

开源软件之——git 学习篇

主讲人 朱亚菲

幻灯片制作 赵海伟、朱亚菲

CVBIOUC

<http://vision.ouc.edu.cn/~zhenghaiyong>

2014 年 9 月 19 日

Contents

① Git 简介

- Git 是什么?
- Git 的诞生

② Git 使用

- Github
- My Project
- 分支管理

Agenda

1 Git 简介

- Git 是什么?
- Git 的诞生

2 Git 使用

- Github
- My Project
- 分支管理

Agenda

1 Git 简介

- Git 是什么?
- Git 的诞生

2 Git 使用

- Github
- My Project
- 分支管理

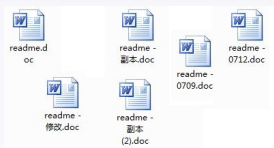


Git 是目前世界上最先进的分布式版本控制系统!

Git

Git 是目前世界上最先进的分布式版本控制系统!

版本控制

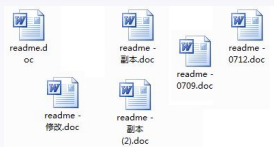


版本	用户	说明	日期
1	张三	删除了实验室规则说明 5	7/12 10:38
2	李四	加入一幅图片	7/12 18:09
3	张三	修改了 README.md	7/13 9:51
4	李四	新增一种绘图颜色	7/14 15:17

Git

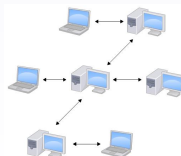
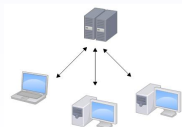
Git 是目前世界上最先进的分布式版本控制系统!

版本控制



版本	用户	说明	日期
1	张三	删除了实验室规则说明 5	7/12 10:38
2	李四	加入一幅图片	7/12 18:09
3	张三	修改了 README.md	7/13 9:51
4	李四	新增一种绘图颜色	7/14 15:17

集中式 vs 分布式



Agenda

1 Git 简介

- Git 是什么?
- Git 的诞生

2 Git 使用

- Github
- My Project
- 分支管理

- 1991-2002 年间，世界各地的志愿者把源代码文件通过 diff 的方式发给 Linus，然后由 Linus 本人通过手工方式合并代码！
- 到 2002 年，整个项目组开始启用分布式版本控制系统 BitKeeper 来管理和维护代码。
- 到了 2005 年，开发 BitKeeper 的商业公司同 Linux 内核开源社区的合作关系结束，他们收回了免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区不得不吸取教训，只有开发一套属于自己的版本控制系统才不至于重蹈覆辙。
- Linus 花了两周时间自己用 C 写了一个分布式版本控制系统，这就是 Git！一个月之内，Linux 系统的源码已经由 Git 管理了！

Agenda

1 Git 简介

- Git 是什么?
- Git 的诞生

2 Git 使用

- Github
- My Project
- 分支管理

使用 Git 前的准备工作

- 安装
- 设置

```
$git config --global user.name "Your Name"
```

```
$git config --global user.email "email@example.com"
```

- 创建版本库

```
[16:20]fei@CVBI:~/repository[0]  
$mkdir LearnGit  
  
[16:20]fei@CVBI:~/repository[0]  
$cd LearnGit/  
  
[16:20]fei@CVBI:~/repository/LearnGit[0]  
$git init  
初始化空的 Git 版本库于 /home/fei/repository/LearnGit/.git/
```

查看某个文件的修改历史

\$git log --pretty=oneline 文件名

```
[14:33]fei@CVBI:~/repository/test[0]
$git log --pretty=oneline README.md
aa31cf4f2fc4b468241de7202c333059a00cb6cc conflict fixed
a22d1f72a6bd9b13a5bcc8c65efbac30d420dd72 & simple
d0d738b829d9ab1cb816fad544d572c2b8e48de AND simple
53be7f2bdf0ec0780234f26d6d21f0e06bb6f938 Add 修改
c125b7465eab1c335eacd7ee973fbf5158e352f3 hh
755fd4f397934e0d0a76d591dafdda7c1ab981a4 jj
2b64fac008025782679f6ed9f5f78b8921eaf74e Update README.md
2e312da51059aad2b2c97e95ec77af028e516f54 Update README.md
d4e34909e1b65a928503c2df875a45c5f46a4921 Update README.md
9bccae45c4426e9cf9cb9f966e9508d68dcefc update
125142ba4ef7192cb88df72a03886318a7622f13 Initial commit
```

\$git show aa31c

```
[14:34]fei@CVBI:~/repository/test[0]
$git show aa31c
commit aa31cf4f2fc4b468241de7202c333059a00cb6cc
Merge: a22d1f7 d0d738b
Author: zhuyafei <zhuyafei4520@163.com>
Date: Wed Sep 17 15:13:19 2014 +0800

    conflict fixed

diff --cc README.md
index 7f832ec,bd4ed88..eb02556
--- a/README.md
+++ b/README.md
@@@ -1,3 -1,3 +1,3 @@@
 修改hhhhhh
  hehe attempt
- Creating a new branch is quick & simple.
- Creating a new branch is quick AND simple.
++Creating a new branch is quick and simple.
```

Agenda

1 Git 简介

- Git 是什么?
- Git 的诞生

2 Git 使用

- Github
- My Project
- 分支管理

Git 的杀手级功能之一：远程仓库

服务器

找一台电脑充当服务器的角色，每天 24 小时开机，其他每个人都从这个“服务器”仓库克隆一份到自己的电脑上，并且各自把各自的提交推送到服务器仓库里，也从服务器仓库中拉取别人的提交。

Github

Github 是一个共享虚拟主机服务，用于存放使用 Git 版本控制的软件代码和内容项目。

Agenda


1 Git 简介

- Git 是什么?
- Git 的诞生



2 Git 使用














- Github
- My Project
- 分支管理

My Project


branch: **master** ▼
Saliency-Benchmark-Frequency / +
☰

Save the GroundTruth as png format instead of jpg format, for there a... ...


zhuyafei authored 2 days ago
latest commit **a466916187** 

 Datasets	Save the GroundTruth as png format instead of jpg format, for there a...	2 days ago
 Models	Update dataset related code and directory structure for Models	a month ago
 Results	Remove prf.txt	3 days ago
 SaliencyMaps	Initial commit	a month ago
 .gitignore	Update .gitignore to ignore 'tmp' folder	a month ago
 README.md	Update README.md	a month ago
 SaliencyBenchmark.m	Change function name 'bianli' to 'traverse'	a month ago
 benchPR.m	Makesure the input SaliencyMap and GroundTruth be image format	4 days ago
 benchPRF.m	Save the results of benchPRF.m to files of the format '.txt'	4 days ago
 drawPR.m	Modify drawPR.m to preallocate the size of modelCell	3 days ago
 drawPRF.m	Change numModel to numMat	3 days ago
 prCount.m	Update for the binarization of the dataset groundtruth	a month ago
 prfCount.m	Update for the binarization of the dataset groundtruth	a month ago

README

📖 README.md

Saliency-Benchmark

State-of-the-art Spectral Saliency Detection Models on state-of-the-art Datasets.

Installation

1. In `Datasets/GroundTruth/` and `Datasets/Images/`, I only put two example images in each dataset folder. If you want to test on the real datasets, you need to replace them.
2. Run `./SaliencyBenchmark.m` to generate saliency maps for all the models on all the datasets, the results are saved in `./SaliencyMaps/`.
3. Run `./benchPR.m` to save Precision and Recall results in `./Results/pr/`.
4. Run `./benchPRF.m` to save Precision, Recall and F-measure results in `./Results/prf/`.
5. Run `./drawPR.m` to show pr curves on all the datasets.
6. Run `./drawPRF.m` to show histogram of Precision, Recall and F-measure results on all the datasets.

Agenda

1 Git 简介

- Git 是什么?
- Git 的诞生

2 Git 使用

- Github
- My Project
- 分支管理

Git 的杀手级功能之二：分支模型

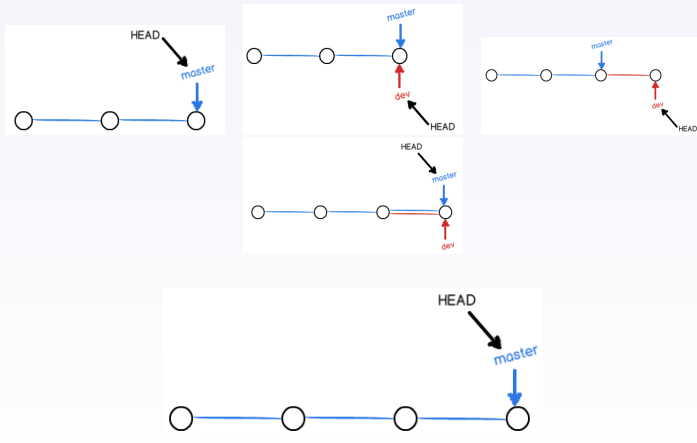
分支

每次提交，Git 都把它们串成一条时间线，这条时间线就是一个分支。

应用场景

假设你准备开发一个新功能，但是需要两周才能完成，如果代码还没写完就提交，不完整的代码库会导致别人无法工作，如果等代码全部写完再一次提交，又存在丢失每天进度的巨大风险。

创建与合并分支



HEAD

- HEAD 文件是一个指向你当前所在分支的引用标识符。

```
1  $cat .git/HEAD
2  ref: refs/heads/master
```

- 如果你执行 `git checkout dev`, Git 就会更新这个文件:

```
1  $cat .git/HEAD
2  ref: refs/heads/dev
```

创建与合并分支

- 查看分支: `git branch`
- 创建分支: `git branch name`
- 切换分支: `git checkout name`
- 创建 + 切换分支: `git checkout -b name`
- 合并某分支到当前分支: `git merge name`
- 删除分支: `git branch -d name`

分支管理策略

- master 分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面工作。
- dev 分支是不稳定的，平时在 dev 分支上工作，到某个时候，比如 1.0 版本发布时，再把 dev 分支合并到 master 上，在 master 分支发布 1.0 版本。
- 团队中的每个人都在 dev 分支上干活，每个人都有自己的分支，时不时地往 dev 分支上合并。

Bug 分支

应用场景

当你接到修复代号 100 的 bug 的任务时，很自然地，你想创建一个分支 issue-100 来修复它，但是当前正在 dev 分支上进行的工作还没有提交，并不是你不想提交，而是工作只进行到一半，还没法提交，预计完成还需 1 天时间，但是必须在两个小时内修复该 bug。

- 使用 git stash 命令把当前工作现场“储藏”起来，等以后恢复现场后继续工作。

Thanks

Yafei Zhu

Ocean University of China

2014.09