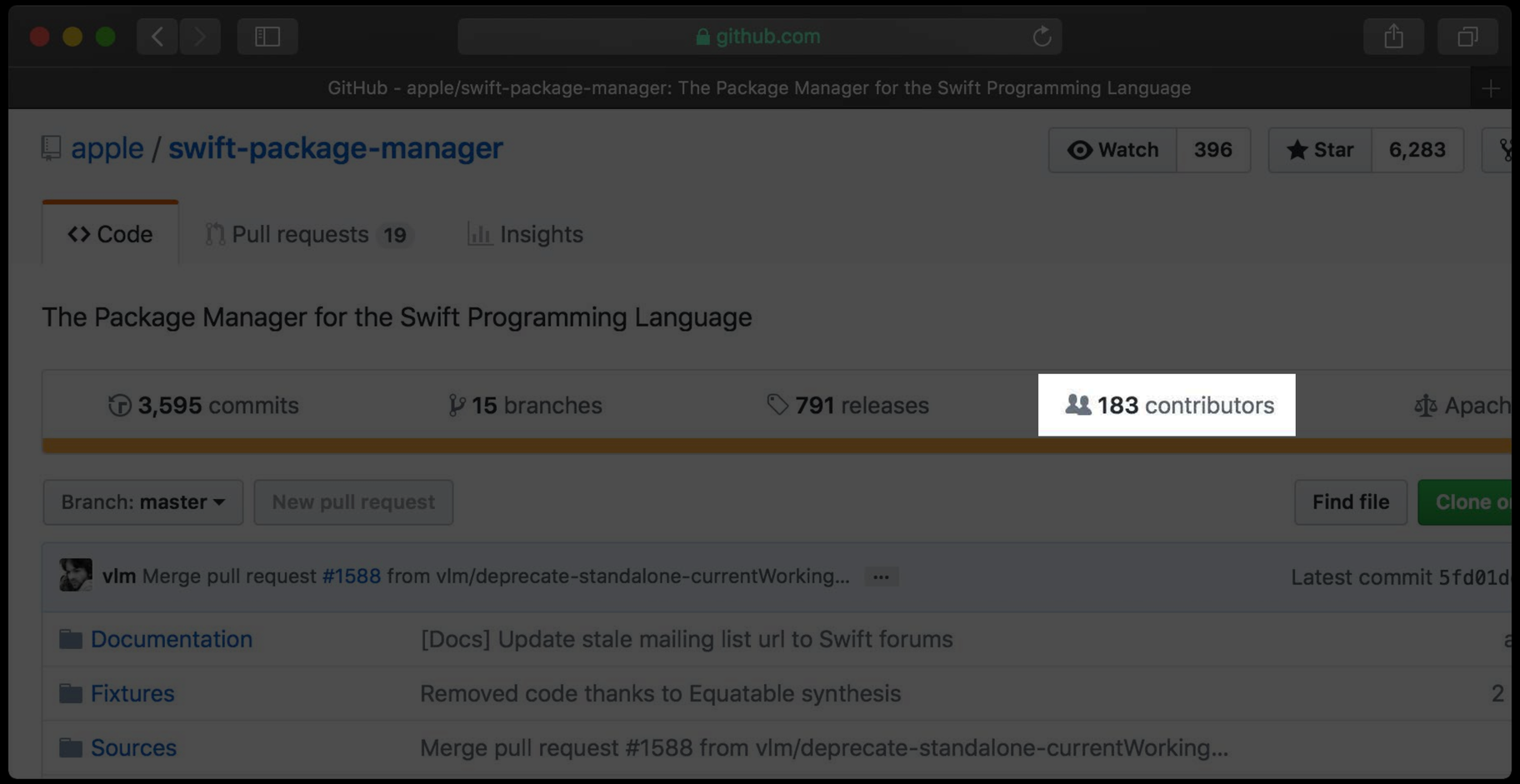


https://swift.org/download/#snapshots

Growing Community

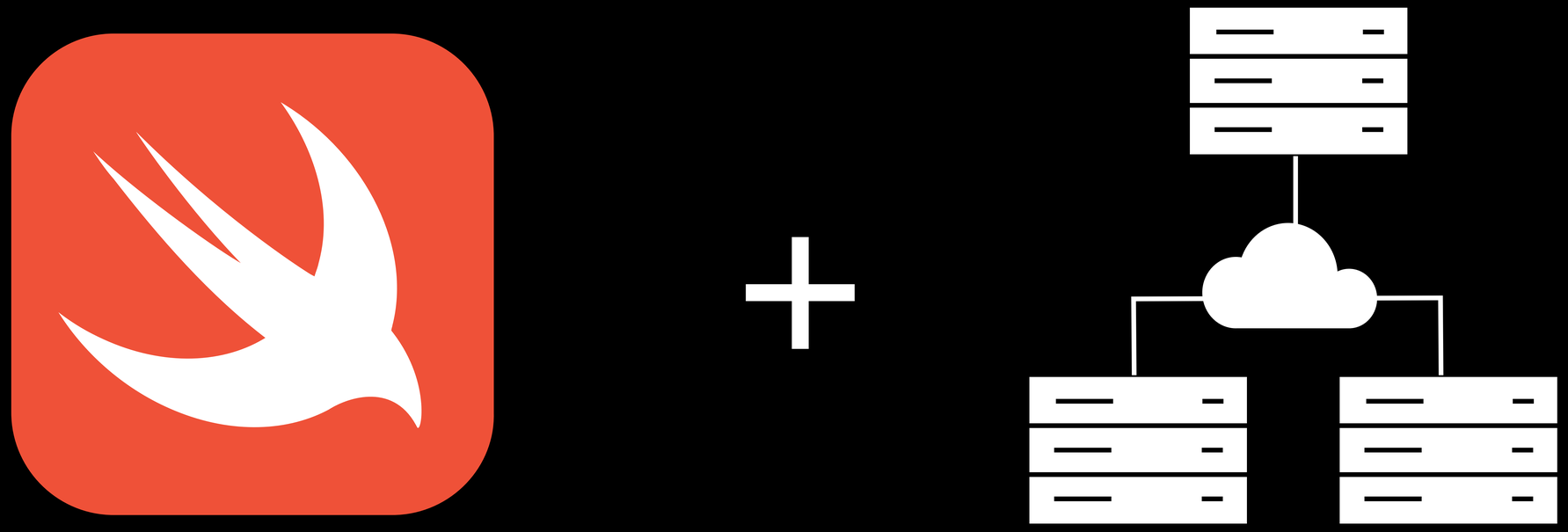


Growing Community

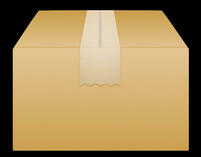


Server-Side Swift

Server-Side Swift



Command-Line Utilities



$ swift package init

More Information

https://developer.apple.com/wwdc18/411

Swift Open Hours	Technology Lab 10	Friday 3:00PM


WWDC18