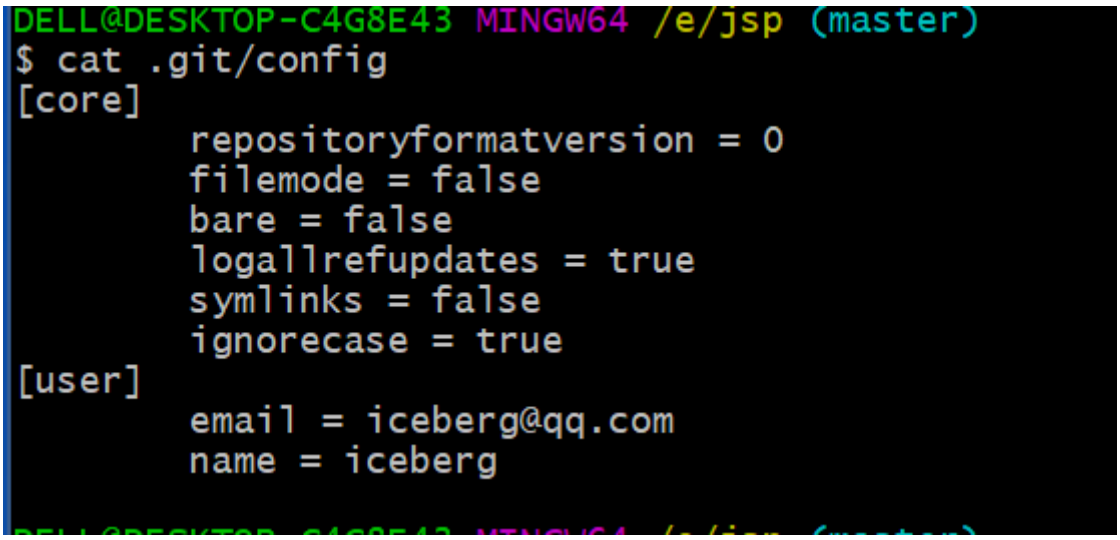
第一节、Git本地

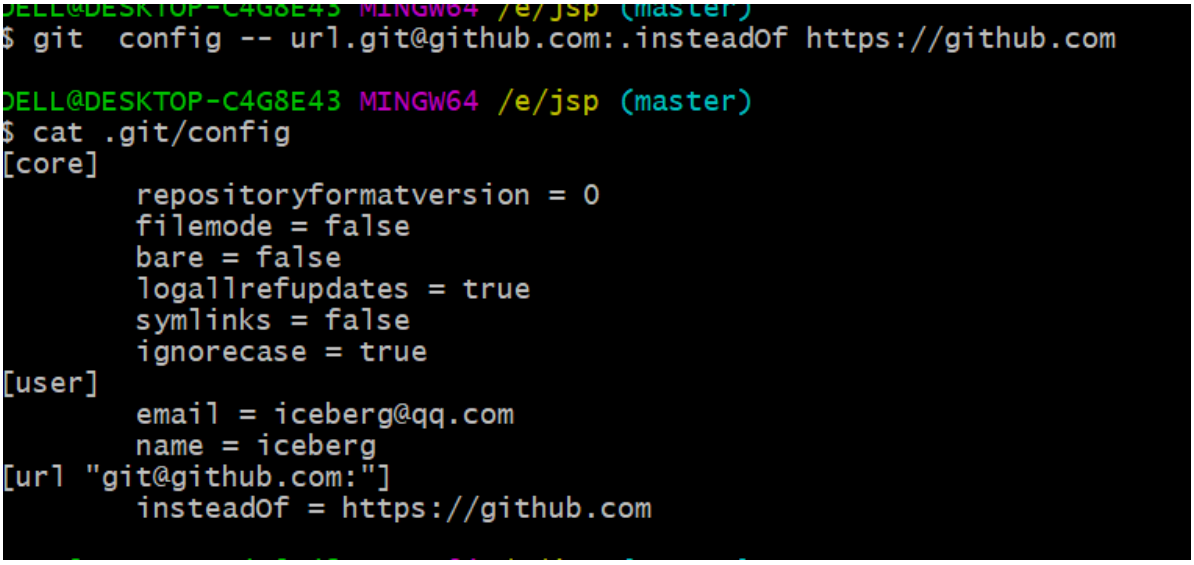
```
 git init  :git的初始化
其他参数：
--initial-branch 初始化的分支，不想要默认的master
--bare 创建一个裸仓库（纯git目录，没有工作目录）
--template 可以通过模板创建预先建好的自定义git目录
```

```
低配置会覆盖高配置
local-->global-->system
```

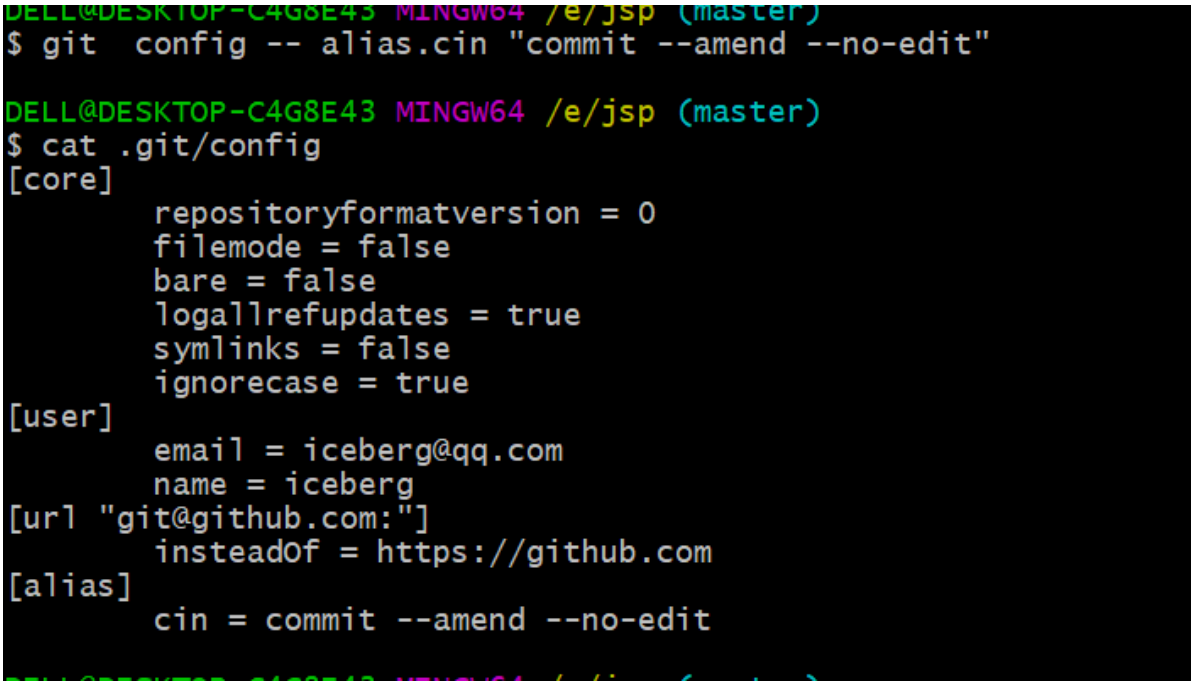
```
改变的是config文件
git config 用户名配置
-- user.name "iceberg"
-- user.email iceberg@qq.com
```




```
instead of 配置（ssh >>https）
git config -- url.git@github.com:.insteadOf https://github.com/
```



git 命令别名:
git config -- alias.cin "commit --amend --no-edit"
就相当于
cin==commit --amend --no-edit



git Remote: 本地和远程一些关联信息
查看 /帮助 remote
git remote -v /-h
添加 remote <命令> [name] [url]
git remote <add> origin_ssh git@github.com:git/git.git
git remote <add> origin_http https://github.com/git/git.git
同一个Origin设置不同的Push和Fetch URL
git remote add origin git@github.com:git/git
git remote set-url --add --push origin git@github.com:xiao/git.git

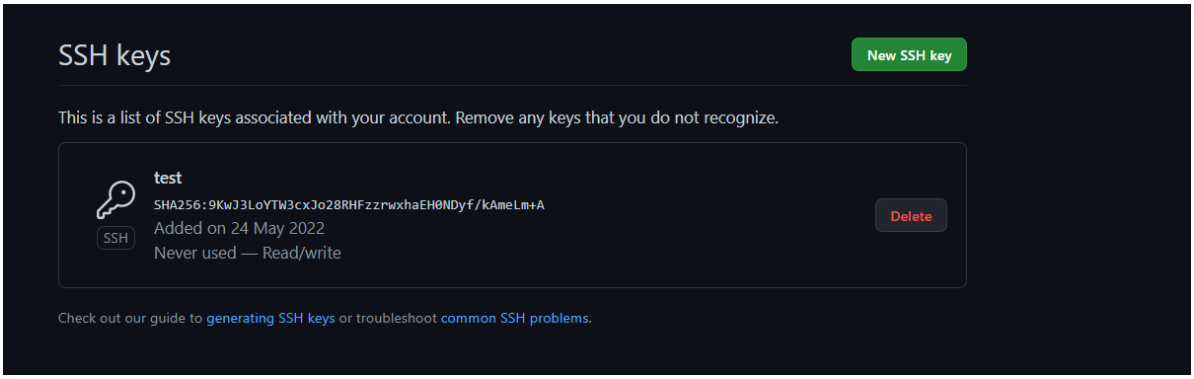
```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ git remote -v
origin  git@github.com:git/git.git (fetch)
origin  git@github.com:xiao/git.git (push)
origin_http     git@github.com:/git/git.git (fetch)
origin_http     git@github.com:/git/git.git (push)
origin_ssh      git@github.com:git/git.git (fetch)
origin_ssh      git@github.com:git/git.git (push)

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = false
        bare = false
        logallrefupdates = true
        symlinks = false
        ignorecase = true
[user]
        email = iceberg@qq.com
        name = iceberg
[url "git@github.com:"]
        insteadOf = https://github.com
[alias]
        cin = commit --amend --no-edit
[remote "origin_ssh"]
        url = git@github.com:git/git.git
        fetch = +refs/heads/*:refs/remotes/origin_ssh/*
[remote "origin_http"]
        url = https://github.com/git/git.git
        fetch = +refs/heads/*:refs/remotes/origin_http/*
[remote "origin"]
        url = git@github.com:git/git.git
        fetch = +refs/heads/*:refs/remotes/origin/*
        pushurl = git@github.com:xiao/git.git

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
```

不安全，不推荐使用

HTTP Remote
URL:https://github.com/git/git.git
免密配置：
内存：git config --global credential.helper 'cache --timeou=3600'
硬盘：git config --global credential.helper "store --fil /path/to/credential-file"
不指定目录的情况默认是 ~/.git-credentials
将密钥信息指定文件中配置：
${scheme}://${password}@github.com

SSH Remote
URL:git@github.com:git/git.git
免密配置
ssh可以通过公私钥的机制，将生成公钥存放在服务端，从而实现 免密访问。
dsa ,rsa,ecdsa,ed25519
默认使用rsa，现在优先ed25519
原因是由于一些安全问题，rsa有时候配置了公密钥，还是不能访问，是因为win改变了一些权限而导致这种问题的出现。
配置：
ssh-keygen -t ed25519 -C "your email@example.com" 密钥默认存放在 ~/.ssh/id_ed25519.pub

```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ ssh-keygen -t ed25519 -C "iceberg@qq.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/DELL/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/DELL/.ssh/id_ed25519
Your public key has been saved in /c/Users/DELL/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:9KwJ3LoYTW3cxJo28RHFzzrwxhaEHONDyf/kAmeLm+A iceberg@qq.com
The key's randomart image is:
+--[ED25519 256]--+
|          .B=.   |
|         ...B.   |
|        o +o *   |
|     . = X..+ * .|
|      + S =+ B = |
|     o = +. O o o|
|    . o o. + + . |
|     o .  E o    |
|    . .          |
+----[SHA256]-----+

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$
```



第二节、git命令原理

四种objects组装,获取当前版本代码

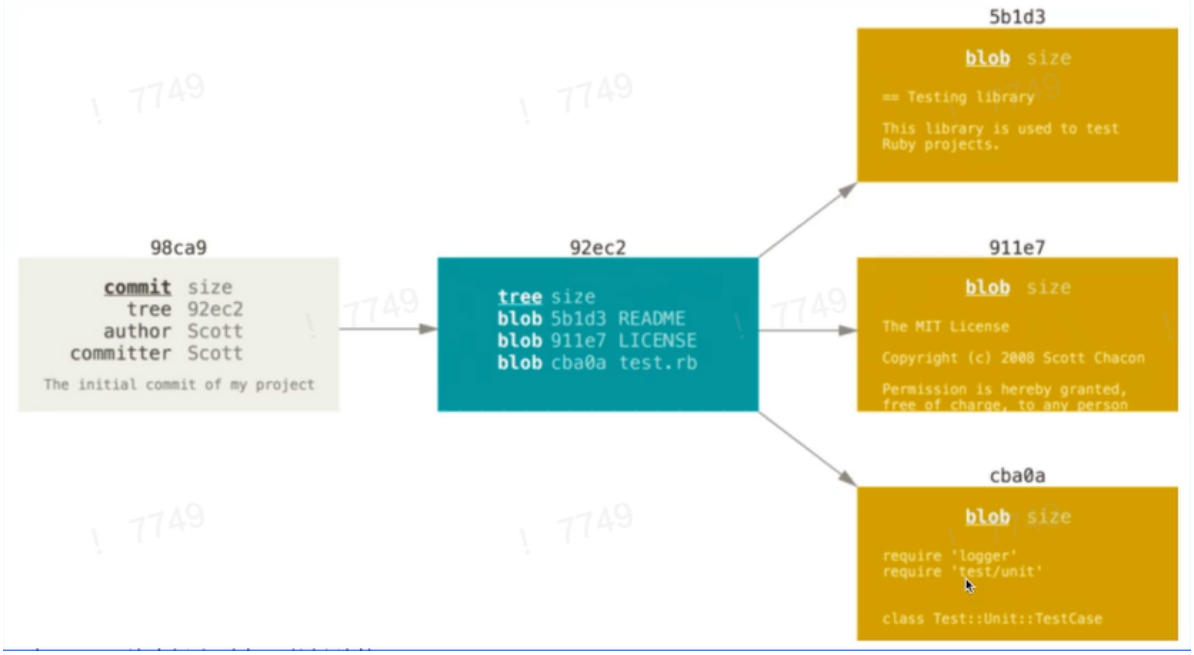
把文件添加进objects，也就是栈存区
`git add <文件>`
需要删除之前的操作
`git rm -r --cached .`
查看文件内容，虽然加密，但是还是可以通过id查看
`git cat-file -p 文件名命令`

```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ git rm --cached .
fatal: not removing '.' recursively without -r

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ git rm -r --cached .
rm 'html_css_js.md'
rm 'readme.md'
rm '前端/.idea/.gitignore'
rm '前端/.idea/artifacts/_war_exploded.xml'
rm '前端/.idea/codeStyles/Project.xml'
rm '前端/.idea/codeStyles/codeStyleConfig.xml'
rm '前端/.idea/description.html'
rm '前端/.idea/encodings.xml'
rm '前端/.idea/misc.xml'
rm '前端/.idea/modules.xml'
rm '前端/.idea/project-template.xml'
rm '前端/out/artifacts/_war_exploded/WEB-INF/classes/com/company/Main.
class'
rm '前端/out/artifacts/_war_exploded/WEB-INF/web.xml'
rm '前端/out/artifacts/_war_exploded/index.jsp'
rm '前端/out/artifacts/_war_exploded/page.jsp'
rm '前端/src/com/company/Main.java'
rm '前端/web/WEB-INF/web.xml'
rm '前端/web/classes/com/company/Main.class'
rm '前端/web/index.jsp'
rm '前端/web/page.jsp'
```

```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   readme.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        html_css_js.md
        "\345\211\215\347\253\257/"
        "\346\223\215\344\275\234/"


DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ git cat-file -p 37e7b9a4ee0fd905e02b640cb0a7368ce4e13f12
hello word git!!

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
```

然后可以提交到目录
git commit -m "add/update <文件名>"
Blob 存储文件的内容
Tree 存储文件的目录
Commit 存储提交信息，一个Commit可以对应唯一版本的代码
《由下往上找信息》

```
hello world git!!

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$ git commit -m "add readme"
[master (root-commit) 976927c] add readme
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (master)
$
```



第三节、修改refs文件

```
创建一个新分支：
git checkout -b <分支名称>
创建一个标签：
git tag <版本名称>
```

refs的内容对应的就是**Commit ID**,因此把ref当作指针，指向对应的Commit来表示当前Ref对应的版本。

不同种类的ref

refs/heads 前缀表示的是**分支**指向相同的commit，用于开发阶段，是可以不断添加Commit进行迭代

还有其他种类的ref，比如refs/tags 前缀表示的是**标签**，表示一个稳定版本，指向的commit是不会变更的,版本迭代。

```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (test)
$ cat .git/refs/heads/master
976927cb8b0de5e931cae56a9421a979a7dbd684

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (test)
$ cat .git/refs/heads/test
976927cb8b0de5e931cae56a9421a979a7dbd684

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (test)
```

Annotation Tag

Annotation Tag 附注标签，可以提供一些额外的信息

```
创建附注标签
git tag -a
```

第四节、获取历史版本代码

Commit里面存有parent commit字段通过commit的串联获取历史代码

```
因为内容变了，那么tree下的blob就会发生改变，不会影响之前的，就相当于copy从新记录了一遍。
```

```
修改历史版本
1、commit --amend
通过这个命令可以修改最近一次的commit信息，修改之后commit id会发生变化
2、rebase
git rebase -i HEAD~3 可以实现对最近三个commit的修改
1）合并commit
2）修改具体commit message
3）删除某个commit
3、filter --branch
 删除所有提交中的某个文件或全局修改邮箱地址等操作
```

新增的Object
修改后Commit后我们发现git object又出现新的，原先的并没有被删除。所有引出**悬空的Object**,顾名思义就是没有ref指向的object

```
git fsck --lost-found
```

Git GC

GC :删除一些不需要的object

Reflog :用于记录操作日志，泛指误操作后数据丢失，手动将日志设置为过期

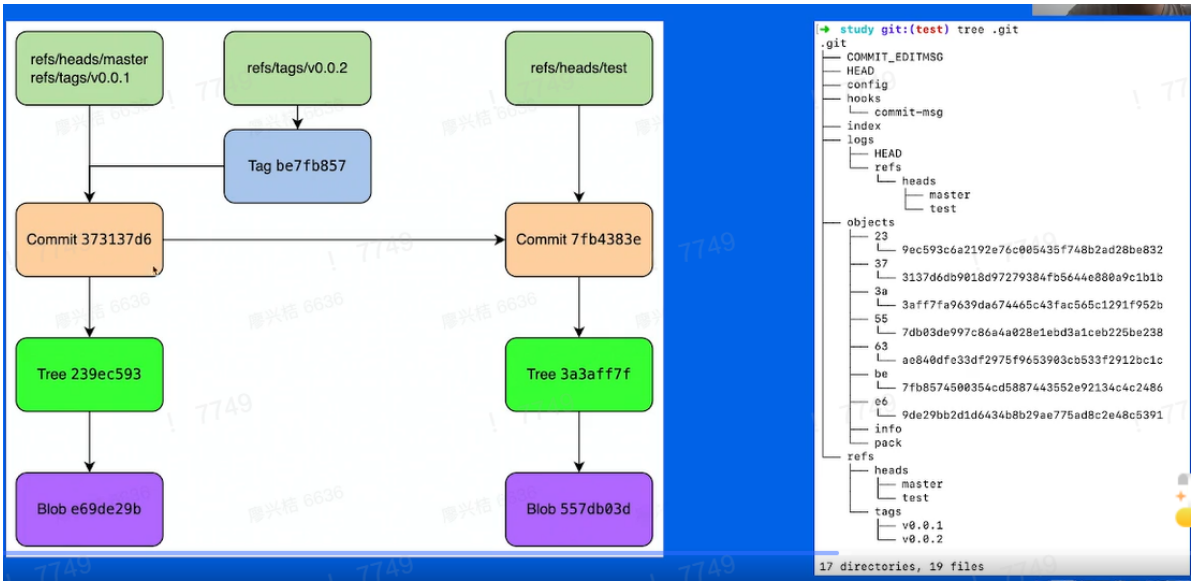
```
git reflog expire -expire=now --all
```

指定时间

git gc prune=now 指定修建多久之前的时间，默认是两周前

```
update readme

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (test)
$ git reflog expire --expire=now --all

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (test)
$ git  gc --prune=now
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp (test)
```

第五节、总结git原理图：



第六节、git 远程与多人协作

```
Clone：拉取完整创库到本地目录。
Fetch： 将远端某些分支最新代码拉取到本地，不会执行merge操作
pull：拉取远端某分支，并和本地代码进行合并，操作等同于git fetch +git merge,或者可以git
pull --rebase 完成git fetch+git rebase,这种可能会存在冲突
```

```
push:将本地同步远程
git push origin masters
```

第7节、常见问题

1、为什么我配置了git,但是还是无法拉取代码?

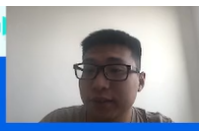
- 免密认证没有配
- instead of 没有配置，配置是ssh，但是用的是HTTP协议

2、为什么我Fetch了远端分支，但是我看到本地当前的分支历史还是没有变化?

- fetch 会把代码拉取到本地的远端分支，但是并不会合并到当前分支，所以当前分支无变化，需要自己自行rebase操作

团队合作

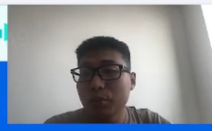
3.1 不同的工作流



类型	代表平台	特点	合入方式
集中式工作流	Gerrit / SVN	只依托于主干分支进行开发，不存在其他分支	Fast-forward
分支管理工作流	Github / Gitlab	可以定义不同特性的开发分支，上线分支，在开发分支完成开发后再通过 MR/PR 合入主干分支	自定义，Fast-Forward or Three-Way Merge 都可以

集中式

3.2 集中式工作流

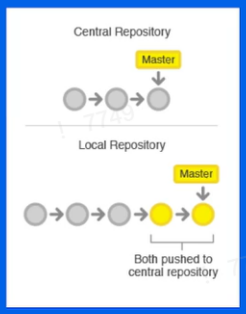


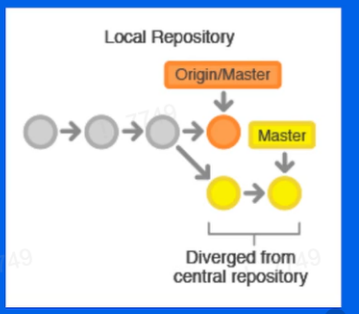
什么是集中式工作流?

只依托于 master 分支进行研发活动

工作方式

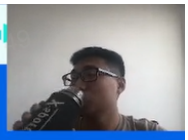
1. 获取远端 master 代码
2. 直接在 master 分支完成修改
3. 提交前拉取最新的 master 代码和本地代码进行合并 (使用 rebase)，如果有冲突需要解决冲突
4. 提交本地代码到 master





分支

3.3 分支管理工作流



分支管理工作流	特点
Git Flow	分支类型丰富，规范严格
Github Flow	只有主干分支和开发分支，规则简单
Gitlab Flow	在主干分支和开发分支之上构建环境分支，版本分支，满足不同发布 or 环境的需要

第8节、本地代码push到远程

1、创建一个master分支

git add <文件名>

git commit -m “add <文件名>”

git push origin master

```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study (test)
$ git clone https://github.com/Sunny-fairy/js.git
Cloning into 'js'...
warning: You appear to have cloned an empty repository.

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study (test)
$ cd js

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ tourch readme.md
bash: tourch: command not found

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ vim readme.md

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ git add readme.md
warning: LF will be replaced by CRLF in readme.md.
The file will have its original line endings in your working directory

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ git commit -m "add readme"
[master (root-commit) bca0a39] add readme
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:/Sunny-fairy/js.git
 * [new branch]      master -> master

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$
```

2、“更新”创建一个feature

git checkout -b feature

git add <文件名>

git commit -m "update <文件名> "

git push origin feature

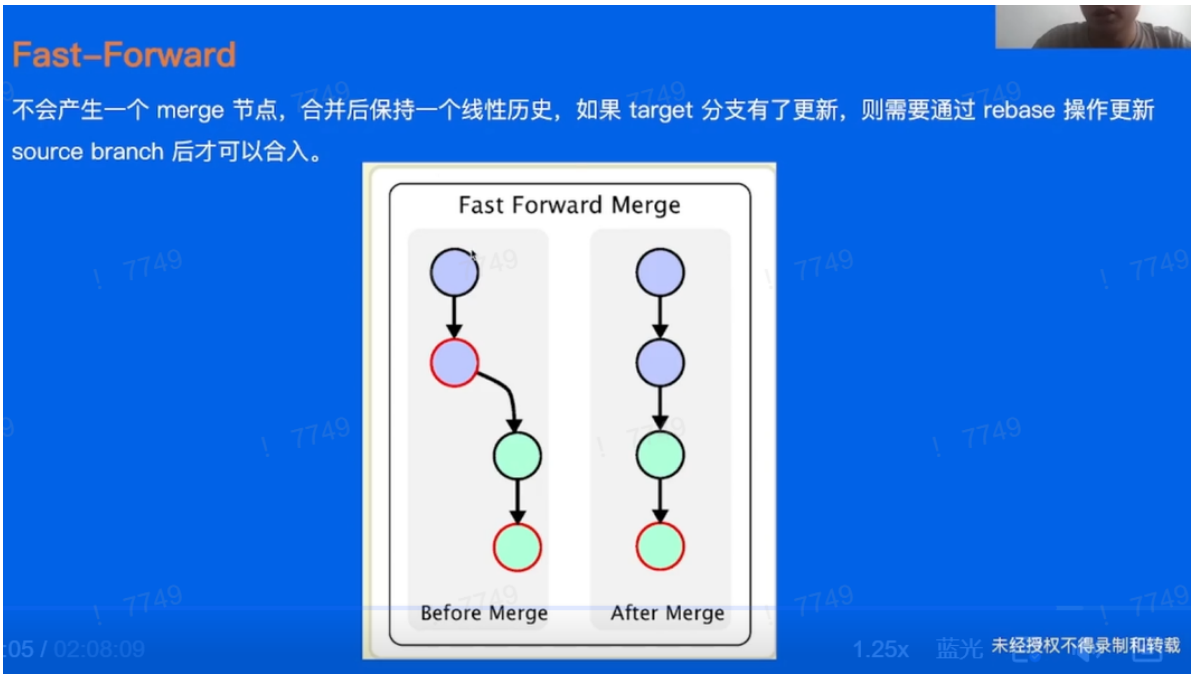
```
 * [new branch]      master -> master

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ git checkout -b feature
Switched to a new branch 'feature'

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (feature)
$ vim readme.md

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (feature)
$ git add readme.md
warning: LF will be replaced by CRLF in readme.md.
The file will have its original line endings in your working directory

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (feature)
$ git commit -m "update readme"
[feature 00e3702] update readme
 1 file changed, 1 insertion(+)

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (feature)
$ git push origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 276 bytes | 276.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/Sunny-fairy/js/pull/new/feature
remote:
To github.com:/Sunny-fairy/js.git
 * [new branch]      feature -> feature
```

第九节、代码合并

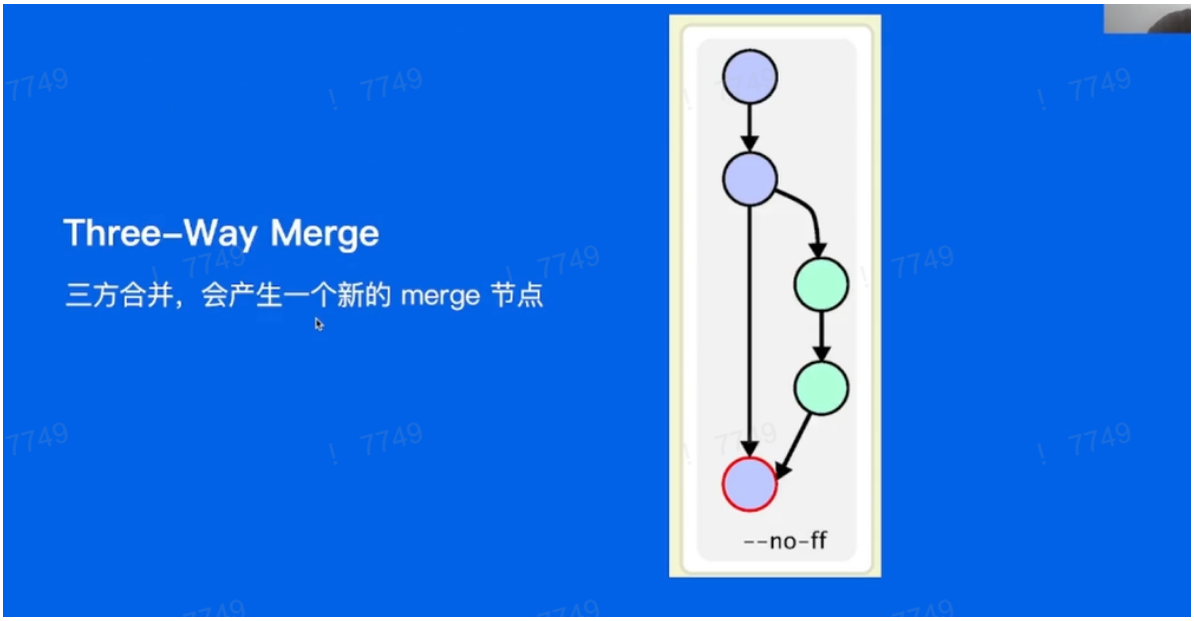
Fast-forward

```
git checkout -b <分支名>：创建一个分支
git add <文件> :添加文件进入obstal
git commit -m "分支名字" ：提交
git checkout <分支名字> ：然后切换分支
git merge <分支名字>  --ff-only :合并分支
```

```
DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (test)
$ git add readme.md
warning: LF will be replaced by CRLF in readme.md.
The file will have its original line endings in your working directory

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (test)
$ git commit -m "test"
[test 3b13c9a] test
 1 file changed, 1 insertion(+)

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (test)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (test)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ git merge test --ff-only
Updating bca0a39..3b13c9a
Fast-forward
 readme.md | 2 ++
 1 file changed, 2 insertions(+)

DELL@DESKTOP-C4G8E43 MINGW64 /e/jsp/study/js (master)
$ git log
```



Three-Way Merge




```
git merge <节点名称> --no-ff :把节点名称与当前节点合并为一个
```