

# CocoaPods了解

📖 CocoaPods 使用指南

[https://study.bytedance.net/course\\_play/131](https://study.bytedance.net/course_play/131) bootcamp课程，讲得很细

<https://juejin.im/post/5aed54876fb9a07ab37953b6> 你真的会写PodFile吗？

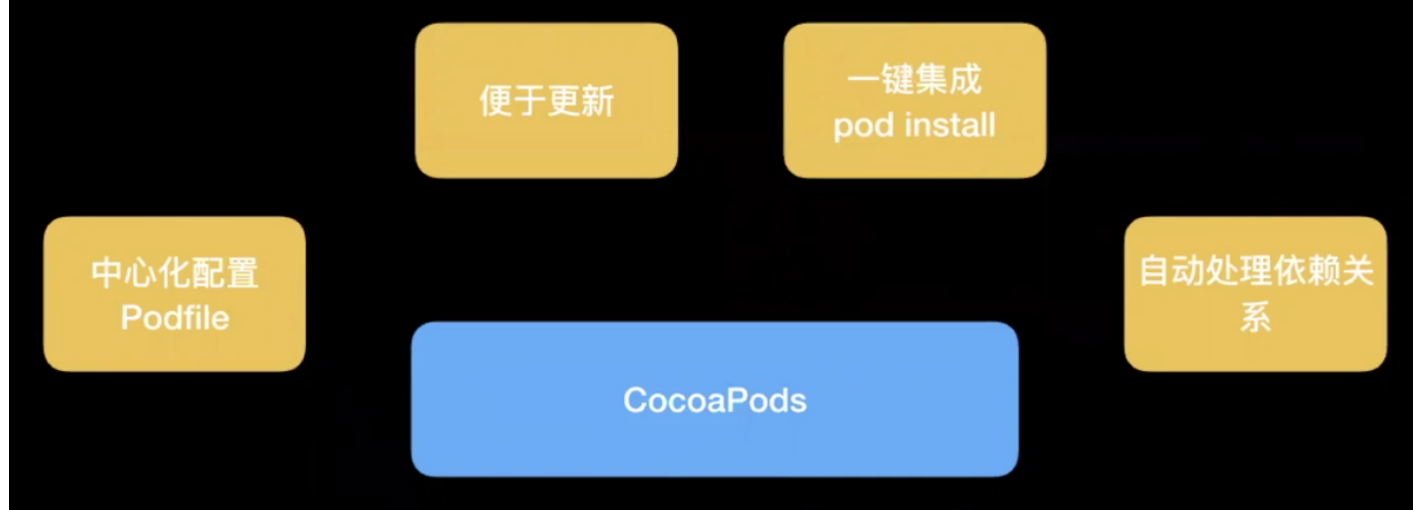
- 什么是CocoaPods？

iOS 和 OS X下的一个第三方库管理工具。

作用：依赖库版本管理、库依赖自动配置；

- 为什么要使用CocoaPods？

## 为什么要使用CocoaPods



安装依赖库 pod install

Pod install 后，文件里多了 `lock,pods/,xcworkspace`

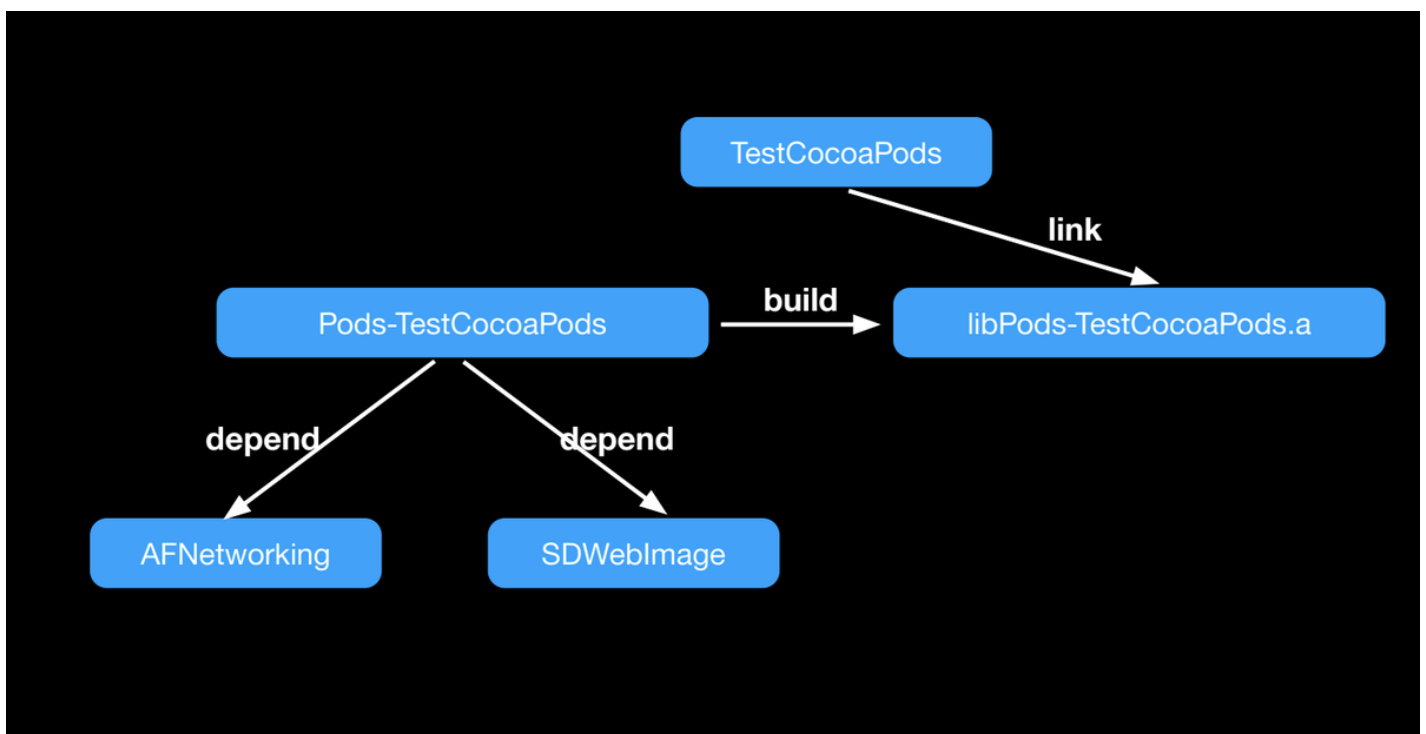
`xcworkspace`组织了项目和pods的`xcodeproj`，后续就在这个workspace开发

描述本工程与依赖库的关系 podfile 中心化配置

保证各个开发人员使用同版本依赖库 podfile.lock

- 1 sequenceDiagram
- 2 A->>Podfile: pod install
- 3 Podfile->>Podfile.lock: 描述了准备的依赖库版本号
- 4 Podfile.lock->>Repo: 版本控制
- 5 B->>Repo: 拉去代码，包括Podfile、Podfile.lock;
- 6
- 7 这个时候B拉去代码，pod install，会根据podfile.lock里的版本去下载依赖库

编译过程：



Pod install

确定项目架构；

找到podfile的change

解析podfile，下载依赖文件

生成proj和lock文件

与原有项目集成

Pod update

如果你运行pod update，后面没有跟库的名字，CocoaPods就会更新每一个Podfile里面的库到尽可能的最新版本。

版本控制：  
pods/文件夹需要加入gitignore，不需要进行版本控制；  
而podfile.lock需要加入项目版本控制，因为lock文件维护的依赖版本信息可以让所有开发人员同步依赖库版本。

以下内容摘抄自使用指南 [CocoaPods 使用指南](#)

## .podspec

### Spec 规格

一直到 1.7 版本之前，CocoaPods 一直是使用的本地 Source 仓来查询对应组件版本。我们每次执行 pod repo update 或者 pod update 其实都会对 Source 仓进行 git pull 的操作。Source 仓就是我们之前所说的中心化管理的特征。所以在我们刚开始使用 CocoaPods 的时候，都会执行 pod setup 将最大的 master 仓库 git clone 下来，这个仓库也就是 CocoaPods 的公网仓，其中收纳了公网所有组件的 podspec。

## 关键目录及文件

Podfile	描述依赖的文件
Podfile.lock	依赖版本的锁存文件
Pods/	Sandbox 目录
Pods/Manifest.lock	Podfile.lock 的备份，项目 build 的时候进行比对
Pods/Headers/	HEADER_SEARCH_PATHS 的搜索目录
Pods/Target Support Files/	存放所有 Pod Target 的 xcconfig
\$HOME/.cocoapods/repos/	设备中所有项目用到的所有 Spec Source 仓
\$HOME/Library/Caches/CocoaPods/	CocoaPods 的 Cache 目录

- 关于常见错误，如spec不是最新，需要pod update，或者是xcode在编译阶段检测出没有manifest.lock文件时提醒要pod install，可参考指南。


## Podfile怎么用？

# 如何编写Podfile文件

```
source 'https://github.com/CocoaPods/Specs.git' #podspec存放地址
platform :ios, '9.0' #版本支持>=9.0的Pod才可以被引入
inhibit_all_warnings! #忽略依赖库中的警告
workspace 'TestCocoaPods'
target 'TestCocoaPods' do
  # pod 'AFNetworking' #默认使用最新的tag
  # pod 'AFNetworking', '3.2.1'
  # pod 'AFNetworking', '~> 3.2.1' #[3.2.1, 3.3) > >= < <=
  # pod 'AFNetworking', :git => 'https://github.com/AFNetworking/AFNetworking.git', :branch => 'master'
  # pod 'AFNetworking', :git => 'https://github.com/AFNetworking/AFNetworking.git', :tag => '3.2.1'
  # pod 'AFNetworking', :git => 'https://github.com/AFNetworking/AFNetworking.git', :commit =>
  '5890adc404d0187dcc452f467c8fc3cedcl9869d'
  # pod 'AFNetworking', :path => '../AFNetworking' #使用本地目录下
  pod 'AFNetworking', :subspecs => ['NSURLSession']
  pod 'SDWebImage', '~> 4.0'
end
```

无论是使用Git还是local，都是外部引入。内部和外部引入的区别，就是其podspec的获取方式是通过source仓库还是非source仓库得到的。

## Source 声明



```
source 'git@code.byted.org:ugc/AWESpecs.git'  
source 'git@code.byted.org:iOS_Library/IES-UGC_binary_repo.git'  
source 'git@code.byted.org:ugc/UGCSpecs.git'  
source 'git@code.byted.org:iOS_Library/douyin_binary_repo.git'  
source 'git@code.byted.org:iOS_Library/douyin_source_repo.git'  
source 'git@code.byted.org:iOS_Library/publicthird_binary_repo.git'  
source 'git@code.byted.org:iOS_Library/publicthird_source_repo.git'  
source 'git@code.byted.org:iOS_Library/privatethird_binary_repo.git'  
source 'git@code.byted.org:iOS_Library/privatethird_source_repo.git'  
source 'git@code.byted.org:iOS_Library/toutiao_binary_repo.git'  
source 'git@code.byted.org:iOS_Library/toutiao_source_repo.git'  
source 'git@code.byted.org:iOS_Library/livebroadcast_binary_repo.git'  
source 'git@code.byted.org:iOS_Library/livebroadcast_source_repo.git'
```

用来制定 Podfile 中所有的 Pod 要从哪些 Source 去搜索。如果一个 Pod 在多个 Source 中存在，会按照顺序取第一个选用。