
ONEVCAT (ONEV@ONEVCAT.COM)


ONEVCAT (ONEV@ONEVCAT.COM)

# 如何打造一个让人愉快的

# 小技

会议主席


我

但考虑到这是一次开发者会议...

# 如何打造一个让人愉快的

# 框架

# 一个故事

# 武田君的轮子

- 网络请求
- 模型解析
- 导航效果
- 视图动画

...

珍藏本

汉译世界学术名著丛书

# 自杀论

〔法〕埃米尔·迪尔凯姆 著

商务印书馆
The Commercial Press

以上故事纯属虚构

如有雷同实属巧合

如何又好又快地开发 APP

# 迅速反应

# 代码重用

# 使用框架

# 远古时代

# COCOA TOUCH LIBRARY

忘不了 那些年

被手动引用和 .a 文件

所支配的恐惧

# 什么是静态库

## (STATIC LIBRARY)

```
GoogleAdMobAdsSdkiOS-6.5.1 % tree .
.
├── GADAdMobExtras.h
├── GADAdNetworkExtras.h
├── GADAdSize.h
├── GADBannerView.h
├── GADBannerViewDelegate.h
├── GADInterstitial.h
├── GADInterstitialDelegate.h
├── GADRequest.h
├── GADRequestError.h
├── README.txt
└── libGoogleAdMobAds.a

0 directories, 11 files
GoogleAdMobAdsSdkiOS-6.5.1 %
```

# 什么是静态库

## (STATIC LIBRARY)

.a + .h

**目标文件(OBJECT FILES. .O)的集合**

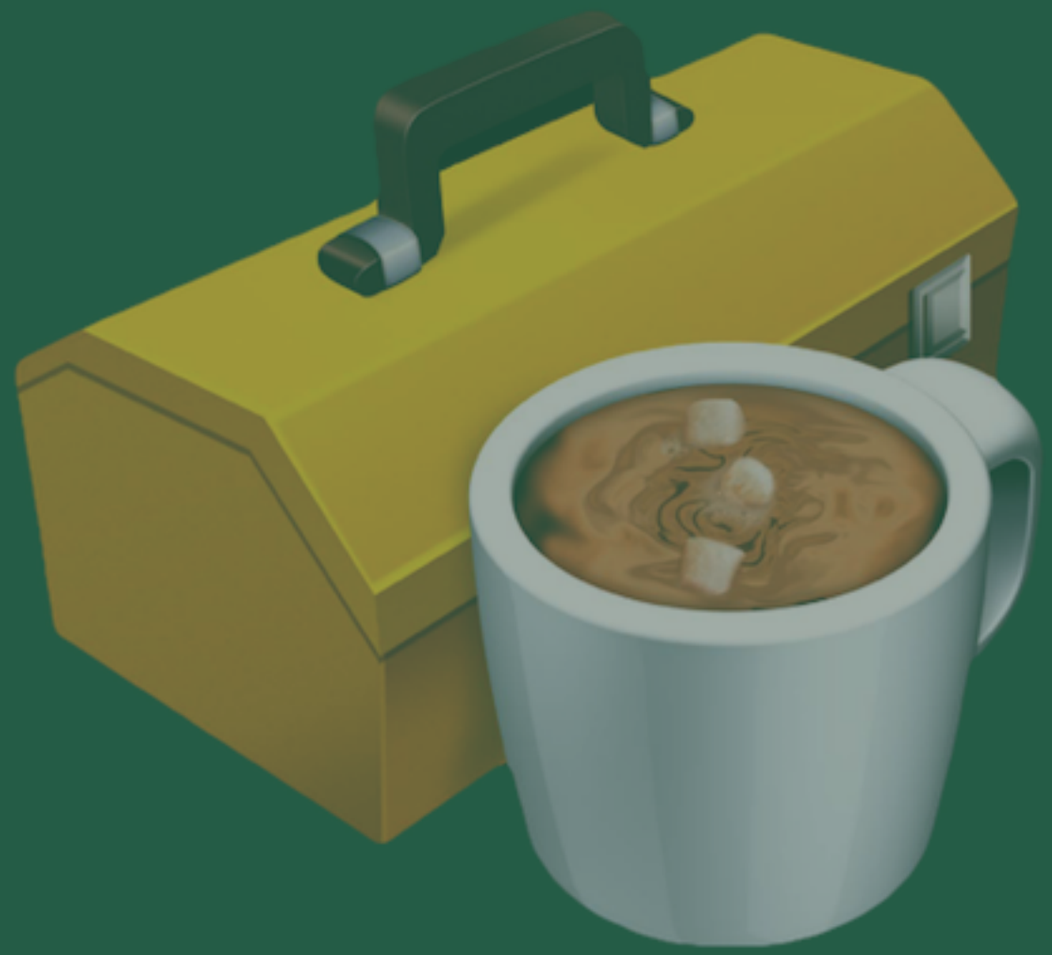
**配合若干头文件(.H)使用**

**链接到最终的可执行文件中**

**在一些上了年头的第三方库中可能还能见到**

AddressBookUI.framework
AdSupport.framework
AssetsLibrary.framework
AudioToolbox.framework
AudioUnit.framework
AVFoundation.framework
AVKit.framework
CFNetwork.framework
CloudKit.framework
Contacts.framework
ContactsUI.framework
CoreAudio.framework
CoreAudioKit.framework
CoreBluetooth.framework
CoreData.framework
CoreFoundation.framework
CoreGraphics.framework
CoreImage.framework
CoreLocation.framework
CoreMedia.framework
CoreMIDI.framework

# 系统框架 .FRAMEWORK



# 什么是动态框架

# (DYNAMIC FRAMEWORK)

存在于系统内部
已经链接好的 IMAGE
**运行时通过 dyld 加载**
**不需要重新加载**

直到 iOS 8，只有 Apple 制作的框架才能使用动态方式

# UNIVERSAL FRAMEWORK

以前的一些第三方框架也提供 .FRAMEWORK 文件

实质上是打包的静态库[1][2]

[1] HTTPS://GITHUB.COM/KSTENERUD/IOS-UNIVERSAL-FRAMEWORK

[2] HTTPS://GITHUB.COM/JVERKOEY/IOS-FRAMEWORK

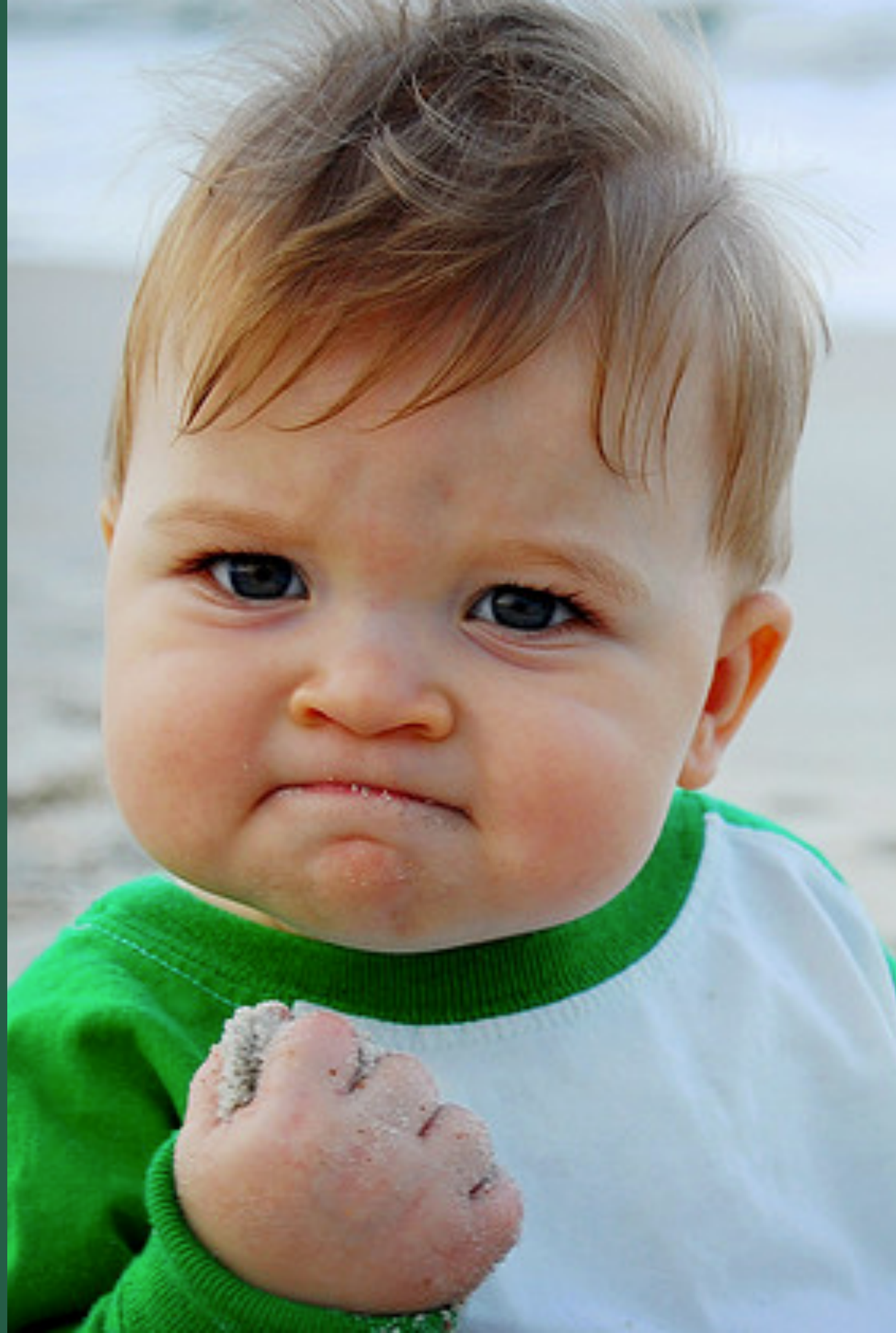
# LIBRARY V.S. FRAMEWORK

# COCOA TOUCH FRAMEWORK

IOS 8 开始能创建动态框架

SWIFT 框架的唯一选择

(因为 RUNTIME 限制)



# 包和依赖管理

# COCOAPODS[3]

[3] HTTP://COCOAPODS.ORG

# 以前

workspace + libPod.a

## 通过 PODSPEC 来标识公开项目的信息

## 和位置

## 修改项目文件、BUILD PHASE 和脚本构建静态库

## '侵入式'的集成方式

# 最近

## FROM 0.36.0 - SWIFT 和动态框架支持

`use_frameworks!`

# 最近

## FROM 0.36.0 - SWIFT 和动态框架支持

`use_frameworks!`

## 这个选项会将项目依赖全部改为

## FRAMEWORK

NONE OR ALL

# 使用 COCOAPODS

```
# Podfile
platform :ios, '8.0'
use_frameworks!

target 'MyApp' do
  pod 'AFNetworking', '~> 2.6'
  pod 'ORStackView', '~> 3.0'
  pod 'SwiftyJSON', '~> 2.3'
end

$ pod install
```

# CARTHAGE[4]

[4] https://github.com/Carthage/Carthage

仅支持 FRAMEWORK

非侵入，不改变项目文件

去中心化，直接从 GIT 仓库获取

# 使用 CARTHAGE

```
# Cartfile
github "ReactiveCocoa/ReactiveCocoa"
github "onevcat/Kingfisher" ~> 1.8
github "https://enterprise.local/hello/repo.git"
```

```
$ carthage update
```

Kingfisher | Build Kingfisher-Demo: **Succeeded**

Kingfisher

General | Capabilities | Resource Tags | Inf

**PROJECT**

Kingfisher

**TARGETS**

Kingfisher-Demo

Kingfisher

KingfisherTests

Kingfisher-tvOS-De...

Kingfisher-tvOS

Device Orientation

Status Bar Style

**App Icons and Launch Images**

App Icons Source

Launch Images Source

Launch Screen File

**Embedded Binaries**

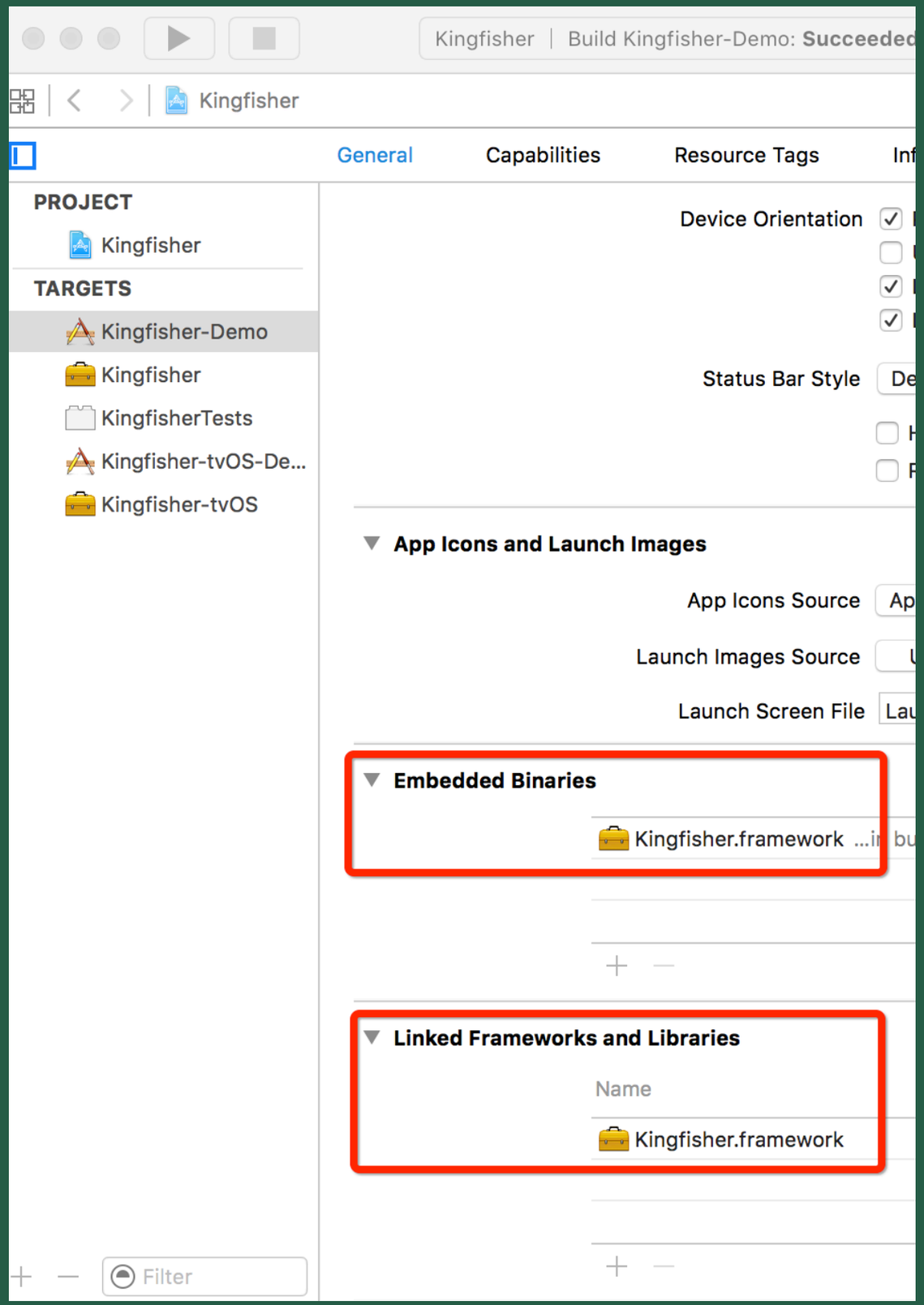
Kingfisher.framework

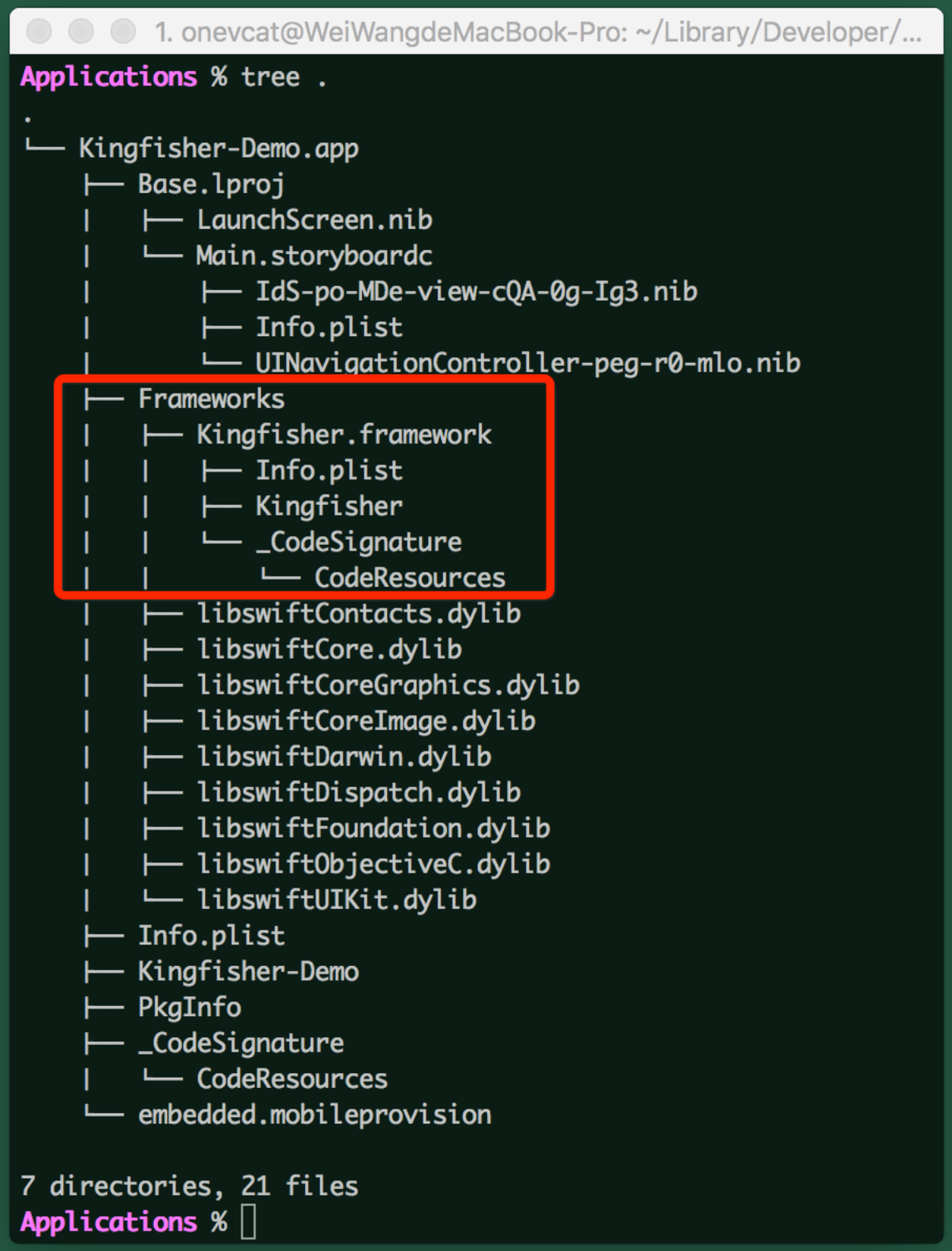
**Linked Frameworks and Libraries**

Name

Kingfisher.framework

Filter


Kingfisher | Build Kingfisher-Demo: Succeeded
Kingfisher
General
Capabilities
Resource Tags
PROJECT
Kingfisher
TARGETS
Kingfisher-Demo
Kingfisher
KingfisherTests
Kingfisher-tvOS-De...
Kingfisher-tvOS
Device Orientation
Status Bar Style
App Icons and Launch Images
App Icons Source
Launch Images Source
Launch Screen File
Embedded Binaries
Kingfisher.framework
Linked Frameworks and Libraries
Name
Kingfisher.framework
Filter


1. onevcat@WeiWangdeMacBook-Pro: ~/Library/Developer/...
Applications % tree .
.
└── Kingfisher-Demo.app
    ├── Base.lproj
    │   ├── LaunchScreen.nib
    │   └── Main.storyboardc
    │       ├── IdS-po-MDe-view-cQA-0g-Ig3.nib
    │       ├── Info.plist
    │       └── UINavigationController-peg-r0-mlo.nib
    ├── Frameworks
    │   ├── Kingfisher.framework
    │   │   ├── Info.plist
    │   │   ├── Kingfisher
    │   │   └── _CodeSignature
    │   │       └── CodeResources
    │   ├── libswiftContacts.dylib
    │   ├── libswiftCore.dylib
    │   ├── libswiftCoreGraphics.dylib
    │   ├── libswiftCoreImage.dylib
    │   ├── libswiftDarwin.dylib
    │   ├── libswiftDispatch.dylib
    │   ├── libswiftFoundation.dylib
    │   ├── libswiftObjectiveC.dylib
    │   └── libswiftUIKit.dylib
    ├── Info.plist
    ├── Kingfisher-Demo
    ├── PkgInfo
    ├── _CodeSignature
    │   └── CodeResources
    └── embedded.mobileprovision
7 directories, 21 files
Applications %

# SWIFT PACKAGE MANAGER[5]

[5] https://swift.org/package-manager/

# 在 SWIFT 中使用框架

- `.framework` - 直接 `import XYZ` 导入 MODULE
- `.a + .h` - BRIDGING HEADER #IMPORT "XYZ.H"

# 創建框架

Choose a template for your new project:

- iOS
  - Application
  - Framework & Library
- watchOS
  - Application
  - Framework & Library
- tvOS
  - Application
  - Framework & Library
- OS X
  - Application
  - Framework & Library
  - System Plug-in
- Other

Cocoa Touch Framework


Cocoa Touch Static Library

**Cocoa Touch Framework**

This template creates a framework that uses UIKit.

Cancel | Previous | Next

# 一些诀窍

# API 设计

## 考虑提供给开发者的内容

# API 设计

## 考虑提供给开发者的内容

### 尽可能小的访问权限

```
// Do this
public func mustMethod() { ... }
func onlyUsedInFramework() { ... }
private func onlyUsedInFile() { ... }

// Don't do this
public func mustMethod() { ... }
public func onlyUsedInFramework() { ... }
public func onlyUsedInFile() { ... }
```

# API 设计

## 命名，是否清晰易懂完整

# API 设计

## 命名，是否清晰易懂完整

### 清晰明确的命名

```
// Do this
public mutating func removeAt(position: Index) -> Element
public func recursivelyFetch(urls: [(String, Range<Version>)]) throws -> [T]
public var displayName: String
public var screenName: String // <- Better

// Don't do this
public mutating func remove(i: Int) -> Element // <- index or element?
public func fetch(urls: [(String, Range<Version>)]) throws -> [T] // <- how?
public func displayName() -> String // <- none or verb?
```

# API 设计

## 命名，是否清晰易懂完整

## 尝试为你的方法写注释文档

一句话无法表述清楚这个方法具体做了什么 -> 请考虑重构

## 理想状态：代码不需要文档就能被看懂

# API 设计

## 命名，是否清晰易懂完整

# SWIFT API DESIGN GUIDELINES

HTTPS://SWIFT.ORG/DOCUMENTATION/API-DESIGN-GUIDELINES.HTML

# API 设计

## 优先测试，测试驱动开发

# API 设计

## 优先测试，测试驱动开发

```
// In Test Target
import XCTest
@testable import YourFramework
class FrameworkTypeTests: XCTestCase {
    // ...
}
```

# 开发时的选择

# 命名冲突

```
// F1.framework
extension UIImage {
    public method() { print("F1") }
}

// F2.framework
extension UIImage {
    public method() { print("F2") }
}
```

# 开发时的选择

# 命名冲突 - CASE 1

```
// app
import F1
import F2
UIImage().method()
// Ambiguous use of 'method()'
```

# 开发时的选择

# 命名冲突 - CASE 2

```
// app
import F1
UIImage().method()
// 输出 F2 （结果不确定）
```

# 开发时的选择

# 命名冲突 - CASE 3

```
// app
extension UIImage {
    public func method() {
        print("app")
    }
}

// F1
UIImage().method()
// app
```

开发时的选择

# 对于 COCOA 类型的 extension

# 必须添加前缀

# 开发时的选择

# 命名冲突

```
// Don't do this
// F1.framework
extension UIImage {
    public method() { print("F1") }
}

// F2.framework
extension UIImage {
    public method() { print("F2") }
}
```

# 开发时的选择

# 命名冲突

```
// Do this
// F1.framework
extension UIImage {
    public f1_method() { print("F1") }
}

// F2.framework
extension UIImage {
    public f2_method() { print("F2") }
}
```

# 开发时的选择

## 资源 BUNDLE

.framework 可以包含资源文件

通过 NSBundle 进行访问

# 开发时的选择

## 资源 BUNDLE

```
let bundle =
    NSBundle(forClass: ClassInFramework.self)
let path =
    bundle.pathForResource("resource", ofType: "png")
```

YourApp.app/Frameworks/YourFramework.framework/
resource.png

# 发布框架

# GITHUB

# COCOAPODS

```
pod spec create MyFramework
```

```
Pod::Spec.new do |s|
  s.name             = "MyFramework"
  s.version          = "1.0.2"
  s.summary          = "My first framework"
  s.description      = <<-DESC
                        It's my first framework.
                       DESC
  s.ios.deployment_target = "8.0"
  s.source           = { :git => "https://github.com/onevcat/myframework.git",
                         :tag => s.version }

  s.source_files     = "Class/*.{h,swift}"
  s.public_header_files = ["MyFramework/MyFramework.h"]
end
```

# COCOAPODS

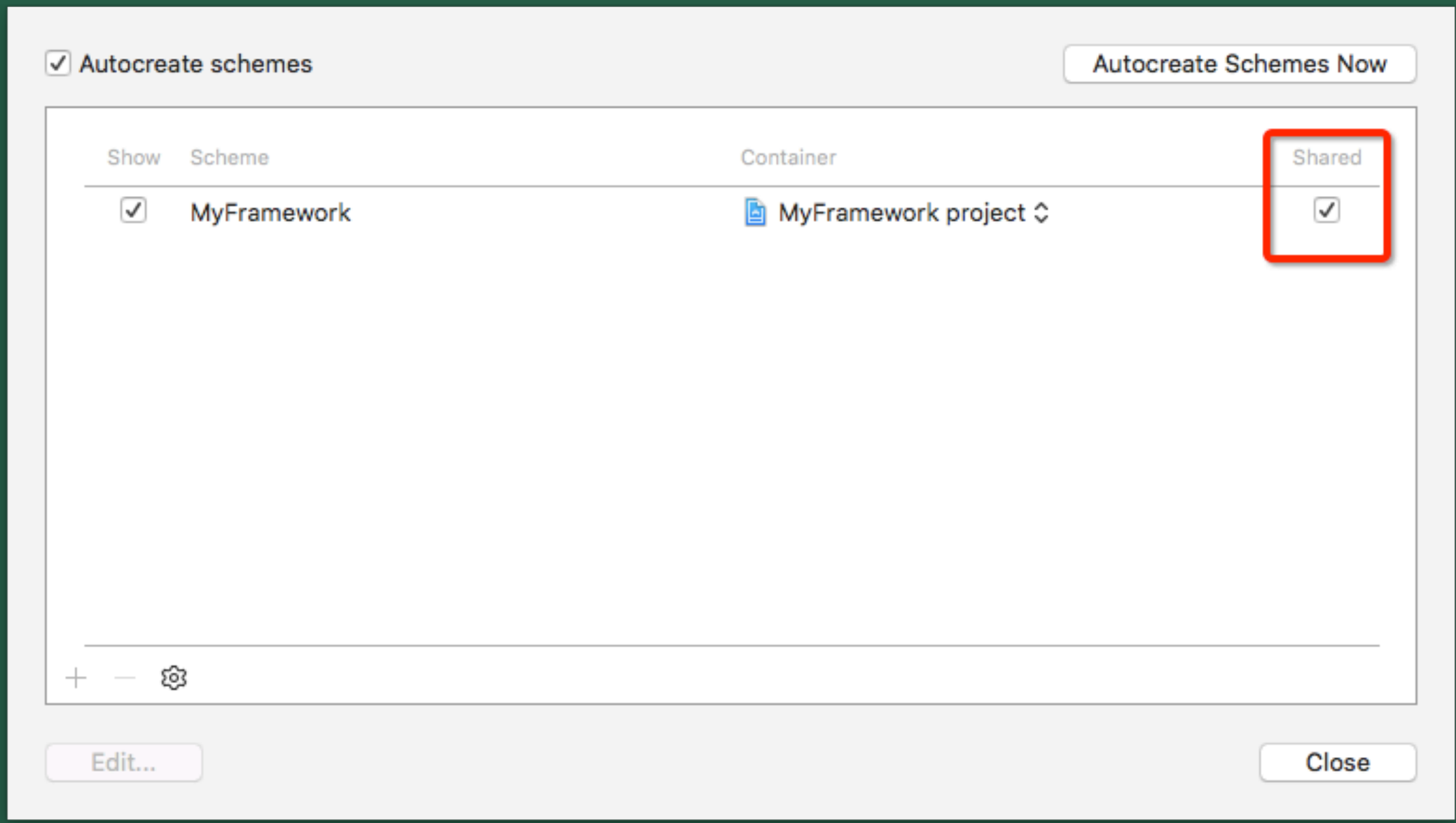
## 提交到 COCOAPODS

```
# 打 tag
git tag 1.0.2 && git push origin --tags

# podspec 文法检查
pod spec lint MyFramework.podspec

# 提交到 CocoaPods 中心仓库
pod trunk push MyFramework.podspec
```

# CARTHAGE


Autocreate schemes
Autocreate Schemes Now
Show
Scheme
Container
Shared
MyFramework
MyFramework project
Edit...
Close

# SWIFT PM

## 将源文件放入 Sources 中

```
// Package.swift
import PackageDescription
let package = Package(
    name: "MyKit",
    dependencies: [
        .Package(url: "https://github.com/onevcat/anotherPacakge.git",
                 majorVersion: 1)
    ]
)
```

# 版本管理

```
# Podfile
pod 'AFNetworking', '~> 2.6.1'
# 2.6.x 兼容 (2.6.1, 2.6.2, 2.6.9 等，不包含 2.7)
```

```
# Podfile
pod 'AFNetworking', '~> 2.6'
# 2.x 兼容 (2.6.1, 2.7, 2.8 等，不包含 3.0)


# Cartfile
github "Mantle/Mantle" >= 1.1
# 大于等于 1.1 (1.1，1.1.4, 1.3, 2.1 等)
```

# 版本兼容

## SEMANTIC VERSIONING[8]

x(major).y(minor).z(patch)

[8] HTTP://SEMVER.ORG

# 版本兼容

- MAJOR - 公共 API 改动或者删减
- MINOR - 新添加了公共 API
- PATCH - BUG 修正等

# 版本兼容

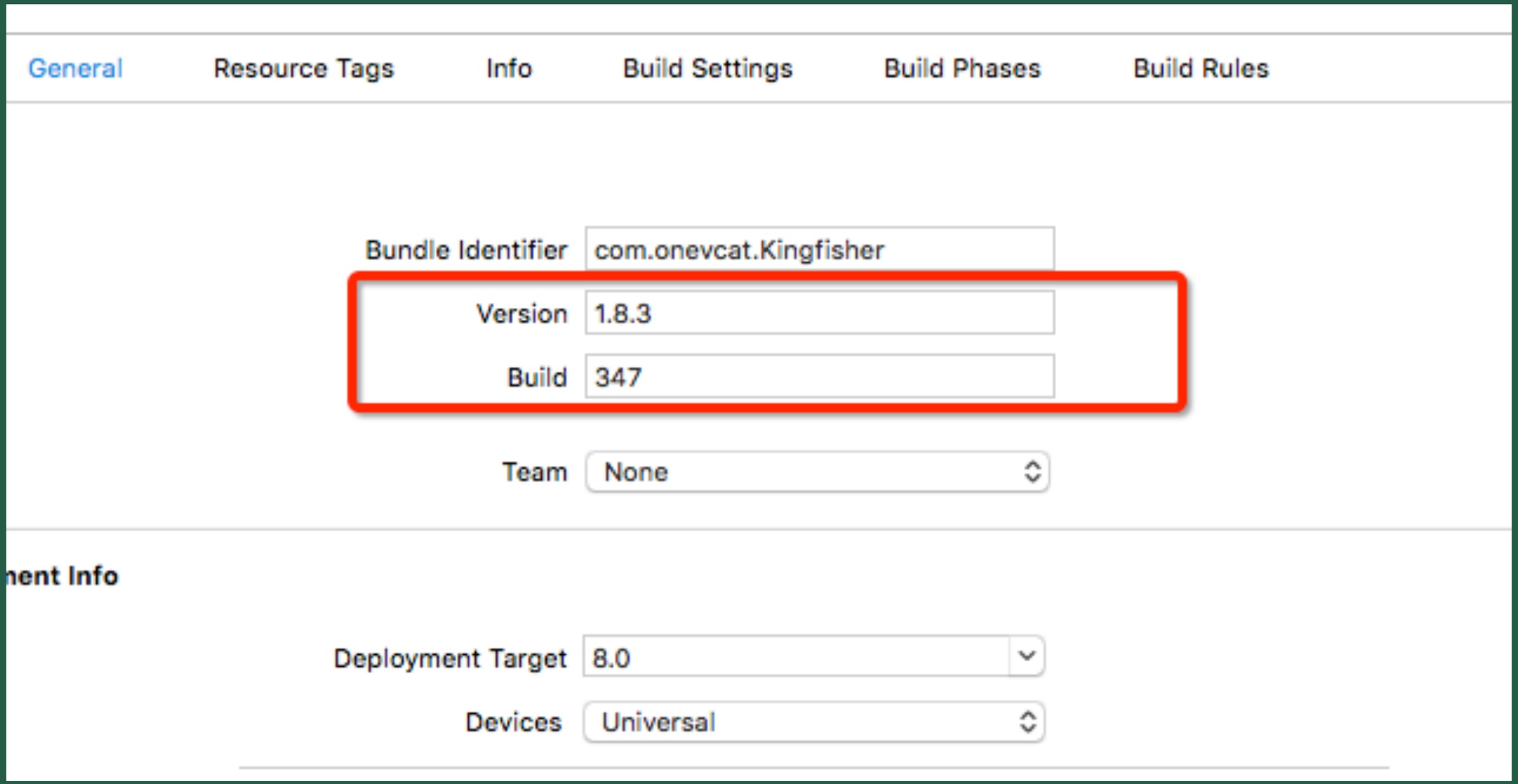
- MAJOR - 公共 API 改动或者删减
- MINOR - 新添加了公共 API
- PATCH - BUG 修正等

0.x.y 只遵守最后一条

# 版本兼容

使用 GIT TAG 进行版本标记
你的用户/包管理系统期望合理的兼容版本

# 版本兼容




我们在设置版本的时候可能会注意

# 版本兼容

```
// MyFramework.h
//! Project version string for MyFramework.
FOUNDATION_EXPORT const unsigned char MyFrameworkVersionString[]; // 1.8.3
//! Project version number for MyFramework.
FOUNDATION_EXPORT double MyFrameworkVersionNumber; // 347
```

```
// Exported module map
//! Project version number for MyFramework.
public var MyFrameworkVersionNumber: Double
// 并没有导出 MyFrameworkVersionString
```

# 持续集成

## 选择合适的 CI 环境

TRAVIS CI. CIRCLE CI. COVERALLS. CODECOV...

# 自动化的发布流程

# FASTLANE[9]

[9] HTTPS://FASTLANE.TOOLS

# 自动化的发布流程

```
# Fastfile
desc "Release new version"
lane :release do |options|
  target_version = options[:version]
  raise "The version is missed." if target_version.nil?
  ensure_git_branch                                                    # 确认 master 分支
  ensure_git_status_clean                                              # 确认没有未提交的文件
  scan                                                                 # 运行测试

  sync_build_number_to_git                                             # 将 build 号设为 git commit 数
  increment_version_number(version_number: target_version)             # 设置版本号

  version_bump_podspec(path: "Kingfisher.podspec",
             version_number: target_version)                           # 更新 podspec
  git_commit_all(message: "Bump version to #{target_version}")         # 提交版本号修改
  add_git_tag tag: target_version                                      # 设置 tag
  push_to_git_remote                                                   # 推送到 git 仓库
  pod_push                                                             # 提交到 CocoaPods
end

$ fastlane release version:1.8.4
```

# 自动化的发布流程

## 另外的例子：

## AFNETWORKING/FASTLANE

# 创建一个优秀的框架

文档，注释，测试，代码质量，
更新日志，ISSUE 响应速度…

# 不服跑个分

Steve Leibos

# 创建一个优秀的框架

文档，注释，测试，代码质量，
更新日志，ISSUE 响应速度…

# COCOAPODS QUALITY

HTTPS://COCOAPODS.ORG/PODS/KINGFISHER/QUALITY

# QUALITY INDEXES GUIDE

# 可能的问题

## 兼容性保证 (DATABASE， KEY-ARCHIVING)

# 可能的问题

## 重复包含

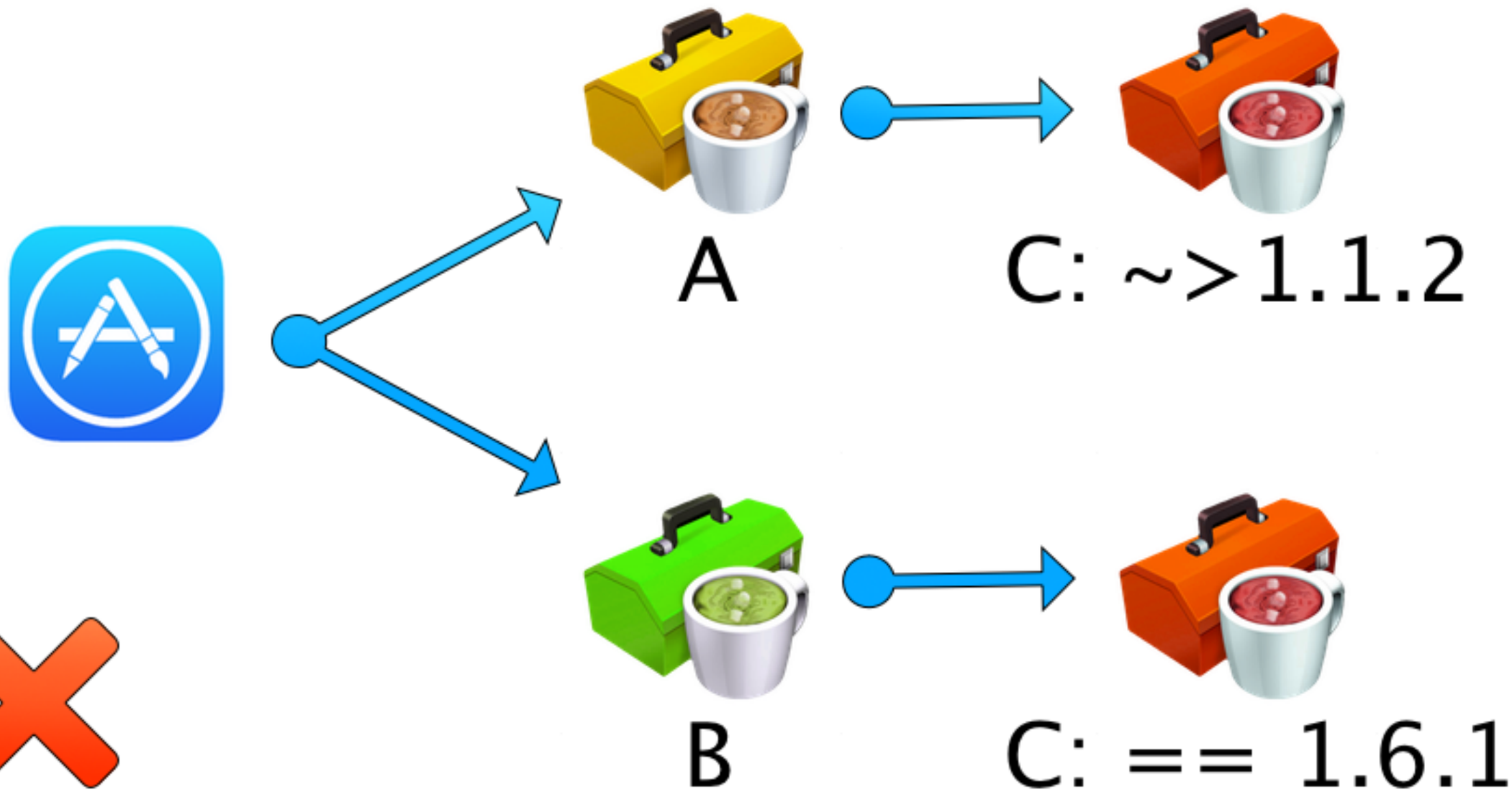
将 EMBEDDED_CONTENT_CONTAINS_SWIFT
设置为 NO

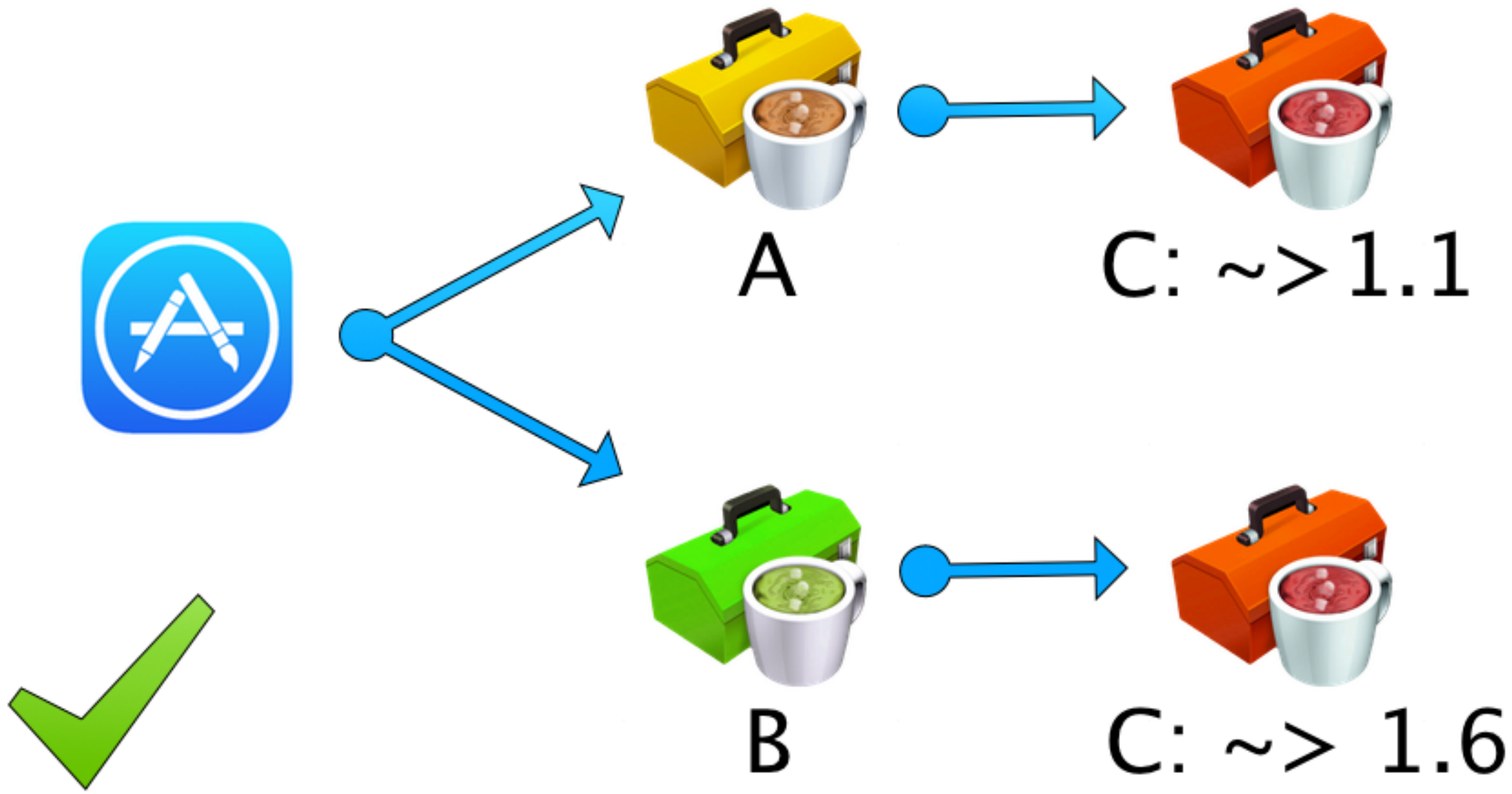
## 不要将依赖的 FRAMEWORK COPY 到 FRAMEWORK 中

# 可能的问题

## 无法兼容的框架依赖[10]

[10] DEPENDENCY HELLS


A
C: ~>1.1.2
B
C: == 1.6.1


A
C: ~>1.1
B
C: ~> 1.6

# 可能的问题

## FRAMEWORK 加载速度[11] [12]

[11] APP LAUNCH TIME INCREASED

[12] RDAR://22948371

# 可能的问题

## SWIFT 版本更新

# 从今天开始开发框架

# WRITE THE CODE
# CHANGE THE WORLD

# QUESTIONS

ONEVCAT (ONEV@ONEVCAT.COM)