

git

利用git 提交 把本地网站提交到 码云新建的仓库里面

- 在网站根目录右键-- Git Bash Here
- 如果是第一次利用git提交，请配置好全局选项

```
git config --global user.name "用户名"  
git config --global user.email "你的邮箱地址"
```

- 初始化仓库

```
git init
```

- 把本地文件放到暂存区

```
git add .
```

- 把本地文件放到本地仓库里面

```
git commit -m '提交黑马面面网站'
```

- 链接远程仓库

```
git remote add origin 你新建的仓库地址
```

- 把本地仓库的文件推送到远程仓库 push

```
git push -u origin master
```

利用push提交可能会弹出windows的弹出窗口输入的用户名是git里面的用户名不是名字

git学习新手入门:

4.配置作者信息:

ctrl+l 清屏

subl

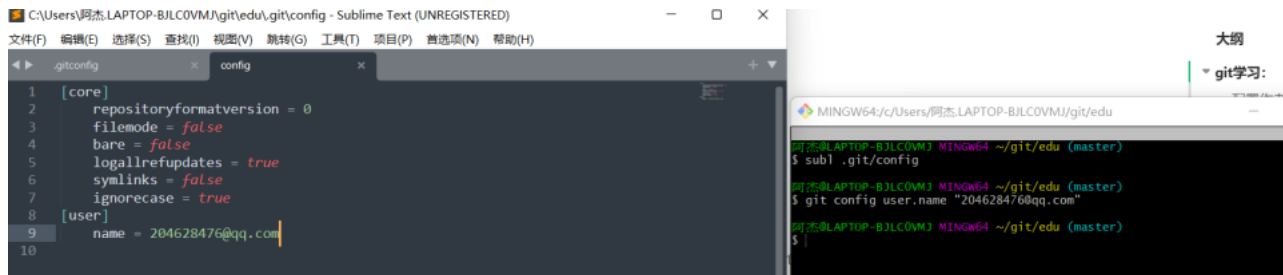
ll 显示详细列表 mkdir创建目录 pwd 查看当前在那个文件夹 cd 回家目录 ls目录列表

git init: 在本地创建了一个版本仓库

这个版本仓库也可以配置信息比如我在 edu 是我创建的目录在它里面创建了版本仓库

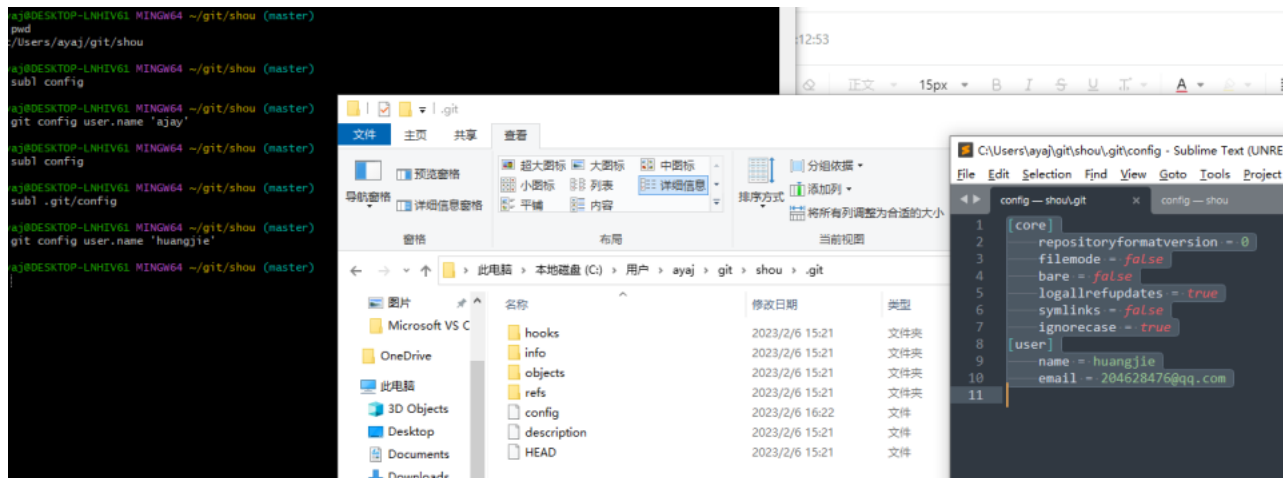
我在edu这个目录下面创建了一个本地版本库，它里面也会有一个git 是edu里面的git

要先标识自己是谁，否则git没法用



global 全局的意思

我们创建的shou目录里面还有一个版本库 这个版本库里面有一个git文件 只要 git init 就会出现这个文件 然后给这个文件声明一下 信息



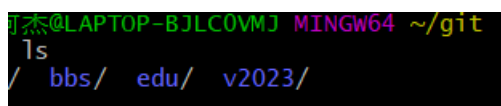
当我们写了很多项目的时候，提交都可以用全局这个帐号。

```
git config --global user.email "204628476@qq.com"
git config --global user.name "ayaj"
```

创建新仓库与维护久仓库：

rm -rf 是删除所有的文件夹 用的时候谨慎

记住 git下面的文件一般都是项目的文件夹，项目文件夹才会放一些项目 所以主目录是不会存放项目文件的。



维护久仓库 比如我们现在去网上想维护一个久仓库 直接复制http地址 用git命令克隆下来就可以

```

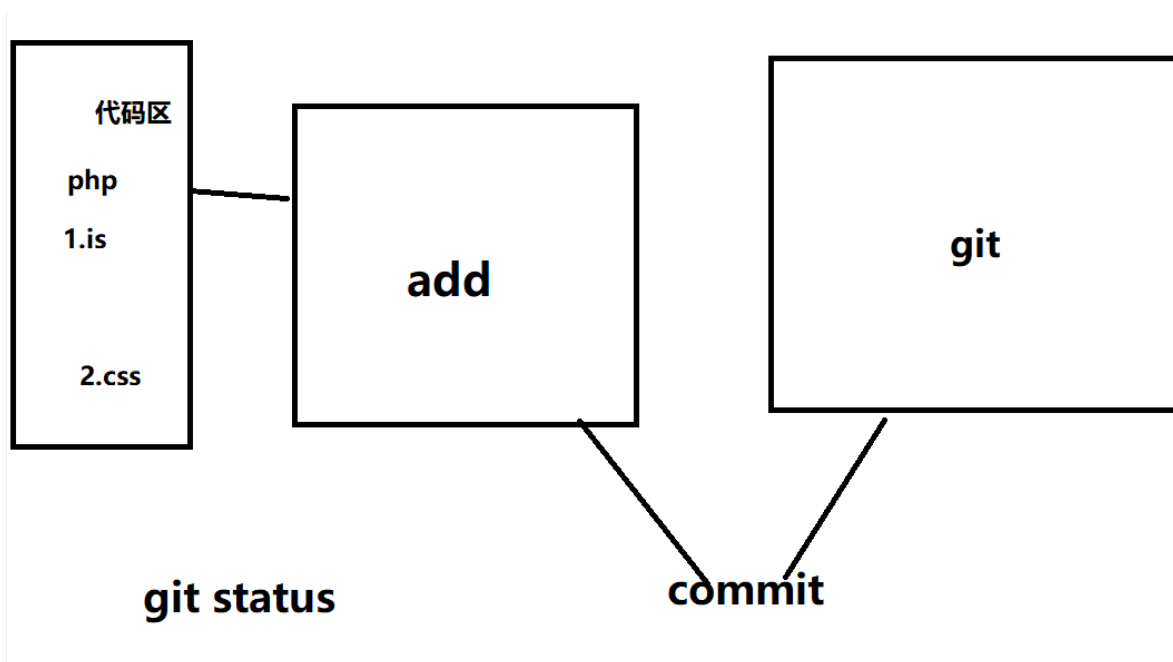
阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ git clone https://github.com/houdunwang/v2023.git
Cloning into 'v2023'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 59 (delta 4), reused 53 (delta 2), pack-reused 0
Receiving objects: 100% (59/59), 1.14 MiB | 54.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ ls
a/  bbs/  edu/  v2023/

```

可以看到我去向军大叔网址扒了一个项目下来 项目的文件夹名字会以v2023保存，当我们查看git下面就会有这个文件其他两个 是自己创建的

git流水线操作分析：



git就是一个仓库，我们所写的一些代码都会放到git里面，来保存我们的代码

那么我们就想知道 代码是怎么放到仓库里面去的，它有个怎么样的流程。

好比现实当中 git是仓库 我们写的代码都是一件件货物我们要把货物放到仓库里面去 就要有一个车子，把货物放到车子里面运到仓库里去 那么 add 就是这个车子 把我们的代码放到车子里面 然后通过 commit 发送到git。

git status查看代码区，代码区可以理解就是生产的车间 生产完以后装到车然后运往仓库

使用命令完成git流水线操作分析：

touch 创建一个空白的文件夹

```
阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/aj (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        p.txt

nothing added to commit but untracked files present (use "git add" to track)
```

查看代码区的状态 上面代码标识 这个文件没有被跟踪 就是没有提交到git仓库

查看提交到车上样子

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   p.txt
```

就是绿色的 下一步就是使用 commit发送到git

```
阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/aj (master)
$ git commit -m "测试用的"
[master (root-commit) 5720014] 测试用的
 2 files changed, 1 insertion(+)
 create mode 100644 a.php
 create mode 100644 p.txt
```

利用 commit发送到git 仓库，我们货物送到仓库都要登记吧，这是什么货物 程序也一样 用-m写一张表。

如果我修改了本地的文件会发生上面呢？

```
$ git status
On branch master

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working dir)
        modified:   a.php
```

modified a.php 表示这个文件修改了，我们重新把它放到车里发送到git仓库

有时候我创建了很多文件不可能一个个打上去提交 此时可以利用 git add . 就可以全部提交到车里，车里就可以直接全部发送到git仓库

```

a.css
b.css
c.css

nothing added to commit but untracked
阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/a
$ git add .

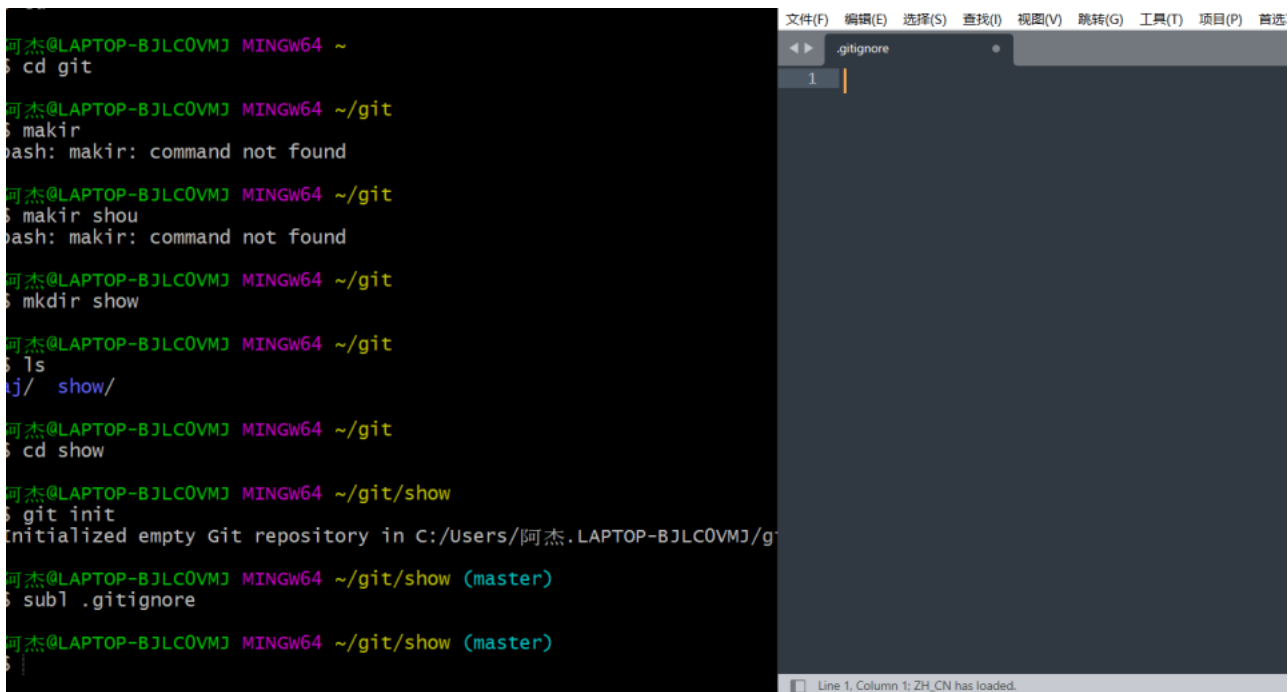
阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/a
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   a.css
    new file:   b.css
    new file:   c.css

```

那么又会有一个疑问 有时候我并不要文件全部上传有一些我不想上传 这就会引出另一个问题

gitignore详解控制版本库文件管理：

接着上面留的疑问，有些文件我们不想上传 就需要一个版本库忽略文件配置



```

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~
$ cd git

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ mkdir
bash: mkdir: command not found

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ mkdir show
bash: mkdir: command not found

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ mkdir show

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ ls
./ show/

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git
$ cd show

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/show
$ git init
Initialized empty Git repository in C:/Users/阿杰.LAPTOP-BJLC0VMJ/git/show/

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/show (master)
$ subl .gitignore

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/show (master)
$

```

通过subl .gitignore 创建这个版本库配置，这里就可以配置那些文件不要提交

```
Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
a.txt
b.txt
c.txt

nothing added to commit but untracked files present (use "git add" to track)
```

此时可以看到我们创建了三个文件，我们放到版本库忽略文件当中

```
$ git status
On branch master

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
```

可以看到这样只会存放后缀名txt的文件。

反过来我只想 存放 b 和 c 不想存放 a

```
.gitignore
a.txt
index.css
```

如果我在这个目录里面又创建了一个目录，那么如果不添加也会现实在外面所以我们也要加上

```
On branch master

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
a.txt
verdor/

nothing added to commit but untracked files present (use "git add" to track)

阿杰@LAPTOP-BJLC0VMJ MINGW64 ~/git/show (master)
$ git status
On branch master

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
a.txt
```

从版本库中删除资源技巧： rm 移除版本库

我们提交了文件到版本库，如果使用rm删除 这时候本地的文件也会一起删除的

那么有时候我提交的到版本库的文件想删除掉但是本地不想删除怎么办 不慌有这个命令git rm --cached

```
ayaj@DESKTOP-LNHIV61 MINGW64
$ touch index.html

ayaj@DESKTOP-LNHIV61 MINGW64
$ ls
index.html

ayaj@DESKTOP-LNHIV61 MINGW64
$ git add .

ayaj@DESKTOP-LNHIV61 MINGW64
$ git commit -m '提交index.html'
[master 2bd91cd] 提交index.html
1 file changed, 0 insertions, 0 deletions
rename a.txt => index.html (100%)

ayaj@DESKTOP-LNHIV61 MINGW64
$ ls
index.html

ayaj@DESKTOP-LNHIV61 MINGW64
$ git rm --cached index.html
rm 'index.html'

ayaj@DESKTOP-LNHIV61 MINGW64
$ ls
index.html
```

可以看到本地还是存在的，只是把版本库里面的删除了

版本库中修改资料名称：

mv 改名

```

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aa (master)
$ git commit -m 'a.js'
[master 8b9d989] a.js
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 a.js

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aa (master)
$ ls
a.js  index.html

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aa (master)
$ git mv a.js houdunren.js

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aa (master)
$ gis status
bash: gis: command not found

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aa (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    a.js -> houdunren.js

```

11.使用log日志查看历史操作:

每次提交都会给一个哈希字符串

```

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/ajie (master)
$ git log
commit cf79d0253c7ef688a592d336098cddd4a9b191f9 (HEAD -> master)
Author: ajay <204628476@qq.com>
Date:   Mon Feb 6 21:58:42 2023 +0800

    two

commit a67473d1463ad3af460b5c54f4edae9539bc3283
Author: ajay <204628476@qq.com>
Date:   Mon Feb 6 21:49:14 2023 +0800

    ajie

```

加山-p 就可以看到详细信息 还可以看到我们修改里什么内容


```

$ git log -p
commit cf79d0253c7ef688a592d336098cddd4a9b191f9 (HEAD -> master)
Author: ajay <204628476@qq.com>
Date:   Mon Feb 6 21:58:42 2023 +0800

    two

diff --git a/a.php b/a.php
index e69de29..f3eab73 100644
--- a/a.php
+++ b/a.php
@@ -0,0 +1 @@
+www.huangjie.site
diff --git a/b.php b/b.php
new file mode 100644
index 0000000..e69de29

commit a67473d1463ad3af460b5c54f4edae9539bc3283
Author: ajay <204628476@qq.com>
Date:   Mon Feb 6 21:49:14 2023 +0800

    ajie

diff --git a/a.php b/a.php
new file mode 100644
index 0000000..e69de29

```

git log -p -1 --oneline --name--only --name-status

后面的 p 是显示段落 还可以跟 数字几就是几条 --oneline 简短 --name--only 那些文件发生变化

--name-status 文件发生了什么变化 修改还是删除

git log 查看最近提交的日志 简单来说就是你最近提交里什么东西 谁提交的 什么时候提交的

12.利用amend修改最新一次提交事件

MINGW64:/c/Users/ayaj/git/ajie

我利用amend修改第二次事件two

```
# Please enter the commit message for your changes
# with '#' will be ignored, and an empty message a
#
# Date:      Mon Feb 6 21:58:42 2023 +0800
#
# On branch master
# Changes to be committed:
#       modified:   a.php
#       new file:   b.php
#
```

调出vim界面 然后修改事件的名称就可以里

此时我们想把这两个文件也放到刚刚的日志里面怎么办

```
ayaj@DESKTOP-LNHIV61 MING
$ touch a.css b.css
```

```
ayaj@DESKTOP-LNHIV61 MIN
$ git add a.css b.css
```

先把这两个放到暂存区

```
ayaj@DESKTOP-LNHIV61 MIN
$ git commit --amend
```

利用--amend 进入vim加入这两个文件就可以

查看日志看到 添加成功

```
$ git log -1 --name-only
commit 421aa6299faad5481e0c5dc5f2c25cd4
Author: ajay <204628476@qq.com>
Date:   Mon Feb 6 21:58:42 2023 +0800
```

我利用amend修改第二次事件two
增加 a.css b.css

```
a.css
a.php
b.css
b.php
```

13.占存区中文件的管理:

有时候呢, 我们可能有些文件不想放到占存区里面都是手快了, 怎么移回去呢, 利用

```
$ git rm --cached a.txt  
rm 'a.txt'
```

git rm --cached 文件名

这个时候如果我们这个文件已经提交到仓库里了，我们在本地里又改了一下这个文件，那么我们第二次上传这个文件又想移出来怎么办

```
(use "git restore --staged <file>..." to unstage  
modified: a.txt  
ayaj@DESKTOP-LNHIV61 MINGW64 ~  
$ git restore --staged a.txt  
ayaj@DESKTOP-LNHIV61 MINGW64 ~  
$ git status  
On branch master  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes in working directory)  
modified: a.txt
```

第二次提交 利用 git restore --staged 文件名就可以移

出来

```
ayaj@DESKTOP-LNHIV61 MINGW64 ~  
$ git checkout --
```

如果想让这次文件里面的内容也变成第一次提交的内容就可以用

14.利用alias命令起别名提高效率:

```
[user]
name = ayaj
email = 204628476@qq.com

[alias]
a = add
s = status
l = log
c = commit
t = touch

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ git c
[master (root-commit) 0f929a6] '测试利
1 file changed, 0 insertions(+), 0 de
create mode 100644 a.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ git config --global alias.t
touch

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ t b.txt
bash: t: command not found

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ touch b.txt

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ pwd
/c/Users/ayaj/git/aj

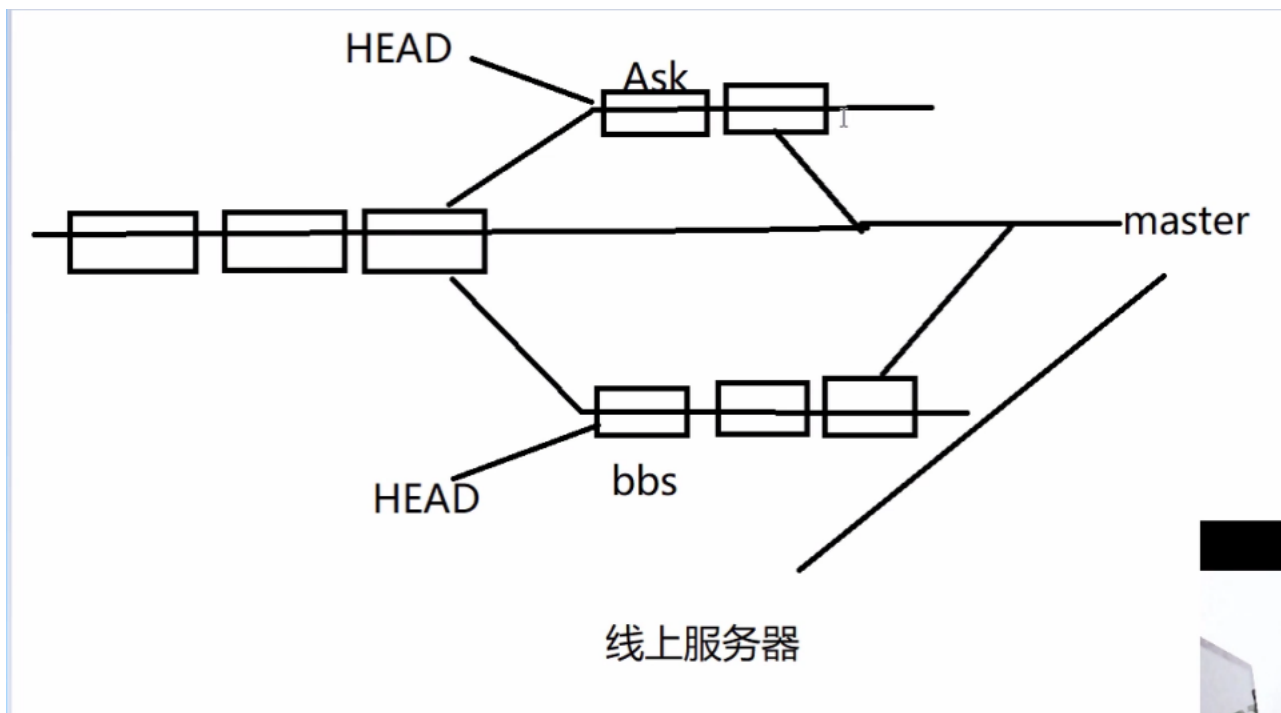
ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ ls
a.php b.txt

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ rm -rf alias.t touch

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj
$ git config --global alias.t touch
```

可以提高效率，节省这些重复利用的关键字

15.git分支:



分支就是我们一次次提交代码会形成一条主分支 master 然后我们可能会遇到一些bug或者错误可以开辟一些分支来添加功能 每个分支也是隔开的，完成以后重新添加到主分支

16.branch分支基本管理操作：

使用branch可以查看分支 前面的星号就是表示你当前在那个分支：

```
$ git branch
* master
```

创建分支 git branch 分支名

```
$ git branch ask

ayaj@DESKTOP-LNHIV61
$ git branch
ask
* master
```

切换分支

```
$ git checkout ask
Switched to branch 'ask'

ayaj@DESKTOP-LNHIV61 MINGW64
$ git branch
* ask
master
```

不同分支下面创建的代码都是不相通的 在ask下面创建的文件 提交里 回到主分支里面是不会显示的

```

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ touch ask.html

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git a .

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git s
On branch ask
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    a.php -> ask.html

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git commit -m 'ask commit'
[ask 9fe2aa0] ask commit
1 file changed, 0 insertions(+), 0 deletions(-)
rename a.php => ask.html (100%)

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ ls
ask.html

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git checkout master
Switched to branch 'master'

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ git branch
ask
* master

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ ls
a.php

```

这里就执行里两个命令创建bbs分支 并切换到这个分支

```

$ git checkout -b bbs
Switched to a new branch 'bbs'

```

17.合并分支删除分支：

总结当我们这块分支代码已经结束写完了我们就可以把它添加到主分支里面，添加完这个分支就没用了然后删除就可。

利用 merge 跟上分支名 就可以合并这个分支到主分支里面

```

ayaj@DESKTOP-LNHIV61 MINGW64
$ git merge ask
Updating 0f929a6..9fe2aa0
Fast-forward
 a.php => ask.html | 0
 1 file changed, 0 insertion
 rename a.php => ask.html (1
ayaj@DESKTOP-LNHIV61 MINGW64
$ ls
ask.html

```

删除分支

```


ayaj@DESKTOP-LNHIV61 MINGW64 ~/gi
$ git branch -d ask
Deleted branch ask (was 9fe2aa0).

ayaj@DESKTOP-LNHIV61 MINGW64 ~/gi
$ git branch
bbs
* master

```

18.正确处理分支冲突：

怎么看待分支冲突 主分支提交里一个文件 然后我创建里两个分支 ask 和bbs 然后在这两个分支里面都改变了这个文件注意一点 这个时候文件里面的内容就是 你主分支提交的内容 我们在两个分支改这个文件是没有问题的 但是当我们合并的话就会出现问题，比如下方我先合并里 ask 发现正常 然后合并bbs里面就报错里 显示有冲突让你解决，然后我们使用vim打开就可以看到 我们可以选择要保存那个文件，如果都保存也可以。



```
ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (bbs)
$ git checkout master
Switched to branch 'master'

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ git merge ask
Updating 00abdc3..9da84d6
Fast-forward
 a.php | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ ls
a.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ cat a.php
我是ask分支里面的php1

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ git merge bbs
Auto-merging a.php
CONFLICT (content): Merge conflict in a.php
Automatic merge failed; fix conflicts and then commit the result.

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ vim a.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (master)
$ cat a.php

我是ask分支里面的php1

我是bbs里面的a.php
```

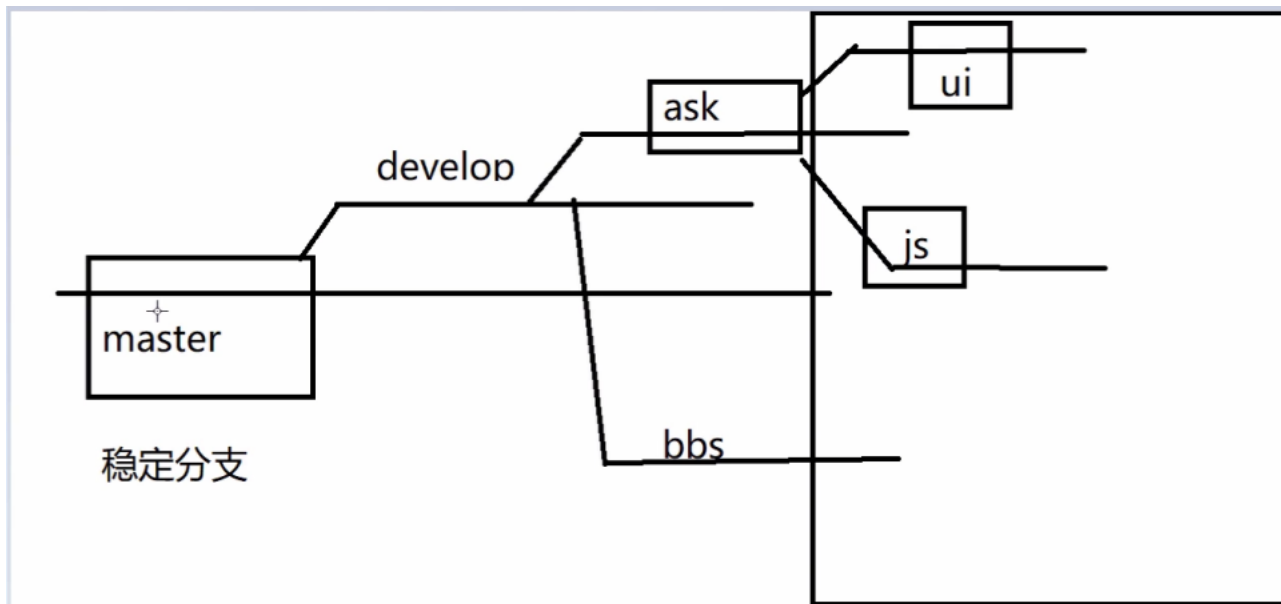
19.分支管理merged及分支强制删除:

git branch --merged //查看已经合并的分支

git branch --no-merger 查看没有合并的分支

git branch -D 一般来说没有合并的分支是不可以删除的 这是git提醒你有时候怕你删错但如果你是真的不想这个分支就加上大D

20.标准的分支操作 workflow:



首先肯定有稳定分支，然后稳定分支会有一个develop这个一看就是项目 然后这个项目肯定还有各种各样的功能在生成各种各样的分支 每个分支 写完以后在放到项目分支里面

git进阶:

stash临时储存区:

```

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git add .

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git commit -m'ask.php'
[ask 79fcd7b] ask.php
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ask.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ vim ask.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git add .
warning: in the working copy of 'ask.php', LF will be

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git s
On branch ask
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   ask.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git checkout bbs
error: Your local changes to the following files would
      ask.php
Please commit your changes or stash them before you s
Aborting

```

我们在ask分支里面已经提交里一次，但是这个时候我想修改一下然后修改完以后提交第二次的时候 我想切换到别的分支 是切换不了的 会报错 让你解决ask分支，这个时候我不想结束这个暂存区里面的文件，但是我又想切换到别的分支该怎么办，就出现了临时储存区。

利用git stash就临时暂存起来了，git stash list 查看

```

$ git stash
Saved working directory and index state WIP on ask: 79fcd7b ask.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/aj (ask)
$ git stash list
stash@{0}: WIP on ask: 79fcd7b ask.php

```

此时我们就可以切换到别的分支进行操作，操作完以后回到分支 然后利用

git stash apply 就可以恢复分区

```

ayaj@DESKTOP-ENH1V61 MINGW64 ~/git/aj (ask)
$ git stash apply
On branch ask
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   ask.php

no changes added to commit (use "git add" and/or "git commit -a")

```

如果不要这个暂存区可以利用git stash drop stash@{0} 后面的是第几个暂存

```

$ git stash drop stash@{0}
Dropped stash@{0} (5e1484b3a1992a6200547a1ff6bb62269ecd7076)

```

生成zip代码发布压缩包：

```

ayaj@DESKTOP-ENH1V61 MINGW64 ~/git/aj (ask)
$ git archive master --prefix='huangjie/' --format=zip > huangjie.zip

```

使用系统别名定义git全局指令：

The screenshot shows a Sublime Text editor window titled "C:\Users\ayaj\.bash_profile - Sublime Text (UNREGISTERED)". The editor has three tabs open: ".gitconfig", ".bash_profile", and "config". The ".bash_profile" tab is active, displaying a list of git aliases. The text in the editor is as follows:

```

1  alias gs="git status"
2  alias gc="git commit -m "
3  alias gl="git log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset :
4  alias gb="git branch"
5  alias ga="git add -A"
6  alias go="git checkout"
7  alias gp="git push;git push github"

```

The status bar at the bottom indicates "Line 7, Column 36", "Tab Size: 4", and "Bash".

gs 查看代码状态
gc 提交代码 并填信息
gl 查看日志
gb 查看分支
ga 提交到暂存区
go 切换分支 后面跟分支名
gp

合并分支产生的问题：

我们在主分支master写了一个master.php的文件 然后提交 提交完以后创建里一个分支交ask分支 然后ask分支也写了一个文件 叫ask.php文件 我们也提交 这个时候主分支合并是没有什么问题的，那么怎么样会产生分支合并的问题呢，我们回到里主分支 又修改里一些内容提交 这个时候我们在合并分支就会产生分支合并的问题

这张图是没有问题的 主分支没有修改：

```
$ git merge ask
Updating 41ec195..6960bc5
Fast-forward
 ask.php | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ask.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/huangjie (master)
$ gl
* 6960bc5 - (HEAD -> master, ask) ask.php commit (4 minutes ago) <ayaj>
* 41ec195 - master commit (5 minutes ago) <ayaj>
```

修改里主分支内容并提交，并合并ask子分支，可以看到下面发生里变化合并了

```
ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/hd (master)
$ touch master2.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/hd (master)
$ ga

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/hd (master)
$ gc 'master2.php'
[master e6846e8] master2.php
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master2.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/hd (master)
$ git merge ask
Merge made by the 'ort' strategy.
 ask.php | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ask.php

ayaj@DESKTOP-LNHIV61 MINGW64 ~/git/hd (master)
$ gl
* 63f9b3b - (HEAD -> master) huangjie.site Merge branch 'ask' (24 seconds ago) <ayaj>
|
| * 5dc46eb - (ask) ask ask.php (3 minutes ago) <ayaj>
* | e6846e8 - master2.php (44 seconds ago) <ayaj>
|/
* 2bb5ffd - master hd.php (4 minutes ago) <ayaj>
```

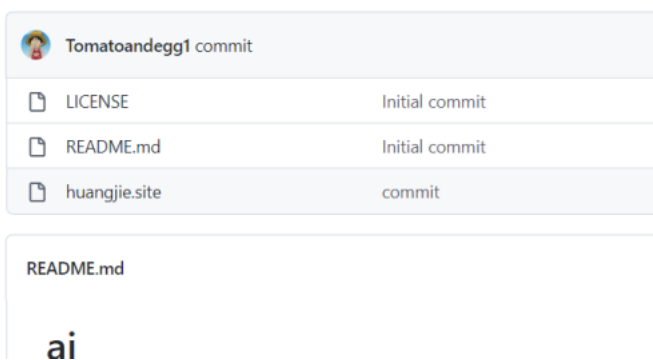
利用rebase把提交点往后面移

```
hdxj@DESKTOP-475TQVI MINGW64 ~/git/houdunren (ask)
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: ask commit

hdxj@DESKTOP-475TQVI MINGW64 ~/git/houdunren (ask)
$ gl
* 0757698 - (HEAD -> ask) ask commit (2 seconds ago) <hdxj>
* d0f3085 - (master) m2 commit (72 seconds ago) <hdxj>
* 2cf735e - master commit (2 minutes ago) <hdxj>
```

国内国外项目托管平台：

给github添加ssh密钥 然后拉取到本地 本地提交一些文件肯定是没有用的 要利用git push推送的服务器上面就有里，这里只是克隆下来



```
ayaj@DESKTOP-LNHIV61 MINGW64 ~/Desktop/aj (main)
$ git commit -m 'commit'
[main 7350e7d] commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 huangjie.site

ayaj@DESKTOP-LNHIV61 MINGW64 ~/Desktop/aj (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 299.00 K
Total 3 (delta 0), reused 0 (delta 0), pack-reuse 0
To github.com:Tomatoandegg1/aj.git
57681b2..7350e7d main -> main
```

ls 查看当前目录

touch 创建一个空白的文件夹

ll 显示详细列表 mkdir创建目录 pwd 查看当前在那个文件夹 cd 回家目录

git init: 在本地创建了一个版本仓库

git status查看代码区

subl .gitignore 版本库