bobby · 更新于 2022-11-03

上一节 4. go项目目录... 6. 微服务的目录... 下一节

Multi-repo 和 Mono-repo 是 Git 托管代码的两种策略，我们讨论下两者的策略以及其利弊。

引言

大多数现代项目都是在 Git 上管理和托管的。Git 已经成为来自世界各地的分布式源代码管理、版本控制和协作的标准平台。Git 是快速和高效的，主要有两种方法来托管和管理 Git 代码:

- Mono-repo
- Multi-repo

在深入研究这些方法之前，让我们先了解一下 Repo 是如何工作的。

Repos 是什么?

仓库(Repo)包含项目的所有文件夹和文件。它还包含关于用户、人和计算机的信息。
Git 仓库数据受版本控制，Repo 可以由个人或团队成员拥有。
Git 仓库可以是公开的，私人的，或者是内部的。GitHub 是 Git 仓库的一个托管服务，并且有一个用户界面。
Git 提供了版本控制和代码共享功能，Git 的特别之处在于，如果开发人员想对他们的文件做一些修改，他们可以将整个存储库复制到他们的本地系统中。因此，即使开发人员没有对特定项目的写入权限，他们也可以在本地复制内容并修改它们(我们称为 forking)。
此外，如果开发人员希望共享本地所做的更改，他们可以向项目所有者发送一个 “pull request”。
一个项目可以只有一个服务。如果你的项目有多个工作流，你可以为每个工作流创建多个服务。大多数开发人员喜欢将较大的项目拆分为具有一个或多个功能的较小的独立服务。每个服务都可以解决各种业务问题。随着 serverless 框架的流行，用户可以将功能作为服务访问。
一旦你创建了这些函数——作为服务并部署它们，下一步就是对它们构造和版本控制——你可以将所有的服务放在一个存储库(mono-repo)中，或者为你拥有的每个服务拥有一个单独的存储库(multi-repo) !

什么是 Mono-repo?

在 mono-repo 方法中，你可以将所有服务保存在单一(mono)存储库中。你仍然可以独立地部署和管理每个服务。这些服务可以共享公共库和代码。
像 Facebook、 Google 和 Dropbox 这样的公司都使用 Mono-repo。

如果你把一个代码打包成公共的代码供大家go get
那么你会面临一个问题:
版本维护
假设A项目用了你的1.x 版本
B项目用了你的2.x的版本
c项目用了你的3.x的版本

所以随着项目的反正，你这个库会越来越大，越来越多的无用代码
最好的基本办法是什么：强制升级，有个问题 你很难知道谁用了你的代码， 别人不愿意升级

Mono-repo 的优势

Mon-repo 方式有许多优点:

- 存储所有项目代码的单独位置，团队中的每个人都可以访问。
- 易于重用和共享代码，与团队合作。
    - 就意味着基本上没有版本管理，只有最新版，我升级了版本，引用的地方就会报错
- 很容易理解你的变更对整个项目的影响。
- 代码重构和代码大变更的最佳选择。
- 团队成员可以获得整个项目的总体视图。
- 易于管理依赖关系。

Mono-repo 的劣势

当然，Mono-repo 也有一些缺点，主要表现在性能上。如果你的项目增长，每隔一天都会添加更多的文件，那么 git checkout、pull 和其他操作可能变得缓慢，以及文件搜索可能需要更长的时间。
此外，如果你为你的项目雇佣了许多独立的承包商，那么让他们访问整个代码库可能不那么安全。
此外，实现持续部署(Continuous deployation，CD)也很困难，因为许多人可以合入他们的更改，而持续集成(Continuous Integration，CI)系统可能需要进行多次重构。
使用 Mono-repo 的大公司都有自定义工具来处理扩展问题。例如，Facebook 使用自定义文件系统和源代码控制。

什么是 Multi-repo?

在 Multi-repo 方法中，存在多个存储库，它们承载一个项目的多个库和服务。如果服务发生更改，开发人员只需重新构建该服务，而不需要构建整个项目。个人和团队可以从事他们特定的服务，他们只能访问他们有权限的服务。
像 Netflix 和 Amazon 这样的公司使用 Multi-repo。

Multi-repo 的优势?

采用 Multi-repo 的公司数量远远多于采用 Mono-repo 的公司，原因如下:

1. 每个服务和库都有自己的版本控制。
2. 代码 checkout 和 pull 是小型且独立的，因此即使项目规模增大，也不存在性能问题。
3. 团队可以独立工作，不需要访问整个代码库。
4. 更快的开发和灵活性。
5. 每个服务都可以单独发版，并有自己的部署周期，从而使 CI 和 CD 更易于实现。
6. 更好的权限访问控制——所有的团队不需要完全访问所有的库——需要的时候，再获得读访问权限。

Multi-repo 的劣势

- 跨服务和项目使用的公共依赖和库必须定期同步以获得最新版本。
- 某种程度上鼓励孤立文化，导致重复代码和各个团队试图解决相同问题。
- 每个团队可能遵循不同的一组最佳实践来编写代码，从而导致难以遵循通用的最佳实践。

Mono Repo 和 Multi Repo 的区别

让我们来概括 Mono Repo 和 Multi Repo 的区别:

Mono-repo	Multi-repo
一个组织的所有项目的所有代码都驻留在中央存储库中	每个服务和项目都有一个单独的存储库
团队可以一起协作和工作; 他们可以看到彼此的变化	团队可以自主工作; 个人的变更不会影响其他团队或项目的变更

Mono-repo	Multi-repo
每个人都可以访问整个项目结构	管理员可以将访问控制限制到开发人员需要访问的项目或服务
如果项目规模不断增长，则可能会出现并放大问题	良好的性能，因为有限的代码和较小的服务单元
难以实现持续部署(CD)和持续集成(CI)	开发人员可以很容易地实现 CD 和 CI，因为他们可以独立地构建服务
开发人员可以轻松地共享库、 api 和其他在中央存储库中更新的公共代码	对库和其他常见代码的任何更改都应该定期同步，以避免以后出现问题

注： 技术栈统一的话建议采用mono-repo， 不同服务技术栈不统一就采用multirepo， 目前采用mono-repo的知名公司： google、bilibili

总结

Mono-repo 和 Multi-repo 同样流行，哪一个更好取决于你的项目大小、项目需求以及你需要的版本控制和访问控制级别。
Mono-repo 侧重一致性，而 Multi-repo 侧重于解耦。在 Mono-repo 中，整个团队可以看到某一个人完成的更改，而 multi-repo 为每个团队创建一个单独的 repo，这些团队只能访问所需的仓库。如果你想为你的项目使用 mono-repo 和 multi-repo 的组合，你可以使用 meta，一个管理多个项目和库的工具。

4. go项目目录规范 上一节 下一节 6. 微服务的目录结构（mono-repo）

我要提出意见反馈