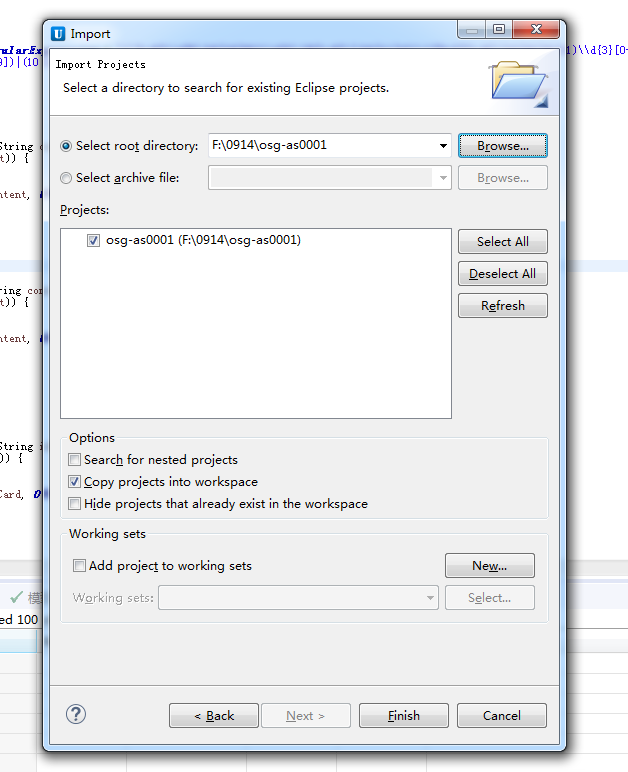
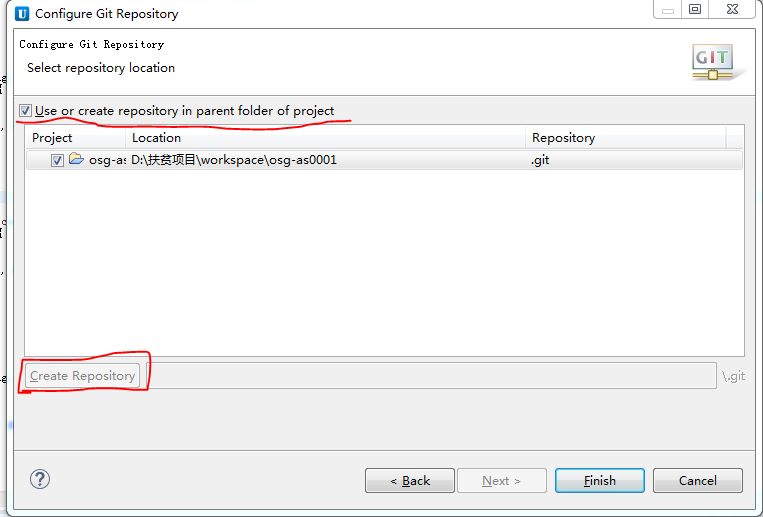
# 一、首次将本地代码上传到Git上的流程：

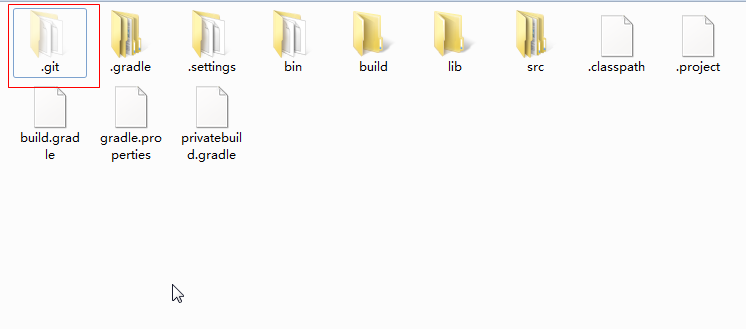
1. 在目前UAP的工作空间下import导入要上传的微服务模块；



备注：勾选上Copy到工作空间下

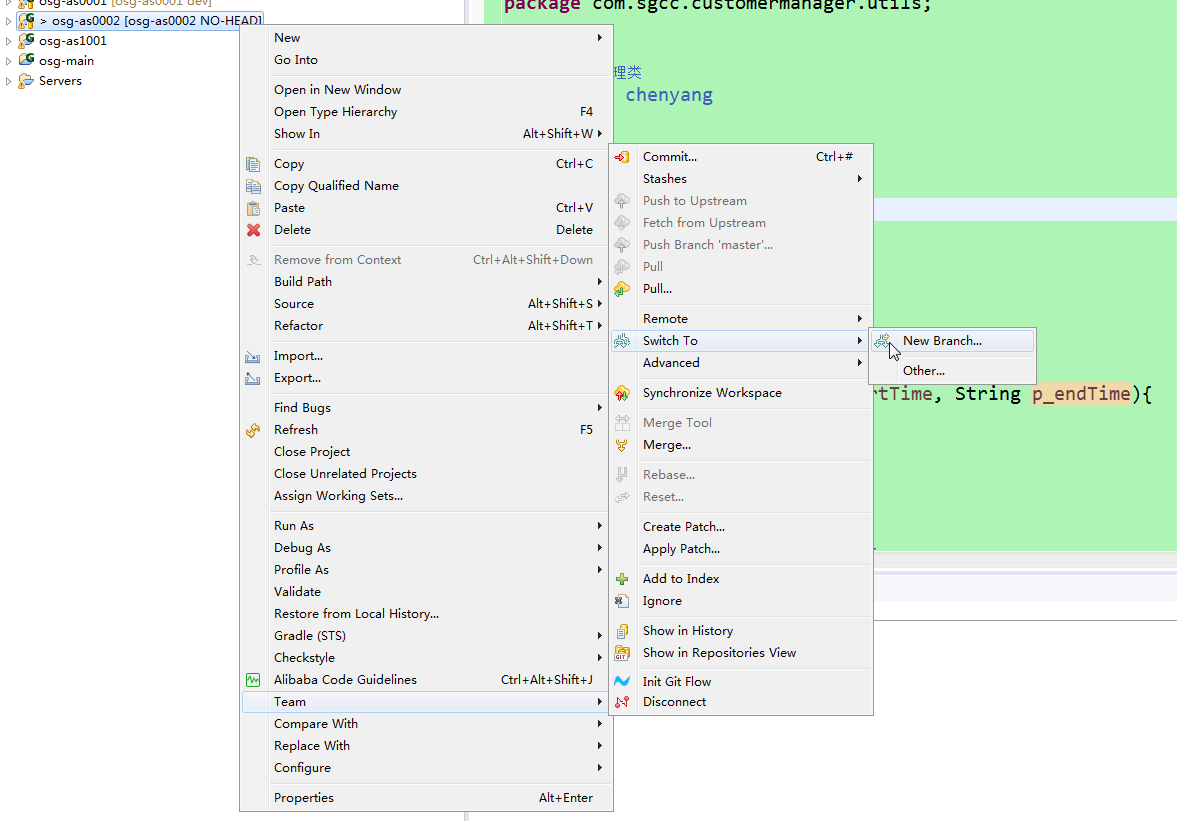
1. 导入代码后需要新建本地仓库，具体为选中微服务项目右键-->Team-->Share Project 弹出框中选择Git，点击Next操作



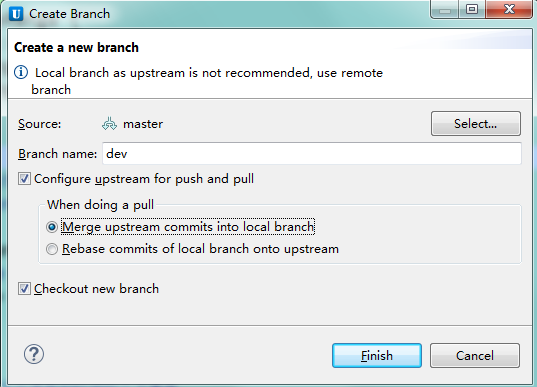
若是首次操作则需要勾选如图所示的Use or create repository in...选项，然后点击下方的“Create Repository”按钮，此步骤相当于是在当前微服务代码文件夹下创建一个.git的目录（有些类似于SVN管理代码一样，会产生一个.svn的文件目录），然后点击”Create Repository”按钮，如下截图所示：

最后Finish操作即可，操作完成以后，项目名称后会出现[NO HEAD]的后缀，此种情况是说明本地仓库中不存在任意一个分支，需要先右键项目Team-->Commit提交（Commit and Push 或者Commit按钮均可）一个或者几个文件，操作完成后，[NO HEAD]的后缀会变成[master]后缀

3、此时右键项目Team-->Switch To-->New Branch...



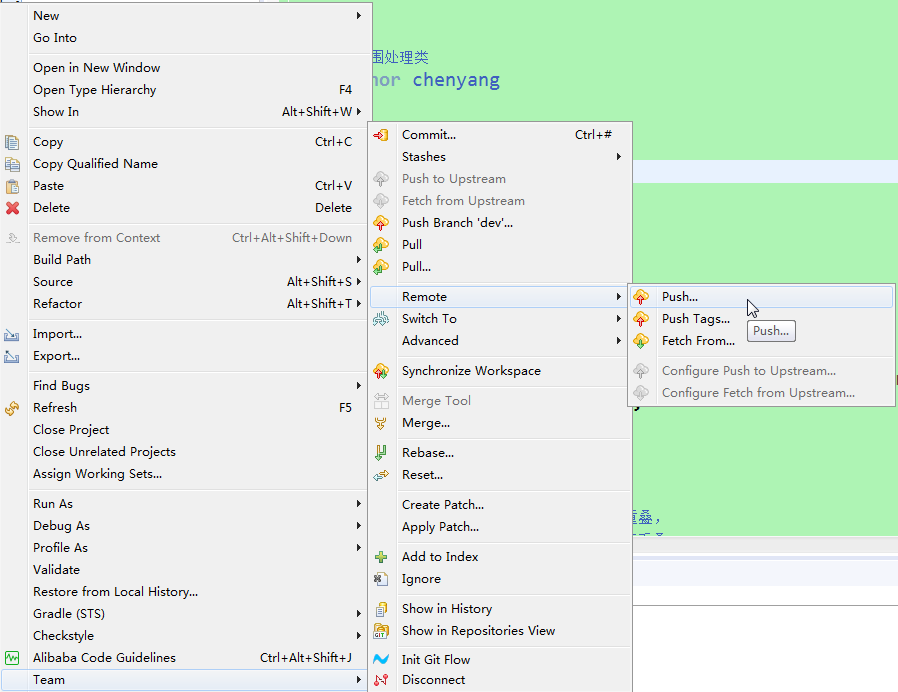
然后新建Switch仓库分支，如下图：



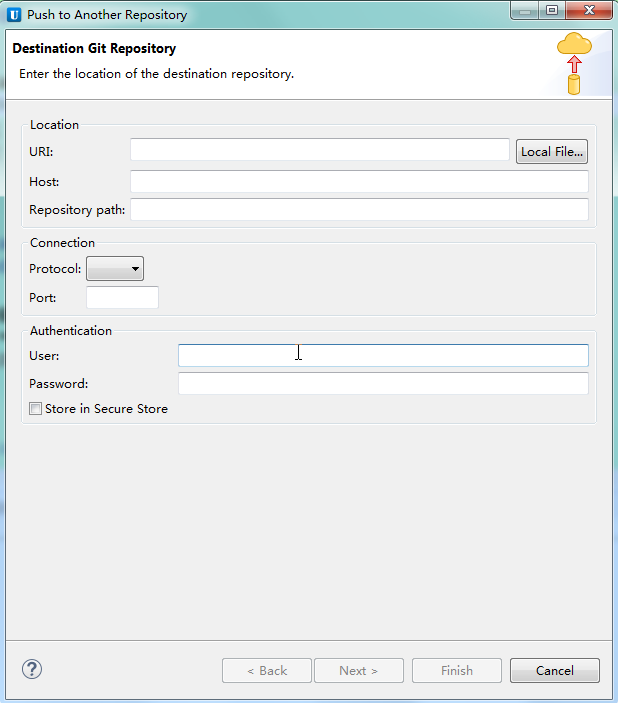
通常dev是开发分支，master是默认分支，test是灰度测试发版分支，默认新建dev分支即可，然后Finish操作成功后，项目后缀即变成了[dev]



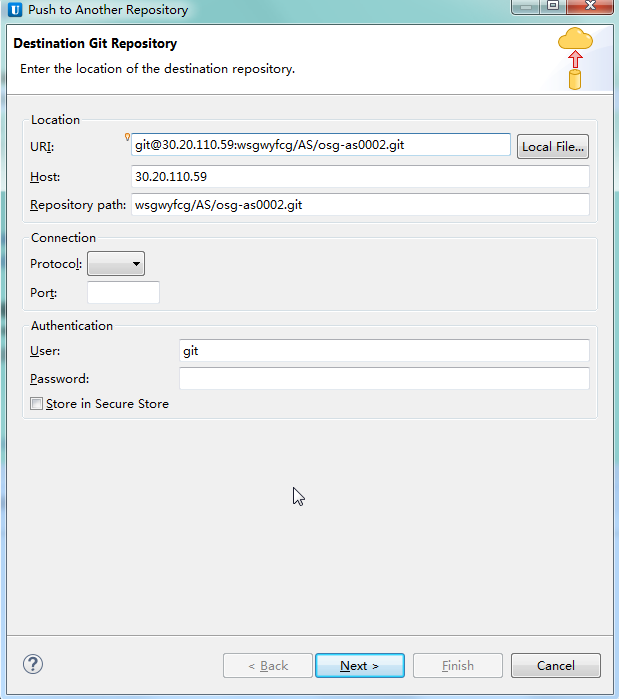
4、然后项目右键Team-->Remote-->Push...关联至远程git仓库，

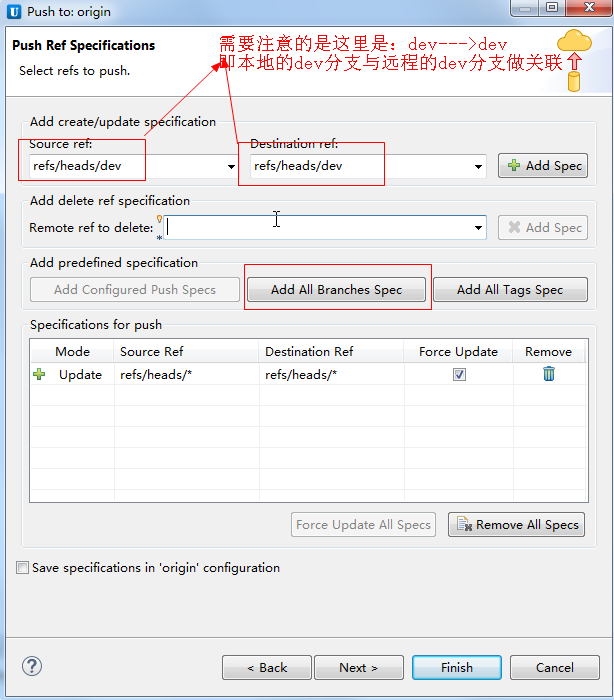


1. 出现如下图所示界面：

其中URI输入从浏览器中拷贝出的微服务git地址，类似于这样：[git@30.20.110.59:wsgwyfcg/AS/osg-as0002.git](mailto:git@30.20.110.59:wsgwyfcg/AS/osg-as0002.git)

其它选项都是输入URI以后默认自动填充的，User和Password都不用填写（因为之前生成过私钥公钥，已经记住了用户名和密码，详细介绍可参见**第五部分**）如下图：



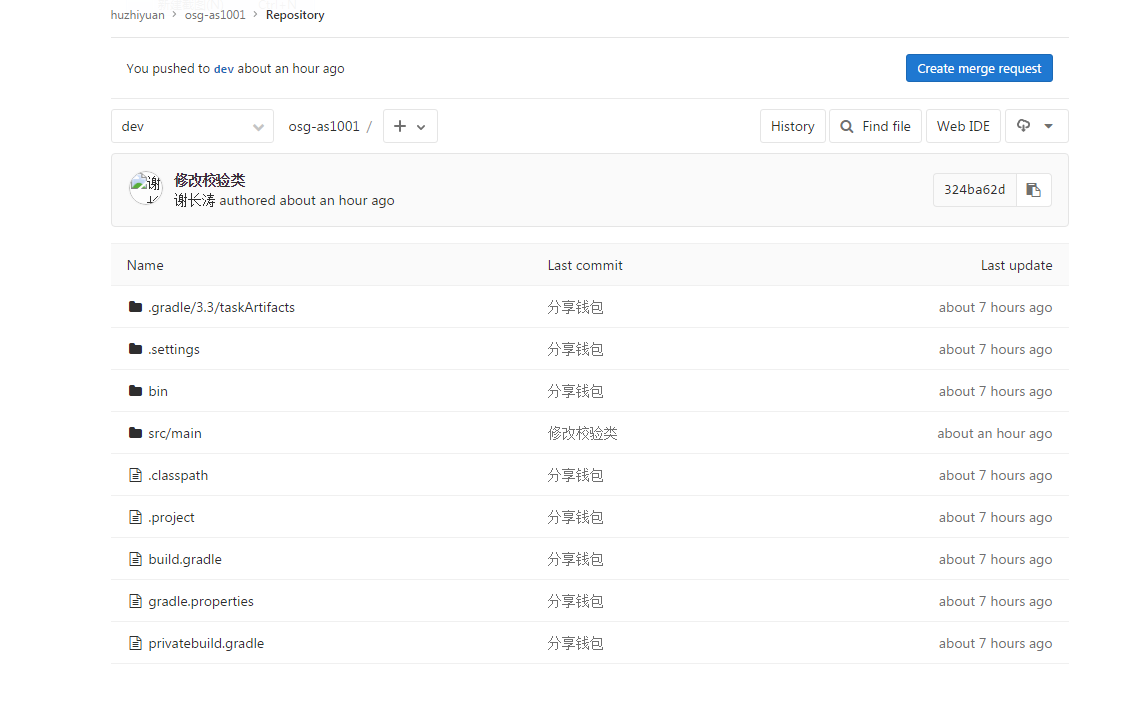
点击“Next”，出现如下界面：  


其中Source ref、Destination ref 等各个选项均按照截图红色框体所示填写即可，然后点击Next，最终Finish操作

1. 接下来右键Team-->Push to Upstream操作，如果未出现异常，则说明推送成功，多数情况下第一次Push操作是失败的，此时具体解决办法以及操作命令可参考**第四部分**，通过该部分的介绍可以解决这个问题

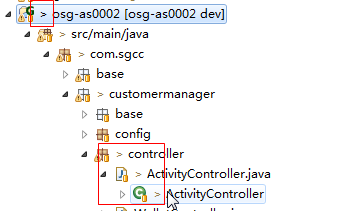
7、操作完成后登陆访问内网Git地址查看是否成功（刷新目录，查看是否存在新上传的微服务项目名称，并且查看最后的更新时间是否正确）

Git网址：http://10.71.12.38:18001/【内网】

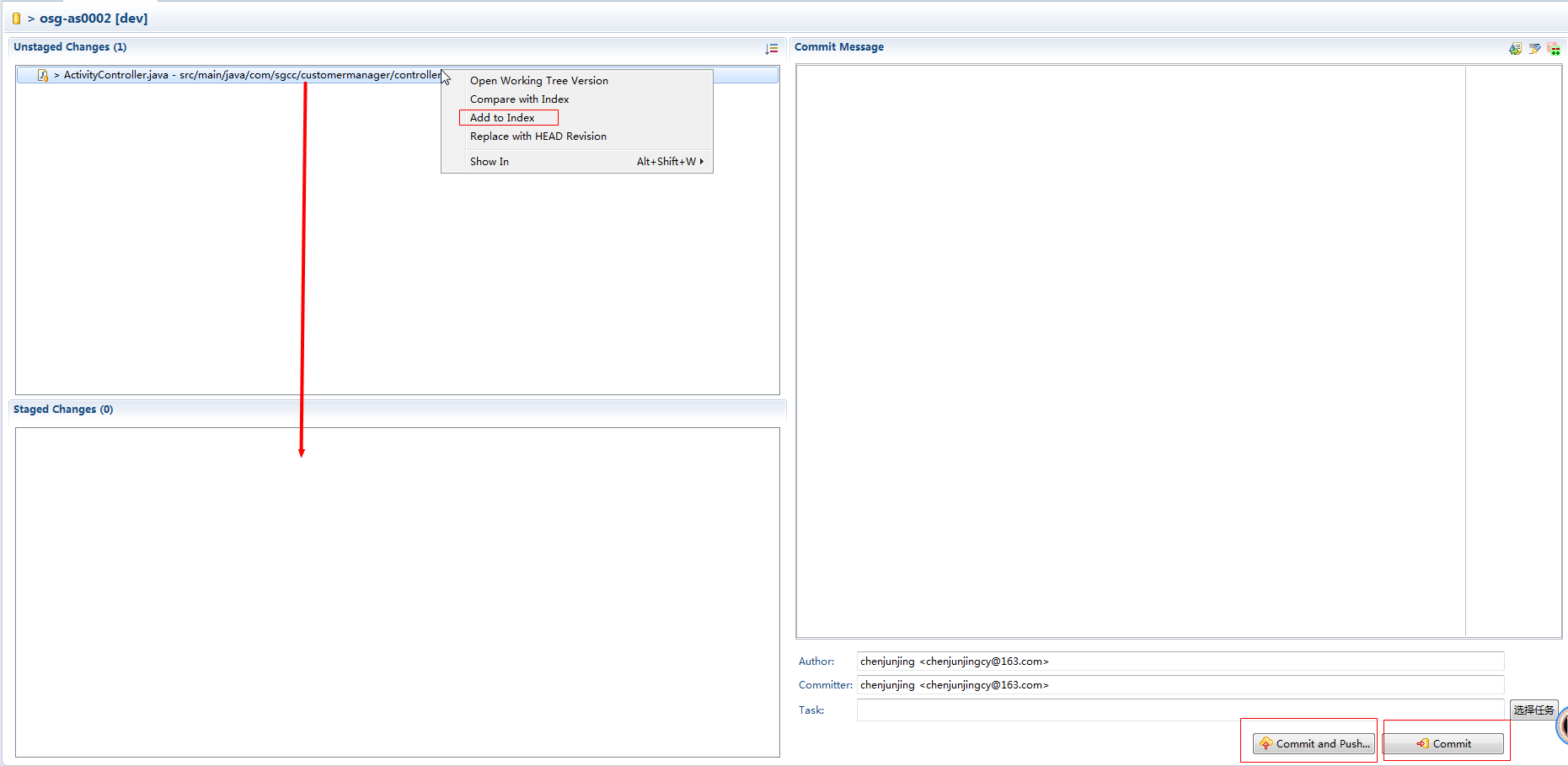


# **二、后续代码变更提交至Git上的流程：**

1. 如果有变更的代码，项目上会有**>**的符号箭头，形如：



1. 提交到Git上的步骤是：
2. 可以选择项目右键Team-->Commit...



左上角区域为待提交的代码，可以选中一个或多个文件右键，Add To Index 操作，然后会出现在左下角区域，此时提交代码有种方式，具体说明如下：

1. 如果使用“Commit”按钮提交的话，操作完成后会在项目上出现一个↑1 标记的



这时还需要“Push to Upstream”操作，这样才相当于是提交成功至分支；

（2）如果使用“Commit and push...”按钮提交的话，若不存在代码冲突的问题，则可以直接一步到位的提交成功至分支，但若存在代码冲突等未知问题，则此按钮功能会失效。

B、除了上述操作方法外，也可以采用单个文件（或多个文件）选中直接右键Team-->Add To Index增加到staged Changes区域，最后选择“Commit and push...”或“Commit”直接提交

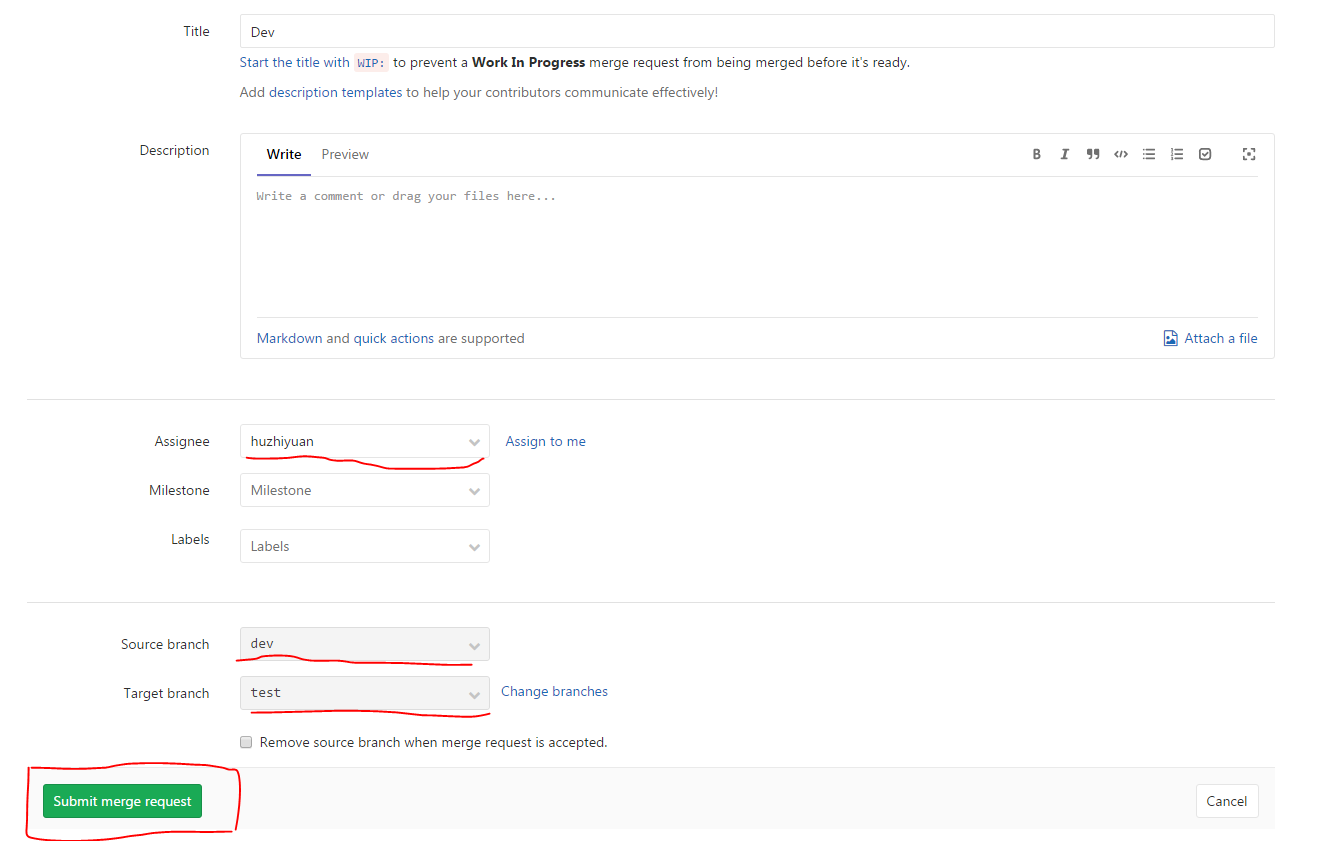
# **三、浏览器Git上新建合并请求的流程：**

首先说明，新建合并请求的目的是为了让管理组帮我们把git上dev分支下的开发代码同步到test测试环境，以便联调测试用

打开Git网址，找到如下图所示界面



然后点击右上角绿色按钮“New merge request”,出现如下界面



按照红色划线部分填写即可，输入标题和描述，点击绿色按钮提交，等待审批

需要注意的是：合并请求均是dev源分支到test目标分支的

# **四、解决IDE中Push to Upstream失效问题：**

第一次将微服务代码与Git关联后在右键Push to Upstream操作后，提示失败或者操作成功但是在浏览器git上未查看到有最新的变更史，则说明工具中Push不好使，这个时候就要使用Git命令行来操作解决该问题，具体解决步骤如下：

1. 强推到远程分支：git push origin --force
2. 如果遇到如下报错信息，意思是说，本地分支暂未与上流（远程）分支建立联系，按照提示命令操作：git push --set-upstream origin dev



1. 如果输入2的命令后，仍然提示报错信息如下图：



这个时候查看下远程分支则会发现并不存在，

远程分支查看命令：git branch -r

所有分支查看命令：git branch -a



此时需要手动同步一下分支（把远程分支拉取到本地），命令：git fetch

然后再查看分支便会出现远程分支信息（红色部分中的remotes属于远程分支），如图：



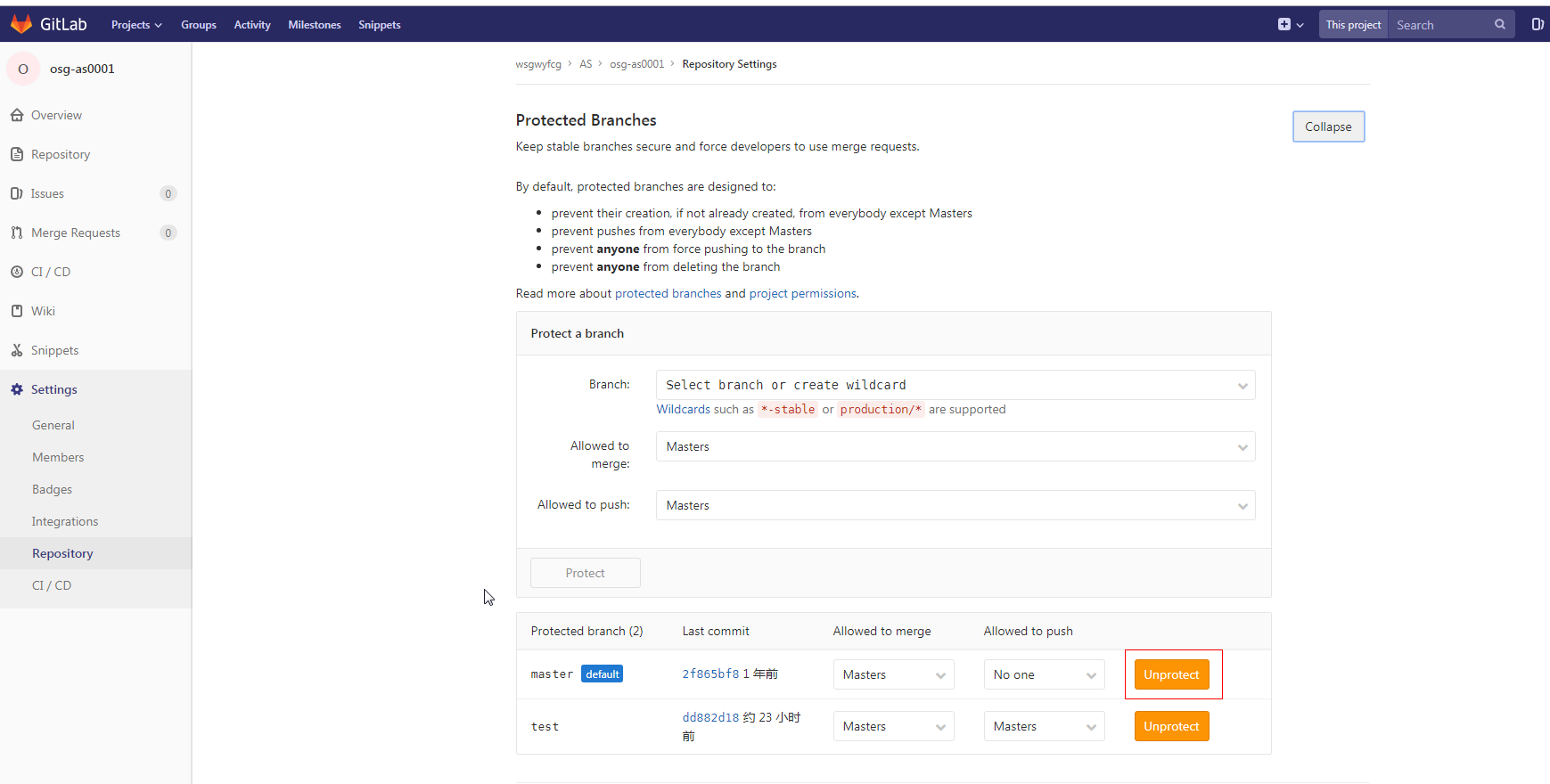
1. 接着再次按照第2条命令进行操作，如果不报错则说明操作成功，提示如下：



1. 再次执行第1条强推命令，如果仍然出现以下报错信息



出现这种报错信息，大概率是因为要push的分支权限不够问题，解决办法是登录Git浏览器地址将该分支权限设置为不受保护状态，具体操作截图如下：



1. 设置完权限后，再次强推，操作成功后如下图所示：

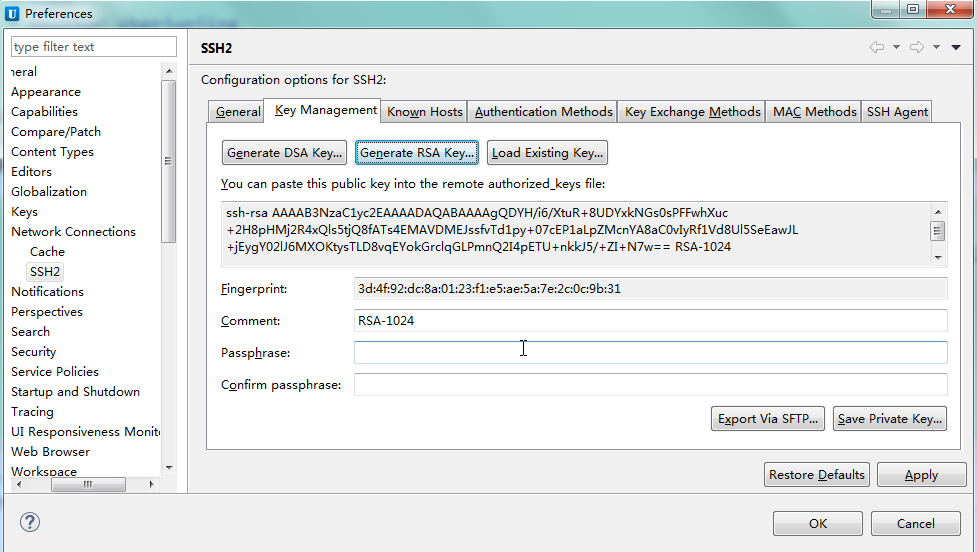


1. 然后在浏览器端查看是否提交成功

# **五、关于如何生成SSH秘钥公钥：**

在第一次项目Team-->Remote-->Push...操作时，关联远程分支界面除了需要填写远程Git地址外，初次操作还会提示输入用户名和密码，并且在以后每次Push操作都会有这一步骤令人感到繁琐，所以，可以使用生成秘钥公钥的方式来一劳永逸的避免这种情形，具体的方式如下：

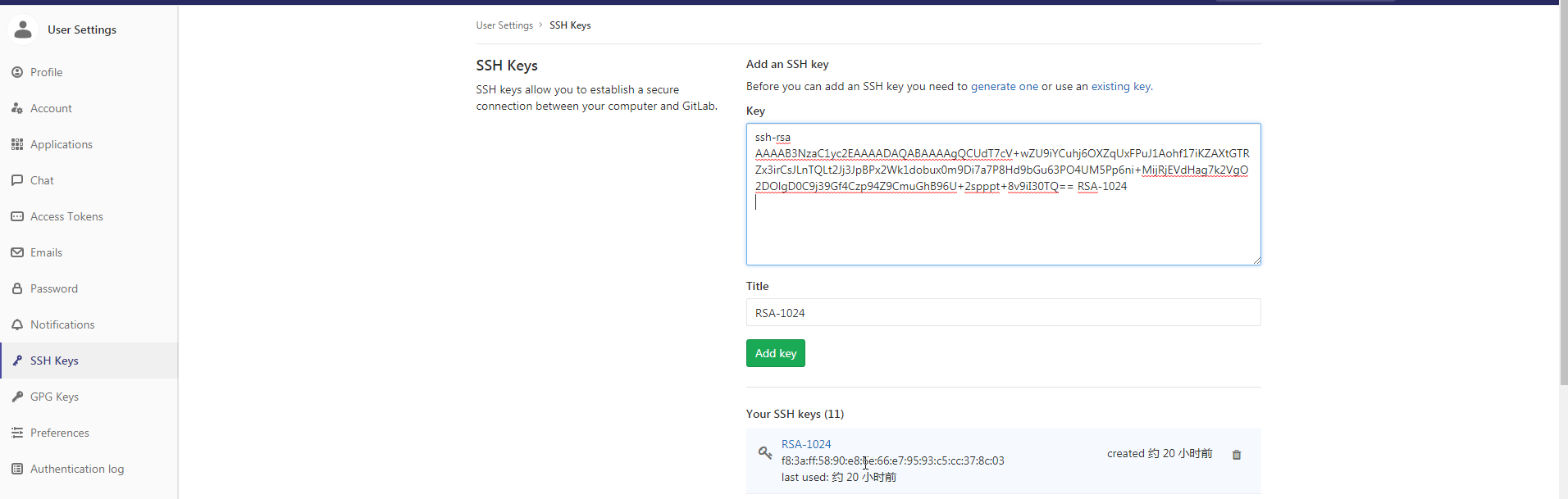
1. 可以使用UAP生成公私秘钥，Window-->Preferences-->Nerwork Connections-->SSH2，界面如下：



点击“Generate RSA Key...”生成秘钥，密码两项不用填写，然后点击“Save Private Key”保存秘钥

PS：C:\Users\Administrator\.ssh文件夹下已经存在id\_rsa、id\_rsa.pub这两个文件说明之前已生成过秘钥公钥，手动删掉再保存新生成的秘钥

1. 然后打开Git的浏览器地址，依次找到个人中心-Settings-SSH Keys选项，



将.ssh目录下的id\_rsa.pub文件中的公钥复制到截图所示框中，然后点击Add key按钮即可，下次在UAP中Team-->Remote-->Push...的操作时便无需再重复输入Git地址和用户名密码了（对应第一部分第5小节的操作步骤）

# **Git基本操作：**

Git的工作就是创建和保存你项目的快照及与之后的快照进行对比，常用的有以下6个命令：**git clone、git add、git commit、git push、git checkout、git pull**



说明：

* workspace：工作区
* staging area：暂存区/缓存区
* local repository：版本库/本地仓库
* remote repository：远程仓库



# **Git常用命令示例：**

1、git status （查看文件状态命令）

2、git add index.html

1.1、git add . （把目录下所有的文件都添加到暂存区）

3、git commit -m‘第一次提交’

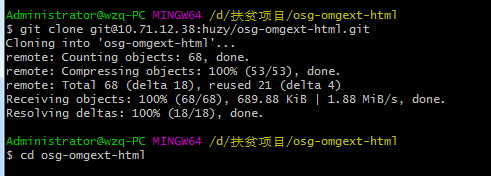
4、git remote add origin（或者origin\_ssh）git@github.com:chenjunjing/git-demo.git（为远程仓库地址起一个别名，下次直接push origin即可）

5、git push origin master

6、git push -u origin master （-u的意思是记住要提交到的远程仓库地址，下次就不用再输入仓库地址了，即：下次再输入，只需要输入git push 即可提交到远程仓库）

7、git clone远程仓库地址（把远程仓库地址克隆到本地）

8、git clone [git@10.71.12.38:huzy/osg-omgext-html.git](mailto:git@10.71.12.38:huzy/osg-omgext-html.git)[远程仓库地址]



使用git clone直接跟远程仓库地址的话会把远程仓库的所有分支董给克隆到本地，但只会在本地默认创建一个master分支，可以查看所有分支

9、如果单独克隆某个分支可以使用命令：

git clone -b dev [git@10.71.12.38:huzy/osg-omgext-html.git](mailto:git@10.71.12.38:huzy/osg-omgext-html.git)[远程仓库地址]

10、git branch -r [查看远程仓库分支]

11、git branch -a [查看所有仓库分支]

12、如果克隆的是全部分支，可以使用checkout命令来把远程分支取到本地，并自动与远程仓库的同名分支建立关联，可以使用如下命令：

git checkout -b dev origin/dev

或者使用t参数，它默认会在本地建立一个和远程分支名字一样的分支

git checkout -t origin/dev

# **git push推送当前分支与.gitconfig文件中的push default**

## 【一】、git push origin与git push -u origin master的区别

### 1、git push origin

* 该命令表示，将当前分支推送到origin主机的对应分支；
* 如果当前分支只有一个追踪分支，那么主机名都可以省略；

### 2、git push -u origin master

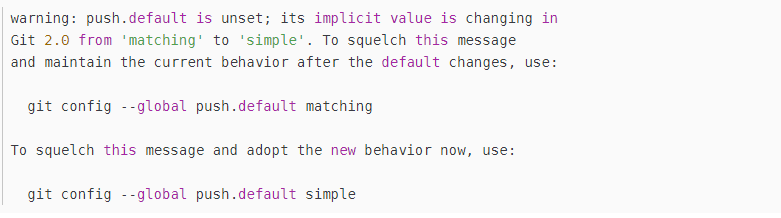
* 该命令将本地的master分支推送到origin主机，同时指定origin为默认主机，后面就可以不加任何参数使用git push了；

### 3、git push

* 如果当前分支与多个主机存在追踪关系，那么这个时候-u选项会指定一个默认主机，这样后面就可以不加任何参数使用git push；
* ****不带任何参数的git push，默认只推送当前分支，这叫做simple方式。此外，还有一种matching方式，会推送所有有对应的远程分支的本地分支。Git 2.0版本之前，默认采用matching方法，现在改为默认采用simple方式。****

## 【二】、Git 2.0更改为push default为’simple’

### 如果你最近更新了Git版本，你可能会在执行git push时看到如下消息：



### 1、Matching

* matching参数是Git 1.x的默认行为，其意是如果你执行git push但没有指定分支，它将push所有你本地的分支到远程仓库中对应匹配的分支；

### 2、Simple

* 而Git 2.x默认的是simple，意味着执行git push没有指定分支时，只有当前分支会被push到你使用git pull获取代码时的分支；

### 修改默认设置

从上述截图提示消息中，我们可以修改全局配置，使之不会每次push的时候都提示 [注意push与default之间的符号.]：

* 对于matching输入以下命令：

git config --global push.default matching

* 对于simple输入以下命令：

git config --global push.default simple

* ****git config --global命令实际上是在操作用户目录下的.gitconfig文件，其内容如下：****

[user]

name = alice

email = [alice@aol.com](mailto:alice@aol.com)

[push]

default = simple

[credential]

helper = store

****git config --global user.email "alice@aol.com" 操作的就是上面的email  
git config --global push.default matching 操作的就是上面的push段中的default字段  
git config --global credential.helper store 操作的就是上面最后一行的值****