一个没有package管理器的遗留项目是如何运作的

### 

[**背景介绍 1**](#_22ubyi4wmz49)

[**NuGet工作方式 1**](#_avajx75k5s16)

[**我们面临的问题？ 2**](#_lyryjja5gd0y)

[**我们是如何实现？ 2**](#_qet9i2uk81h7)

[**Library Repo在现实中表现如何？ 4**](#_nugmz3oh05bk)

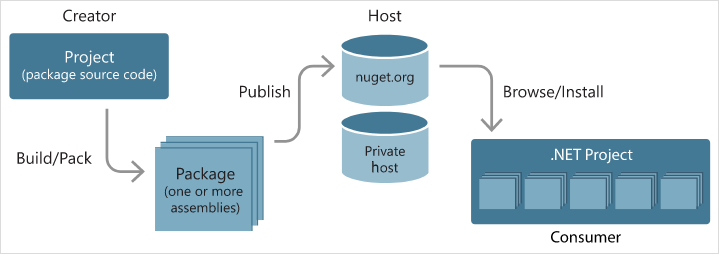
### 

### 背景介绍

我们团队正在维护一个有将近15年历史遗留的产品，按照产品业务将code拆分在10多个Repo，Reop之间存在相互的引用依赖，每一个repo有自己独立的pipeline可以完成编译，运行单元测试，通过一个外部总pipeline将所有单独运行的pipeline串联起来，最终生成一个Windows桌面应用setup安装包。由于没有使用.Net常用的Nuget包管理器，那么它是如果解决不同Repo之间的Library依赖呢?

### NuGet工作方式

我们在.Net开发过程中，NuGet是我们常用的包管理器，简单方便并且支持私有的包管理服务器。



首先创建者生成有用的 NuGet 包并将其发布到主机。 然后，使用者可以在可访问的包管理服务器上搜索有用且兼容的包，下载包并将其包含在项目中。 在项目中安装包后，包的 API 将可用于其余项目代码。

### 我们面临的问题？

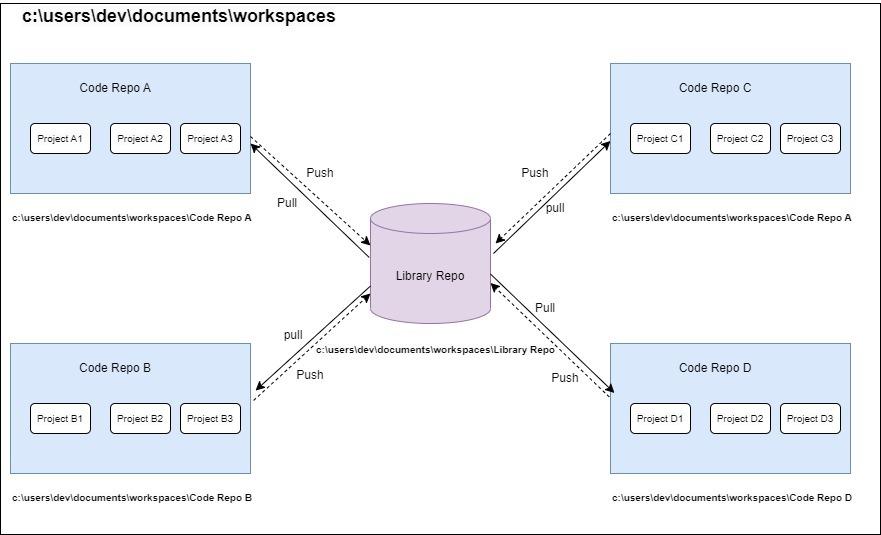
如果没有Nuget我们需要解决，

1. 如何从Library源获取依赖对应的Library
2. Library源管理多个版本的Library
3. Package文件目录
4. Library依赖项管理
5. Library定向兼容性

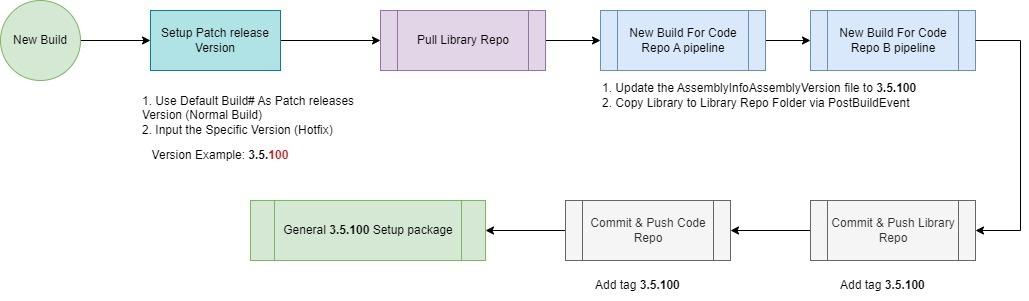
### 我们是如何实现？

我们需要指定一个Library源，10多个Repo都可以从这里获取自己所需要的Library和发布自己的Library，还需要能维护不同版本的Library记录，因此我们可以创建一个Library Repo充当Library源，提交记录对应不同版本。

我们需要指定一个Package文件目录，保存下载的Library。这个Package文件目录和NuGet的Package会有差异，并不是每一个repo有一个Package文件目录。由于所有的Repo需要保持版本的一致性，Library是不存在版本差异，所以只需要一个Package文件目录，10多个Repo共同使用。指定一个工作目录，Clone 对应Code Repo并保持一个平级的目录结构，同时Clone Library Repo保持一致目录结构，然后Code Repo可以从Library Repo中通过相对路径方式获取所需的Library。这种方式也带来了一个便捷，本地开发过程中，如果Code Repo中一个Library Project需要被其他的Code Repo Project所引用，那么需要将编译后通过PostBuildEvent 方式将Library拷贝到Library Repo中，其他Repo就可以随时获取到新的修改Library。参考下图



我们要解决Library 版本及提交Library到Library Repo. 我们需要通过pipeline来完成。每一个Code Repo下包含一个AssemblyInfoAssemblyVersion文件记录当前版本，这个文件将会有pipeline在运行过程中进行修改。当Pipeline完成所有的编译工作后，Code Repo的AssemblyInfoAssemblyVersion文件的修改会被提交，并且Library Repo中对用的新版本Library也会提交，每次的提交都会对应的version tag标记。这个版本也将会成为Setup的版本，当用户安装后应用版本也与其一致。如果出现的线上的问题，我们也能快速根据版本快速定位问题。



上述过程体现了Library Repo在现实Pipeline中如何运作。由于Repo之间依赖关系，我们需要按照依赖关系来编排pipeline的steps，对没有依赖关系repo将采用并行的方式执行来提升效率。

### Library Repo在现实中表现如何？

* 优点

1. 在没有Nuget包管理器的条件下，满足现有项目Library的管理。
2. 同时工作在多个有依赖关系Code Repo，Library引用可以无缝衔接，无需等待依赖Repo Libray提交发布。
3. Library版本一致性，避免解决Library定向兼容性。

* 缺点

1. 如果Code Repo切换到之前版本节点，Library Repo也需要进行切换到与之对应版本的节点。
2. 如果多个Code Repo在不同版本节点，Library Repo也需要根据具体Code Repo版本频繁切换。
3. 容易出现错误版本Library引用问题，编译期间也及时不能发现。
4. Code Repo不能单独编译，需要借助Library Repo才能获取到依赖的Library。
5. 对SonarQube支持不够友好，SonarQube对C#分析，前提需要能够编译成功。单个Repo编译需要借助Library Repo提供外部依赖才能编译成功，因此SonerQube对Repo分析前，也需要获取Library Repo以确保当前Repo编译成功。