1.从远程仓库克隆

首先，登录github，创建一个新的库，名字叫做gitskills

勾选Initialize this repository with a README

GitHub会自动为我们创建一个README.md文件。创建完毕后，可以看到README.md文件：

用命令git clone克隆一个本地库：

$ git clone git@github.com:michaelliao/gitskills.git

Cloning into 'gitskills'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3

Receiving objects: 100% (3/3), done.

把Git库的地址换成你自己的，然后进入gitskills目录看看，已经有README.md文件了。

从远程库克隆

阅读: 753302

上次我们讲了先有本地库，后有远程库的时候，如何关联远程库。

现在，假设我们从零开发，那么最好的方式是先创建远程库，然后，从远程库克隆。

首先，登陆GitHub，创建一个新的仓库，名字叫gitskills：

github-init-repo

我们勾选Initialize this repository with a README，这样GitHub会自动为我们创建一个README.md文件。创建完毕后，可以看到README.md文件：

github-init-repo-2

现在，远程库已经准备好了，下一步是用命令git clone克隆一个本地库：

$ git clone git@github.com:michaelliao/gitskills.git

Cloning into 'gitskills'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3

Receiving objects: 100% (3/3), done.

注意把Git库的地址换成你自己的，然后进入gitskills目录看看，已经有README.md文件了：

$ cd gitskills

$ ls

README.md

如果有多个人协作开发，那么每个人各自从远程克隆一份就可以了。

你也许还注意到，GitHub给出的地址不止一个，还可以用https://github.com/michaelliao/gitskills.git这样的地址。实际上，Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只能用https。

2.创建与合并分支

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

截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。

HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

master分支是一条线，Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点



每次提交，master分支都会向前移动一步，这样，随着你不断提交，master分支的线也越来越长

当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev上：



Git创建一个分支很快，因为除了增加一个dev指针，改改HEAD的指向，工作区的文件都没有任何变化！

不过，从现在开始，对工作区的修改和提交就是针对dev分支了，比如新提交一次后，dev指针往前移动一步，而master指针不变：



假如我们在dev上的工作完成了，就可以把dev合并到master上。

直接把master指向dev的当前提交，就完成了合并：



合并完分支后，甚至可以删除dev分支。删除dev分支就是把dev指针给删掉，删掉后，我们就剩下了一条master分支：



实战

首先，我们创建dev分支，然后切换到dev分支：

$ git checkout -b dev

Switched to a new branch 'dev'

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

$ git branch dev

$ git checkout dev

Switched to branch 'dev'

然后，用git branch命令查看当前分支：

$ git branch

\* dev

master

git branch命令会列出所有分支，当前分支前面会标一个\*号。

然后，我们就可以在dev分支上正常提交，比如对readme.txt做个修改，加上一行：

Creating a **new** branch is quick.

然后提交：

$ git add readme.txt

$ git commit -m "branch test"

[dev b17d20e] branch test

1 file changed, 1 insertion(+)

现在，dev分支的工作完成，我们就可以切换回master分支：

$ git checkout master

Switched to branch 'master'

切换回master分支后，再查看一个readme.txt文件，刚才添加的内容不见了！因为那个提交是在dev分支上，而master分支此刻的提交点并没有变：



把dev分支的工作成果合并到master分支上：

$ git merge dev

Updating d46f35e..b17d20e

Fast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

git merge命令用于合并指定分支到当前分支。合并后，再查看readme.txt的内容，就可以看到，和dev分支的最新提交是完全一样的。

注意到上面的Fast-forward信息，Git告诉我们，这次合并是“快进模式”，也就是直接把master指向dev的当前提交，所以合并速度非常快。

当然，也不是每次合并都能Fast-forward，我们后面会讲其他方式的合并。

合并完成后，就可以放心地删除dev分支了：

$ git branch -d dev

Deleted branch dev (was b17d20e).

删除后，查看branch，就只剩下master分支了：

$ git branch

\* master

因为创建、合并和删除分支非常快，所以Git鼓励你使用分支完成某个任务，合并后再删掉分支，这和直接在master分支上工作效果是一样的，但过程更安全。

Tips

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>[登录](https://aws.amazon.com/)

删除分支：git branch -d <name>

1.1

$ echo "hello, world!"

hello, world!

命令启动本地 Web 开发服务器：

$ rails server

与命令行提示符一样，本书也使用 Unix 惯用的目录分隔符（即斜线 /）。例如，演示应用的配置文件 production.rb，它的路径是：

config/environments/production.rb

上述文件路径相对于应用的根目录而言。在不同的系统中，根目录会有差别。在云端 IDE 中，根目录像下面这样：

/home/ec2-user/environment/sample\_app/

production.rb 文件的完整路径是：

/home/ec2-user/environment/sample\_app/config/environments/production.rb

通常省略应用的路径，简写成 config/environments/production.rb。

1.2搭建环境

先安装rvm

brew install gnupg

完成gnupg之后安装mpapis public keys

gpg --keyserver hkp://keys.gnupg.net

--recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3

1.3 第一个应用

创建环境文件夹

$ cd

$ mkdir environment

$ cd environment/

执行 rails new 命令（明确指定版本号）

$ cd ~/environment

$ rails \_5.1.6\_ new hello\_app

| Rails 目录结构简介 | |
| --- | --- |
| **文件/文件夹** | **作用** |
| app/ | 应用的核心文件，包含模型、视图、控制器和辅助方法 |
| app/assets | 应用的静态资源文件，例如层叠样式表（CSS）、JavaScript 文件和图像 |
| bin/ | 可执行的二进制文件 |
| config/ | 应用的配置 |
| db/ | 数据库文件 |
| doc/ | 应用的文档 |
| lib/ | 代码库模块文件 |
| lib/assets | 代码库的静态资源文件，例如层叠样式表（CSS）、JavaScript 文件和图像 |
| log/ | 应用的日志文件 |
| public/ | 公共（如浏览器）可访问的文件，例如错误页面 |
| bin/rails | 生成代码、打开终端会话或启动本地服务器的程序 |
| test/ | 应用的测试 |
| tmp/ | 临时文件 |
| vendor/ | 第三方代码，例如插件和 gem |
| vendor/assets | 第三方静态资源文件，例如层叠样式表（CSS）、JavaScript 文件和图像 |
| README.md | 应用简介 |
| Rakefile | 使用 rake 命令执行的实用任务 |
| Gemfile | 应用所需的 gem |
| Gemfile.lock | gem 列表，确保这个应用的副本使用相同版本的 gem |
| config.ru | [Rack 中间件](http://rack.github.io/)的配置文件 |
| .gitignore | Git 忽略的文件模式 |

1.3bundler

1.4 hello\_app 目录中默认生成的 Gemfile 文件

source 'https://rubygems.org'

git\_source(:github) **do** **|**repo\_name**|**

repo\_name **=** "#{repo\_name}/#{repo\_name}" **unless** repo\_name.**include?**("/")

"https://github.com/#{repo\_name}.git"

**end**

*# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'*

gem 'rails', '~> 5.1.6'

*# Use sqlite3 as the database for Active Record*

gem 'sqlite3'

*# Use Puma as the app server*

gem 'puma', '~> 3.7'

*# Use SCSS for stylesheets*

gem 'sass-rails', '~> 5.0'

*# Use Uglifier as compressor for JavaScript assets*

gem 'uglifier', '>= 1.3.0'

*# See https://github.com/rails/execjs#readme for more supported runtimes*

*# gem 'therubyracer', platforms: :ruby*

*# Use CoffeeScript for .coffee assets and views*

gem 'coffee-rails', '~> 4.2'

*# Turbolinks makes navigating your web application faster.*

*# Read more: https://github.com/turbolinks/turbolinks*

gem 'turbolinks', '~> 5'

*# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder*

gem 'jbuilder', '~> 2.5'

*# Use Redis adapter to run Action Cable in production*

*# gem 'redis', '~> 3.0'*

*# Use ActiveModel has\_secure\_password*

*# gem 'bcrypt', '~> 3.1.7'*

*# Use Capistrano for deployment*

*# gem 'capistrano-rails', group: :development*

group :development, :test **do**

*# Call 'byebug' anywhere in the code to stop execution*

*# and get a debugger console*

gem 'byebug', platforms: [:mri, :mingw, :x64\_mingw]

*# Adds support for Capybara system testing and selenium driver*

gem 'capybara', '~> 2.13'

gem 'selenium-webdriver'

**end**

group :development **do**

*# Access an IRB console on exception pages or by using*

*# <%= console %> anywhere in the code.*

gem 'web-console'

gem 'listen', '>= 3.0.5', '< 3.2'

*# Spring speeds up development by keeping your application running*

*# in the background. Read more: https://github.com/rails/spring*

gem 'spring'

gem 'spring-watcher-listen', '~> 2.0.0'

**end**

*# Windows does not include zoneinfo files, so bundle the tzinfo-data gem*

gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64\_mingw, :jruby]

##### 1.5在 Gemfile 文件中为所有 gem 指定精确的版本号

source 'https://rubygems.org'

gem 'rails', '5.1.6'

gem 'puma', '3.9.1'

gem 'sass-rails', '5.0.6'

gem 'uglifier', '3.2.0'

gem 'coffee-rails', '4.2.2'

gem 'jquery-rails', '4.3.1'

gem 'turbolinks', '5.0.1'

gem 'jbuilder', '2.7.0'

group :development, :test **do**

gem 'sqlite3', '1.3.13'

gem 'byebug', '9.0.6', platform: :mri

**end**

group :development **do**

gem 'web-console', '3.5.1'

gem 'listen', '3.1.5'

gem 'spring', '2.0.2'

gem 'spring-watcher-listen', '2.0.1'

**end**

*# Windows does not include zoneinfo files, so bundle the tzinfo-data gem*

gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64\_mingw, :jruby]

写入应用的 Gemfile 文件之后，执行 bundle install 命令[[8](https://railstutorial-china.org/book/chapter1.html#fn-8)]安装这些 gem：

$ cd hello\_app/

$ bundle install

Fetching source index **for** https://rubygems.org/

2.1规划应用