**1.版本控制器的方式**

集中式版本控制工具

SVN和CVS

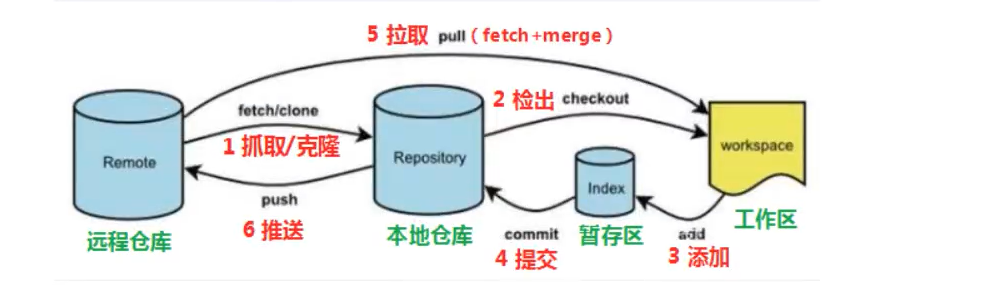
集中式版本控制工具,版本库集中存放在中央服务器,team里每个人work时,从中央服务器下载代码,必须联网才能工作,局域网或互联网,个人修改后提交到版本库

分布式版本控制工具

GIT

分布式版本控制系统没有中央服务器,每个人的电脑都可以是一个完整的版本库,这样工作的时候,无需联网,因为版本库就在你自己电脑中,多人协作只需要将各自的修改推送给对方,就能互相看到对方的修改

**2.Git工作流程**



clone:从远程仓库克隆代码到本地仓

checkout:从本地仓库检出一个仓库分支然后进行修订

add添加:在提交前,先将代码提交到缓存区

commit:提交到本地仓库,本地仓库保存修改的各个版本历史

fetch抓取:从远程库,抓取到本地库,不进行任何合并操作

pull拉取:从远处库拉取到本地库,自动合并然后放到工作区,相当于fench+merge

push:修改完成推送到远程仓库

**3.Git环境配置**

**3.1Git安装**

下载地址:<https://git-scm.com/download>

下载完成后直接安装出现以下两个代表安装成功

截图.png

Git基本配置

打开GitBash

设置用户信息

git config --global user.name"zgq"

git config --global user.email"zhanggaoqi@qq.com"

查看用户信息

git config --global user.name

git config --global user.email

为常用指令配置别名

有些常用的命令参数非常多,每次都要输入好多参数,我们可以使用别名

打开用户目录创建 .bashrc文件有部分系统不支持可以使用~/.bashrc

vi .bashrc文件 输入别名

alias git-log= 'git log --pretty=oneline --all --graph --abbrev-commit'

alias ll='ls -al'

执行source ~/.bashrc 或 source .bashrc

解决GitBash乱码问题

打开gitBash执行下面命令

git config --global core.quotepath false

${git\_home}/etc/bash.bashrc文件最后加入下面两行

export LANG="zh\_CN.UTF-8"

export LC\_ALL="zh\_CN.UTF-8"

**3.2获取本地仓库**

1.在电脑任意位置创建目录作为本地Git仓库

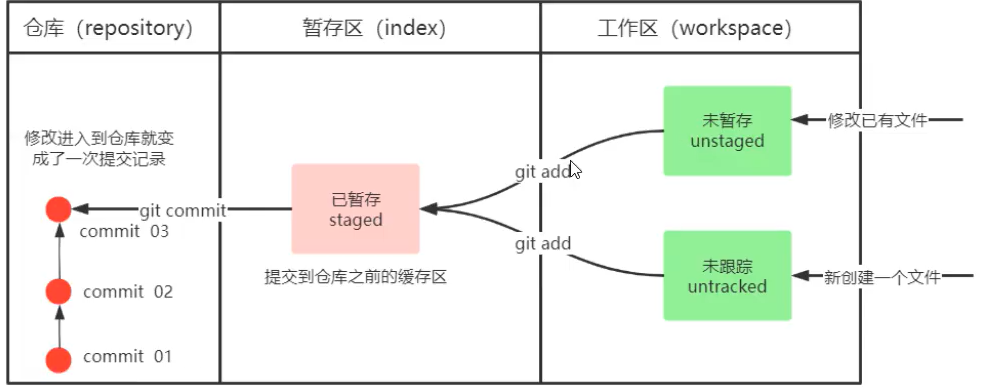
右键点击Git窗口

执行命令 git init

如果创建成功可以在文件下看到隐藏的git目录

**3.3基础操作命令**

Git工作目录下对于文件的修改增加删除更新会存在几个状态,这些修改的状态会随着我们执行Giy的命令而发生变化



1.git add (工作区-->暂存区)

2.git commit(暂存区-->本地仓库)

查看修改状态

作用:查看修改的状态(暂存区,工作区)

命令形式

git status

添加工作区到暂存区

作用:添加工作区一个或多个文件到暂存区

命令形式:git add 单个文件或通配符.

将所有修改加入到暂存区 git add .

提交暂存区到本地仓库

作用:提交暂存区内容到本地仓库的当前分支

命令行事:git commit -m "注释内容"

查看提交日志

作用:查看提交记录

命令形式:git log[option]

options:

--all 显示所有分支

--pretty=oneline 将提交信息显示为一行

--abbrev-commit 是的输出的commitd更简短

--graph 以图的形式显示

版本回退

作用:版本切换

命令形式:git reset -hard commitID

commitID可以通过git-log 或git log命令查看

如何查看已经删除的记录

git reflog

这个指令可以看到已经删除的提交记录

添加文件到忽略列表

一般来说我们总有一些文件不需要纳入Git的管理,也不希望他们总是出现在未跟踪文件列表如日志文件或者便是过程中的其他临时文件,这种情况下我们可以在工作目录下创建一个名为.gitignore的文件列出要忽略的文件

如 \*.txt 忽略所有以txt结尾的文件

**3.4分支**

几乎所有版本控制系统都以某形式支持分支,使用分支意味着你可以把你的工作从开发主线上分离开来进行重大Bug的修改开发新的功能以免影响开发主线

查看本地分支

命令:git branch

创建本地分支

命令:git branch 分支名

切换分支

命令 git check out 分支名

我们还可以切换到不存在的分支 git checkout -b 分支名

合并分支（有合并的快速模式，如果主分支没有，从分支有，在主分支合并就会快速合并）

一个分支上的提交可以合并到另一个分支

命令 git merge 分支名称

删除分支

不能删除当前分支,只能删除其他分支

git branch -d 分支名 删除分支时需要做各种检查

git branch -D 分支名 删除分支时不需要做各种检查（此分支东西比主分支多没有合并，可以使用强制删除）

查看分支绑定远程分支

git branch -vv

解决冲突（主分支和从分支同时修改同一个文件会产生冲突）

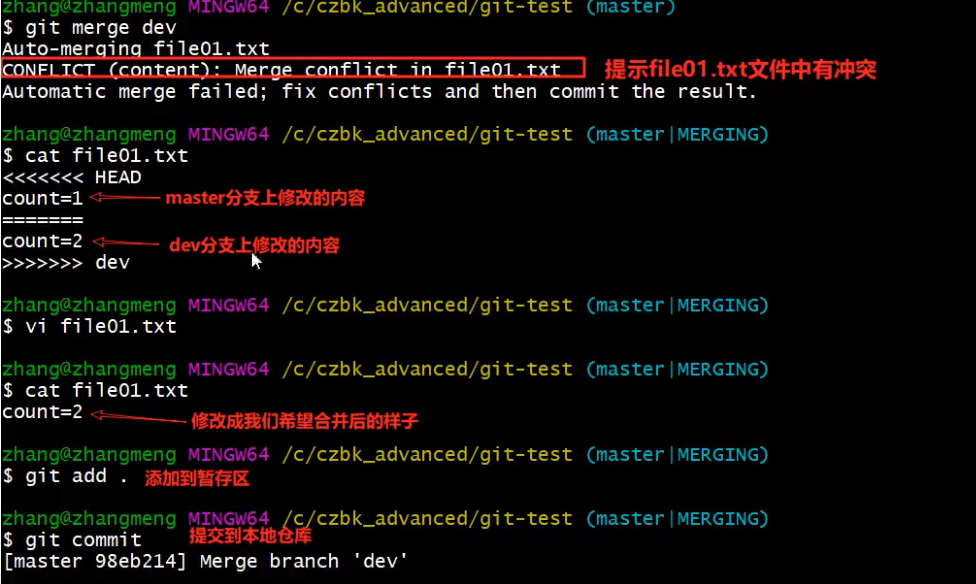
当两个分支对文件的修改可能存在冲突，例如同时修改同一个文件的同一行，这时就需要解决冲突，解决冲突步骤如下：

处理文件中冲突的地方

将解决完冲突的文件加入暂存区git add

然后提交到仓库git commit -m"提示信息"

这样当前所在分支冲突的地方就会根据自己处理的来，而另外的分支还是原来的数据



开发中分支使用原则与流程

几乎所有版本控制系统都以某种形式支持分支，使用分支意味着你可以把你的工作从主线上分离开来进行重大bug修改，开发新功能，以免影响开发主线

master生产分支

线上分支，主分支，中小规模项目作为线上运行的应用对应的分支

develop开发分支

是从master创建的分支，一般作为开发部门的主要开发分支，如果没有其他并行开发不同期上线要求都可以在此版本上进行开发，阶段开发完成后，需要和master分支合并

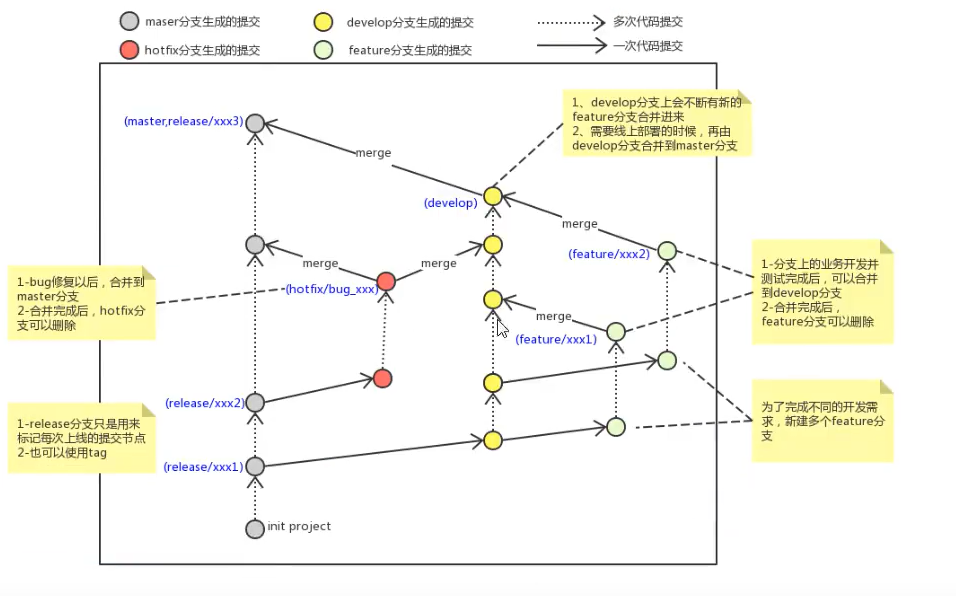
feature/xxx分支

从develop分支创建的分支，一般是同期并行开发，但不同期上线时创建的分支，分支上的研发任务完成后提交到develop分支

hotfix/xxx分支

从master分支派生的分支，一般作为线上bug修复使用，修复完成后需要合并到master，test，develop分支

还有其他一些分支如test（测试）分支 pre（预上线）分支



**4.Git远程仓库**

常用的托管服务（远程仓库）

GitHub 地址：<https://github.com>

码云 地址：<https://gitee.com>

GitLab 地址：<https://about.gitlab.com>

Git提交本地仓库到远程仓库

在本地仓库位置右击gitbash

git push https://gitee.com/cxyxz/git\_test.git

配置SSH公钥

生产SSH公钥

ssh-keygen -t rsa

回车

如果公钥已存在自动覆盖

Gitee设置 账户公钥

获取公钥

cat ~/.ssh/id\_rsa.pub



**验证是否配置成功**

**ssh -T** [**git@gitee.com**](http://git@gitee.com)

**操作远程仓库**

**添加远程仓库**

**命令 git remote add <远端名称> <仓库路径>**

**远端名称默认为origin取决于服务器设置**

**仓库路径：从远端服务器获取URL**

**例如：git remote add origin** git@gitee.com:cxyxz/git\_test.git

查看远程仓库

git remote

推送到远程仓库

命令行：git push [-f][--set-upstream][远端地址][本地分支名:远端分支名]

git push origin master:master

如果远程分支名和本地分支名相同

git push origin master

--set-upstream 推送到远端的同时和远端分支建立联系

git push --set-upstream origin master

如果当前分支已经和远程分支有关联可以省略分支名和远端名

git push 将master分支推送到已关联的远程分支

-f强制覆盖以提交为准

查看本地分支与远程分支关系

命令： git branch -vv

从远程仓库克隆

如果已经有一个远程仓库，我们可以直接clone到本地

从远程仓库抓取和拉取

远程分支和我们本地分支一样我们可以进行merge操作，只是需要先把远程仓库里的更新下载到本地在进行操作

抓取命令：git fetch [远程地址][远程分支名]

抓取指令就是将仓库里的更新抓取到本地不会合并

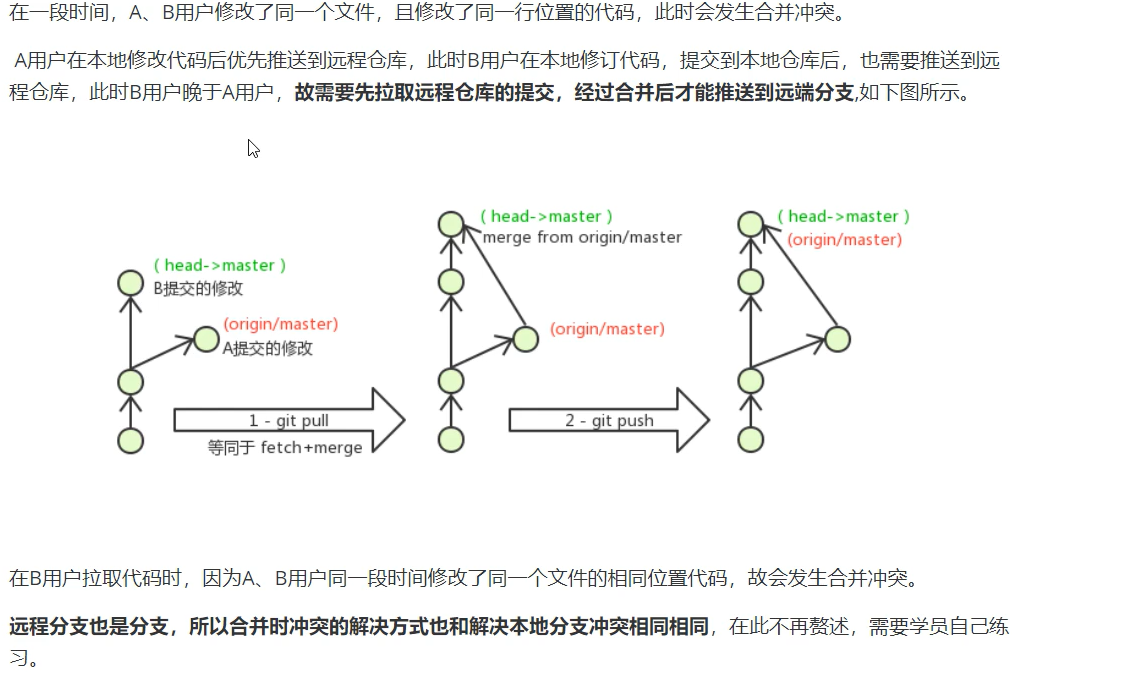
如果不能指定远程名称和分支名 就抓取所有分支

拉取命令：git pull [远程地址][远程分支名]

拉取指令就是将远程仓库的修改拉去到本地并自动进行合并等同于fetch merge

如果不指定远端名称和分支名 拉取所有并更新当前分支

解决合并冲突



先拉取远程仓库文件，发现冲突解决冲突提交到远程仓库

**5.在Idea中使用Git**

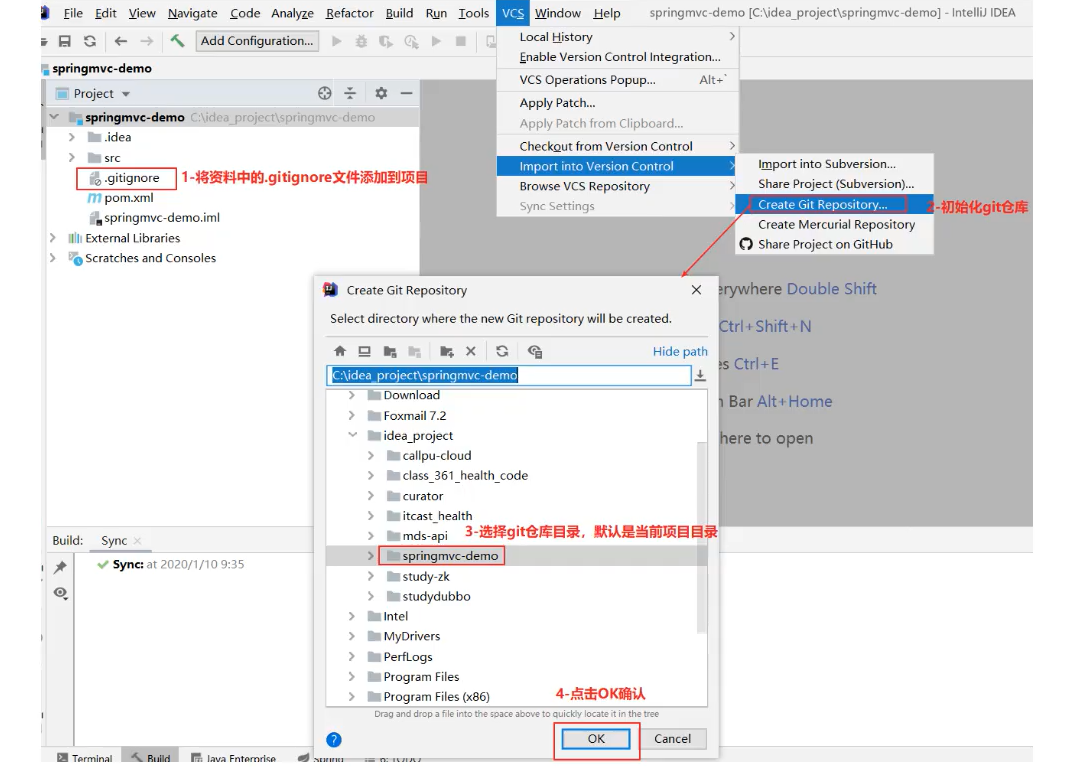
在Idea中配置Git

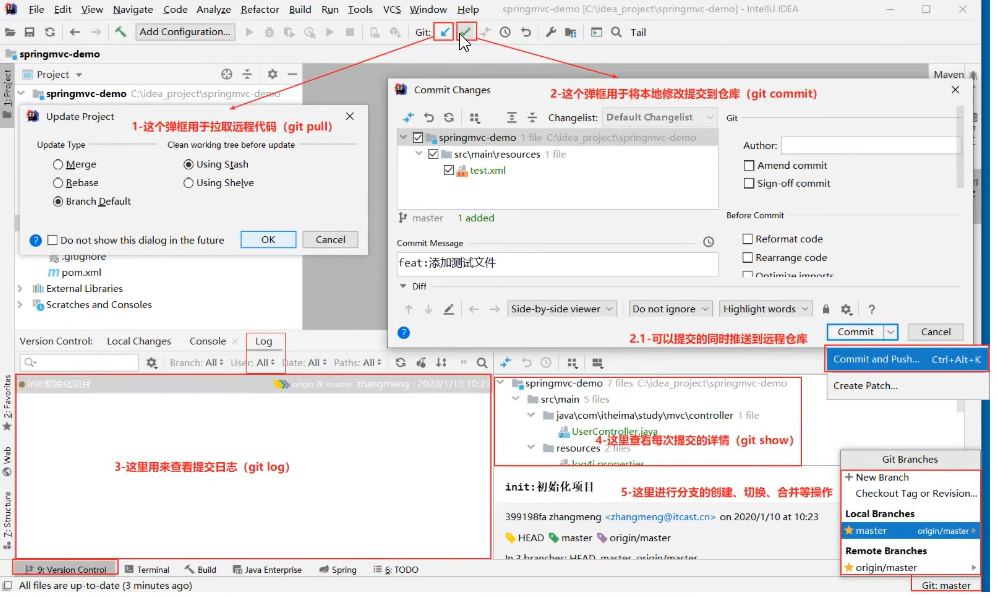
安装好idea后 setting--> version Control -->Git

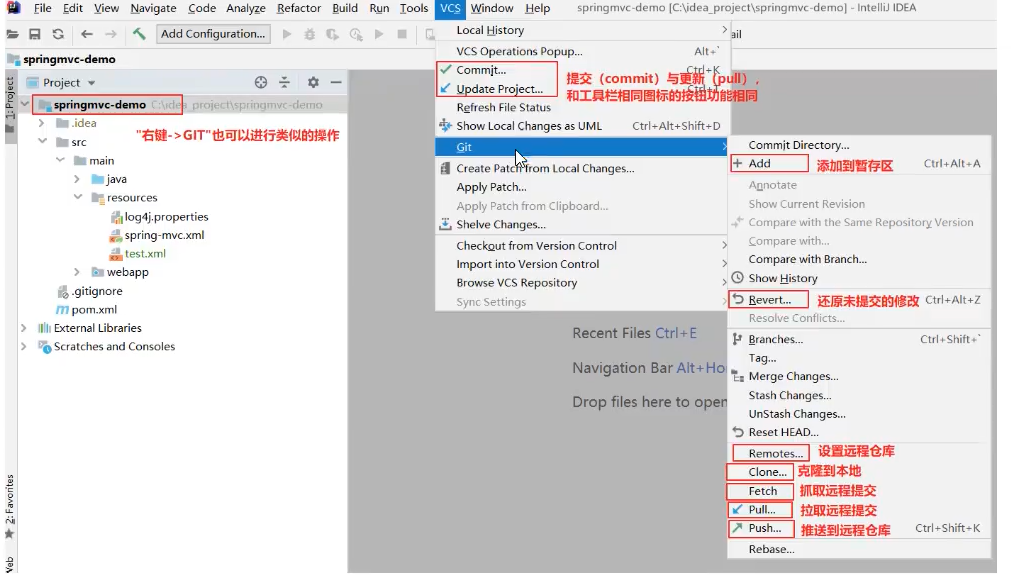
在idea操作GIt

第一步在码云上创建远程仓库

初始化本地仓库







集成gitbash到idea 将idea的setting下的terminal 的所指位置改为${git\_home}/bin/bash.exe