大纲：

1.前言

2.创建分支

3.切换分支

4.合并分支（快速合并）

5.删除分支

6.分支合并冲突

7.合并分支（普通合并）

8.分支管理策略

9.团队多人开发协作

10.总结

注，测试机 [CentOS](http://www.linuxidc.com/topicnews.aspx?tid=14) 5.5 x86\_64，Git 服务器版本：git version 1.8.2.1，客户端版本：git version 1.9.2.msysgit.0。所有软件请到这里下载：<http://msysgit.github.io/>。

1.前言

在上一篇博客中我们主要讲解了Git 远程仓库，相信大家对远程的Git仓库有一定的了解，嘿嘿。在这一篇博客中我们来在大家讲解一下Git 分支管理，这可以说是Git的又一大特点。下面我们就来学习一下Git分支管理吧。

我们先来说一个简单的案例吧，你们团队中有多个人再开发一下项目，一同事再开发一个新的功能，需要一周时间完成，他写了其中的30%还没有写完，如 果他提交了这个版本，那么团队中的其它人就不能继续开发了。但是等到他全部写完再全部提交，大家又看不到他的开发进度，也不能继续干活，这如何是好呢？

对于上面的这个问题，我们就可以用分支管理的办法来解决，一同事开发新功能他可以创建一个属于他自己的分支，其它同事暂时看不到，继续在开发分支 （一般都有多个分支）上干活，他在自己的分支上干活，等他全部开发完成，再一次性的合并到开发分支上，这样我们既可知道他的开发进度，又不影响大家干活， 是不是很方便呢？

大家可能会说了，你Git的分支功能人家SVN也有啊，也没什么特殊的嘛。但我想说你那个创建和切换速度怎么样呢？嘿嘿，我想说谁用谁知道啊！但Git呢，无论你创建还是切换或者删除都很快哦！

分支本质上其实就是一个指向某次提交的可变指针。Git 的默认分支名字为 master 。而我们是怎么知道当前处于哪个分支当中呢？答案就是在于 HEAD 这个十分特殊的指针，它专门用于指向于本地分支中的当前分支。我们可以简单理解为：commit <- branch <- HEAD （注，本来我们详细说一下HEAD的，这个东西真不好说，goole了一下也没几个大神说这个，嘿嘿。下面我就通过讲解帮助大家理解，简单的说HEAD就 是指向于本地分支中的当前分支，如下图：） 下面我们来创建分支。

**GitHub 教程系列文章**：

GitHub 使用教程图文详解  <http://www.linuxidc.com/Linux/2014-09/106230.htm>

Git 标签管理详解 <http://www.linuxidc.com/Linux/2014-09/106231.htm>

Git 分支管理详解 <http://www.linuxidc.com/Linux/2014-09/106232.htm>

Git 远程仓库详解 <http://www.linuxidc.com/Linux/2014-09/106233.htm>

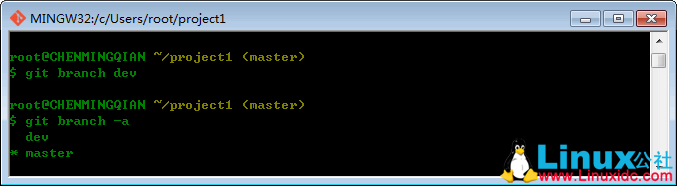
Git 本地仓库（Repository）详解 <http://www.linuxidc.com/Linux/2014-09/106234.htm>

Git 服务器搭建与客户端安装  <http://www.linuxidc.com/Linux/2014-05/101830.htm>

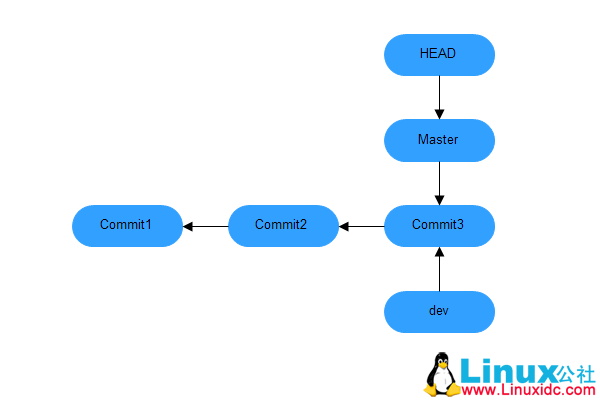
Git 概述 <http://www.linuxidc.com/Linux/2014-05/101829.htm>

2.创建分支

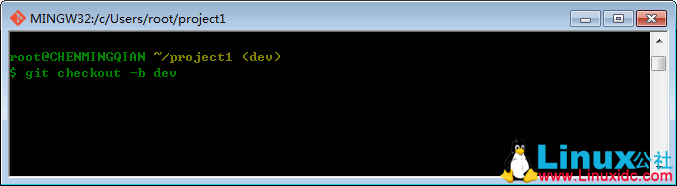
当我们需要调试某个Bug或者尝试添加或修改程序中的某个模块，而又不能影响主分支的开发时。就可以通过创建分支来满足需求。创建分支相当于是创建一个新的分支指针指向当前所在的提交。我们在Commit3上创建dev分支：



如下图所示，dev分支指向Commit3。



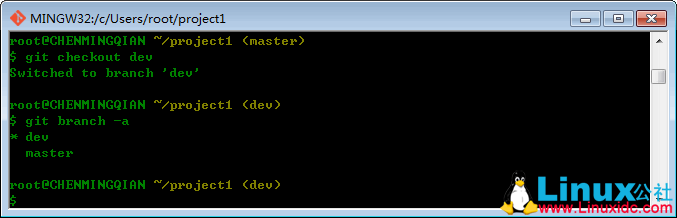
从上图可知，虽然我们创建了一个新分支，但是 HEAD 仍然指向 master 。如果希望在创建分支的同时切换到新分支上，我们可以通过以下命令实现：



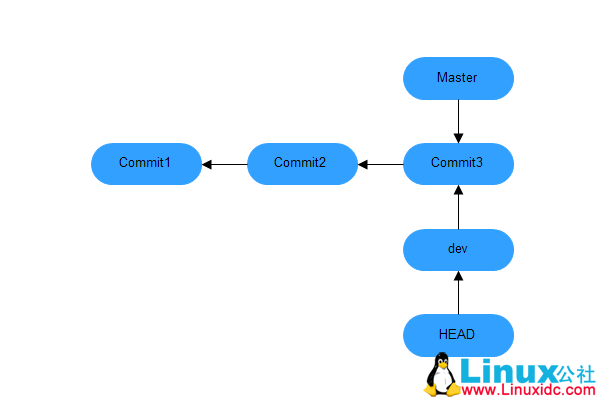
git checkout命令加上-b参数表示创建并切换分支上。

3.切换分支

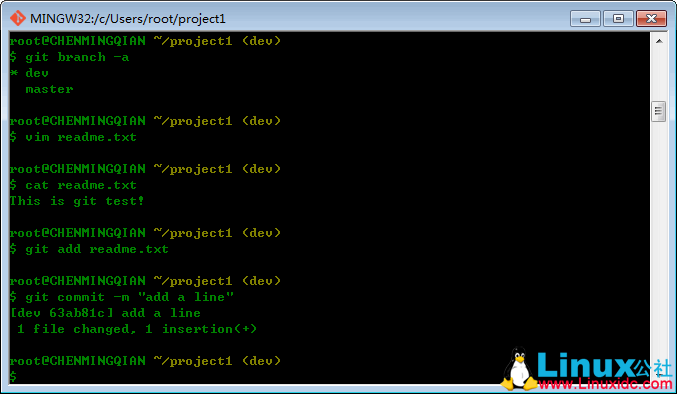
切换当前分支我们可以用以下命令实现：



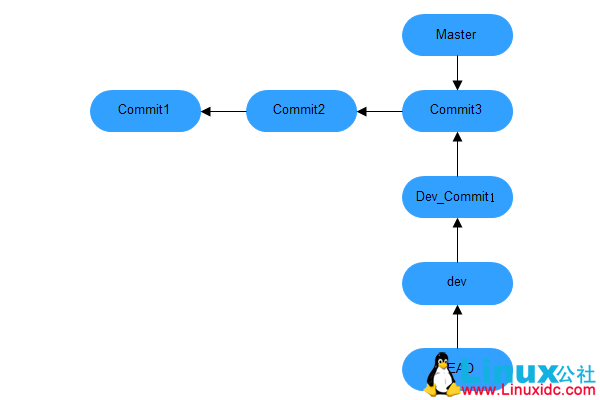
git branch -a 命令可以查看所有分支，现在我们HEAD指针便指向dev分支，大家可以在上图中看到dev分支上有个\*号。



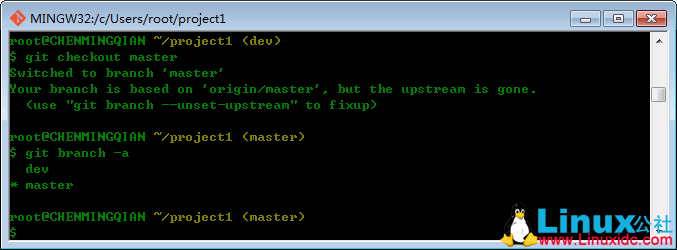
下面我们修改一下readme.txt中的内容，并在dev分支上提交一下。如下图：



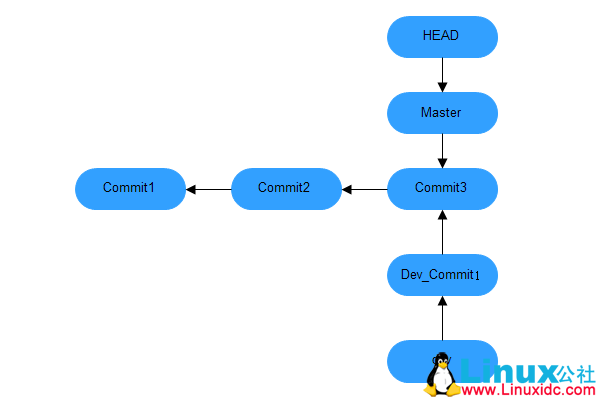
用流程图演示上述过程如下：



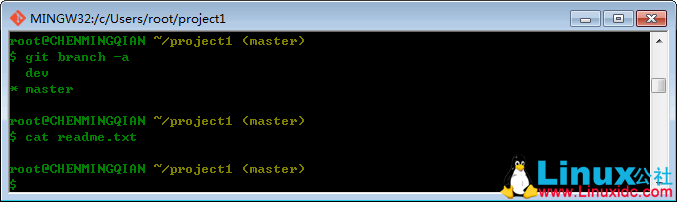
现在我们在dev分支上完成工作，现在到master分支上。如下图：



流程图表示如下：



切换回master分支后，再查看一个readme.txt文件，如下图：



大家可以看到我们刚才增加的一行内容不见了，嘿嘿。因为那个提交是在dev分支上，而master分支没有变化。好了，下面我们来演示一下合并分支。