inspur

公司名称 | [公司地址]

Git Quick Manual

[文档副标题]

**说明**

**Doc. Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Ver.** | **Note** | **Modified By** | **Reviser** |
| 2016/7/3 | V0.1 | 新建，初置文件结构 |  |  |
| 2016/7/9 | V0.2 | 添加部分基本操作内容，添加第三章内容，主要涉及log |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

约定说明：

1、$ <**加粗**、*斜体*> 该部分内容为输入的命令

例：$ ***git add git-readme***

## 第一章 简介

### 一、版本控制

### 二、Git简介

### 三、Windows下安装Git

## 第二章 基本操作

### 一、基本配置

进行本地Git的基本设置，包括用户名、邮箱、字体、默认编辑器器的设置，其中用户名、邮箱是重要配置，用以区分库中文件的归属，其他配置可根据个人喜好选择。

命令如下：

git config [--global] user.name <username>

git config [--global] user.email <email>

git config --list ##列出所有Git的配置信息

### 二、创建Git库基本操作

1、创建并初始化Git本地仓库

$ ***mkdir learngit***

$ ***cd learngit/***

$ ***git init***

Initialized empty Git repository in E:/git-len/learngit/.git/

$ ***ls -al***

total 8

drwxr-xr-x 1 inspur 197121 0 7月 3 15:25 ./

drwxr-xr-x 1 inspur 197121 0 7月 3 15:25 ../

drwxr-xr-x 1 inspur 197121 0 7月 3 15:25 .git/

2、创建文件，提交Git控制

$ ***pwd***

/e/git-len/learngit

$ ***vim git-readme***

$ ***cat git-readme***

Git是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目.

$ ***ls -al***

total 12

drwxr-xr-x 1 inspur 197121 0 7月 3 15:35 ./

drwxr-xr-x 1 inspur 197121 0 7月 3 15:25 ../

drwxr-xr-x 1 inspur 197121 0 7月 3 15:25 .git/

-rw-r--r-- 1 inspur 197121 929 7月 3 15:35 git-readme

$ ***git add git-readme***

$ ***git commit -m "创建git-readme文件"***

[master (root-commit) 0ad9537] 创建git-readme文件

The file will have its original line endings in your working directory.

1 file changed, 3 insertions(+)

create mode 100644 git-readme

3、查看git提交日志

$ ***git log git-readme***

commit 0ad9537e99f528405ab11eaa07d99efe707cdc64

Author: inspur <251871598@qq.com>

Date: Sun Jul 3 15:37:56 2016 +0800

创建git-readme文件

4、版本库内的文件删除重命名

$ ***ls***

git-readme next-file oth-file

$ ***git rm oth-file***

rm 'oth-file'

$ ***git commit -m "remove oth-file"***

[master aed76e2] remove oth-file

1 file changed, 1 deletion(-)

delete mode 100644 oth-file

$ ***ls***

git-readme next-file

$ ***mv next-file mv\_next-file***

$ ***git rm next-file***

rm 'next-file'

$ ***git add mv\_next-file***

$ ***git commit -m "moved next-file to mv\_next-file"***

[master d7f21eb] moved next-file to mv\_next-file

1 file changed, 0 insertions(+), 0 deletions(-)

rename next-file => mv\_next-file (100%)

$ ***ls***

git-readme mv\_next-file

### 三、Git的标签

Git有两种基本的标签类型：

lightweight：轻量级的

annotatated：带标注的

inspur@inspur-PC MINGW64 /e/git-len/learngit (master)

$ ***git tag -a var1.0 -m "initial version"***

$ ***git tag***

var1.0

### 四、创建分支并修改提交

1、创建分支br01并切换到新分支上

$ ***git branch br01***

$ ***git branch***

br01

\* master

$ ***git checkout br01***

Switched to branch 'br01'

$ ***git branch***

\* br01

master

$ ***git status***

On branch br01

nothing to commit, working directory clean

2、向新分支提交修改

$ ***vim git-readme***

$ ***cat git-readme***

Git是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目.

你是否喜欢Git？

$ ***git add .***

$ ***git commit***

[br01 cd0d84d] ▒▒ӡ▒▒▒▒Ƿ▒ϲ▒▒Git▒▒▒▒▒▒▒▒▒һ▒У▒▒▒▒ύ▒▒br01▒▒֧

1 file changed, 1 insertion(+)

### 五、将分支的修改合并回主干

$ git checkout master

$ git merge br01

## 第三章 提交

### 一、原子变更集

### 二、识别提交

### 三、提交历史记录

1、基本log

$ git log git-readme

commit cd0d84dade2e854ff7fcbf9687da1acdbeef2ded

Author: inspur <251871598@qq.com>

Date: Sun Jul 3 15:54:44 2016 +0800

添加“你是否喜欢Git？”到最后一行，并提交到br01分支

git log 默认按提交时间列出所有的更新，最近的更新排在最上面。每次更新都有一个 SHA-1 校验和、作者的名字和电子邮件地址、提交时间，最后缩进一个段落显示提交说明

2、-n 仅显示最近n次的log

$ ***git log -1***

3、-p 选项展开显示每次提交的内容差异

$ ***git log -p -1***

commit d7f21eb36483318f2bf74fed8bf07b191ca60e8e

Author: inspur <251871598@qq.com>

Date: Sat Jul 9 15:31:03 2016 +0800

moved next-file to mv\_next-file

**diff --git a/mv\_next-file b/mv\_next-file**

**new file mode 100644**

**index 0000000..9c86ee2**

**--- /dev/null**

**+++ b/mv\_next-file**

@@ -0,0 +1 @@

+next file

**diff --git a/next-file b/next-file**

**deleted file mode 100644**

**index 9c86ee2..0000000**

**--- a/next-file**

**+++ /dev/null**

@@ -1 +0,0 @@

-next file

4、--stat仅显示简要的增删行数统计

$ ***git log --stat -1***

commit d7f21eb36483318f2bf74fed8bf07b191ca60e8e

Author: inspur <251871598@qq.com>

Date: Sat Jul 9 15:31:03 2016 +0800

moved next-file to mv\_next-file

mv\_next-file | 1 +

next-file | 1 -

2 files changed, 1 insertion(+), 1 deletion(-)

5、--pretty 按指定格式显示

--pretty=oneline 将每个提交放在一行显示

$ ***git log --pretty=oneline -3***

d7f21eb36483318f2bf74fed8bf07b191ca60e8e moved next-file to mv\_next-file

aed76e229e19a1a1d6a2fbe7171cec9da8786aa6 remove oth-file

987031c001eaa94a60f1c581810d9ac99a62cd5c Merge branch 'master' into br01

--pretty=short 显示哈希、作者、提交说明

--pretty=full 显示哈希、作者、提交者、提交说明

--pretty=fuller 显示哈希、作者、作者日期、提交者、提交者日期、提交说明

--pretty=format 定制要显示的格式

例：显示最近一次提交的简短哈希、作者、作者修改日期、提交说明

$ ***git log --pretty=format:"%h - %an, %ar : %s" -1***

d7f21eb - inspur, 3 hours ago : moved next-file to mv\_next-file

|  |  |
| --- | --- |
| --pretty=format的选项及说明 | |
| %ad | 作者修订日期（可以用 -date= 选项定制格式） |
| %ae | 作者的电子邮件地址 |
| %an | 作者（author）的名字 |
| %ar | 作者修订日期，按多久以前的方式显示 |
| %cd | 提交日期 |
| %ce | 提交者的电子邮件地址 |
| %cn | 提交者(committer)的名字 |
| %cr | 提交日期，按多久以前的方式显示 |
| %h | 提交对象的简短哈希字串 |
| %H | 提交对象的完整哈希字串 |
| %p | 父对象（parent）的简短哈希字串 |
| %P | 父对象（parent）的完整哈希字串 |
| %s | 提交说明 |
| %t | 树对象（tree）的简短哈希字串 |
| %T | 树对象（tree）的完整哈希字串 |

6、git log的其他参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -p | 按补丁的格式显示每个更新的差异 |
| --stat | 显示每次更新文件的修改统计信息 |
| --shortstat | 只显示—stat最后的行数修改添加移除统计 |
| --name-only | 仅在提交信息后显示已修改的文件清单 |
| --name-status | 显示新增、修改、删除的文件清单 |
| --abbrev-commit | 仅显示 SHA-1 的前几个字符 |
| --relative-date | 使用较短的相对时间显示 |
| --graph | 显示 ASCII 图形表示的分支合并历史 |
| --pretty | 使用其他格式显示历史提交信息 |
| -<n> | 仅显示最近的 n 条提交 |
| --since, --after | 仅显示指定时间之后的提交  git log --since="2008-10-01"  git log --since=2.weeks |
| --until, --before | 仅显示指定时间之前的提交  --since="2008-10-01"  --since=2.weeks |
| --author | 仅显示指定作者相关的提交 |
| --committer | 仅显示指定提交者相关的提交 |
| --reverse | 按照逆序显示提交记录 |
|  |  |

提交范围“开始..结束”定义为：从“结束”的提交可到达的和从“开始”提交不可达的一组提交。等价于：“^开始 结束”，例：git log ^X Y 等价于 git log X..Y

例：选择在master分支但不在dev、topic、bugfix分支上的提交

git log ^dev ^topic ^bugfix master

### 四、查找提交

1、使用git bisect

git bisect start

git bisect good

git bisect bad

git bisect log

git bisect reset

2、使用git blame

显示一个文件的每一行最后是由谁修改的和哪次提价做出的变更

git blame <filename>

3、使用Pickaxe

带有-S命令的git log命令称为pickaxe

git log –Sinclude <filename>

提交的内容中包含增加或删除了include

## 第四章 分支

### 一、使用分支的原因