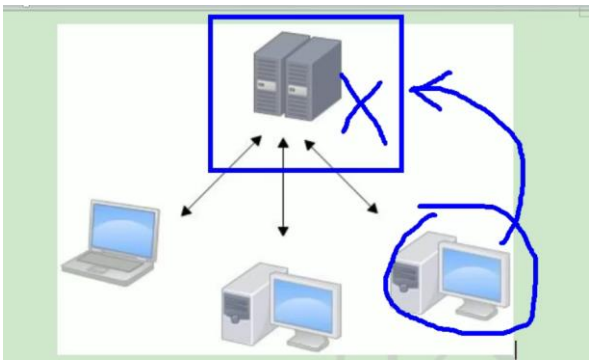
Git 版本控制

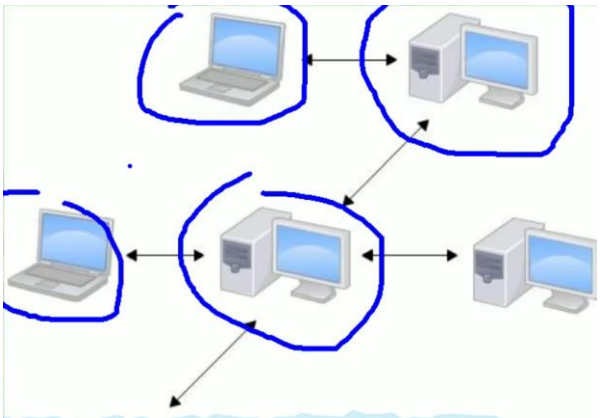
集中式版本控制工具　　客户端和服务器进行交互

缺点：当服务器挂掉，会损失历史记录



CVS, SVN, VSS

分布式版本控制工具 - 避免了单点故障



Git, Mercurial, Bazaar, Darcs……

版本控制工具应该剧本的功能

团队协作

数据备份

版本管理

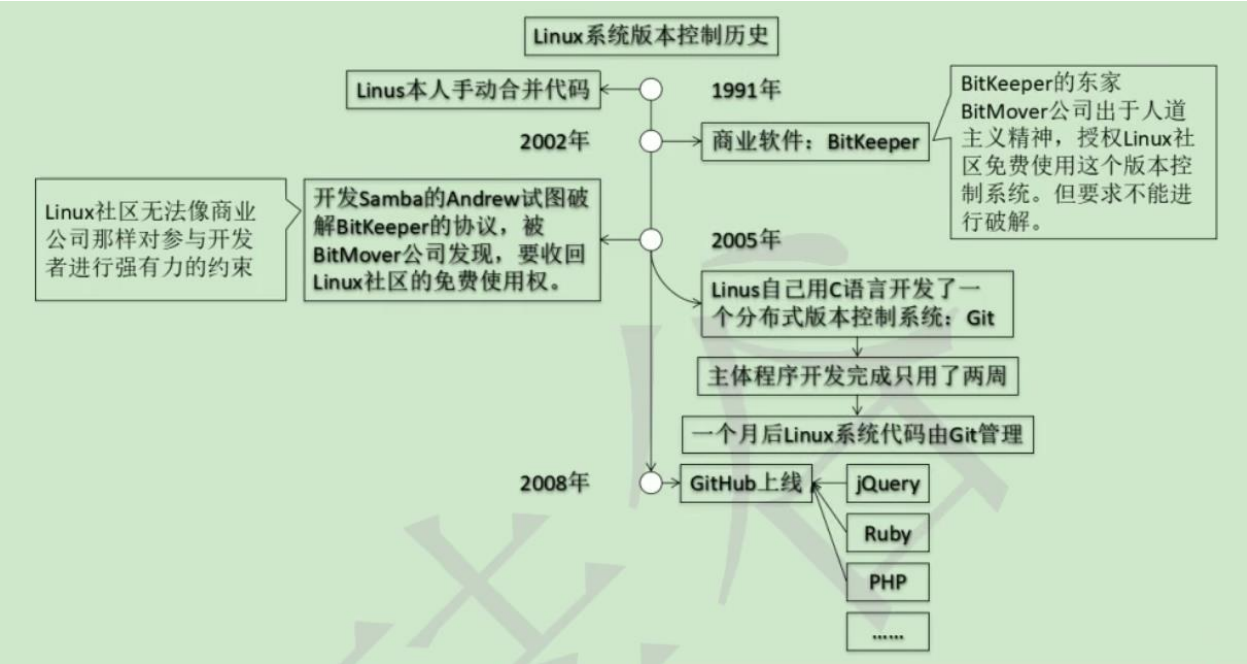
不断迭代、不断更新

权限控制

历史记录

分支管理

Git 发展历史：



个人开发改进迭代

Git 优势

大部分操作在本地完成，不需要联网

完整性保证

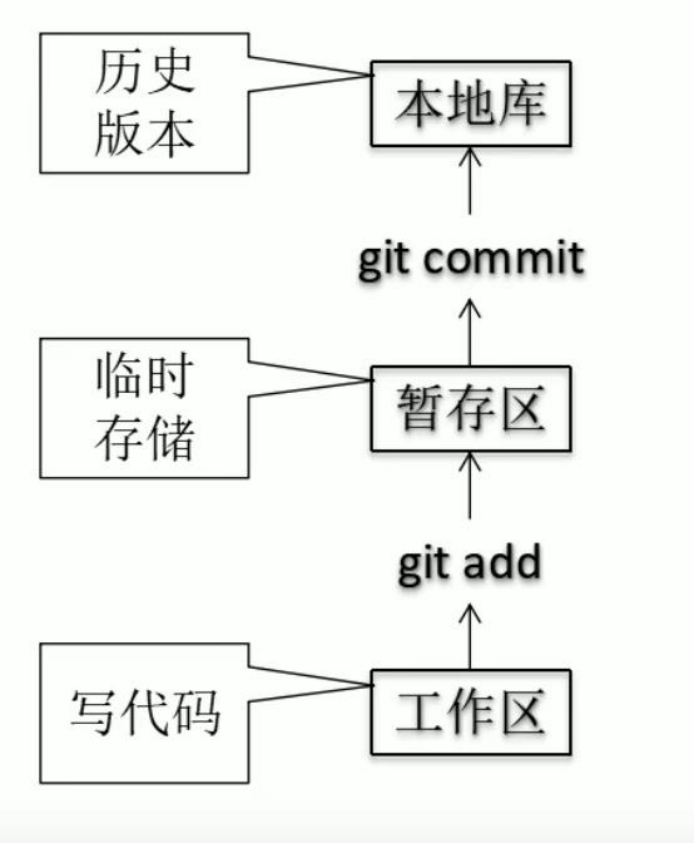
尽可能添加数据而不是删除或修改数据

分支操作非常快捷流畅

与 Linux 命令全面兼容

Git 结构

Git 和代码托管中心



Git 和代码托管中心

代码托管中心：维护远程库

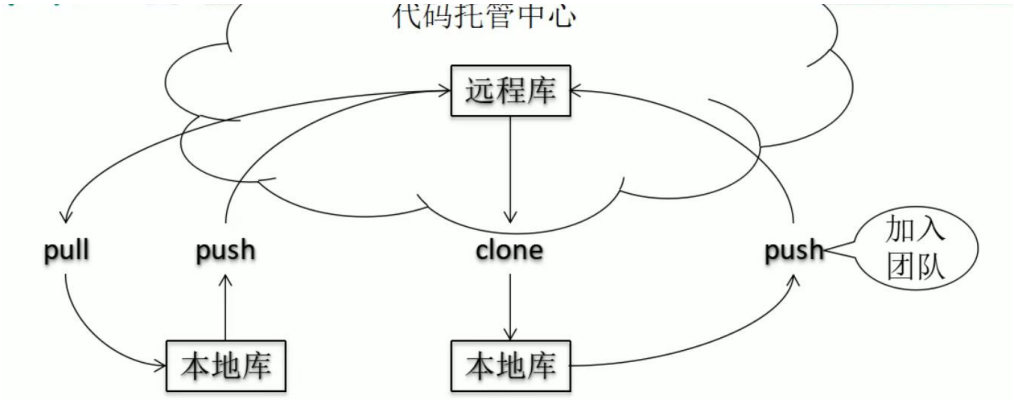
局域网环境： Gitlab

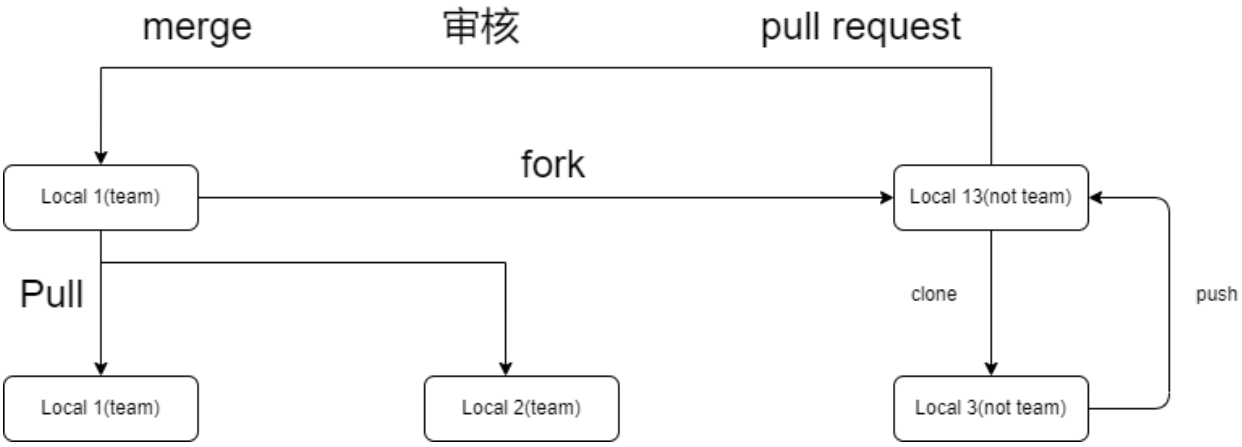
外网环境：Github、码云

本地库 & 远程库：

团队内部协作：

跨团队协作





Git 命令行操作

本地库初始化

命令：Git add

.git 目录中存放的是本地库相关的子目录和文件，不要删除或修改

设置签名

User: yiwen

Email: yiwen@gmail.com

用字符串标志开发人员的身份

辨析：设置的签名和登录远程库无任何关系

命令：

项目/仓库级别：尽在当前本地库范围内有效

git config user.name yiwen

git config user.email yiwen@gmail.com

信息保存位置：~/.git/config 文件

系统用户级别：登录当前操作系统的用户范围

git config --global user.name yiwen

git config –global user.email yiwen@gmail.com

级别优先级：

就近原则：项目级别优先于系统用户级别，二者都有时候采用项目级别的签名

如果只有系统级别的签名，就一系统用户级别的签名为准

不可以都没有

Git 命令:

```
yiwen@DESKTOP-QKNALJE MINGW64 ~
$ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local
 branch
   push              Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

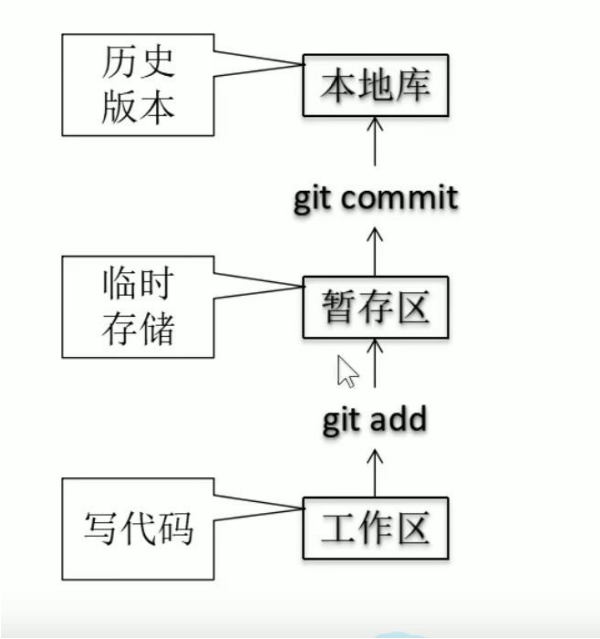
Git status:　　查看工作区状态、暂存区状态

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

```
$ git status
On branch master
nothing to commit, working tree clean
```

Git commit　　提交操作：提交文件：将暂存区的内容提交到本地库

Git add [file name]　　添加操作，把工作区的新建/修改添加到暂存区



查看历史：

Git log --pretty=online

Git log –oneline

Git reflog

多屏显示控制方式

空格：向下翻页　　B：向上翻页　　q：退出

在查看历史记录上前进/后退的版本

Head 指针

基于索引值操作

Git reset –hard [局部索引值]

```
Lenovo@DESKTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (master)
$ git reflog
fd83eb9 (HEAD -> master) HEAD@{0}: commit: insert qqqqqqqq edit
7bf0e31 HEAD@{1}: commit: insert pppppp edit
2679109 HEAD@{2}: commit: insert oooooo edit
9a9ebe0 HEAD@{3}: commit: insert nnnnnnnnn edit
49f1bd3 HEAD@{4}: commit: insert mmmmmmmm edit
a6ace91 HEAD@{5}: commit: insert lllllll edit
3dd95d7 HEAD@{6}: commit: insert kkkkkkkkk edit
42e7e84 HEAD@{7}: commit: insert jjjjjjjj edit
7c265b1 HEAD@{8}: commit: insert iiiiiiii
c309b92 HEAD@{9}: commit: insert hhhhhh edit
7305cd8 HEAD@{10}: commit: insert ggggggggg edit
ede116d HEAD@{11}: commit: insert fffffff edit
6325c55 HEAD@{12}: commit: insert eeeeeee edit
a709ad9 HEAD@{13}: commit: for test history
bfb79b7 HEAD@{14}: commit: My second commit,modify good.txt
ac5c801 HEAD@{15}: commit (initial): My first commit.new file good.txt

Lenovo@DESKTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (master)
$ git reset --hard 9a9ebe0
HEAD is now at 9a9ebe0 insert nnnnnnnnn edit
```

删除文件并找回：

前提：删除前： 文件存在时的状态提交到了本地库

操作：git reset –hard HEAD[指针位置]

删除操作已经提交到本地库：指针位置指向历史记录

删除操作尚未提交到本地库：指针位置使用 HEAD

比较文件：

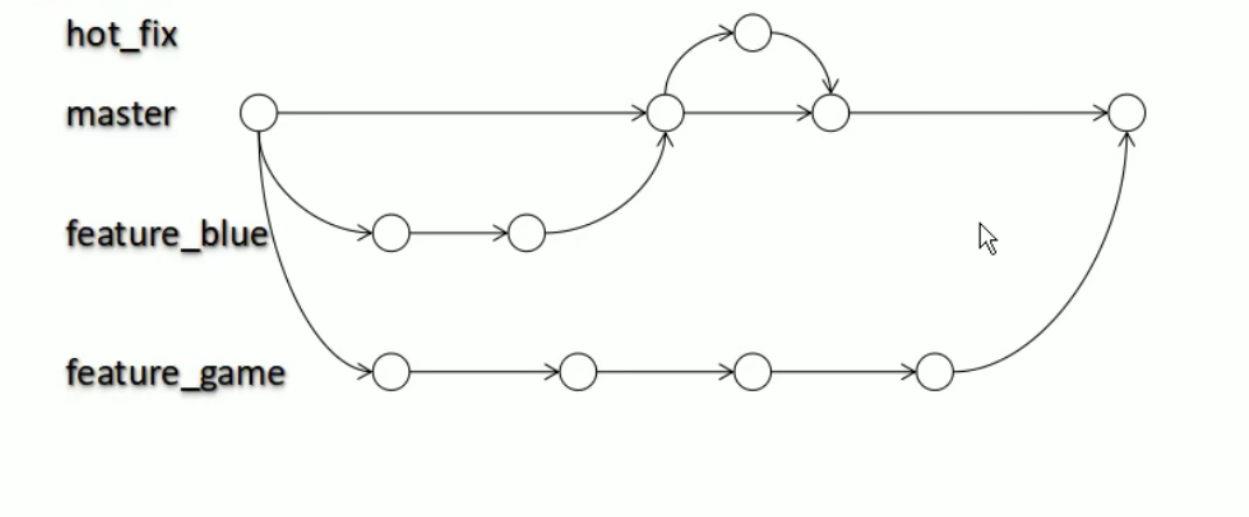
Git diff[文件名] 将工作区中的文件何暂存区的文件进行比较

Git diff[本地库中历史版本][文件名] 将工作区中的文件何本地库历史记录比较

```
$ vim apple.txt

Lenovo@DESKTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (master)
$ git diff apple.txt
diff --git a/apple.txt b/apple.txt
index 255c39f..27a377a 100644
--- a/apple.txt
+++ b/apple.txt
@@ -1,6 +1,6 @@
 apple
 apple
-apple
+apple @@@@@@@@@@@@
 apple
 apple
 apple
```

分支管理:

Git 分支：在版本控制过程中，使用多条线同时推进多个任务



分支的好处:

同时并行推进多个功能开发，提高开发效率

各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响。失败的分支删除重新开始即可。

分支操作

Git status

查看分支　　Git branch -v

创建分支　　Git branch [分支名]

切换分支　　Git checkout[分支名字]

合并分支

第一步：切换到接受修改的分支上（被合并，增加新内容）上

Git checkout [被合并分支名]

第二步：执行 merge 命令　　git merge [有新内容的分支名]

解决冲突

第一步：编辑文件，删除特殊符号

第二步：把文件修改到满意的程度，保存退出

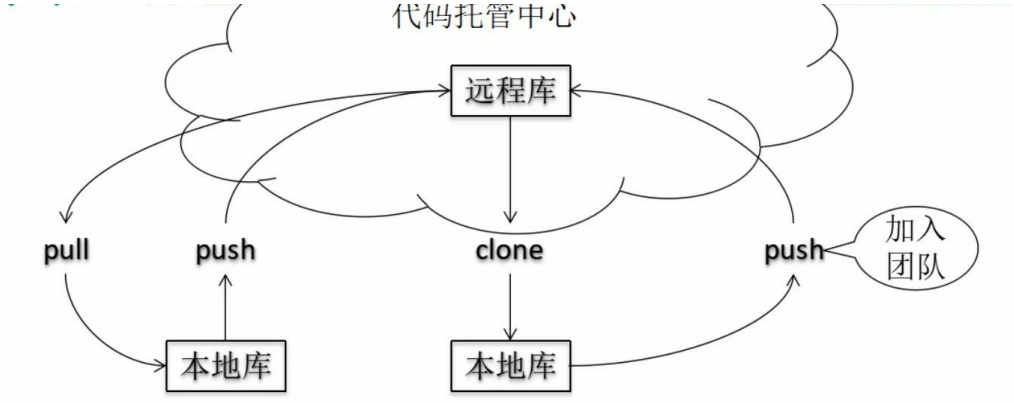
第三步：git add[文件名]

第四步：git commit -m “日志信息”

注意：此时 commit 一定不能带具体文件名

和另一个分支的作者商讨，修改好后，写入

本地库和远程库的交互



HTTP 地址

SSH 地址

Git remote -v 查看远程地址

Git remote add origin http://…….git 给地址赋值为 origin

推送：

Git push [别名][分支名]

Git Push origin master 添加 github 账号密码

从本地 master 到远程库 master

克隆：

命令：Git clone https://…….git

把远程库克隆到本地

完整的把远程库下载到本地

创建 origin 远程地址别名

初始化本地库

邀请成员加入团队

Collaborators, invite by adding email

成为团队成员后，git push origin master

拉取 pull = (fetch + merge)

Git fetch origin master　　拉取把远程库

Git checkout origin/master　　可以看到拉取的新的内容

再切换回去

Git merge origin/master　　远程 master 合并到本地 master

Git fetch [远程库地址别名][远程分支名]

Git Merge [远程地址别名/远程分支名]

Git pull [远程库地址别名]

协同开发时发生冲突的解决

如果不是 github 远程库的最新版本所作的修改，不能推送，必须先拉取

拉取后如果进入冲入状态，按照分支冲突状态解决

SSH 免密登录

进入当前用户的 home 目录

~cd

删除.ssh 目录

$rm -rvf.ssh

运行命令生成.ssh 密钥目录

Ssh-keygen -t rsa -C email

进入.ssh 目录查看文件列表

$cd.ssh

$ls -lF

查看 id_wsa.pub

复制 id_rsa.pub 文件内容，登录 github, setting——SSH and GPG keys

New SSH key

输入复制的蜜月信息

回到 Git bash 创建远程地址别名

Git remote add origin_ssh git@github……

推送文件进行测试