目录

[1. Git和SVN有什么区别？ 1](#_Toc49969623)

[2. Git和GitHub有什么区别 1](#_Toc49969624)

[3. 什么是Git 2](#_Toc49969625)

[4. Git跟其他版本控制器有啥区别？ 2](#_Toc49969626)

[5. Git是用什么语言编写的？ 3](#_Toc49969627)

[6. Git中的一些基本命令并解释其用法 3](#_Toc49969628)

[7. Git中的存储库是什么？ 3](#_Toc49969629)

[8. 如何在Git中创建存储库 3](#_Toc49969630)

[9. 在Git中提交的命令是什么？ 3](#_Toc49969631)

[10. 提交对象包含什么？ 4](#_Toc49969632)

[11. 如果本次提交误操作,如何撤销? 4](#_Toc49969633)

[12. Git config 的功能是什么？ 4](#_Toc49969634)

[13. 如何把本地仓库的内容推向一个空的远程仓库？ 4](#_Toc49969635)

[14. 如何查看文件的提交历史和分支的提交历史 5](#_Toc49969636)

[15. 描述以下你所使用的分支策略 5](#_Toc49969637)

[16. 使用过Git merge 和 Git 和 rebase吗？它们之间有什么区别？ 5](#_Toc49969638)

[17. 如果分支是否已合并为master，你可以通过什么手段知道？ 6](#_Toc49969639)

[18. "冲突"是如何产生的？你是如何解决Git中的冲突的 6](#_Toc49969640)

[19. Git fetch 和 Git pull命令之间的区别 6](#_Toc49969641)

[20. 你使用过git stash命令吗？你一般什么情况下会使用它？ 7](#_Toc49969642)

[21. 我们在本地工程常会修改一些配置文件，这些文件不需要被提交，而我们又不想每次执行git status时都让这些文件显示出来，我们该如何操作？ 7](#_Toc49969643)

[22. 能说一下git系统中HEAD、工作树和索引之间的区别吗？ 7](#_Toc49969644)

[23. 在Git中，你如何还原已经push并公开的提交？ 8](#_Toc49969645)

[24. 什么是 Git 中的“裸存储库”？ 8](#_Toc49969646)

[25. 什么是工作区、暂存区、本地仓库、远程仓库 8](#_Toc49969647)

[26. 请写出查看分支、创建分支、删除分支、切换分支、合并分支的命令以及写出解决冲突的思路？ 9](#_Toc49969648)

[27. 之前项目中是使用的GitFlow工作流程吗？它有什么好处？ 9](#_Toc49969649)

[28. 请写出团队内部协作开发的流程 10](#_Toc49969650)

[29. 请写出远程跨团队协作开发的流程？ 11](#_Toc49969651)

[30. 请写出配置ssh的思路？ 11](#_Toc49969652)

[31. 请写出你们公司团队内部协作开发的流程？ 12](#_Toc49969653)

[32. 请描述什么是GitLab,或者说出你对GitLab的理解？ 12](#_Toc49969654)

[33. 请写出你所参与的多人协同开发时候，项目都有哪些分支，分支名是什么，每个分支代表什么，以及分支是由谁合并？ 12](#_Toc49969655)

本地仓库：建立在本地的文件夹。 远程仓库：建立在互联网的服务器内的文件夹。

**分布式版本控制系统**

配有上述两个仓库，电脑上有一个本地仓库 在远程的服务器有一个远程仓库

我们在提交文件的时候会先提交到本地仓库，然后在有网络的情况下，再从本地仓库提交到网络上的远程仓库。 Git 就是一个典型的分布式版本控制系统

**集中式版本控制系统**

只配有上述的远程仓库，如果你的这台电脑就充当远程服务器的角色，那远程仓库其实就在你这台电脑上。 我们在提交文件的时候是直接提交到远程仓库。 SVN 就是一个典型的集中式版本控制系统

### 1. Git和SVN有什么区别？

| **GIt** | **SVN** |
| --- | --- |
| 1.Git是一个分布式的版本控制工具 | 1.SVN 是集中版本控制工具 |
| 2.它属于第3代版本控制工具 | 2.它属于第2代版本控制工具 |
| 3.客户端可以在其本地系统上克隆整个存储库 | 3.版本历史记录存储在服务器端存储库中 |
| 4.即使离线也可以提交 | 4.只允许在线提交 |
| 5.Push/pull 操作更快 | 5.Push/pull 操作较慢 |
| 6.工程可以用 commit 自动共享 | 6.没有任何东西自动共享 |

### 2. Git和GitHub有什么区别

* **Git**
  + [Git](https://bit.ly/31MeW9b)是一个分布式的版本控制系统，用于在软件开发过程中跟踪源代码的更改。它有助于协调程序员之间的工作，但可用于跟踪任何文件集中的更改。Git的主要目标是速度，数据完整性以及对分布式非线性工作流的支持。
* **GitHub**
  + GitHub是Git存储库托管服务，此外它还添加了许多自己的功能。GitHub提供了基于Web的图形界面。它还为每个项目提供访问控制和一些协作功能，基本任务管理工具。

### 3. 什么是Git

* Git 是分布式版本控制系统（DVCS）。它可以跟踪文件的更改，并允许你恢复到任何特定版本的更改。
* 它是一种分布式体系结构，与诸如SVN的其他版本控制系统（VCS）相比，具有许多优势。主要优点之一是它不依赖中央服务器来存储项目文件的所有版本。
* 每个开发人员都可以“克隆”储存库的副本，并且在他的硬盘驱动器上具有项目的完整历史记录，因此当服务器中断时候，你需要的所有恢复数据都在你队友的本地Git存储库中。
* 还有一个中央云存储库，开发人员可以向其提交更改，并与其他团队成员进行共享，如图所示，所有协作者都在提交更改“远程存储库”。



### 4. Git跟其他版本控制器有啥区别？

GIT是分布式版本控制系统，其他类似于SVN是集中式版本控制系统。

分布式区别于集中式在于：每个节点的地位都是平等，拥有自己的版本库，在没有网络的情况下，对工作空间内代码的修改可以提交到本地仓库，此时的本地仓库相当于集中式的远程仓库，可以基于本地仓库进行提交、撤销等常规操作，从而方便日常开发。

### 5. Git是用什么语言编写的？

Git使用 C 语言。Git速度很快，而 C 语言通过减少运行时的开销来做到这一点。

### 6. Git中的一些基本命令并解释其用法

| **命令** | **用法** |
| --- | --- |
| $ git clone <url> | 克隆远程仓库 |
| $ git init | 初始化本地版本库 |
| $ git add . | 跟踪所有改动过的文件 |
| $ git status | 查看状态 |
| $ git commit -m "commit message" | 提交所有更新过的文件 |
| $ git branch | 查看分支 |
| $ git branch <branch> | 创建分支 |
| $ git checkout <branch> | 切换分支 |
| $ git merge <branch> | 合并分支 |

### 7. Git中的存储库是什么？

Git中的存储库是Git存储所有文件的地方。Git可以将文件存储在本地存储库或远程存储库上。

### 8. 如何在Git中创建存储库

要创建存储库，先为项目创建一个目录（如果该目录不存在），然后运行命令 **git init**。通过运行此命令，将在项目的目录中创建 .git 目录。

### 9. 在Git中提交的命令是什么？

* 写入提交命令是git commit -a
* 如果是第一次需要提交新文件，可以在git commit -a 之前先git add <file>

### 10. 提交对象包含什么？

commit对象包含以下组件:

1. 一组文件，表示给定时间点的项目状态
2. 引用父提交对象
3. SHAI名称，一个40个字符的字符串，提交对象的唯一标识

### 11. 如果本次提交误操作,如何撤销?

* 如果想撤销提交到索引区的文件，可以通过git reset HEAD file
* 如果想撤销提交到本地仓库的文件，可以通过git reset –soft HEAD^n恢复当前分支的版本库至上一次提交的状态，索引区和工作空间不变更
* 可以通过git reset –mixed HEAD^n恢复当前分支的版本库和索引区至上一次提交的状态，工作区不变更
* 可以通过git reset –hard HEAD^n恢复当前分支的版本库、索引区和工作空间至上一次提交的状态。

### 12. Git config 的功能是什么？

Git使用你的用户名将提交与身份相关联。Git config命令可用来更改你的Git配置，包括你的用户名。

假设你要提供用户名和电子邮件 ID 用来将提交与身份相关联，以便你可以知道是谁进行了特定提交。

**git config –global user.name "Your Name":** 此命令将添加用户名。

**git config –global user.email "Your E-mail Address"**: 此命令将添加电子邮件ID。

### 13. 如何把本地仓库的内容推向一个空的远程仓库？

首先确保本地仓库与远程之间是连同的。如果提交失败，则需要进行下面的命令进行连通：

git remote add origin 远程仓库地址

如果是第一次推送，则进行下面命令：

git push -u origin master

注意：-u 是指定origin为默认主分支 之后的提交，只需要下面的命令：

git push origin master

### 14. 如何查看文件的提交历史和分支的提交历史

* 使用git log 查看文件提交历史
  + git log –number：表示查看当前分支前number个详细的提交历史记录；
  + git log –number –pretty=oneline：在上个命令的基础上进行简化，只显示sha-1码和提交信息；
* 使用git log查看分支提交历史
  + git reflog –number: 表示查看所有分支前number个简化的提交历史记录；
  + git reflog –number –pretty=oneline：显示简化的信息历史信息；

**注意：**如果没有number则显示全部的提交次数

### 15. 描述以下你所使用的分支策略

* 功能分支
  + 要素分支模型将特定要素的所有更改保留在分支内。当通过自动化测试对功能进行全面测试和验证时，该分支将合并到主服务器中。
* 任务分支
  + 在此模型中，每个任务都在其自己的分支上实现，任务键包含在分支名称中。很容易看出哪个代码实现了哪个任务，只需在分支名称中查找任务键。
* 发布分支
  + 一旦开发分支获得了足够的发布功能，你就可以克隆该分支来形成发布分支。创建该分支将会启动下一个发布周期，所以在此之后不能再添加任何新功能，只有错误修复，文档生成和其他面向发布的任务应该包含在此分支中。一旦准备好发布，该版本将合并到主服务器并标记版本号。此外，它还应该再将自发布以来已经取得的进展合并回开发分支。

### 16. 使用过Git merge 和 Git 和 rebase吗？它们之间有什么区别？

简单来说，git merge 和 git rebase 都是合并分支的命令。

git merge branch会把branch分支的差异内容pull到本地，然后与本地分支的内容一并形成一个committer对象提交到主分支上，合并后的分支与主分支一致；

git rebase branch会把branch分支优先合并到主分支，然后把本地分支的commit放到主分支后面，合并后的分支就好像从合并后主分支又拉了一个分支一样，本地分支本身不会保留提交历史。

### 17. 如果分支是否已合并为master，你可以通过什么手段知道？

要知道某个分支是否已合并为master，你可以使用以下命令：

**git branch –merged** 它列出了已合并到当前分支的分支。

**git branch –no-merged** 它列出了尚未合并的分支。

### 18. "冲突"是如何产生的？你是如何解决Git中的冲突的

Git可以使用其自动合并功能自行处理大多数合并。当两个单独的分支对文件中的同一行进行编辑时，或者在一个分支中删除了一个文件却在另一个分支中对其进行了编辑时，就会发生冲突。在团队环境中工作时，冲突最有可能发生。

**如何解决冲突**

1. 发生冲突，在IDE里面一般都是对比本地文件和远程分支的文件
2. 把远程分支上文件的内容git pull 到本地文件
3. 再提交冲突的文件使其保证和远程分支的文件一致
4. 这样就消除冲突，然后提交自己修改的部分

特别要注意下，修改本地冲突文件使其与远程仓库的文件保持一致后，需要提交后才能消除冲突，否则无法继续提交。必要时可与同事交流，消除冲突。

发生冲突，也可以使用命令: 通过git stash命令，把工作区的修改提交到栈区，目的是保存工作区的修改； 通过git pull命令，拉取远程分支上的代码并合并到本地分支，目的是消除冲突； 通过git stash pop命令，把保存在栈区的修改部分合并到最新的工作空间中；

### 19. Git fetch 和 Git pull命令之间的区别

简单来说：git fetch branch是把名为branch的远程分支拉取到本地；

而git pull branch是在fetch的基础上，把branch分支与当前分支进行merge；因此pull = fetch + merge

git pull 命令从中央存储库中提取特定分支的新更改或提交，并更新本地存储库中的目标分支。

### 20. 你使用过git stash命令吗？你一般什么情况下会使用它？

命令git stash是把工作区修改的内容存储在栈区。

以下几种情况会使用到它：

解决冲突文件时，会先执行git stash，然后解决冲突； ​ 遇到紧急开发任务但目前任务不能提交时，会先执行git stash，然后进行紧急任务的开发，然后通过git stash pop取出栈区的内容继续开发； 切换分支时，当前工作空间内容不能提交时，会先执行git stash再进行分支切换；

### 21. 我们在本地工程常会修改一些配置文件，这些文件不需要被提交，而我们又不想每次执行git status时都让这些文件显示出来，我们该如何操作？

1. 首先利用命令touch .gitignore新建文件：

touch .gitignore

1. 然后往文件中添加需要忽略哪些文件夹下的什么类型的文件：

/target/class  
.settings  
.imp  
\*.ini

注意：忽略/target/class文件夹下所有后缀名为.settings，.imp的文件，忽略所有后缀名为.ini的文件。

### 22. 能说一下git系统中HEAD、工作树和索引之间的区别吗？

**HEAD文件**包含当前分支的引用（指针）；

**工作树**是把当前分支检出到工作空间后形成的目录树，一般的开发工作都会基于工作树进行；

**索引index文件**是对工作树进行代码修改后，通过add命令更新索引文件；GIT系统通过索引index文件生成tree对象

### 23. 在Git中，你如何还原已经push并公开的提交？

这个问题可以有两个答案，你回答时也要保包含这两个答案，因为根据具体情况可以使用以下选项：

* 删除或修复新提交中的错误文件，并将其推送到远程存储库。这是修复错误的最自然方式。对文件进行必要的修改后，将其提交到我将使用的远程存储库

git commit -m "commit message"

* 创建一个新的提交，撤消在错误提交中所做的所有更改。可以使用命令：

git revert <name of bad commit>

### 24. 什么是 Git 中的“裸存储库”？

你应该说明 “工作目录” 和 “裸存储库” 之间的区别。

Git 中的 “裸” 存储库只包含版本控制信息而没有工作文件（没有工作树），并且它不包含特殊的 .git 子目录。相反，它直接在主目录本身包含 .git 子目录中的所有内容，其中工作目录包括：

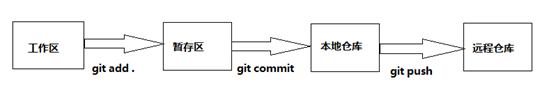
一个 .git 子目录，其中包含你的仓库所有相关的 Git 修订历史记录。 ​ 工作树，或签出的项目文件的副本。

### 25. 什么是工作区、暂存区、本地仓库、远程仓库

* 定义

1. 工作区:即个人克隆项目到本地后,项目所在的文件夹目录.
2. 暂存区:用于储存工作区中的变更(增删改等改动)的文件的地方.操作时使用git add会将本地所有的变更提交到暂存区中
3. 本地仓库:用于储存工作区和暂存区中提交上来的文件,使用git commit -m '提交内容的描述'
4. 远程仓库:当进行到这里的时候即一个人的开发完毕的时,需要将自己开发的功能合并到著主项目中去,但因为是多人开发,要保管好主项目中存储的代码和文件的话,就需要放在搭建好的远程git仓库中,即远程仓库.具体操作:git push origin 远程分支名即可

* 工作区、暂存区、本地仓库、远程仓库关系图:



### 26. 请写出查看分支、创建分支、删除分支、切换分支、合并分支的命令以及写出解决冲突的思路？

$ git branch //查看分支

$ git branch <branch> //创建分支

$ git branch -d <branch> //删除分支

$ git checkout <branch> //切换分支

$ git merge <branch> //合并分支

* 当多个分支代码合并到一个分支时 或 多个分支向同一个远端分支推送代码时会产生冲突
* 关于push和pull其实就分别是用本地分支合并到远程分支 和 将远程分支合并到本地分支 也可能产生冲突

**解决冲突思路:**

* 解决：保留自己修改的数据，同时保留别人的数据信息
  1. 发生冲突，在IDE里面一般都是对比本地文件和远程分支的文件
  2. 把远程分支上文件的内容通过 git pull 到本地文件
  3. 再提交冲突的文件使其保证和远程分支的文件一致
  4. 这样就消除冲突，然后提交自己修改的部分

特别要注意下，修改本地冲突文件使其与远程仓库的文件保持一致后，需要提交后才能消除冲突，否则无法继续提交。必要时可与同事交流，消除冲突。

### 27. 之前项目中是使用的GitFlow工作流程吗？它有什么好处？

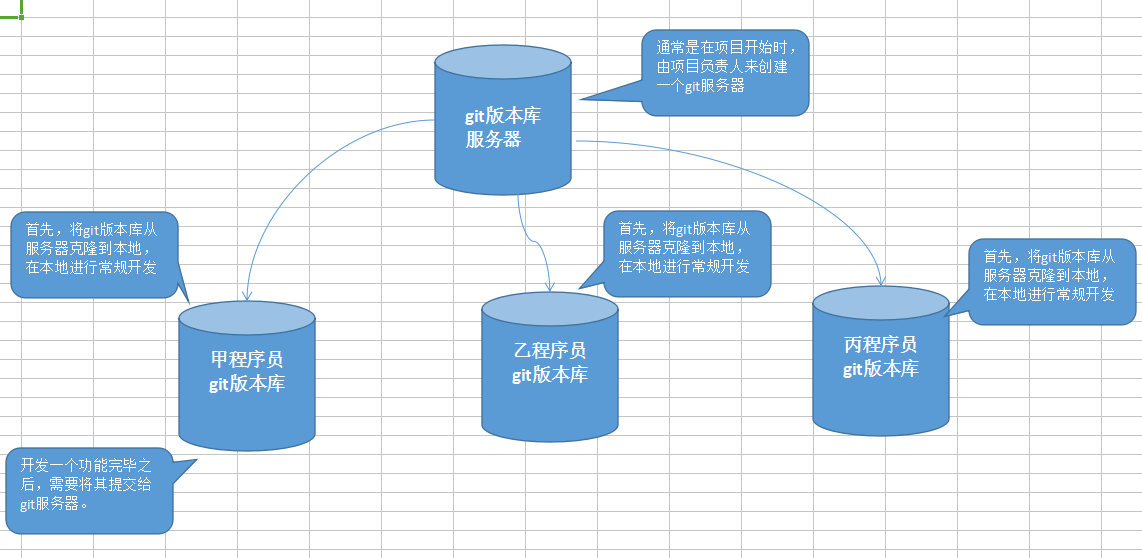
为了记录项目的历史，Gitflow工作流程采用了两个并行且长期运行的分支-掌握和开发：

* Master –该分支始终准备好在LIVE上发布，所有内容都经过全面测试和批准（可投入生产）。
* 修补程序–这些分支用于快速修补生产版本。这些分支与发行分支和功能分支非常相似，只不过它们是基于master而不是development的。
* 开发–这是将所有功能分支合并到其上并执行所有测试的分支。只有对所有内容进行彻底检查和修复后，才能将其合并到主数据库。
* 功能–每个新功能应驻留在其自己的分支中，可以将其作为其父项推入到develop分支。

**GitFlow的好处** 为不同的分支分配一个明确的角色，并定义分支之间如何交互以及什么时间交互；可以帮助大型项目理清分支之间的关系，简化分支的复杂度。

### 28. 请写出团队内部协作开发的流程

如图所示：



* 分为如下几个步骤：

1.创建一个git裸服务器 ,

* + 项目管理员添加用户到群组，赋予权限

2.从裸服务器将版本库克隆至本地（git clone ）

* + 通过终端打开项目目录，创建自己的分支，并推送到远程项目仓库。

3.开发

* + 发布任务、编写代码、提交代码并备注、回复任务、结束任务（每一次开发都要走这个流程，做到每一次代码改动都有迹可循：为什么做、怎么做的、提交结果）

4.推送版本至服务器 （git remote + git push）

5.从远程服务器拉取版本（git pull）

6.项目经理拉取最新内容 合并到master，日后便于上线(git merge)

### 29. 请写出远程跨团队协作开发的流程？

1. 项目组长联系组员，讲解项目，布置任务
2. 组员通过Fork将git远程仓库克隆到自己的仓库中
3. 将自己的git仓库中的项目拉取到本地，对项目进行开发
4. 开发完成后推送版本至服务器 （git remote + git push）
5. 组长浏览组员发送过来的请求，确定无误之后，接受合并请求

### 30. 请写出配置ssh的思路？

1. 首先我们要生成一个新的SSH秘钥，打开终端进入 根目录 （cd ~）然后输入以下命令

ssh-keygen -t rsa -b 4096 -C “email” //后面换成你的GitHub邮箱

1. 如果之前生成过秘钥，会提示是否覆盖：输入y并且回车，如果没有生成过，则跳过这一步
2. 这里提示我们输入秘钥密码，尽量别设置太复杂，以后push/pull时候都要输入这个密码，

不输入密码也可以，点击回车

1. 进入保存秘钥的文件位置

cd .ssh

1. 查看公钥（id\_rsa.pub）,并复制里面全部内容

cat id\_rsa.pub

1. 进入git找到SSH and GpG keys这一项，title信息随便填写，key这一项粘贴刚才复制的内容,就完成了SSH的添加

### 31. 请写出你们公司团队内部协作开发的流程？

### 32. 请描述什么是GitLab,或者说出你对GitLab的理解？

* 什么是GitLab
  + GitLab 是利用 Ruby on Rails 一个开源的版本管理系统，实现一个自托管的 Git 项目仓库，可通过 Web 界面进行访问公开的或者私人项目，它拥有与 Github 类似的功能，能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序 (Wall) 进行交流。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

### 33. 请写出你所参与的多人协同开发时候，项目都有哪些分支，分支名是什么，每个分支代表什么，以及分支是由谁合并？

* 项目有master主分支 和 development 分支
* 成员在development分支操作，最后提交到远程仓库的development分支
* 项目管理员查看development分支，代码符合要求时，合并到master分支