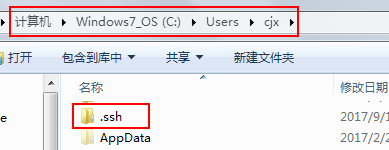
# 一、个人基本操作

1. PC上安装git客户端
2. gitLab上注册账号

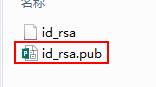
**GitLab 是一个用于仓库管理系统的开源项目**

目的：通过git管理github托管项目代码

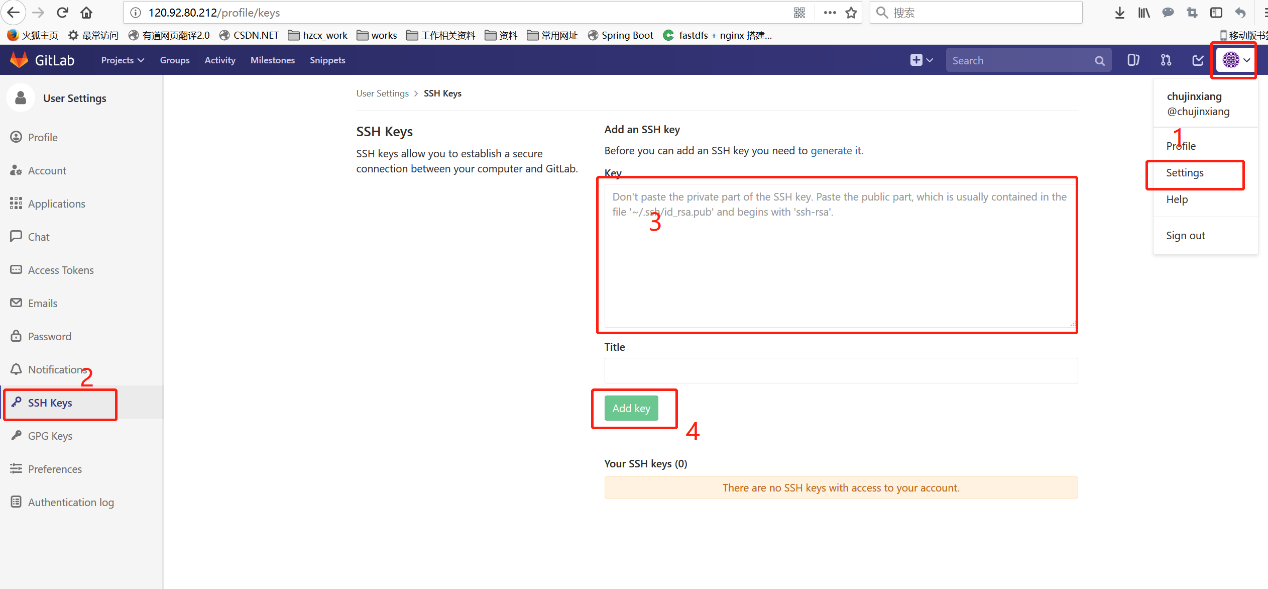
1. 添加sshkey（该方式针对ssh协议操作git）
2. 桌面上鼠标右键点击”git bash here”
3. 输入命令：ssh-keygen -t rsa -C ["chujinxiang@126.com" 然后不断回车。](mailto:\"chujinxiang@126.com\"然后不断回车)
4. 在user目录下会生成.ssh的目录



点击进去后文本打开id\_rsa.pub文件（公钥）

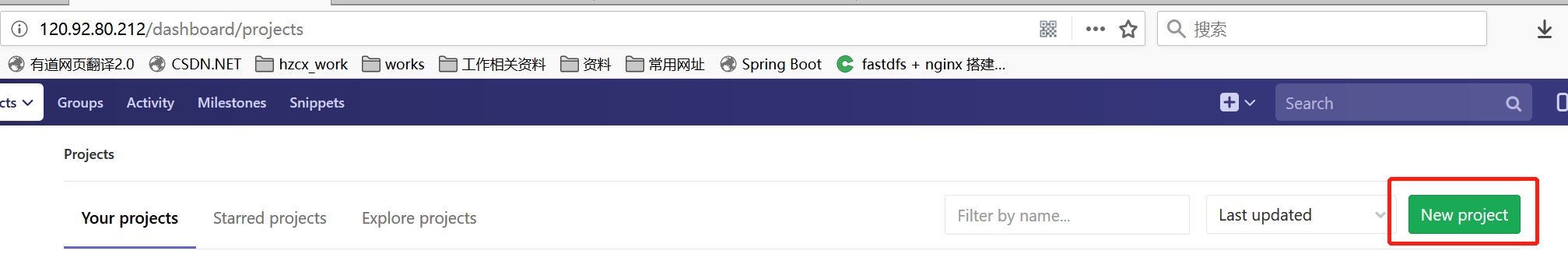


复制里面的内容，打开gitLab

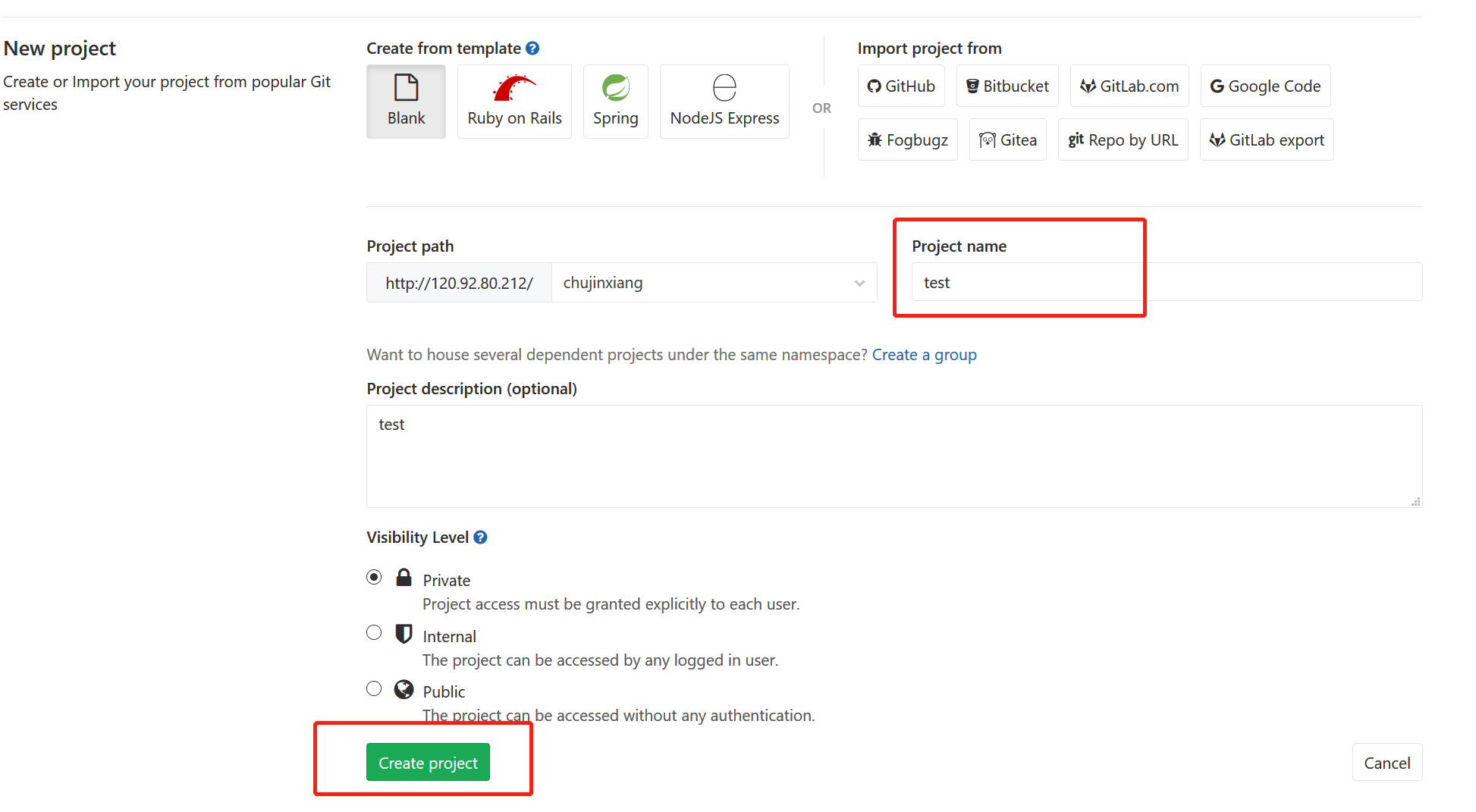


选中1选择settings，点击2，将复制的内容粘贴在3的位置后点击4。

1. gitLab中新增项目：



添加自己的项目名称



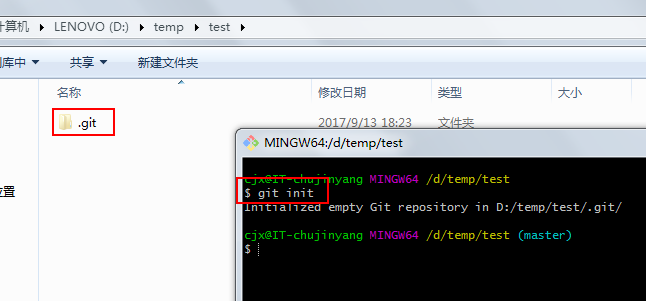
创建项目后：



可以看到git生成的项目地址，需要拷贝此项目地址，通常使用ssh的方式进行git操作。

1. 在本地磁盘创建一个目录，在目录中右键”git bash here”,输入初始化仓库的命令：

**git init**

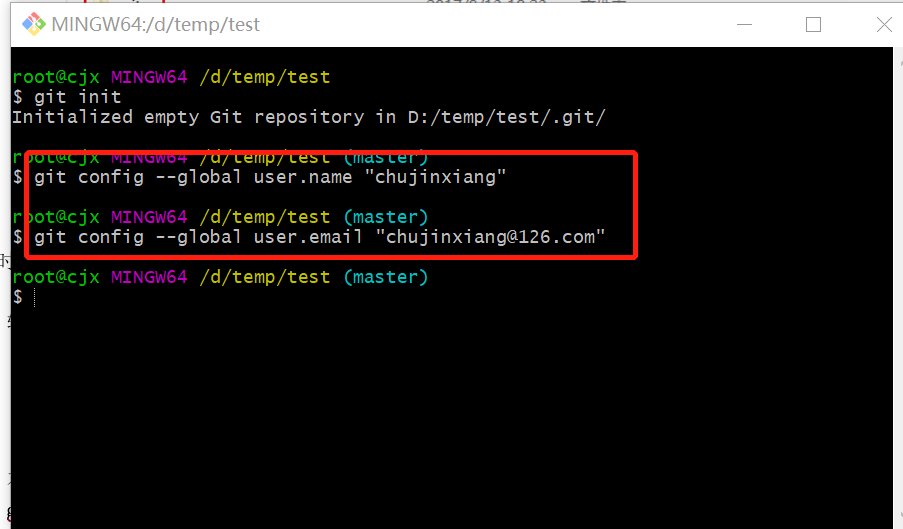


此时在隐藏的文件中可以看到.git的目录。

1. 输入两条命令设置姓名和邮箱：

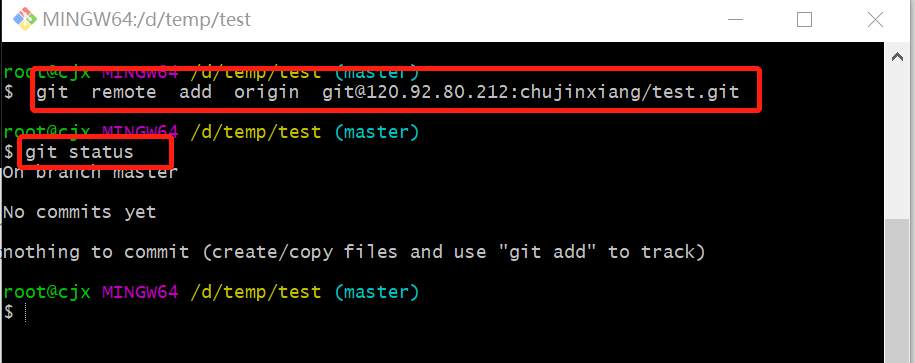
**git config --global user.name "chujinxiang"**

**git config --global user.email ["chujinxiang@126.com"](mailto:%22chujinxiang@126.com%22)**



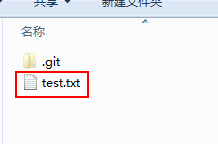
7、本地关联远程仓库（公司内只能用http的地址）：

git remote add origin git@120.92.80.212:chujinxiang/test.git

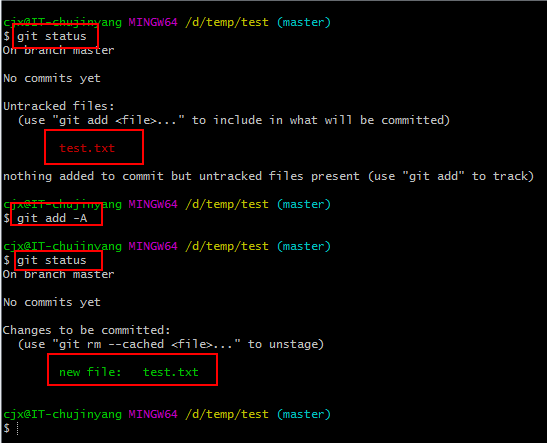


git status 可以查看当前状态

1. 拉取代码命令 git pull origin master
2. 本地新建立一个文本：



在命令窗口中可查看：

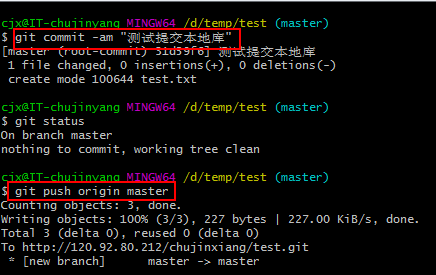


红色的字体是新增的，git add -A 增加文件到库

增加完了以后查看状态会变成绿色

1. 提交到本地仓库： git commit -am “第一次提交测试”
2. 推送到远程分支（第一次推送的时候会弹出一个框，要输入git的用户名和密码）：

git push origin master



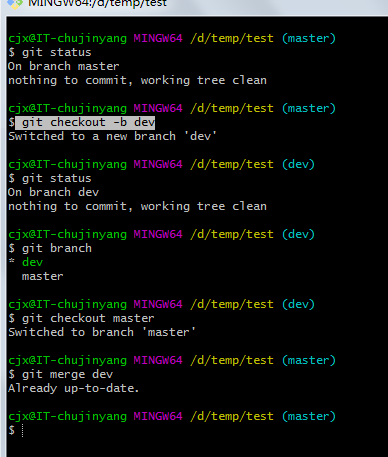
1. 此时代码已经提交到服务器了。

以master分支为基础新建并切换到dev分支： git checkout -b dev

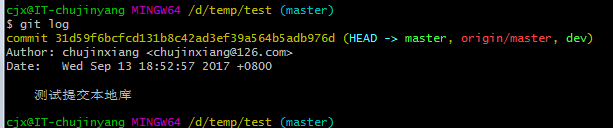
查看已有的分支（查看当前所在分支）： git branch

切换到master分支(已有)：git checkout master

当前分支合并dev分支（把dev的代码合并到当前分支上）：git merge dev



1. 查看日志 git log



# 二、团队分支操作

git服务器上分三个分支，（以刚才创建的test项目为例）：

**master -- 生产**

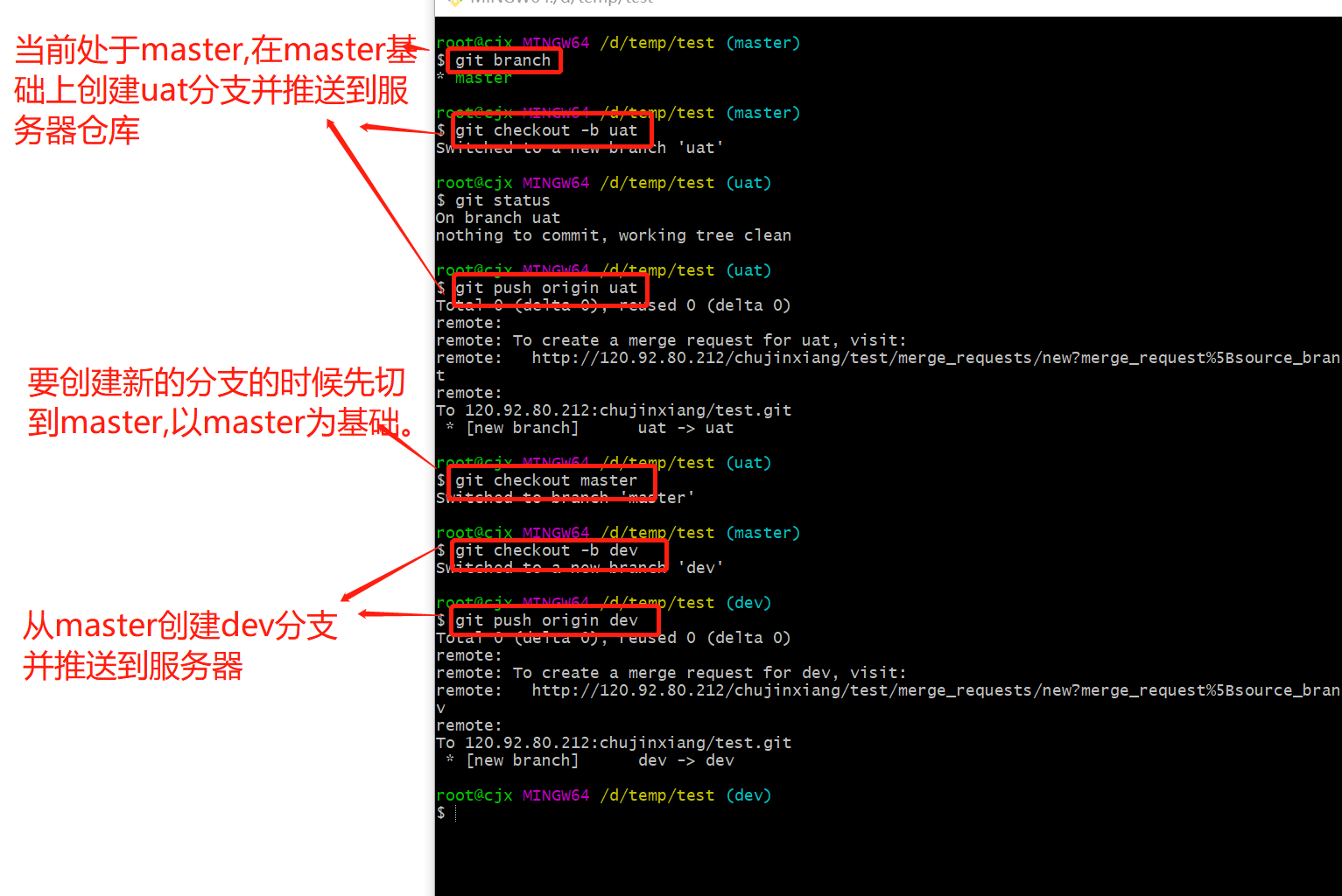
**uat -- 测试**

**dev -- 开发**

**项目初始化：**

分别初始化三个分支（此时三个分支的代码都是一样的）

注意，所有分支都从**master创建**



此时项目已经完成了各个分支的初始化。

git合并分支三条原则：

1. 创建新分支永远只从master进行创建
2. master永远不合uat、不合dev；uat永远不合dev; 服务器仓库上的分支之间没有交集
3. 合并时候千万不要用本地分支去合并dev\uat\master.

比如：

master merge function\_1 -- 正确

master merge function\_2 -- 正确

dev merge function\_1 – 正确

uat merge function\_1 – 正确

master merge dev – 错误

master merge uat – 错误

uat merge dev – 错误

function\_1 merge dev – 错误

function\_2 merge uat – 错误

function\_1 merge master – 错误

dev merge master – 错误

**应用场景一：**

A童鞋开发两个功能function\_one和function\_two,

B童鞋开发功能function\_three,

经过测试后最终要把A的function\_two和B的function\_three上生产。

**A的操作步骤如下：**

**1、先把自己本地的切换到master分支：git checkout master**

**2、更新本地master分支的代码：git pull origin master**

**3、基于master分支创建function\_one分支：git checkout –b function\_one**

**4、开发完成后需要提交dev分支：**

**4.1先提交到本地仓库：git commit –am “function\_one finish”**

**4.2切换到dev分支（如果本地没有dev分支需要创建）git checkout (-b) dev**

**4.3 更新本地dev分支的代码：git pull origin dev**

**4.4 在dev的基础上合并开发的function\_one：git merge function\_one**

**4.5 如果合并后有冲突则解决冲突，提交到本地库：git commit –am “function one”**

**4.6 推送代码到服务器仓库的dev分支：git push origin dev**

**5、把dev打包发送到服务进行测试并准备开发function\_two**

**6、先把本地切换到master分支：git checkout master**

**7、更新master分支代码：git pull origin master**

**8、创建新分支function\_two：git checkout –b function\_two**

**9、开发完成后需要提交dev分支：**

**9.1先提交到本地仓库：git commit –am “function\_two finish”**

**9.2切换到dev分支（如果本地没有dev分支需要创建）git checkout (-b) dev**

**9.3 更新本地dev分支的代码：git pull origin dev**

**9.4 在dev的基础上合并开发的function\_two：git merge function\_two**

**9.5 如果合并后有冲突则解决冲突，提交到本地库：git commit –am “function two”**

**9.6 推送代码到服务器仓库的dev分支：git push origin dev**

**B的操作步骤如下：**

**1、先把自己本地的切换到master分支：git checkout master**

**2、更新本地master分支的代码：git pull origin master**

**3、基于master分支创建function\_three分支：git checkout –b function\_three**

**4、开发完成后需要提交dev分支：**

**4.1先提交到本地仓库：git commit –am “function\_three finish”**

**4.2切换到dev分支（如果本地没有dev分支需要创建）git checkout (-b) dev**

**4.3 更新本地dev分支的代码：git pull origin dev**

**4.4 在dev的基础上合并开发的function\_ three：git merge function\_three**

**4.5 如果合并后有冲突则解决冲突，提交到本地库：git commit –am “function three”**

**4.6 推送代码到服务器仓库的dev分支：git push origin dev**

**5、把dev打包发送到服务进行测试**

**此时A、B两位同学开发的三个功能function\_one\function\_two\function\_three代码都在dev分支上，此时function\_one因为测试未通过或者是需求变更的原因不上了，只上function\_two和function\_three，需要把这两个功能提交到uat环境。**

**A的操作：**

1. **先切换到uat分支：git checkout uat**
2. **更新uat代码：git pull origin uat**
3. **和并自己要上的function\_two分支：git merge function\_two**
4. **如果有冲突解决冲突，然后提交到本地库：git commit –am “function two uat”**
5. **推送到uat： git push origin uat**

**B的操作：（同A,只是合并的分支不同）**

* 1. **先切换到uat分支：git checkout uat**
  2. **更新uat代码：git pull origin uat**
  3. **和并自己要上的function\_two分支：git merge function\_three**
  4. **如果有冲突解决冲突，然后提交到本地库：git commit –am “function three uat”**
  5. **推送到uat： git push origin uat**

**uat验证也通过后需要上生产，上生产的master的分支操作同uat：**

**A的操作：**

**1、先切换到master分支：git checkout master**

**2、更新master代码：git pull origin master**

**3、合并自己要上的function\_two分支：git merge function\_two**

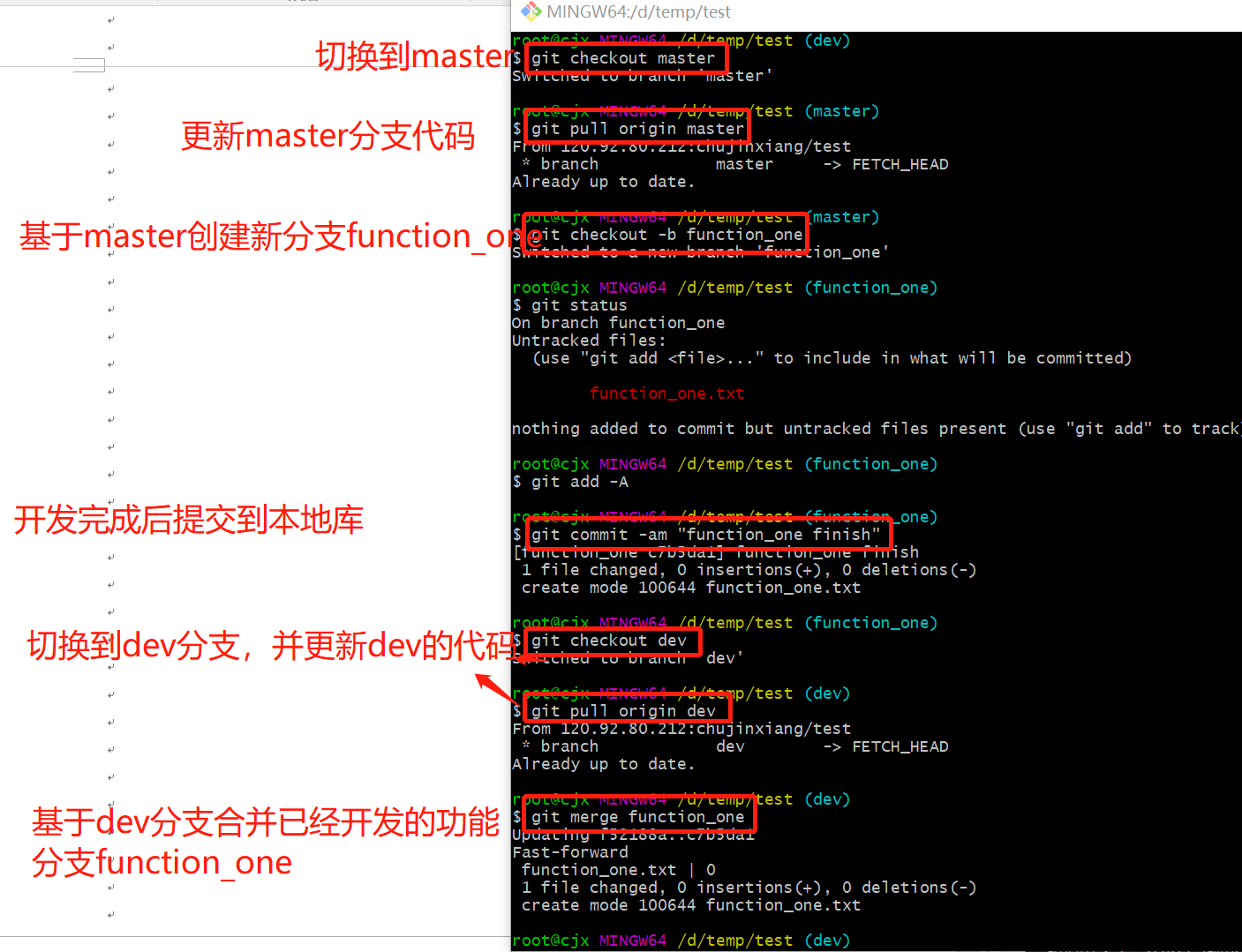
**4、如果有冲突解决冲突，然后提交到本地库：git commit –am “function two master”**

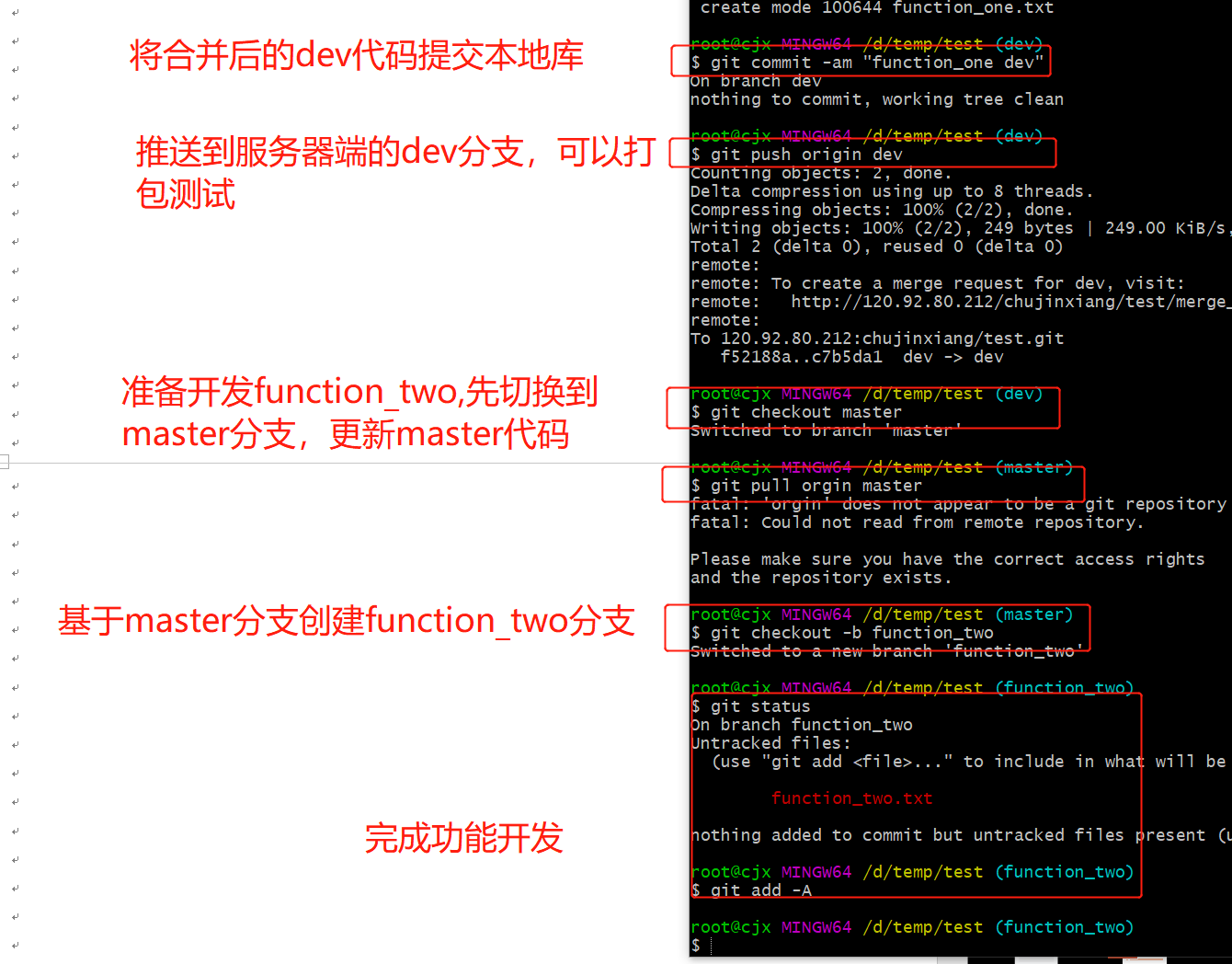
**5、推送到master： git push origin master**

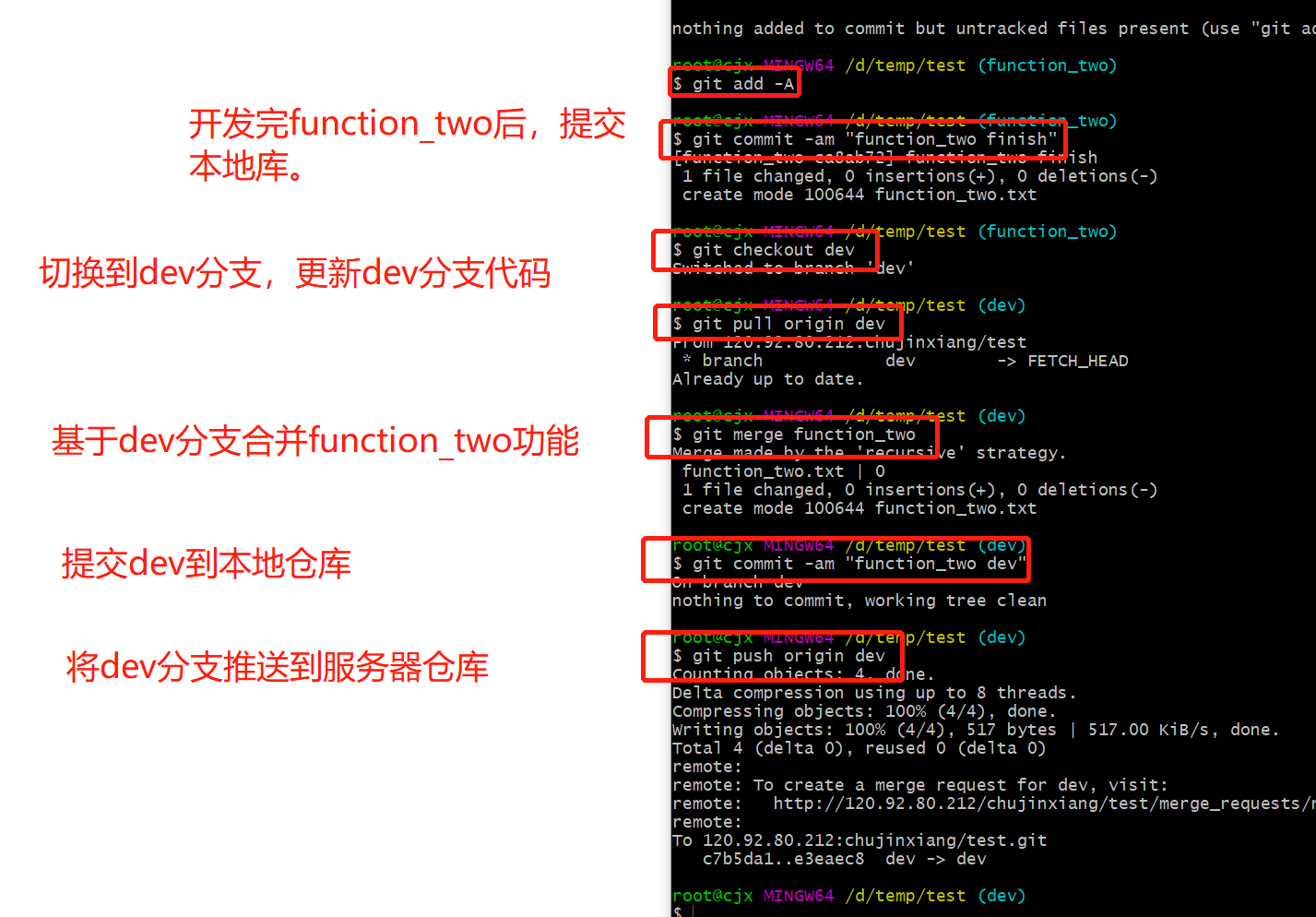
**B的操作：（同A,只是合并的分支不同）**

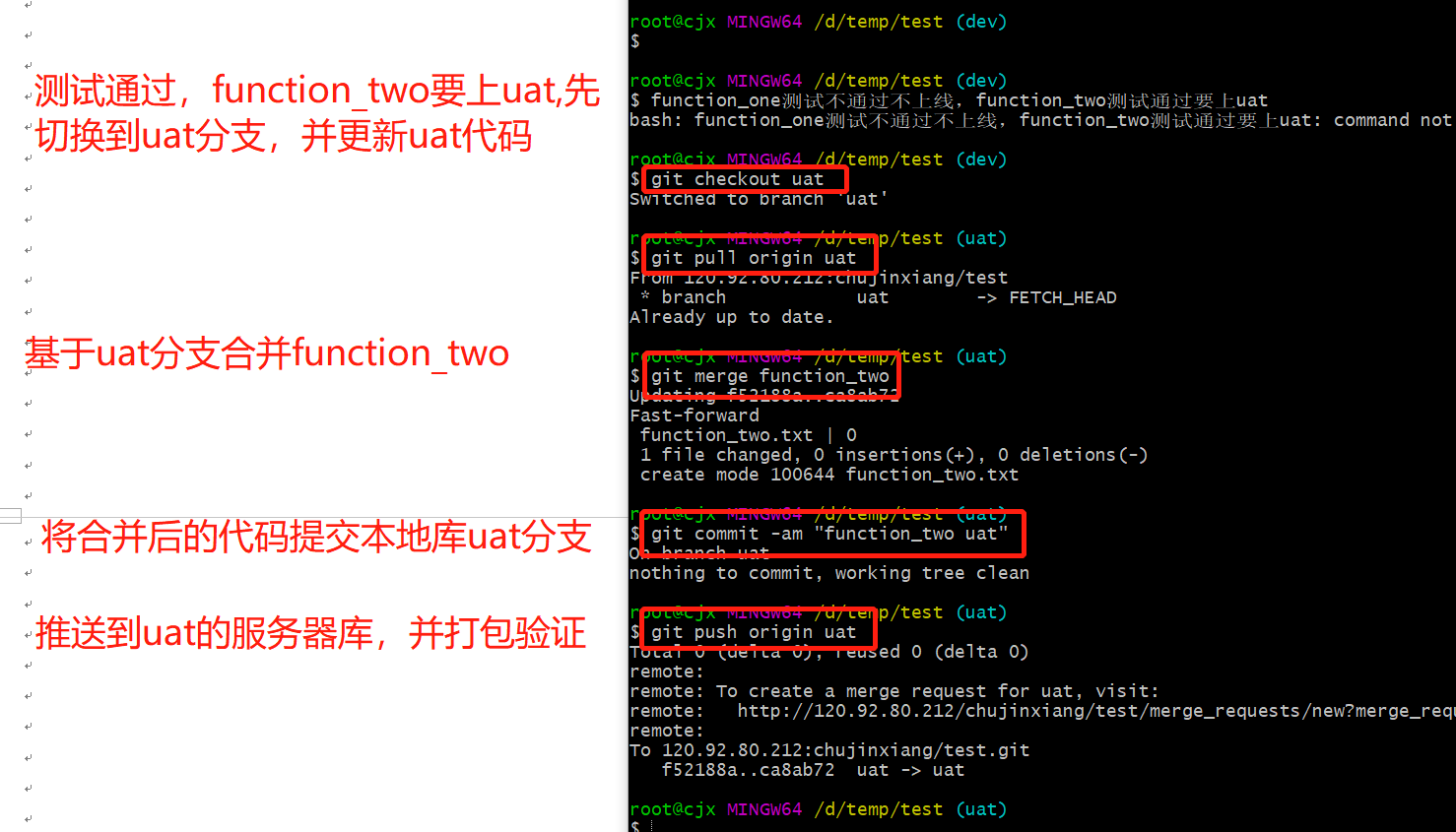
1. **先切换到master分支：git checkout master**
2. **更新master代码：git pull origin master**
3. **和并自己要上的function\_two分支：git merge function\_three**
4. **如果有冲突解决冲突，然后提交到本地库：git commit –am “function three master**
5. **推送到master： git push origin master**

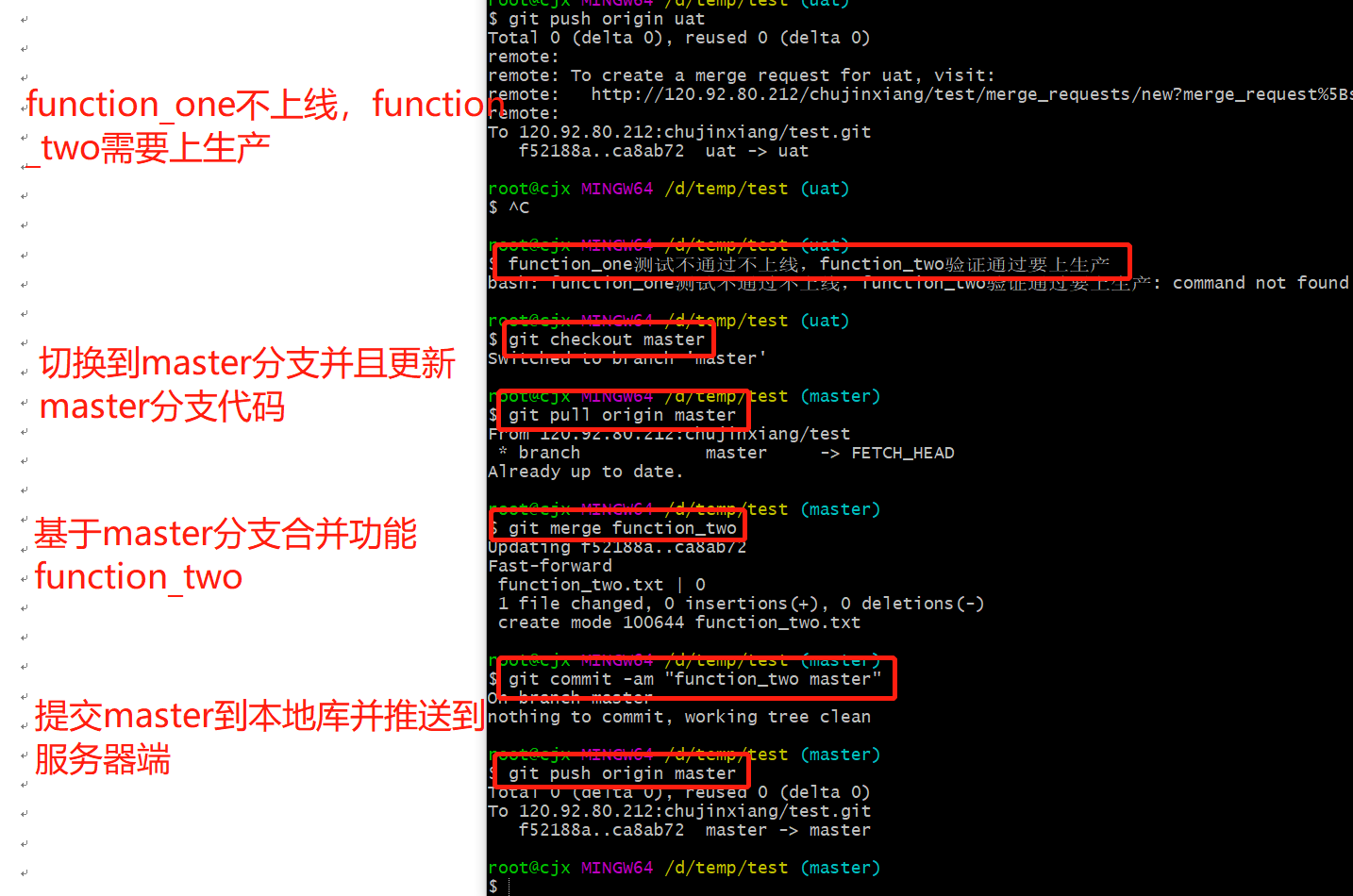
**以下是模拟A同学的git操作：**

****

****

****

****

****

**应用场景二、**

**AB两位同学同时开发一个功能big\_function,此时没办法独立建立自己分支，则两人合用一个分支进行开发，将此分支直接推送到服务器端，开发完成后用dev\uat\master分别合并此big\_function分支**