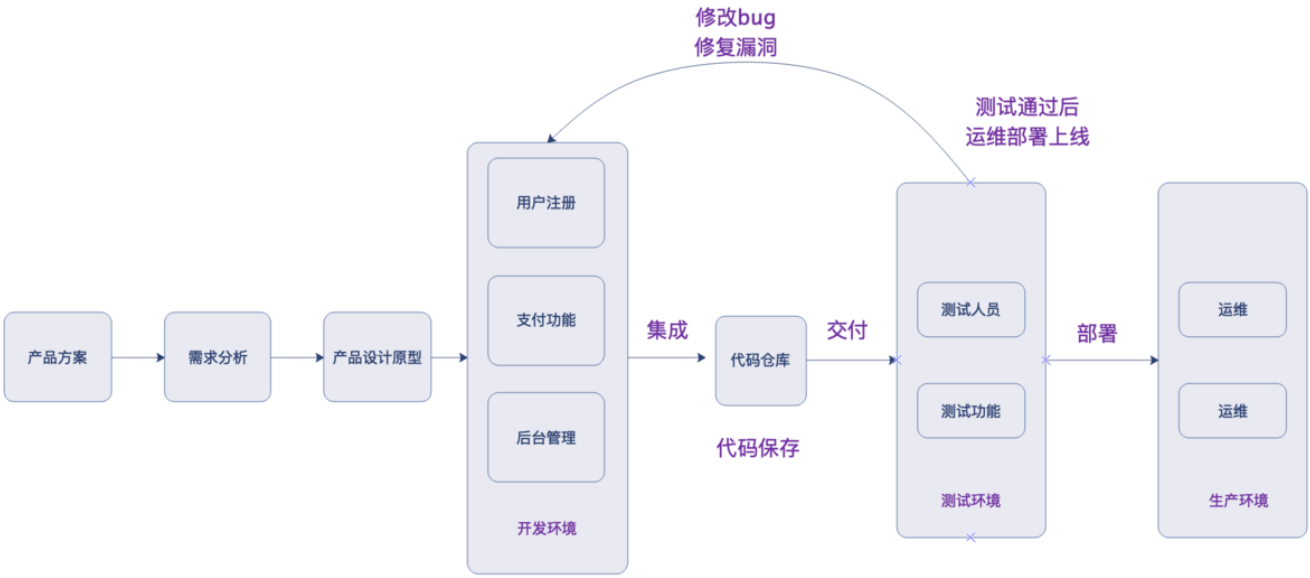
第0章 软件开发生命周期

1.开发流程

```
项目立项-->需求调研-->需求拆解-->交给不同的开发进行开发-->测试环境测试-->部署生产环境。
```



2.环境解释

```
开发环境：一般开发环境是指开发人员自己的电脑环境，不同语言不通的环境，比如python,go,php,java等。
测试环境：开发好的代码先要在测试环境跑通，测试环境的软件版本和生产环境一致，但是数据一般为测试数据，作用主要是测试开发好的代码各个组件之间是否能跑通。
预发布环境：比测试环境更贴近生产环境，数据更接近真实环境，与生产环境的域名不同，主要用于质量检测。
生产环境：真正面向用户的线上环境，一般只有运维有权限进行代码的部署和维护，其他人员一般没有权限。
```

3.手动部署的问题

```
1.上传方式不方便，scp rsync rz ftp等
2.手动部署效率低下，占用大量时间
3.如果服务器多，上线速度慢
4.手动部署容易误操作，不能保证准确率
5.出问题不好回滚，手忙脚乱
```

4.自动部署的优势

4.1 持续集成

```
1.开发的代码持续的集成到代码仓库里就是持续集成，不用等所有人都开发完毕在合并，可以多个开发人员同时工作。
2.开发将代码提交到代码仓库，由ci服务器自动将代码拉下来进行编译，测试，然后将结果返回给开发人员。
3.持续集成的目的是可以频繁的将开发的功能进行合并，提高工作效率。
```

4.2 持续交付

```
1.持续交付就是将编译开发好的代码持续的交付到测试环境进行测试。
2.在预发布环境我们可以对代码进行质量扫描和漏洞扫描，并且将测试结果返回给测试人员。
3.如果测试的代码有问题，测试人员就通知开发人员进行修复，如果没有问题则进入下一个部署环节。
```

4.3 持续部署

```
1.代码测试没有问题之后就可以进入预发布环境进行进一步测试，如果预发布环境也没有问题可以通过jenkins服务器持续的部署到生产服务器
2.如果新部署的服务发现有问题，通过jenkins服务器可以快速的回滚到正常的代码。
```

第1章 准备环境

```
主机名    IP           服务         内存
gitlab   10.0.0.200   Gitlab       2G
jenkins  10.0.0.201   Jenkins      1G
nexus    10.0.0.202   Nexus        2G
sonar    10.0.0.203   SonarQube    2G
web      10.0.0.7     Nginx        1G
```

第2章 git基本配置

0.git版本控制系统介绍

```
vcs `version control system`
版本控制系统是一一种记录一一个或若干干个文文件内容变化，以便便将来查阅特定版本内容情况的系统
记录文文件的所有历史变化  随时可恢复到任何一一个历史状态  多人人协作开发
```

1.安装命令

```
yum install git -y
```

2.查看配置

```
[root@gitlab ~]# git config
用法：git config [选项]

配置文件位置
    --global              使用全局配置文件
    --system              使用系统级配置文件
    --local               使用版本库级配置文件
    -f, --file <文件>      使用指定的配置文件
```

3.配置使用git的用户

```
git config --global user.name "zhangya"
```

4.配置使用git的邮箱

```
git config --global user.email "526195417@qq.com"
```

5.设置语法高亮

```
git config --global color.ui true
```

6.查看配置

```
[root@gitlab ~]# git config --list
user.name=zhangya
user.email=526195417@qq.com
color.ui=true

[root@gitlab ~]# cat .gitconfig
[user]
        name = zhangya
        email = 526195417@qq.com
[color]
        ui = true
```

第3章 git初始化

1.创建工作目录

```
mkdir /git_data
```

2.初始化

```
cd /git_data
git init
```

3.查看状态

```
git status
```

4.隐藏文件介绍

```
[root@gitlab /git_data]# ls .git|xargs -n 1
branches        #分支目录
config          #定义项目的特有配置
description     #描述
HEAD            #当前分支
hooks           #git钩子文件
info            #包含一个全局排除文件
objects         #存放所有数据，包含info和pack两个子文件夹
refs            #存放指向数据（分支）的提交对象的指针
index           #保存暂存区信息，在执行git init的时候，这个文件还没有
```

第4章 基本命令

1.在工作目录创建测试文件

```
[root@gitlab /git_data]# touch a b c
[root@gitlab /git_data]# git status
# 位于分支 master
#
# 初始提交
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#       a
#       b
#       c
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
```

2.提交文件到暂存区

提交a文件到暂存区

```
[root@gitlab /git_data]# git add a
[root@gitlab /git_data]# git status
# 位于分支 master
```

```
#
# 初始提交
#
# 要提交的变更:
#    (使用 "git rm --cached <file>..." 撤出暂存区)
#
#        新文件:     a
#
# 未跟踪的文件:
#    (使用 "git add <file>..." 以包含要提交的内容)
#
#        b
#        c
```

查看隐藏目录

```
[root@gitlab /git_data]# ll .git/
总用量 20
drwxr-xr-x 2 root root    6 5月  11 12:32 branches
-rw-r--r-- 1 root root   92 5月  11 12:32 config
-rw-r--r-- 1 root root   73 5月  11 12:32 description
-rw-r--r-- 1 root root   23 5月  11 12:32 HEAD
drwxr-xr-x 2 root root 4096 5月  11 12:32 hooks
-rw-r--r-- 1 root root   96 5月  11 13:01 index        # git add a 把文件提交到了暂存区
drwxr-xr-x 2 root root   20 5月  11 12:32 info
drwxr-xr-x 5 root root   37 5月  11 13:01 objects
drwxr-xr-x 4 root root   29 5月  11 12:32 refs
```

提交所有文件

```
[root@gitlab /git_data]# git add .
[root@gitlab /git_data]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更:
#    (使用 "git rm --cached <file>..." 撤出暂存区)
#
#        新文件:     a
#        新文件:     b
#        新文件:     c
```

3.撤回提交到暂存区的文件

```
[root@gitlab /git_data]# git rm --cached c
```

```
rm 'c'

[root@gitlab /git_data]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更:
#    (使用 "git rm --cached <file>..." 撤出暂存区)
#
#       新文件:     a
#       新文件:     b
#
# 未跟踪的文件:
#    (使用 "git add <file>..." 以包含要提交的内容)
#
#       c
[root@git
```

4.删除提交到暂存区的文件

方法1: 先从暂存区撤回到工作区，然后直接删除文件

```
[root@gitlab /git_data]# rm -f c
[root@gitlab /git_data]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更:
#    (使用 "git rm --cached <file>..." 撤出暂存区)
#
#       新文件:     a
#       新文件:     b
```

方法2:直接同时删除工作目录和暂存区的文件

```
[root@gitlab /git_data]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月  11 13:00 a
-rw-r--r-- 1 root root 0 5月  11 13:00 b

[root@gitlab /git_data]# git rm -f b
rm 'b'

[root@gitlab /git_data]# git status
# 位于分支 master
#
```

```
# 初始提交
#
# 要提交的变更:
#    (使用 "git rm --cached <file>..." 撤出暂存区)
#
#        新文件:     a
#

[root@gitlab /git_data]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月  11 13:00 a
```

5.提交当前暂存区的所有文件到本地仓库

```
[root@gitlab /git_data]# git commit -m "commit a"
[master（根提交） 1153f56] commit a
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a

[root@gitlab /git_data]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

6.重命名已提交到本地仓库的文件

方法1:手动修改

```
[root@gitlab /git_data]# mv a a.txt
[root@gitlab /git_data]# git status
# 位于分支 master
# 尚未暂存以备提交的变更:
#    (使用 "git add/rm <file>..." 更新要提交的内容)
#    (使用 "git checkout -- <file>..." 丢弃工作区的改动)
#
#        删除:       a
#
# 未跟踪的文件:
#    (使用 "git add <file>..." 以包含要提交的内容)
#
#        a.txt
修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）

[root@gitlab /git_data]# git rm --cached a
rm 'a'
[root@gitlab /git_data]# git status
# 位于分支 master
# 要提交的变更:
#    (使用 "git reset HEAD <file>..." 撤出暂存区)
```

```
#
#       删除:       a
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#       a.txt

[root@gitlab /git_data]# git add a.txt
[root@gitlab /git_data]# git status
# 位于分支 master
# 要提交的变更:
#   (使用 "git reset HEAD <file>..." 撤出暂存区)
#
#       重命名:     a -> a.txt

[root@gitlab /git_data]# git commit -m "commit a.txt"
[master 42ede9c] commit a.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename a => a.txt (100%)
[root@gitlab /git_data]# git status
# 位于分支 master
无文件要提交，干净的工作区

[root@gitlab /git_data]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月  11 13:00 a.txt
```

方法2:git修改

```
[root@gitlab /git_data]# git mv a.txt a
[root@gitlab /git_data]# git status
# 位于分支 master
# 要提交的变更:
#   (使用 "git reset HEAD <file>..." 撤出暂存区)
#
#       重命名:     a.txt -> a

[root@gitlab /git_data]# git commit -m "rename a.txt a"
[master 5c3ddba] rename a.txt a
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename a.txt => a (100%)
[root@gitlab /git_data]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

7.对比工作目录的文件和暂存区文件的差异

```
[root@gitlab /git_data]# echo aaaa > a
[root@gitlab /git_data]# git diff
diff --git a/a b/a
index e69de29..5d308e1 100644
--- a/a
+++ b/a
@@ -0,0 +1 @@
+aaaa
```

8.对比暂存区和本地仓库的文件内容的差异

提交a到本地暂存区，用git diff查看是相同的

```
[root@gitlab /git_data]# git add a
[root@gitlab /git_data]# git diff
```

对比暂存区和本地仓库文件的不同

```
[root@gitlab /git_data]# git diff --cached a
diff --git a/a b/a
index e69de29..5d308e1 100644
--- a/a
+++ b/a
@@ -0,0 +1 @@
+aaaa
```

将暂存区文件提交到本地仓库后再对比

```
[root@gitlab /git_data]# git commit -m "modified a"
[master 8203c87] modified a
 1 file changed, 1 insertion(+)
[root@gitlab /git_data]# git diff --cached a
```

9.查看历史的提交记录

查看详细信息

```
[root@gitlab /git_data]# git log
commit 8203c878bc30c3bd23ee977e5980232fa660ddae
Author: zhangya <526195417@qq.com>
Date:   Mon May 11 13:38:22 2020 +0800

    modified a

commit 5c3ddba7bc8de6b8575e77513ee9805021ffc5ef
Author: zhangya <526195417@qq.com>
Date:   Mon May 11 13:26:10 2020 +0800
```

```
    rename a.txt a

commit 42ede9cc10865b67e4b1e8ad58a601eadf45cd61
Author: zhangya <526195417@qq.com>
Date:   Mon May 11 13:24:35 2020 +0800

    commit a.txt

commit 1153f564c45678cc9d4c265a1b55f5ba7b610ac9
Author: zhangya <526195417@qq.com>
Date:   Mon May 11 13:16:13 2020 +0800

    commit a
```

查看简单的信息一行现实

```
[root@gitlab /git_data]# git log --oneline
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

显示当前的指针指向

```
[root@gitlab /git_data]# git log --oneline --decorate
8203c87 (HEAD, master) modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

显示具体内容的变化

```
[root@gitlab /git_data]# git log -p
```

只显示最新的内容

```
[root@gitlab /git_data]# git log -1
commit 8203c878bc30c3bd23ee977e5980232fa660ddae
Author: zhangya <526195417@qq.com>
Date:   Mon May 11 13:38:22 2020 +0800

    modified a
```

10.回滚到指定版本

提交新内容bbb到文件a

```
[root@gitlab /git_data]# echo bbb >> a
[root@gitlab /git_data]# git add a
[root@gitlab /git_data]# git commit -m "add bbb"
[master b11e0b2] add bbb
 1 file changed, 1 insertion(+)
```

提交新内容ccc到文件a

```
[root@gitlab /git_data]# echo ccc >> a
[root@gitlab /git_data]# git commit -am "add ccc"
[master 4df18d4] add ccc
 1 file changed, 1 insertion(+)
```

查看版本号

```
[root@gitlab /git_data]# git log --oneline
4df18d4 add ccc
b11e0b2 add bbb
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

回滚到指定版本 modified a

```
[root@gitlab /git_data]# git reset --hard 8203c87
HEAD 现在位于 8203c87 modified a
[root@gitlab /git_data]# cat a
aaaa
```

此时发现回滚错了，应该回退到bbb

此时查看历史会发现并没有bbb,因为回到了过去，那时候提交bbb还没发生，所有看不到记录

```
[root@gitlab /git_data]# git log --oneline
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

我们可以使用reflog来查看总的历史记录

```
[root@gitlab /git_data]# git reflog
8203c87 HEAD@{0}: reset: moving to 8203c87
4df18d4 HEAD@{1}: commit: add ccc
b11e0b2 HEAD@{2}: commit: add bbb
8203c87 HEAD@{3}: commit: modified a
5c3ddba HEAD@{4}: commit: rename a.txt a
42ede9c HEAD@{5}: commit: commit a.txt
1153f56 HEAD@{6}: commit (initial): commit a
```

然后再指定回退到bbb版本

```
[root@gitlab /git_data]# git reset --hard b11e0b2
HEAD 现在位于 b11e0b2 add bbb
[root@gitlab /git_data]# cat a
aaaa
bbb
```

第5章 分支

1.查看当前属于什么分支

```
[root@gitlab /git_data]# git branch
* master
```

2.创建分支

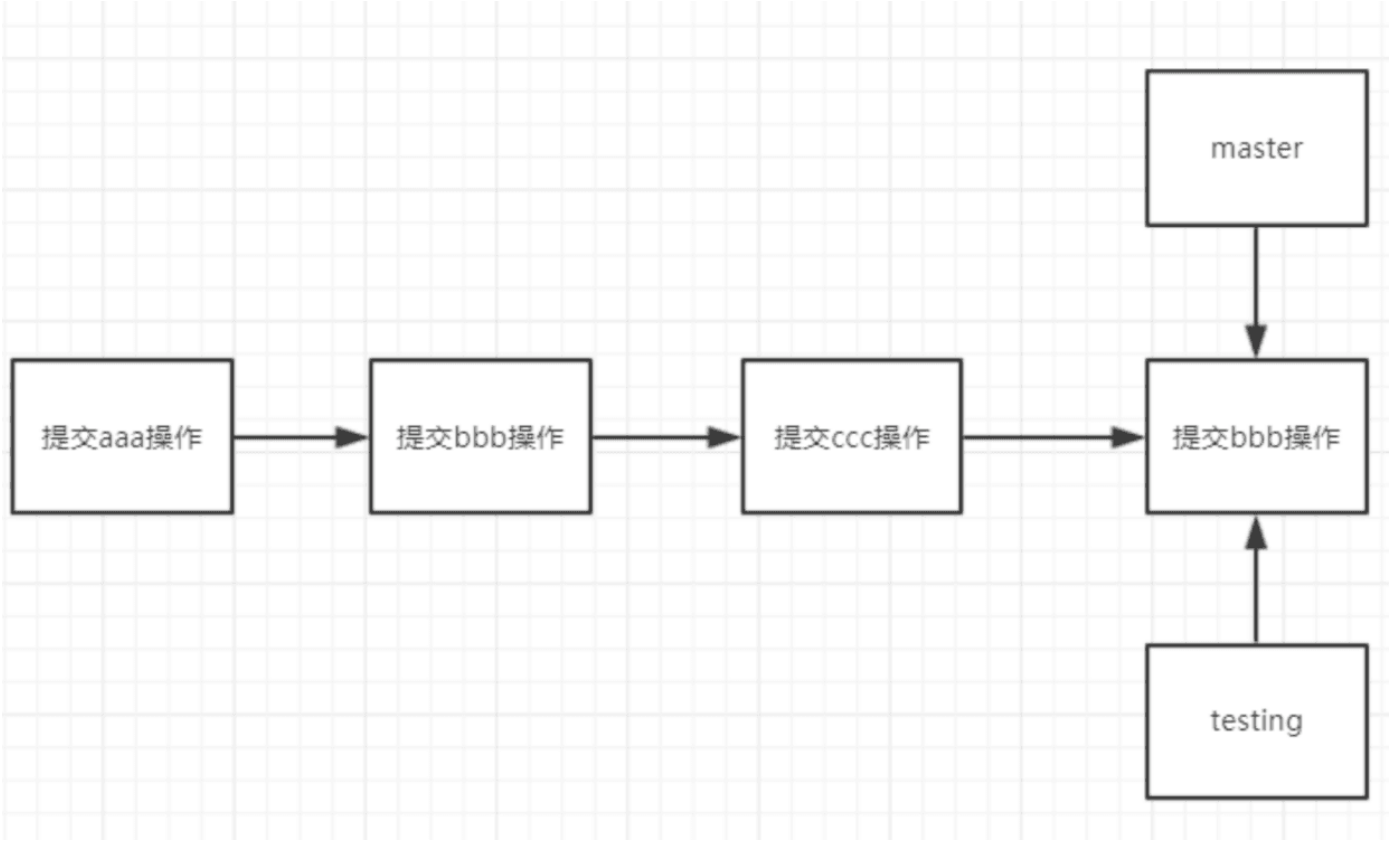
创建新分支

```
[root@gitlab /git_data]# git branch testing
```

创建新分支并切换到指定分支

```
[root@gitlab /git_data]# git checkout -b testing
切换到一个新分支 'testing'
[root@gitlab /git_data]# git branch
  master
* testing
```

图解：



4.查看分支指向

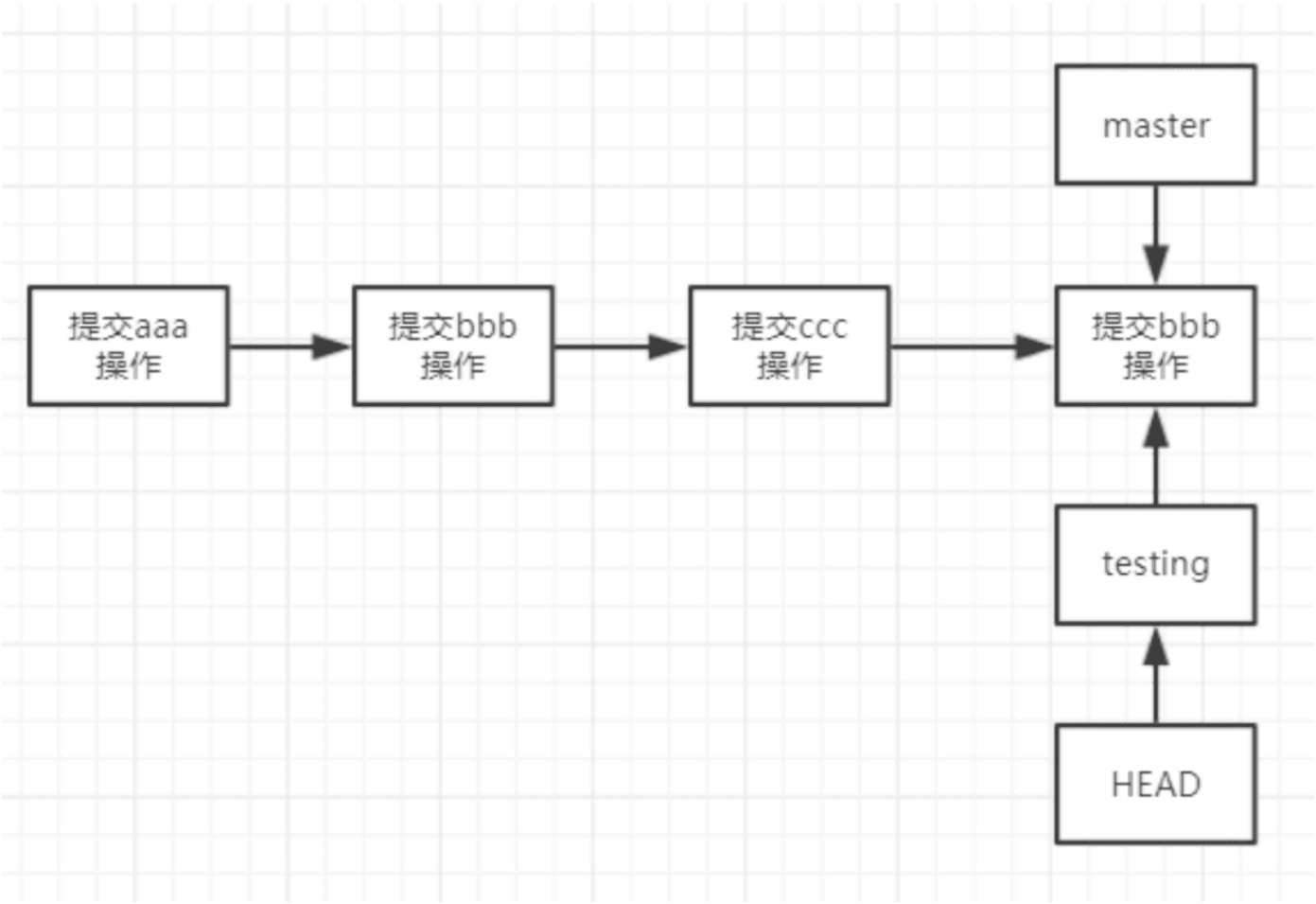
```
[root@gitlab /git_data]# git log --oneline --decorate
b11e0b2 (HEAD, testing, master) add bbb
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

5.切换到指定分支

切换到testing分支

```
[root@gitlab /git_data]# git checkout testing
切换到分支 'testing'
[root@gitlab /git_data]# git branch
  master
* testing
```

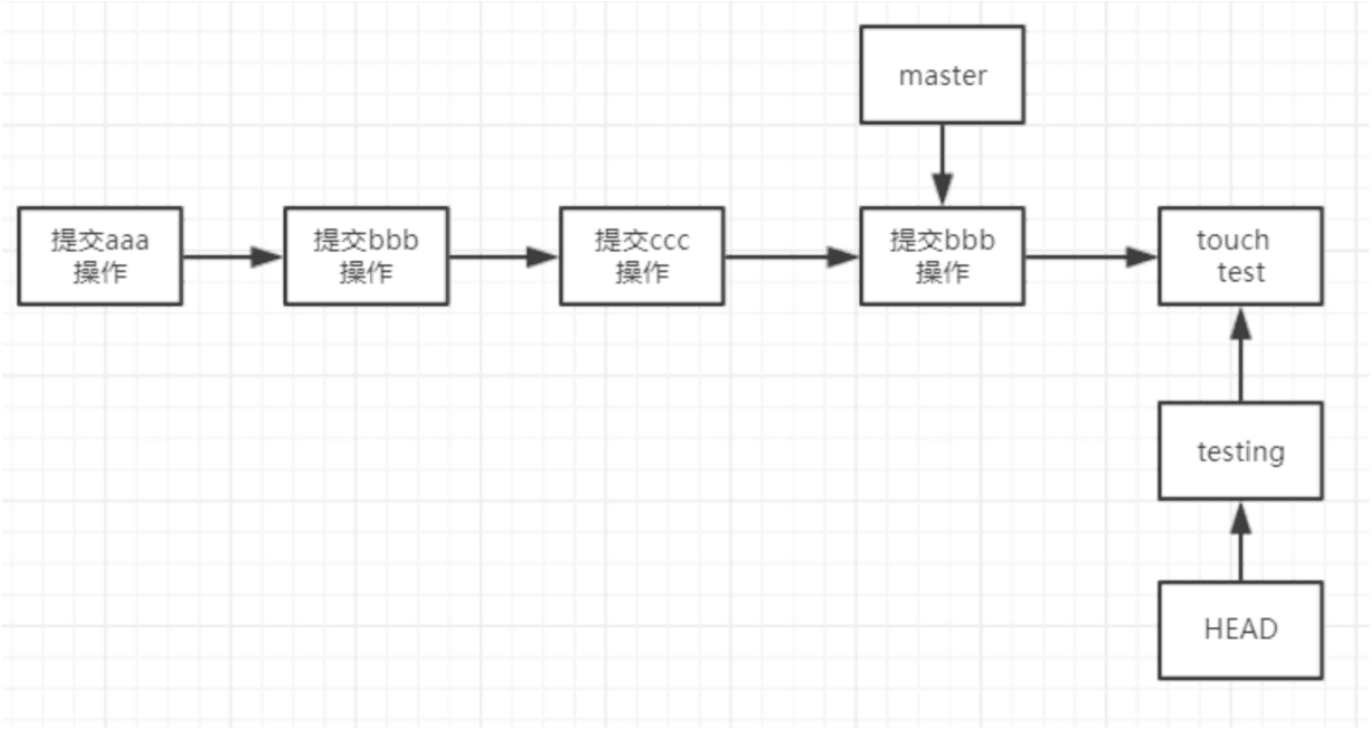
图解：



在当前分支创建文件并提交到本地仓库

```
[root@gitlab /git_data]# touch test
[root@gitlab /git_data]# git add .
[root@gitlab /git_data]# git commit -m "commit test"
[testing d50853d] commit test
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test
[root@gitlab /git_data]# ll
总用量 4
-rw-r--r-- 1 root root 9 5月  11 15:51 a
-rw-r--r-- 1 root root 0 5月  11 16:02 test
```

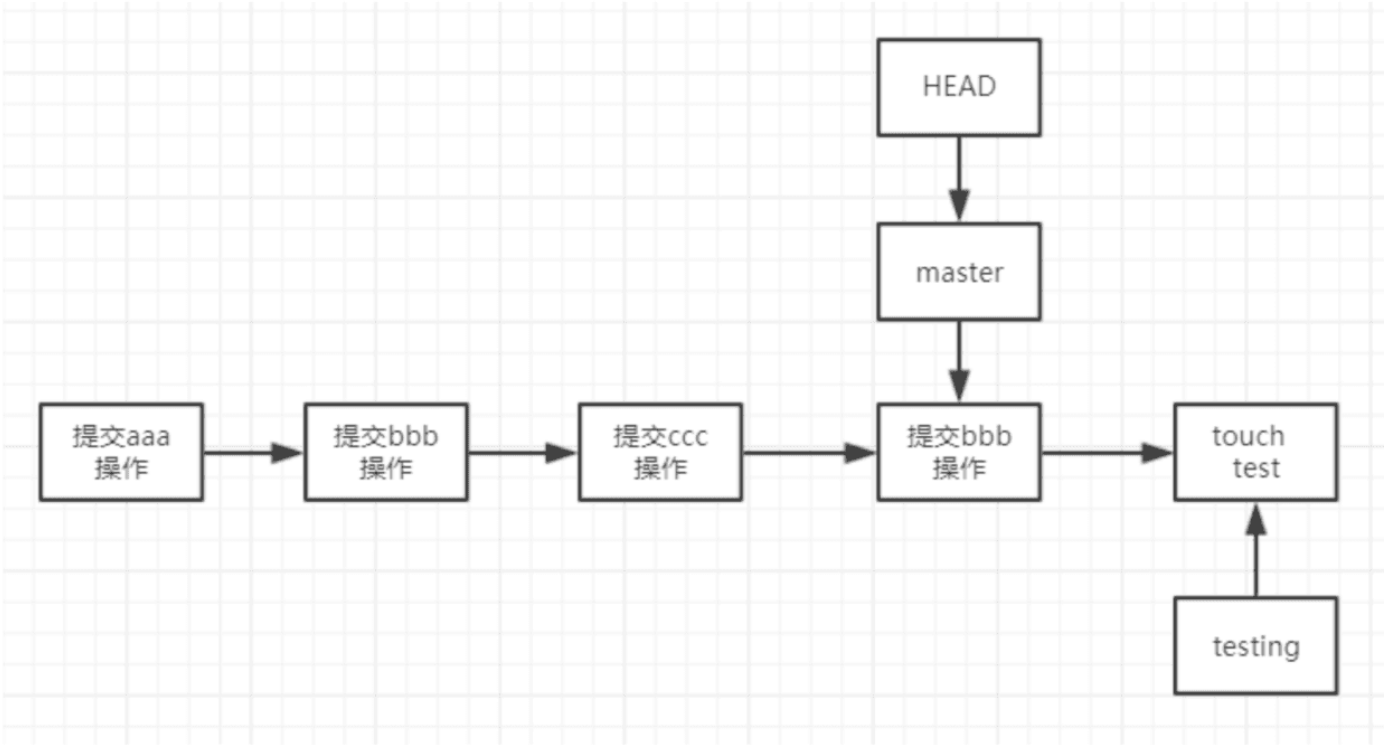
图解：



切换到master分支查看文件

```
[root@gitlab /git_data]# git checkout master
切换到分支 'master'
[root@gitlab /git_data]# git branch
* master
  testing
[root@gitlab /git_data]# ll
总用量 4
-rw-r--r-- 1 root root 9 5月  11 15:51 a
```

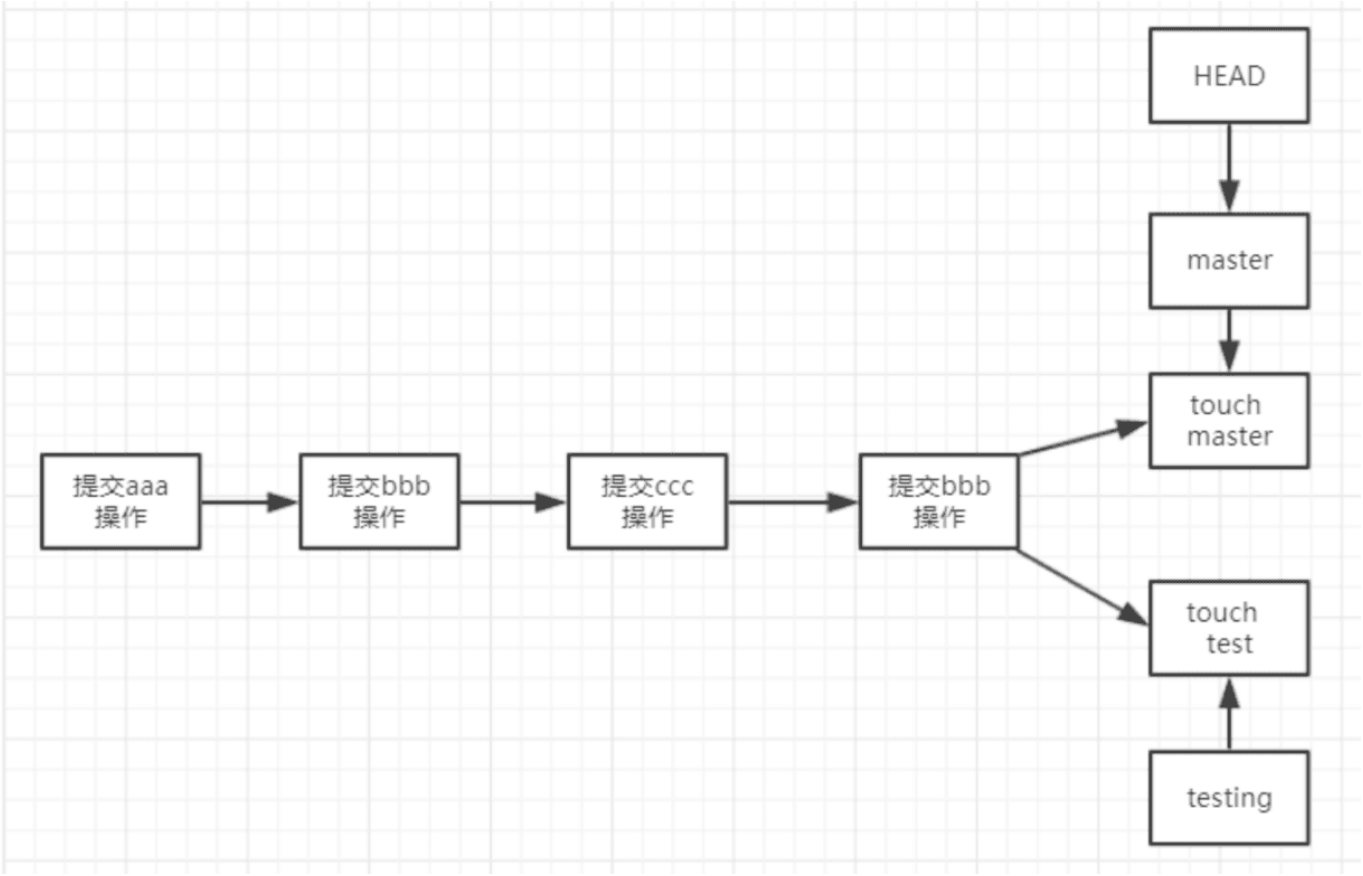
图解：



在master分支下创建文件

```
[root@gitlab /git_data]# touch master
[root@gitlab /git_data]# git add .
[root@gitlab /git_data]# git commit -m "commit master"
[master 6f9e2f0] commit master
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master
[root@gitlab /git_data]# ll
总用量 4
-rw-r--r-- 1 root root 9 5月  11 15:51 a
-rw-r--r-- 1 root root 0 5月  11 16:10 master
```

图解：



6.合并分支

将test和master分支合并

```
[root@gitlab /git_data]# git branch
* master
  testing
[root@gitlab /git_data]# git merge testing
Merge made by the 'recursive' strategy.
 test | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test
```

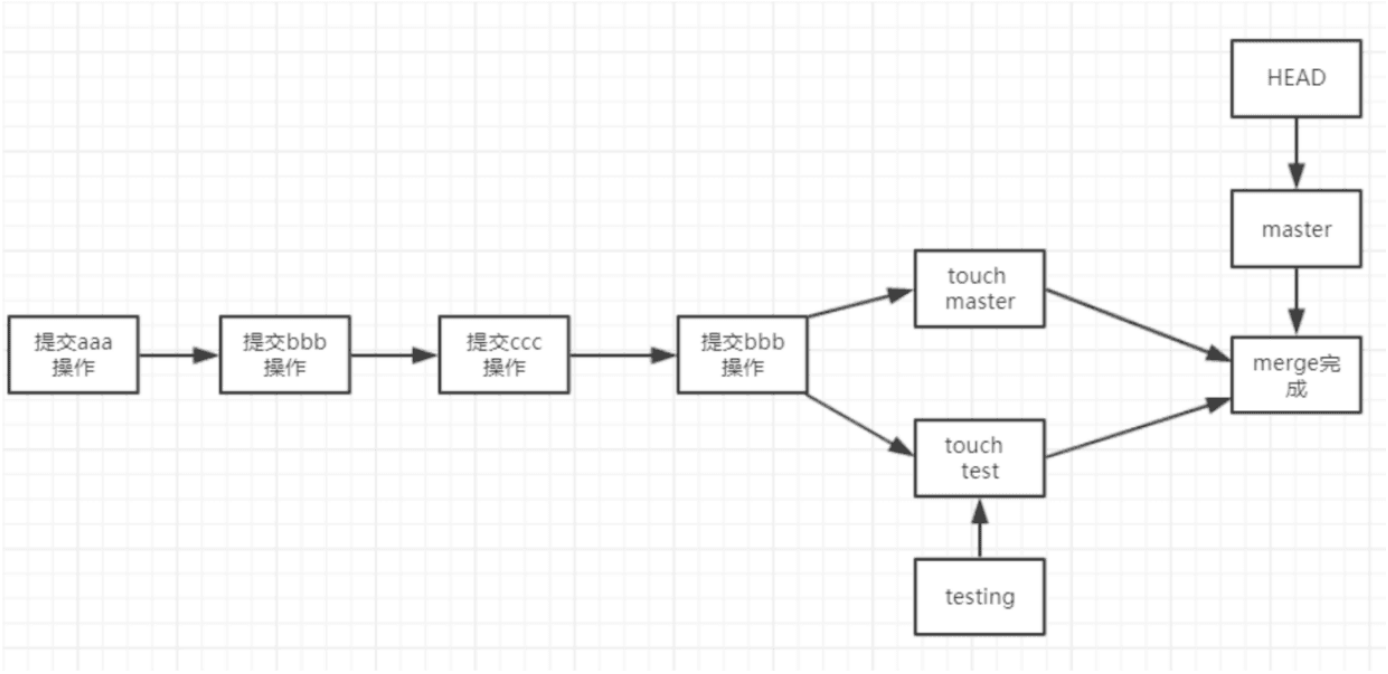
查看提交日志

```
 [root@gitlab /git_data]# git log --oneline --decorate
6f38df1 (HEAD, master) Merge branch 'testing'
6f9e2f0 commit master
d50853d (testing) commit test
b11e0b2 add bbb
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

查看文件

```
[root@gitlab /git_data]# ll
总用量 4
-rw-r--r-- 1 root root 9 5月  11 15:51 a
-rw-r--r-- 1 root root 0 5月  11 16:10 master
-rw-r--r-- 1 root root 0 5月  11 16:11 test
```

图解



7.冲突合并

在master分支下编辑a文件并提交

```
[root@gitlab /git_data]# echo "master" >> a
[root@gitlab /git_data]# git commit -am "modified a master"
[master 38fd841] modified a master
 1 file changed, 1 insertion(+)
[root@gitlab /git_data]# cat a
aaaa
bbb
master
```

切换到test分支下编辑文件并提交

```
[root@gitlab /git_data]# git checkout testing
切换到分支 'testing'
[root@gitlab /git_data]# echo "testing" >> a
[root@gitlab /git_data]# git commit -am "modified a on testing branch"
[testing 71c50c8] modified a on testing branch
 1 file changed, 1 insertion(+)
 [root@gitlab /git_data]# cat a
aaaa
bbb
testing
```

切换到master分支并合并test分支，此时两个分支的文件内容是有冲突的

```
[root@gitlab /git_data]# git checkout master
切换到分支 'master'
[root@gitlab /git_data]# git merge testing
自动合并 a
冲突（内容）：合并冲突于 a
自动合并失败，修正冲突然后提交修正的结果。
```

此时冲突内容会自动写到文件里

```
[root@gitlab /git_data]# cat a
aaaa
bbb
<<<<<<< HEAD
master
=======
testing
>>>>>>> testing
```

要想解决冲突，我们需要手动修改文件，保留最终的文件,然后重新提交

```
[root@gitlab /git_data]# vim a
[root@gitlab /git_data]# cat a
aaaa
bbb
master
[root@gitlab /git_data]# git commit -am "merge testing to master"
[master 921d88e] merge testing to master
```

查看提交日志

```
[root@gitlab /git_data]# git log --oneline --decorate
921d88e (HEAD, master) merge testing to master
71c50c8 (testing) modified a on testing branch
38fd841 modified a master
6f38df1 Merge branch 'testing'
6f9e2f0 commit master
d50853d commit test
b11e0b2 add bbb
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
```

8.删除指定分支

```
[root@gitlab /git_data]# git branch -d testing
已删除分支 testing（曾为 71c50c8）。
[root@gitlab /git_data]# git branch
* master
```

第6章 git标签使用

1.给当前版本创建标签

```
[root@gitlab /git_data]# git tag v1.0 -m "aaa bbb master testing version v1.0"
```

2.给指定版本打标签

```
[root@gitlab /git_data]# git log --oneline
921d88e merge testing to master
71c50c8 modified a on testing branch
38fd841 modified a master
6f38df1 Merge branch 'testing'
6f9e2f0 commit master
d50853d commit test
b11e0b2 add bbb
8203c87 modified a
5c3ddba rename a.txt a
42ede9c commit a.txt
1153f56 commit a
[root@gitlab /git_data]# git tag -a v2.0 b11e0b2 -m "add bbb version v2.0"
```

3.查看标签

```
[root@gitlab /git_data]# git tag
v1.0
v2.0
```

4.回滚到指定标签

首先查看当前版本文件

```
[root@gitlab /git_data]# ll
总用量 4
-rw-r--r-- 1 root root 16 5月  11 16:36 a
-rw-r--r-- 1 root root  0 5月  11 16:33 master
-rw-r--r-- 1 root root  0 5月  11 16:11 test
```

回滚到指定版本

```
[root@gitlab /git_data]# git reset --hard v2.0
HEAD 现在位于 b11e0b2 add bbb
```

再次查看文件

```
[root@gitlab /git_data]# ll
总用量 4
-rw-r--r-- 1 root root 9 5月  11 16:52 a
```

第7章 将代码提交到gitee码云

1.需求

```
讲自己写过的shell,ansible提交到gitee上
```

第8章 总结

1.命令总结

```
git init      #初始化一个目录为git版本库
git add .     #将没有被管理的文件，加入git管理，添加到暂存区
git commit -m "描述"  #将暂存区的文件提交到版本库中，进行版本的管理
git log       #查看提交的历史记录
git reflog    #查看git提交的所有历史记录
git status    #查看当前的文件管理状态（已提交|未提交）
gi tag -a 版本号 -m "版本说明"  #给代码打标签
git reset --hard  commitID  #回退到指定的提交版本记录
git remote add origin 远程仓库地址  #添加远程仓库地址
```

```
git push origin 分支名称     #提交代码
git pull origin 分支名称     #拉取代码
等价于
git fetch origin dev #拉取代码
git merge origin/dev #合并代码
```

2.图解