2016年9月12日

随笔

往事如风，这些只言片语，或许对你的学习有一些参考价值

# 前言

一年前的事，如今又重新来一遍，才知道当时做了多少工作，幸好，当时的系统镜像没有删除，而如今只是做一些修修补补的事情，但就是这样，还是感到有些力不从心，连最基本的命令都需要重新学习，还好基础不错。

每一次重来，势必都会有一些收获，linux这东西就是这么让人琢磨不定，魅力这东西没体会到，折磨人到是真的。

有个习惯，就是喜欢把遇到的问题记录下来，无论是给自己还是给别人，做个参考吧。

Lamp这套组合还真是经典，可是比较搞笑的是，我居然连怎么装centos都不知道了，基础模式最小安装，呵呵居然还要百度一下，分区更是搞笑，居然还要分boo/,有意义吗，交换分区还是要的。

Linux安装完了，apache、mysql、php通常是下载安装包，而我还是选择了最难的源码安装模式，因为这个模式需要做的工作非常的多，从编译方式，到启动方式，都需要你手动配置，这个真的很麻烦，也很复杂。

因为这个编译是以前编译的，也就是这个软件都安装完了，但我需要从新配置。

第一个问题就是apache的问题，因为是源码安装，rpm service这样的命令都查不到，复习一下：

rpm –qa |grep apache或者httpdld 无结果

service –list-all 无结果

好在当时的工作做的比较扎实，把这些东西都做到系统服务里了，这样就有办法查看了，要不就只能一个一个手动查询了。

chkconfig –list

看到了久违的apache mysql php-fpm，而且都已启动，说明他们都在正常工作，那就测试一下吧，

2了吧唧的我直接输入地址测试apache，就是报403 404这种错误，仔细研究才发如下问题：

1 apache主目录搞错 导致不能访问 需要编辑httpd.conf

2 mysql管理工具phpMyAdmin无法访问，主要是config.inc.php和config.defalut.php配置问题

3 高级功能不全，导入create\_phpmyadmin.sql

4 短密码问题，cookie问题

也不知道哪来的那么多问题，反正不顺利。

好在这些问题最后还是都解决掉了，网上的问题答案太不靠谱了，全是扯犊子，就说短密码问题，还是在老外的帖子里解决的，这个短密码不是随便敲几个字符就可以的，而是最少要46个字符，这个错误才能取消掉。奶奶的，网上的教程有多少是随便敲几个就可以，还有甚者居然说空格都行，这些教程都是抄袭的吧，根本就没有人测试，太坑了。

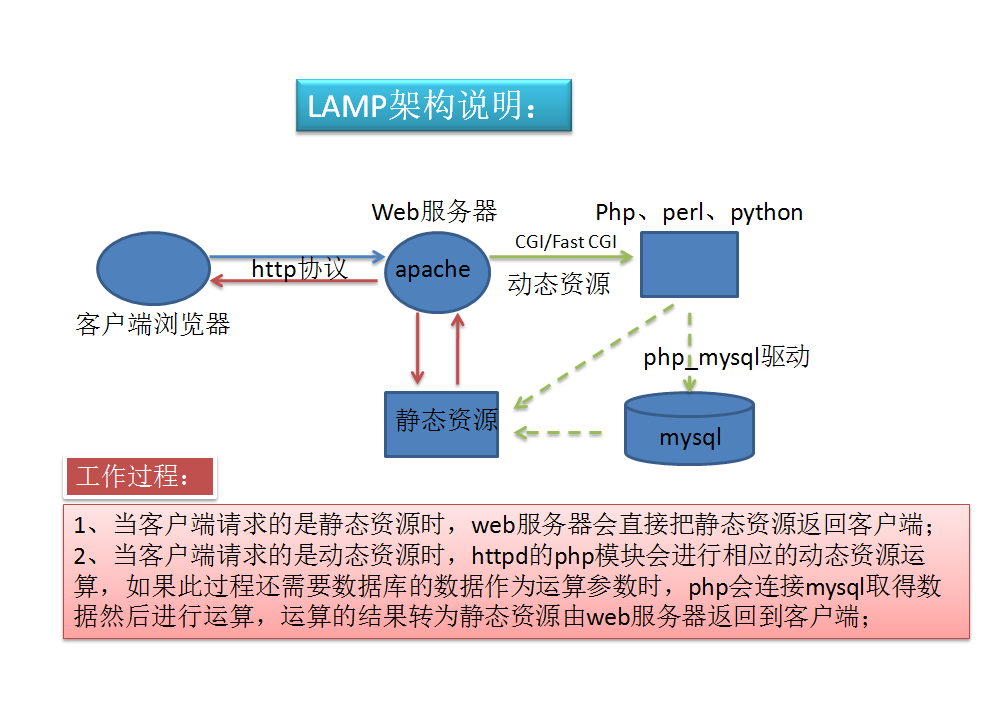
# 1、LAMP架构原理及搭建详解

## 1.1、何为LAMP

LAMP（Linux-Apache-MySQL-PHP）网站架构是目前国际流行的Web框架，该框架包括：Linux操作系统，Apache网络服务器，MySQL数据库，Perl、PHP或者Python编程语言，所有组成产品均是开源软件，是国际上成熟的架构框架，很多流行的商业应用都是采取这个架构，LAMP具有通用、跨平台、高性能、低价格的优势，因此LAMP无论是性能、质量还是价格都是企业搭建网站的首选平台。

## 1.2、LAMP的架构以及通信过程

### 1.2.1、LAMP的架构：



LAMP是一个多C/S架构的平台，最初级为web客户端基于TCP/IP通过http协议发起传送，这个请求可能是动态的，也可能是静态的。

所以web服务器通过发起请求的后缀来判断，如果是静态的资源就由web服务器自行处理，然后将资源发给客户端。如果是动态这时web服务器会通过CGI（Common Gateway interface）协议发起给php。

这里如果php是以模块形式与Web服务器联系。那么他们是通过内部共享内存的方式。如果是php单独的放置与一台服务器，那么他们是通过sockets套接字监听的方式通信（这又是一个C/S架构）。

这时php会相应的执行一段程序，如果在执行程序时，需要用到数据。那么php就会通过mysql协议发送给mysql服务器（也可以看作是一个C/S架构）。由mysql服务器处理，将数据供给php程序。

### 1.2.2、lamp流程：

1、用户发送http请求到达httpd服务器

2、httpd解析url获取需要的资源的路径，通过内核空间读取硬盘资源，如是静态资源，则构建响应报文，发回给用户

3、如果是动态资源，将资源地址发给php解析器，解析php程序文件，解析完毕将内容发回给httpd，httpd构建响应报文，发回给用户

4、如果涉及到数据库操作，则利用php-mysql驱动，获取数据库数据，返回给PHP解析器。

### 1.2.3、A,M,P是怎么联动起来工作的呢：

#### 1.2.3.1、apache + php结合的方式大概几种：

第一种：把php编译时直接编译成apache的模块、module模块化的方式进行工作(apahce默认的这种方式)。

第二种：CGI、通用网关接口、apache基于CGI跟php通信

第三种：fastcgi、也是一种协议、在这种模块下他们两个是这样结合的：

本来php是做为一个模块或者是php解析器运行的，不是监听在某个套接字上接收别人的请求的，而是让别人调用为一个进程使用的，可能是做为别人的子进程在运行，但是工作在fastcgi这种模块下的php自行启用为一个服务进程，它监听在某个套接字上，随时可以接受来自客户端的请求的，它也是有一个主进程的，为了可以响应多个用户的请求，它会启用多个子进程，这些子进程我们也可以称为工作进程，它也是有空闲进程的，一但有客户请求他马上使用空闲的进程响应客户端的请求，将结果返回给前端的调用者，在php5.3.3版本之前它是没有这个能力了，只能工作在模块和CGI的方式下，而在5.3.3之后这个模块直接被收进php模块中，这种模块就叫php-fpm。

所以在以后编译php时，要想跟apache结合，就要编译成php-fpm，这是基于fastcgi工作的模式，并启动这服务进程，也就意味着他是通过套接字跟前端的调用者通信，既然基于套接字通信了，那么前端的web服务器和后面的php服务器完全可以工作在不同的主机上，实现了所谓的分层机制。

apache不会跟数据库打交道，它是个静态web服务器，跟数据库打交道的是应用程序，作为应用程序的源驱动能够基于某个API跟服务器之间建立会话，而后它会通过我们的mysql语句发送给数据库，数据库再将结果返回给应用程序，不是php进程，而是php进程中所执行的代码。

#### 1.2.3.2、php + mysql的通信：

PHP跟mysql怎么整合起来呢，php又怎么被httpd所调用呢

首先httpd并不具备解析代码的能力，他要依赖于php的解析器，接着php本身不依赖于mysql，他只是一个解析器，能执行代码就OK了，那它什么时候用到mysql呢，如果要在mysql中存数据时才用到mysql，只是当php中有运行mysql语句时才用到mysql。

php语言要想联系mysql，通常用到php的驱动，rpm包的叫php\_mysql，php跟mysql没有一点关系，只有程序员在php中编写mysql语句时才连接mysql来执行sql语句的。

基于php-mysql去连接mysql只使用一个函数mysql\_connect();而mysql\_connect()正是php-mysql提供的一个API，只要指明要连接的服务器即可。

## 1.3、问题集锦

phpmyadmin提示：配置文件现在需要绝密的短语密码(blowfish\_secret)解决方法。

问题：在登录phpmyadmin时，有时会出现“配置文件现在需要绝密的短语密码(blowfish\_secret)”这样的错误提示。

解决方法：到phpmyadmin的安装目录里面找到config.inc.PHP文件，用记事本将其打开，通过记事本的查找功能查找“$cfg['blowfish\_secret']=’’”语句。在其单引号里面随便输入至少46个数字或字母（默认为空），最后保存。

总结：出现这样的错误，原因在于我们将config.inc.php和config.default.php里面的$cfg['Servers'][$i]['auth\_type']设置为cookie（默认为config，但这样不是很安全，所以有必要改为cookie）而$cfg['blowfish\_secret']却是空的。

既然将$cfg['Servers'][$i]['auth\_type']设置为cookie，

那么就应该通过某个cookie来访问它，

这正是通过$cfg['blowfish\_secret']来设置。

所以$cfg['blowfish\_secret']不能为空。

请引起注意：config.inc.php 此文件可能在phpmyadmin 搜索或找不到；或许我本人认为可能是：config.sample.inc 这个文件。

但通过本人在config.default.php 里查找：blowfish\_secret 后在找到的

$cfg['blowfish\_secret'] = ' '; 在单双引号里随意添加 符号 并保存文件即可。

最后在网页浏览器地址栏中输入：http://localhost/phpmyadmin 后按回车键；如果还是还没依然提示：

配置文件现在需要绝密的短语密码(blowfish\_secret)那就请再语言栏中选择其他语言即可显示出：

tools | blowfish generator | engage | wpa cracker

Enter the randomly generated set of characters below into your phpMyAdmin config.inc.php file for the $cfg[‘blowfish\_secret’] variable. You can technically use any phrase you like for the blowfish\_secret variable however the below makes it easy so you don’t have to think of one. Every time you refresh this page a new $cfg[‘blowfish\_secret’] will be randomly generated.

$cfg[‘blowfish\_secret’]: Article relating to a blowfish\_secret variable not entered error and some other phpMyAdmin configuration options.

The documentation states that any combination of characters up to 46 total can be used for the blowfish\_secret variable which is used for cookie authentication. The characters below are generated on the fly using lowercase letters, uppercase letters, numbers, and symbols. Each combination is never logged in any way so you can rest assured that your combination of characters is unique and will not be used by anyone else. Just copy the characters at the bottom of the page and enter them into the config.inc.php file located in the phpMyAdmin home directory on your server which by default on CentOS is /usr/share/phpmyadmin.

Example phpMyAdmin Blowfish Secret Variable Entry:

PHP

/\*

\* This is needed for cookie based authentication to encrypt password in

\* cookie

\*/

$cfg['blowfish\_secret'] =

'{^QP+-(3mlHy+Gd~FE3mN{gIATs^1lX+T=KVYv{ubK\*U0V'; /\* YOU MUST FILL IN THIS FOR COOKIE AUTH! \*/

Copy The Output Below To Use For Your phpMyAdmin $cfg[‘blowfish\_secret’]:

rFzsbs5PD#tjTp]@UDW][u-URHFn68f\*4kmE+CV[

If you would like a different character combination than the one above click the refresh button in your browser to generate a new set of characters.

Are there other online tools you would like to see the Question Defense team develop? Let us know about them here.

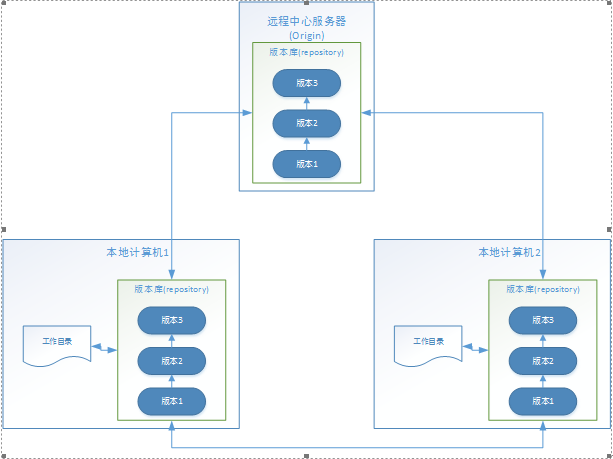
Have questions not answered by the article or this tool then click here to Engage with other technical people.

tools | blowfish generator | engage | wpa cracker

# 2、GitHub

一、Git基本概念

Git是分布式的版本管理系统，每台机器都可以基于远程中心服务器拥有自己的版本，如图：



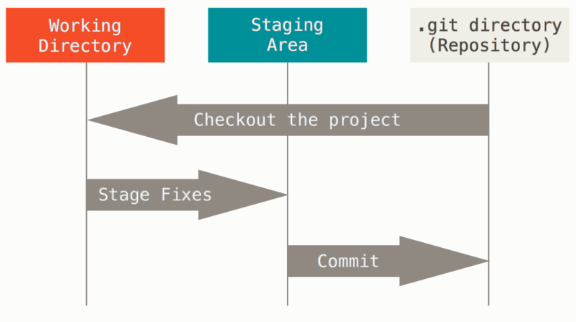
远程中心服务器一般叫做Origin repository。

下面来看一下本地计算机包含哪些内容，图片来自于Git官网，包含三个区域：

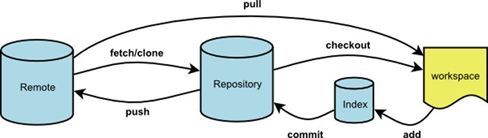
（1）工作目录（working Directory）：我们肉眼能够看到的目录及相应的文件；

（2）暂存区（Staging Area，也有叫Index）：中间过渡区域，可以通过git add命令将工作目录文件放入该区域；

（3）版本库（Repository）：存放版本、分支、Commits的相关信息，通过git commit命令将Staging Area区域的内容放在版本库中；版本库的信息以文件的形式存放在工作目录下的.git隐藏文件夹中；

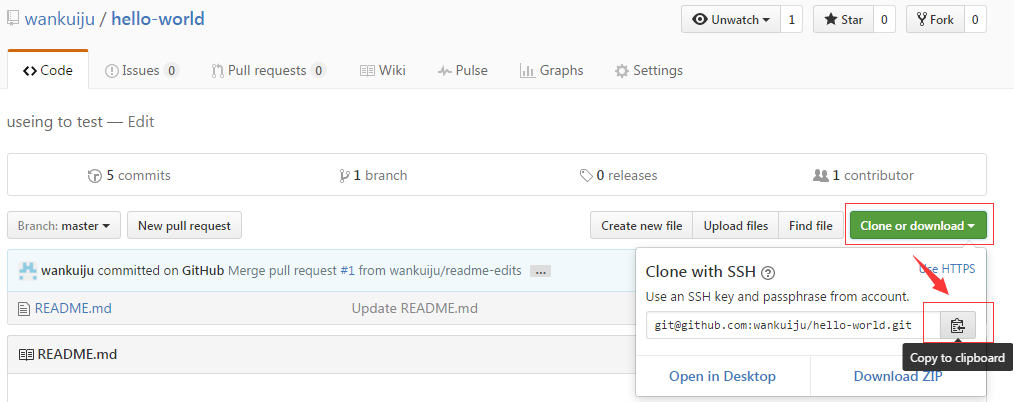


（4）命令集关系



二、Git前期准备

1、找到GitHub中项目地址，点击Clone or download按钮，获取Clone地址



2、由于Clone是通过SSH进行传送的，所以需要生成你机器的认证信息，生成SSH 认证信息，输入如下代码，一路回车：

ssh-keygen -t rsa -b 4096 -C youremail@example.com

具体到我自己就输入：

ssh-keygen -t rsa -b 4096 -C adonis1978@163.com

完成后进入到存储key的文件：

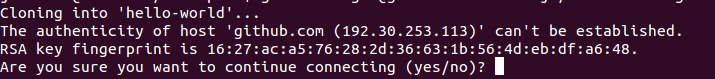
cat ~/.ssh/id\_rsa.pub

copy public key，放在GitHub – setting – SSH and GPG keys

开始Clone，clone前先确认项目存放位置，并进入到相应目录：

juwankui@ubuntu:~/workspace$ git clone git@github.com:wankuiju/hello-world.git

进行clone时会提示如下信息，输入yes，



项目clone完成后，自动创建项目文件夹，我们现在先进入到项目目录：

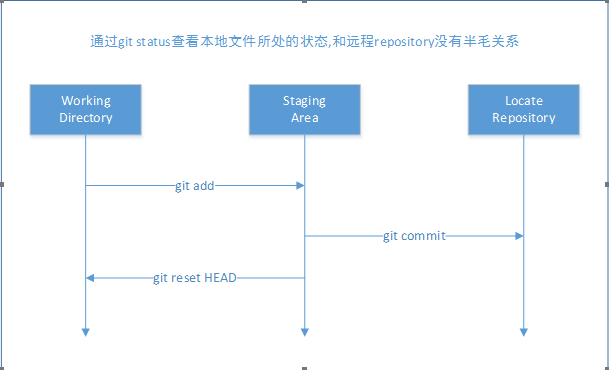
cd hello-world

通过ls –al 可以看到项目目录下存在一个.git的隐藏文件夹,就是我们刚才提到的用于存放当前工作空间的版本信息。

三、不涉及到协作、分支等因素的纯练习内容（不可用于实际项目）

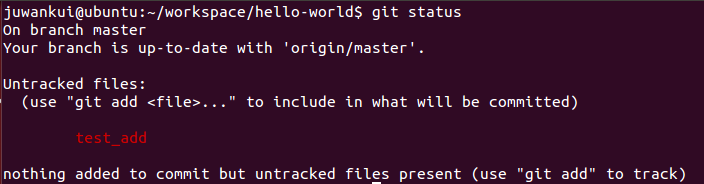
只有在项目第一次从远端下载到本地计算机时才采用clone方式，在后面我会介绍，项目已经存在的情况下，如何根据派发下来的任务进行分支相关的操作，这次暂且不说；

在当前项目目录下新增一个文件名为test\_add的文件，此时我们先不碰触远程版本库，只在本地计算机内玩耍，那么如下图：

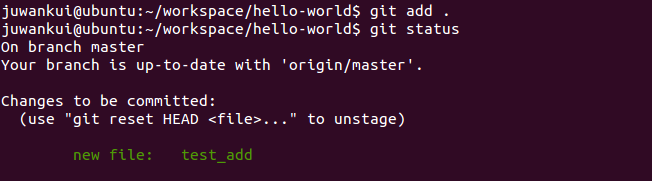


拿刚才我们新建的test\_add文件为例来看一下，文件创建完成后：

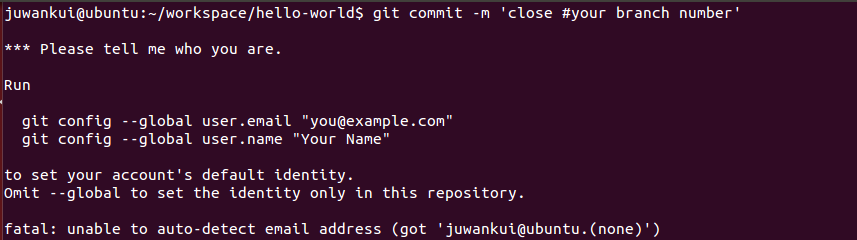
（1）git status看一下本地文件状态，其实git的说明已经非常清晰明了，平时如果出现问题认真看提示说明，就能解决一些问题，如下图告诉我们这个新文件还没有被track，就是没有和版本产生关系，可以通过git add filename，我一般使用 git add .来操作当前目录下所有文件：



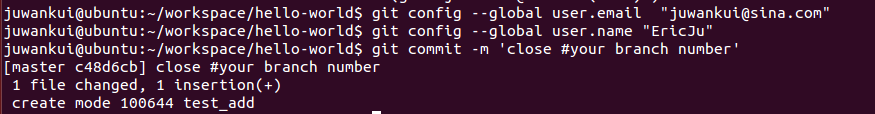
（2）git add 将test\_add纳入版本跟踪，随后再次git status，如下图，文件已经进入staging area：



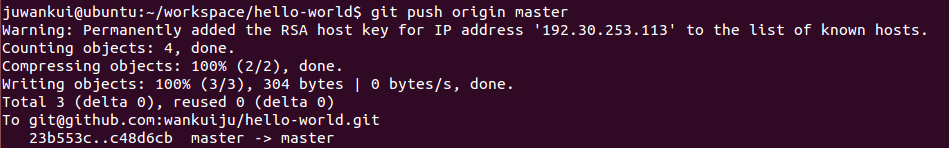
（3）git commit 将test\_add纳入本地版本库，随后执行git status 查看状态



可以看到由于我的本地计算机第一次使用git commit，所以提示需要设置邮箱和用户名（来至GitHub注册信息），按照提示操作，替换成你的信息即可：



（4） 到此，我们在本地玩的很痛快，下一步就需要和远程仓库进行交互了，通过git push将你的版本信息提交到远程仓库（origin repository），我们当前操作的是master分支，一般开发过程中不会直接操作master的。

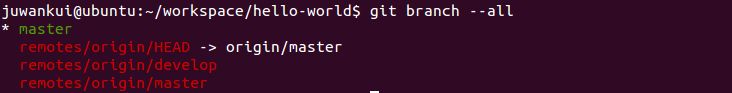


四、 模仿实际项目的操作

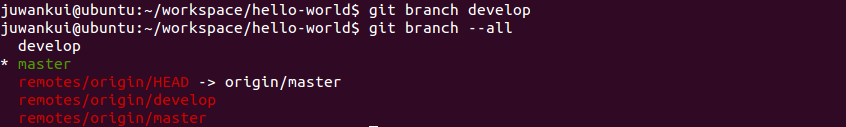
假设我们已经根据上一章的clone方法，将远程仓库（origin repository）中的项目下载本地（默认clone远程的master分支），还有一点需要注意，我们一般不会直接在master主分支上进行项目操作，那怎么办，可以在远程仓库中基于master分支建立一个develop分支，用于开发工作。

1. git branch –all 查看分支

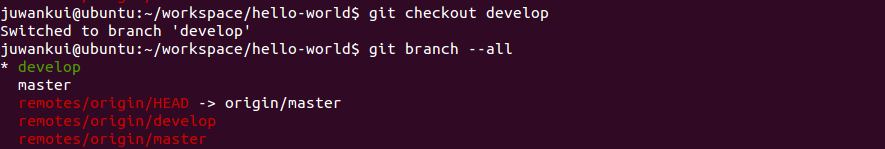
查看远程版本库和本地版本库中的分支信息，目前我本地只有一个分支，分支前面的“\*”代表本地当前分支，远程仓库有两个分支，并且远程仓库当前分支为origin/master（即 远程仓库中的/master分支）：



2. git branch develop创建develop分支



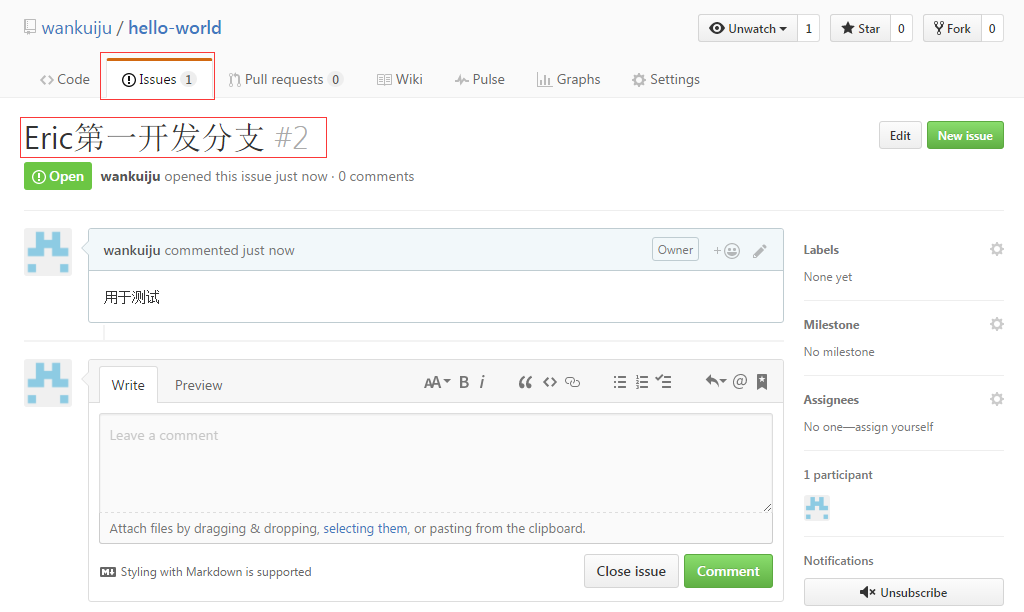
3. git checkout develop切换到develop分支

我们一般使用[**Git**](http://lib.csdn.net/base/git) checkout –b develop 一条语句来实现创建和切换到指定分支的操作，此处是为了演示。   


4. 保证本地develop分支最新

通过执行git fetch（从远程仓库中拉取所有分支信息到本地），git rebase origin/develop（将拉取的origin/develop的内容合并到本地当前分支），这样就保证当前分支develop同步于远程分支。

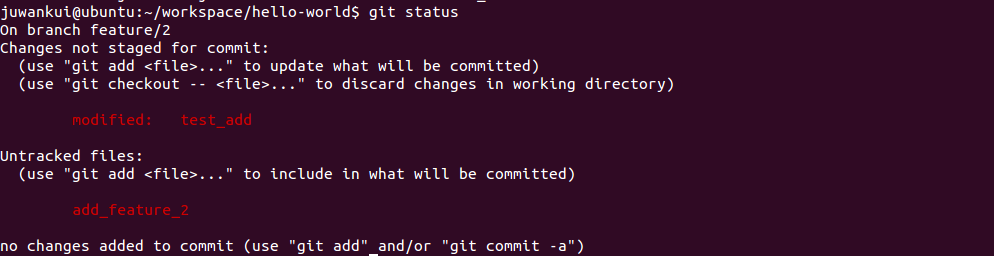
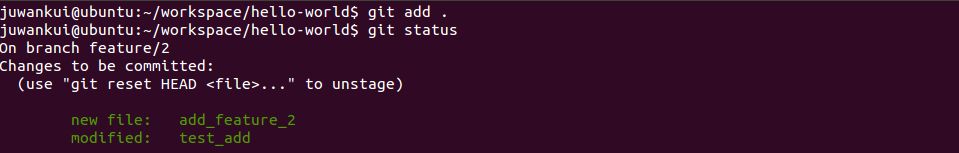
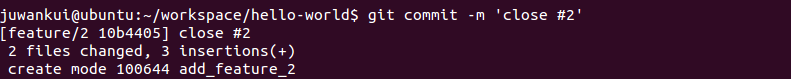
5. 基于本地develop创建你自己的开发分支

此处再次重申，创建你自己的开发分支一定要以develop为基础。   
我们通过git checkout -b feature/2来创建属于我们自己的第一个开发分支（develop开发分支属于大家的），其中“feature”一般为约定俗成的名字,“2”为项目组为你分配的Issues号码，如下图，暂时我们不管Issues操作，只需要知道你任务分支的编码就可。   
  
这里写图片描述

6. 进行你该干的事情吧

真正属于我们的开发分支已经创建完了，现在你可以为所欲为的修改你的代码了。

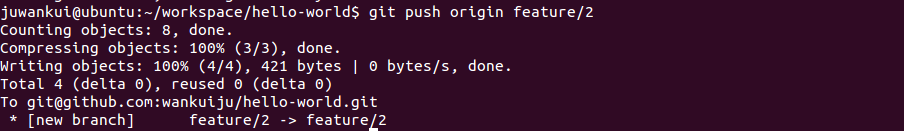
7. 将你修改的内容提交到本地版本库

(1) 新增或修改完内容后通过git status查看一下本地版本库状态   
  
(2) 按照提示,执行git add . (不要忘记后面的”.”哦) 添加当前工作目录文件到暂存区，如果想从暂存区回退到工作目录，可以使用git reset，参考下图   
  
(3) 执行git commit –m ‘close #2’命令将暂存区内容添加到本地仓库，close #2为约定俗成的，即关闭我们当前分支的任务，这只是一个说明代表我们完成了该分支，如果关闭需要进行其他动作，在这，我需要说明一下commit提交方式，git commit -m ‘close #your branch number’①每次使用该语句都会在本地生成一份commit信息，可以通过git log 查看， git commit –amend –no-edit②使用该语句不会创建新的commit信息，会在上一个commit基础上进行覆盖提交，当然，这些commit信息会随着后续push操作，将commit信息随着版本分支一同上传到远程repository，所以要根据自身项目情况决定（通常是第一次commit执行①，以后commit执行②）：   


8. 再次保证你当前分支版本为最新的

通过执行git fetch -p（从远程仓库中拉取所有分支信息到本地，-p就会在本地删除远程已经删除的分支），git rebase origin/develop（将拉取的origin/develop的内容合并到本地当前分支），这样就保证当前分支同步于远程develop分支。

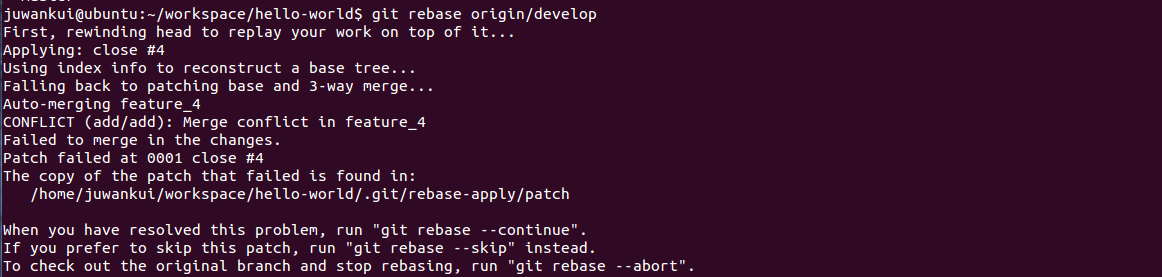
9. 将你的分支提交到远程仓库

通过执行git push origin feature/2将此分支提交到远程仓库，如果已经提交过，在保证你的版本库最新的情况下执行git push origin feature/2 –f   


10. 将远程仓库中你的分支合并到远程仓库的develop中（origin/develop）

一般由管理员或开发经理负责，统一在github上进行操作。

11. 解决冲突

如果远程仓库的develop分支和我们修改了同一个文件，在调用git rebase origin/develop   
命令时就会存在冲突的情况，如下图   
  
(1) 找到冲突文件，并删除冲突内容，主要是<<<<，==== ，>>>>三行的内容   
  
(2) 执行git add . 命令   
(3) 执行git rebase –continue

12. 如何用本地版本库文件覆盖工作目录文件

git checkout – bad\_filename 可以指定分支，默认为当前分支

13. 将commit到本地版本库的内容撤销

通过git log找到要退回的sha（commit后面的一长串字符），记住一定是要退回的sha，不是当前的sha，执行git reset sha（commit后面的一长串字符）

顶

# 3、静态web网页设计

# 4、div+css

# 5、js

# 6、php

对于只有 PHP 代码的文件，结束标志(“?>”)最好不存在，PHP自身并不需要结束符号，不添加结束符号可以很大程度上防止末尾被添加额外的注入内容，让程序更加安全。

这个就比较有意思了，作为php的开篇语，居然是一句和大多数介绍php书籍相悖的概念，其实就是这样，很多东西，都是在以后的更新中不断的认知的，而且像php这种解释型语言，本身就不像编译和预编译那样严格。

这个php教程不是一个严格教程，但会就一些常用问题，进行特别说明。

1、函数调用方法

require include

require\_once include\_once

include 和 require 区别 ( 同时 include\_once 和 require\_once 区别)

相同的地方就是，都是可以引入别的页面.

不同的地方是: include 如果出现了错误,会继续执行，require 出现错误，则会终止程序.

证明:

小结: 我们应当使用哪个?

我们做项目的时候，基本上使用的 require\_once

☞ 注意我们的requir\_once / require ... 应当放在 php 最前面.

php\_uname()函数

1. 说明

php\_uname — 返回运行 PHP 的系统的有关信息。

原型：string php\_uname ([ string $mode = “a” ] )。

返回运行php的操作系统的相关描述，和 phpinfo() 最顶端上输出的是同一个字符串。 如果仅仅要获取操作系统的名称。可以考虑使用常量 PHP\_OS，不过要注意该常量会包含 PHP 构建（built）时的操作系统名。

2. 参数

mode 是单个字符，用于定义要返回什么信息：

‘a’：此为默认。包含序列 “s n r v m” 里的所有模式。

‘s’：操作系统名称。例如： FreeBSD。

‘n’：主机名。例如： localhost.example.com。

‘r’：版本名称，例如： 5.1.2-RELEASE。

‘v’：版本信息。操作系统之间有很大的不同。

‘m’：机器类型。例如：i386。

3. 示例

<?php

$sys\_info = php\_uname() . "\n";

if(strtolower(substr($sys\_info,0,5)) == "linux") {

echo "this is a linux system.\n";

}

echo "Host: " . php\_uname('n') . "\n";

?>

## 1、换行符

浏览器识别不了\n或\r\n，这两个换行符是文本换行符，文本文件有效；在网页中查看HTML源代码可以发现代码b成功实现了换行

总结：如果需要将结果输出到浏览器或打印到显示器，代码中使用</br>;如果只是在源代码中换行，则使用\n或\r\n

<?php

echo'hello</br>';

echo'world!';

?>

<?php

echo'hello\n';//unix系统使用\n；windows系统下\r\n

echo'world!';

?>

前一段代码浏览器中成功实现了换行，后面的代码浏览器中未能识别，所以如果要在浏览器中换行，只能使用</br>,如果只是在程序中使用换行，那么可以使用\n或者\r\n。

# 7、mysql

1、sql语句

这个sql语句本身是一种标准化的语言，就是数据的操作语言，但由于不同数据库在处理字符的方式上的不同，导致sql语言并不通用，也就是说会有一些不同，总的来说是一样的，但会有少许不同。

由于我们使用的是mysql数据库，所以这里说的的都是mysql数据库支持的sql语句，下面看一下它们有什么不同，

/\*创建数据库\*/

CREATE DATABASE IF NOT EXISTS `test`;

/\*选择数据库\*/

USE `test`;

/\*创建表\*/

CREATE TABLE IF NOT EXISTS `user` (name varchar(50), age int);

/\*插入测试数据\*/

INSERT INTO `user` (name, age) VALUES('harry', 20), ('tony', 23), ('harry', 24);

这是一段标准的sql创建数据库和表的操作，但它在mysql数据库中却不能被执行，会报错，原因就是字符格式的问题。修改成如下就可以执行了：

/\*创建数据库\*/

CREATE DATABASE IF NOT EXISTS test;

/\*选择数据库\*/

USE test ;

/\*创建表\*/

CREATE TABLE IF NOT EXISTS user (name varchar(50), age int);

/\*插入测试数据\*/

INSERT INTO user (name, age) VALUES('harry', 20), ('tony', 23), ('harry', 24);

不仔细观察，你会发现没啥区别，实际上我们把数据库名称和表名称的单引号都去掉了，然后它就不会报错了。

# 8、github

写代码的就不用说了，不过，既然代码可以多人共同进行版本控制，其实写文档不是一样吗，只不过，就是文档的格式不是二进制而已，不过用来做更新中转还是不错的注意。

经常在家或者单位写同一个文档，这时就有一个版本问题，如何能自动同步呢，github是一个不错的选择，下面就github的一些用法做一个简单的介绍吧。

这个工具是linux的创始人开发的，主要是应用在linux内核的版本控制上，其功能和易用性上得到大家的认可，现在也是主要的版本控制软件之一。

言归正传，github的工作模式有3种，有网页版、客户端和命令行。

这几种方式大同小异，相比之下客户端可能是最方便的，但作为教程，这几种方式都会简单介绍一下：

先来说一下网页版的吧，这个应该是功能最全的的，包括仓库的创建，克隆仓库，同步和下载仓库，以及删除仓库等。