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
Canonical Package Management Tool
Canonical Package Management Tool

- mac OS
- Ubuntu
- Future Platforms
Code Reuse Beyond the Core Libraries

- Foundation
- Dispatch
- XCTestCase
- 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

Swift.org - Package Manager

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

https://swift.org/package-manager/
Swift 4.2 Development

Swift 4.2 Snapshots are prebuilt binaries that are automatically created from swift-4.2-branch branch. These snapshots are not official releases. They have gone through automated unit testing, but they have not gone through the full testing that is performed for official releases.

<table>
<thead>
<tr>
<th>Download</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xcode</td>
<td>June 5, 2018</td>
</tr>
<tr>
<td>(Debugging Symbols)</td>
<td></td>
</tr>
<tr>
<td>Ubuntu 16.10</td>
<td>June 5, 2018</td>
</tr>
<tr>
<td>(Signature)</td>
<td></td>
</tr>
<tr>
<td>Ubuntu 16.04</td>
<td>June 5, 2018</td>
</tr>
<tr>
<td>(Signature)</td>
<td></td>
</tr>
</tbody>
</table>

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

$ swift build
$ swift run
$ swift test
$ swift package
Packages Use Git
Demo
Creating your first package
Anatomy of a Package
Anatomy of a Package

**Targets**

**Products**

**Dependencies**

Dependencies
Anatomy of a Package

Targets

Dependencies
Anatomy of a Package

- Targets
- Products
- Dependencies
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"],
    ]
)
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"]),
    ]
)
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
Packages Provide Products
Packages Provide Products

Linkage:
- Automatic
- Static
- Dynamic
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"]),
    ]
)
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"]),
    ]
)
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"],
    ]
)
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"]),
    ]
)
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"]),
    ]
)
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"],),
    ]
)
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"]),
    ]
)
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

- Configuration
- Dependencies
- Building
- Workflow Features
- Tools Evolution
Swift Language Manifest Format

Easy to understand

Follows Swift API design guidelines

Supported by existing Swift tools
Prefer Declarative Syntax

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

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

Sources
- dealer
  - main.swift
- libdealer
  - libdealer.swift
Convention versus Configuration

Sources
- dealer
- main.swift
- libdealer
- libdealer.swift

```swift
... products: [
  .library(name: "libdealer",
    targets: ["libdealer"]),
...

... targets: [
  .target(name: "libdealer",
    dependencies: ["DeckOfPlayingCards"]),
...```
Convention versus Configuration

**Sources**
- dealer
  - main.swift
- libdealer
  - libdealer.swift

```swift
... products: [
  .library(name: "libdealer",
            targets: ["libdealer"]),
... 
... targets: [
  .target(name: "libdealer",
           dependencies: ["DeckOfPlayingCards"]),
... 
```
Convention versus Configuration

Sources

- dealer
  - main.swift
  - libdealer
  - libdealer.swift

```swift
... products: [
  .library(name: "libdealer",
    targets: ["libdealer"]),
...
...
... targets: [
  .target(name: "libdealer",
    dependencies: ["DeckOfPlayingCards"]),
...```
Support for Building Other Languages

C  C++  m
Support for Building Other Languages

Swift
C
C++

Targets
The Design of SwiftPM

- Configuration
- Dependencies
- Workflow Features
- Building
- Tools Evolution
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

- **dealer**
  - **swiftpm.git**
    - exact("0.2.1")
  - **deck.git**
    - from: "3.0.0"
Resolving Versions

- `dealer`
  - `swiftpm.git`
    - `.exact("0.2.1")`
  - `deck.git`
    - `from: "3.0.0"`
Resolving Versions

- dealer
  - swiftpm.git
    - tag 0.2.1
  - deck.git
    - from: "3.0.0"
Resolving Versions

- dealer
  - swiftpm.git
    - tag 0.2.1
  - deck.git
    - tag 3.1.4
Transitive Dependencies

- Dealer
  - swiftpm.git (tag: 0.2.1)
  - deck.git (tag: 3.1.4)
Transitive Dependencies

- **dealer**
  - **swiftpm.git** tag 0.2.1
  - **fisher-yates.git** from: "2.0.0"
  - **playing-card.git** upToNextMinor: "3.0.0"
- **deck.git** tag 3.1.4
Transitive Dependencies

- dealer
  - swiftpm.git
    - tag 0.2.1
  - deck.git
    - tag 3.1.4
  - fisher-yates.git
    - from: "2.0.0"
  - playing-card.git
    - upToNextMinor: "3.0.0"
Transitive Dependencies

- dealer
  - swiftpm.git
    - tag 0.2.1
  - deck.git
    - tag 3.1.4
    - fisher-yates.git
      - tag 2.2.5
    - playing-card.git
      - upToNextMinor: "3.0.0"
Transitive Dependencies

- dealer
  - swiftpm.git
    - tag 0.2.1
  - deck.git
    - tag 3.1.4
- fisher-yates.git
  - tag 2.2.5
- playing-card.git
  - tag 3.0.2
Resolving Products

libdealer

swiftpm.git  deck.git

fisher-yates.git  playing-card.git
Resolving Products

libdealer
  └── Utility
      └── FisherYates
  └── DeckOfPlayingCards
      └── PlayingCard
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

- Configuration
- Dependencies
- Building
- Workflow Features
- Tools Evolution
Ilbuild

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

- Configuration
- Dependencies
- Building
- Tools Evolution
- Workflow Features
Edit Mode
Branch Dependencies

from: "1.0.0"

.branch("master")

.branch("master")
Local Package Dependencies
The Design of SwiftPM

- Configuration
- Dependencies
- Building
- Workflow Features
- Tools Evolution
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:4.2

import PackageDescription

let package = Package(..., swiftLanguageVersions: [.v4_2, .v4])
// swift-tools-version:4.2
import PackageDescription

let package = Package(…, swiftLanguageVersions: [.v4_2, .v4])
The Design of SwiftPM

- Configuration
- Dependencies
- Building
- Tools Evolution
- Workflow Features
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
let package = Package(
    name: "Networking",
    products: [
        .library(name: "Networking", targets: ["Networking"])
    ],
    targets: [  
        .target(  
            name: "Networking",
            dependencies: []
        ),
    ]
)
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

$ git tag 2.1.9

$ git push origin 2.1.9
Idea: Automatic Semantic Versioning

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

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

- 1.6.0
- 2.0.0
Idea: Deployment Automation
Idea: Deployment Automation
Support Complex Packages
Idea: Build Settings
Idea: Extensible Build Tools

DocGen
Trust, Manage, and Find Packages
Idea: Package Content Verification
Idea: Cross-Platform Sandboxing
Idea: Fork Support
Idea: Package Index

- AcmeUtils
- NiftyNetworking
- YetAnotherProtocol
- NextGreatThing
Your Contributions

- Build Settings
- Extensible Build Tools
- Resources
- Deployment Automation
- Verify Expected Package Content
- Cross-Platform Sandboxing
- Fork Support
- Automatic SemVer
- Machine-Editable Package.swift
- Tag & Publish
- Leverage libSwiftPM
- Package Index
Why a package manager for Swift?
How to use it
The design of SwiftPM
Evolution ideas
Open source process
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

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

Swift Programming Language Evolution

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.

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

[SR-1402] Check dependency graph for possible collisions - Swift

SR-1402
Check dependency graph for possible collisions.

SR-6978
SwiftPM ignores vX.X.X style tags.

SR-7559
Building swiftpm requires re-sync, ...

SR-4329
SwiftPM should have a feature to ...

SR-7279

Description
If an external dependency has a module name same as a module in root package the root package module is not compiled at all. This should

https://bugs.swift.org
Swift Continuous Integration

@swift-ci please test
Trunk Snapshots

Development Snapshots are prebuilt binaries that are automatically created from mainline development branches. These snapshots are not official releases. They have gone through automated unit testing, but they have not gone through the full testing that is performed for official releases.

<table>
<thead>
<tr>
<th>Download</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xcode (Debugging Symbols)</td>
<td>June 5, 2018</td>
</tr>
<tr>
<td>Ubuntu 16.10 (Signature)</td>
<td>June 5, 2018</td>
</tr>
<tr>
<td>Ubuntu 16.04 (Signature)</td>
<td>June 5, 2018</td>
</tr>
</tbody>
</table>

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

GitHub - apple/swift-package-manager: The Package Manager for the Swift Programming Language

apple / swift-package-manager

Branch: master

Latest commit 5fd81d

Documentation
- [Docs] Update stale mailing list url to Swift forums

Fixtures
- Removed code thanks to Equatable synthesis

Sources
- Merge pull request #1588 from vlm/deprecate-standalone-currentWorking...
Growing Community
Server-Side Swift
Server-Side Swift
Command-Line Utilities

```plaintext
exec
```
$ swift package init
More Information

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

Swift Open Hours

Technology Lab 10

Friday 3:00PM