目录

[1、Git 教程 3](#_Toc109396240)

[1.1 Git 与 SVN 区别 3](#_Toc109396241)

[1.2 Git 快速入门 4](#_Toc109396242)

[2、Git 安装配置 5](#_Toc109396243)

[2.1 Linux 平台上安装 5](#_Toc109396244)

[2.1.1 Debian/Ubuntu 5](#_Toc109396245)

[2.1.2 Centos/RedHat 5](#_Toc109396246)

[2.1.3 源码安装 6](#_Toc109396247)

[2.2 Windows 平台上安装 6](#_Toc109396248)

[2.3 Mac 平台上安装 7](#_Toc109396249)

[2.4 Git 配置 8](#_Toc109396250)

[2.5 用户信息 9](#_Toc109396251)

[2.6文本编辑器 9](#_Toc109396252)

[2.7 差异分析工具 9](#_Toc109396253)

[2.8查看配置信息 9](#_Toc109396254)

[3、Git 工作流程 11](#_Toc109396255)

[4、Git 工作区、暂存区和版本库 13](#_Toc109396256)

[5、Git 创建仓库 15](#_Toc109396257)

[5.1 git init 15](#_Toc109396258)

[5．1.1使用方法 15](#_Toc109396259)

[5.2 git clone 16](#_Toc109396260)

[5.3 配置 16](#_Toc109396261)

[6、Git 基本操作 18](#_Toc109396262)

[6.1 提交与修改 18](#_Toc109396263)

[6.1.1提交日志 19](#_Toc109396264)

[6.1.2远程操作 19](#_Toc109396265)

[Git 分支管理 19](#_Toc109396266)

[7、Git 分支管理 21](#_Toc109396267)

[7.1 列出分支 21](#_Toc109396268)

[7.2 删除分支 23](#_Toc109396269)

[7.3 分支合并 23](#_Toc109396270)

[7.4 合并冲突 24](#_Toc109396271)

[8、Git 查看提交历史 27](#_Toc109396272)

[8.1 git log 27](#_Toc109396273)

[8.2 git blame 29](#_Toc109396274)

[9、Git 标签 30](#_Toc109396275)

[10、Git 远程仓库(Github) 32](#_Toc109396276)

[10.1 添加远程库 32](#_Toc109396277)

[10.2 查看当前的远程库 37](#_Toc109396278)

[10.2.1实例 37](#_Toc109396279)

[10.3 提取远程仓库 38](#_Toc109396280)

[10.4 推送到远程仓库 39](#_Toc109396281)

[10.5 删除远程仓库 40](#_Toc109396282)

[10.5.1 实例 40](#_Toc109396283)

[11、Git Gitee 42](#_Toc109396284)

[12、Git 服务器搭建 49](#_Toc109396285)

[12.1 安装Git 49](#_Toc109396286)

[12.2 创建证书登录 49](#_Toc109396287)

[12.3 初始化Git仓库 49](#_Toc109396288)

[12.4 克隆仓库 50](#_Toc109396289)

[13、Sourcetree 使用教程 51](#_Toc109396290)

[13.1 Sourcetree 安装 51](#_Toc109396291)

[13.2 连接 Github 账户 53](#_Toc109396292)

[13.3 创建本地仓库 56](#_Toc109396293)

[14、Git 测验 59](#_Toc109396294)

[14.1 开始 https://www.runoob.com/git/git-quiz.html# 59](#_Toc109396295)

[14.2 其他相关测试 59](#_Toc109396296)

1、Git 教程

Git 是一个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。

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

Git 与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

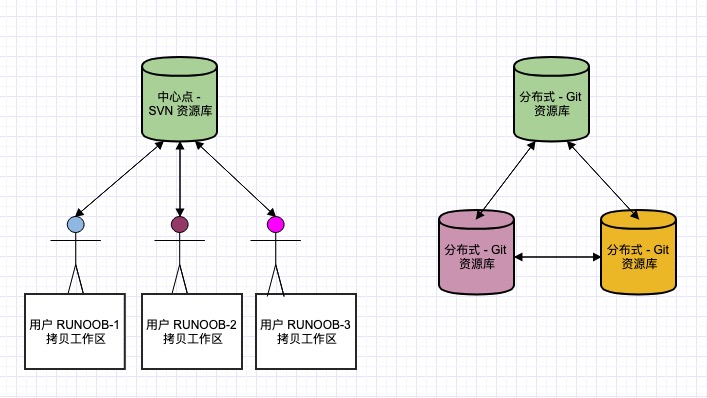
1.1 Git 与 SVN 区别

Git 不仅仅是个版本控制系统，它也是个内容管理系统(CMS)，工作管理系统等。

如果你是一个具有使用 SVN 背景的人，你需要做一定的思想转换，来适应 Git 提供的一些概念和特征。

Git 与 SVN 区别点：

* **1、Git 是分布式的，SVN 不是**：这是 Git 和其它非分布式的版本控制系统，例如 SVN，CVS 等，最核心的区别。
* **2、Git 把内容按元数据方式存储，而 SVN 是按文件：**所有的资源控制系统都是把文件的元信息隐藏在一个类似 .svn、.cvs 等的文件夹里。
* **3、Git 分支和 SVN 的分支不同：**分支在 SVN 中一点都不特别，其实它就是版本库中的另外一个目录。
* **4、Git 没有一个全局的版本号，而 SVN 有：**目前为止这是跟 SVN 相比 Git 缺少的最大的一个特征。
* **5、Git 的内容完整性要优于 SVN：**Git 的内容存储使用的是 SHA-1 哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。



1.2 Git 快速入门

本站也提供了 Git 快速入门版本，你可以点击 [Git 简明指南](https://www.runoob.com/manual/git-guide/) 查看。

入门后建议通过本站详细学习 Git 教程。

Git 完整命令手册地址：<http://git-scm.com/docs>

PDF 版命令手册：[github-git-cheat-sheet.pdf](https://www.runoob.com/manual/github-git-cheat-sheet.pdf)

# 2、Git 安装配置

在使用Git前我们需要先安装 Git。Git 目前支持 Linux/Unix、Solaris、Mac和 Windows 平台上运行。

Git 各平台安装包下载地址为：<http://git-scm.com/downloads>

## 2.1 Linux 平台上安装

Git 的工作需要调用 curl，zlib，openssl，expat，libiconv 等库的代码，所以需要先安装这些依赖工具。

在有 yum 的系统上（比如 Fedora）或者有 apt-get 的系统上（比如 Debian 体系），可以用下面的命令安装：

各 Linux 系统可以使用其安装包管理工具（apt-get、yum 等）进行安装：

### 2.1.1 Debian/Ubuntu

Debian/Ubuntu Git 安装命令为：

$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \

libz-dev libssl-dev

$ apt-get install git

$ git --version

git version 1.8.1.2

### 2.1.2 Centos/RedHat

如果你使用的系统是 Centos/RedHat 安装命令为：

$ yum install curl-devel expat-devel gettext-devel \

openssl-devel zlib-devel

$ yum -y install git-core

$ git --version

git version 1.7.1

### 2.1.3 源码安装

我们也可以在官网下载源码包来安装，最新源码包下载地址：<https://git-scm.com/download>

安装指定系统的依赖包：

########## Centos/RedHat ##########

$ yum install curl-devel expat-devel gettext-devel \

openssl-devel zlib-devel

########## Debian/Ubuntu ##########

$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \

libz-dev libssl-dev

解压安装下载的源码包：

$ tar -zxf git-1.7.2.2.tar.gz

$ cd git-1.7.2.2

$ make prefix=/usr/local all

$ sudo make prefix=/usr/local install

## 2.2 Windows 平台上安装

在 Windows 平台上安装 Git 同样轻松，有个叫做 msysGit 的项目提供了安装包，可以到 GitHub 的页面上下载 exe 安装文件并运行：

安装包下载地址：<https://gitforwindows.org/>

官网慢，可以用国内的镜像：<https://npm.taobao.org/mirrors/git-for-windows/>。

图形用户界面, 文本, 应用程序, Word

描述已自动生成

完成安装之后，就可以使用命令行的 git 工具（已经自带了 ssh 客户端）了，另外还有一个图形界面的 Git 项目管理工具。

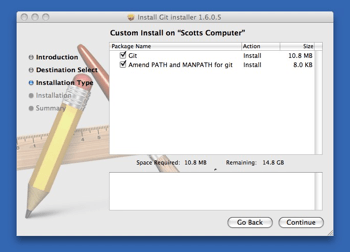
在开始菜单里找到"Git"->"Git Bash"，会弹出 Git 命令窗口，你可以在该窗口进行 Git 操作。

## 2.3 Mac 平台上安装

在 Mac 平台上安装 Git 最容易的当属使用图形化的 Git 安装工具，下载地址为：

<http://sourceforge.net/projects/git-osx-installer/>

安装界面如下所示：



## 2.4 Git 配置

Git 提供了一个叫做 git config 的工具，专门用来配置或读取相应的工作环境变量。

这些环境变量，决定了 Git 在各个环节的具体工作方式和行为。这些变量可以存放在以下三个不同的地方：

* /etc/gitconfig 文件：系统中对所有用户都普遍适用的配置。若使用 git config 时用 --system 选项，读写的就是这个文件。
* ~/.gitconfig 文件：用户目录下的配置文件只适用于该用户。若使用 git config 时用 --global 选项，读写的就是这个文件。
* 当前项目的 Git 目录中的配置文件（也就是工作目录中的 .git/config 文件）：这里的配置仅仅针对当前项目有效。每一个级别的配置都会覆盖上层的相同配置，所以 .git/config 里的配置会覆盖 /etc/gitconfig 中的同名变量。

在 Windows 系统上，Git 会找寻用户主目录下的 .gitconfig 文件。主目录即 $HOME 变量指定的目录，一般都是 C:\Documents and Settings\$USER。

此外，Git 还会尝试找寻 /etc/gitconfig 文件，只不过看当初 Git 装在什么目录，就以此作为根目录来定位。

### 2.5 用户信息

配置个人的用户名称和电子邮件地址：

$ git config --global user.name "runoob"

$ git config --global user.email test@runoob.com

如果用了 **--global** 选项，那么更改的配置文件就是位于你用户主目录下的那个，以后你所有的项目都会默认使用这里配置的用户信息。

如果要在某个特定的项目中使用其他名字或者电邮，只要去掉 --global 选项重新配置即可，新的设定保存在当前项目的 .git/config 文件里。

### 2.6文本编辑器

设置Git默认使用的文本编辑器, 一般可能会是 Vi 或者 Vim。如果你有其他偏好，比如 Emacs 的话，可以重新设置：:

$ git config --global core.editor emacs

### 2.7 差异分析工具

还有一个比较常用的是，在解决合并冲突时使用哪种差异分析工具。比如要改用 vimdiff 的话：

$ git config --global merge.tool vimdiff

Git 可以理解 kdiff3，tkdiff，meld，xxdiff，emerge，vimdiff，gvimdiff，ecmerge，和 opendiff 等合并工具的输出信息。

当然，你也可以指定使用自己开发的工具，具体怎么做可以参阅第七章。

### 2.8查看配置信息

要检查已有的配置信息，可以使用 git config --list 命令：

$ git config --list

http.postbuffer=2M

user.name=runoob

user.email=test@runoob.com

有时候会看到重复的变量名，那就说明它们来自不同的配置文件（比如 /etc/gitconfig 和 ~/.gitconfig），不过最终 Git 实际采用的是最后一个。

这些配置我们也可以在 **~/.gitconfig** 或 **/etc/gitconfig** 看到，如下所示：

vim ~/.gitconfig

显示内容如下所示：

[http]

postBuffer = 2M

[user]

name = runoob

email = test@runoob.com

也可以直接查阅某个环境变量的设定，只要把特定的名字跟在后面即可，像这样：

$ git config user.name

runoob

3、Git 工作流程

本章节我们将为大家介绍 Git 的工作流程。

一般工作流程如下：

* 克隆 Git 资源作为工作目录。
* 在克隆的资源上添加或修改文件。
* 如果其他人修改了，你可以更新资源。
* 在提交前查看修改。
* 提交修改。
* 在修改完成后，如果发现错误，可以撤回提交并再次修改并提交。

下图展示了 Git 的工作流程：



4、Git 工作区、暂存区和版本库

**4.1基本概念**

我们先来理解下 Git 工作区、暂存区和版本库概念：

* **工作区：**就是你在电脑里能看到的目录。
* **暂存区：**英文叫 stage 或 index。一般存放在 .git 目录下的 index 文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。
* **版本库：**工作区有一个隐藏目录 .git，这个不算工作区，而是 Git 的版本库。

下面这个图展示了工作区、版本库中的暂存区和版本库之间的关系：

图示

描述已自动生成

* 图中左侧为工作区，右侧为版本库。在版本库中标记为 "index" 的区域是暂存区（stage/index），标记为 "master" 的是 master 分支所代表的目录树。
* 图中我们可以看出此时 "HEAD" 实际是指向 master 分支的一个"游标"。所以图示的命令中出现 HEAD 的地方可以用 master 来替换。
* 图中的 objects 标识的区域为 Git 的对象库，实际位于 ".git/objects" 目录下，里面包含了创建的各种对象及内容。
* 当对工作区修改（或新增）的文件执行 git add 命令时，暂存区的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的ID被记录在暂存区的文件索引中。
* 当执行提交操作（git commit）时，暂存区的目录树写到版本库（对象库）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时暂存区的目录树。
* 当执行 git reset HEAD 命令时，暂存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。
* 当执行 git rm --cached <file> 命令时，会直接从暂存区删除文件，工作区则不做出改变。
* 当执行 git checkout . 或者 git checkout -- <file> 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区中的改动。
* 当执行 git checkout HEAD . 或者 git checkout HEAD <file> 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改动。

# 5、Git 创建仓库

本章节我们将为大家介绍如何创建一个 Git 仓库。

你可以使用一个已经存在的目录作为 Git 仓库。

## 5.1 git init

Git 使用 **git init** 命令来初始化一个 Git 仓库，Git 的很多命令都需要在 Git 的仓库中运行，所以 **git init** 是使用 Git 的第一个命令。

在执行完成 **git init** 命令后，Git 仓库会生成一个 .git 目录，该目录包含了资源的所有元数据，其他的项目目录保持不变。

### 5．1.1使用方法

使用当前目录作为 Git 仓库，我们只需使它初始化。

git init

该命令执行完后会在当前目录生成一个 .git 目录。

使用我们指定目录作为Git仓库。

git init newrepo

初始化后，会在 newrepo 目录下会出现一个名为 .git 的目录，所有 Git 需要的数据和资源都存放在这个目录中。

如果当前目录下有几个文件想要纳入版本控制，需要先用 git add 命令告诉 Git 开始对这些文件进行跟踪，然后提交：

$ git add \*.c

$ git add README

$ git commit -m '初始化项目版本'

以上命令将目录下以 .c 结尾及 README 文件提交到仓库中。

**注：** 在 Linux 系统中，commit 信息使用单引号 '，Windows 系统，commit 信息使用双引号 "。

所以在 git bash 中 git commit -m '提交说明' 这样是可以的，在 Windows 命令行中就要使用双引号 git commit -m "提交说明"。

## 5.2 git clone

我们使用 **git clone** 从现有 Git 仓库中拷贝项目（类似 **svn checkout**）。

克隆仓库的命令格式为：

git clone <repo>

如果我们需要克隆到指定的目录，可以使用以下命令格式：

git clone <repo> <directory>

**参数说明：**

* **repo:**Git 仓库。
* **directory:**本地目录。

比如，要克隆 Ruby 语言的 Git 代码仓库 Grit，可以用下面的命令：

$ git clone git://github.com/schacon/grit.git

执行该命令后，会在当前目录下创建一个名为grit的目录，其中包含一个 .git 的目录，用于保存下载下来的所有版本记录。

如果要自己定义要新建的项目目录名称，可以在上面的命令末尾指定新的名字：

$ git clone git://github.com/schacon/grit.git mygrit

## 5.3 配置

git 的设置使用 git config 命令。

显示当前的 git 配置信息：

$ git config --list

credential.helper=osxkeychain

core.repositoryformatversion=0

core.filemode=true

core.bare=false

core.logallrefupdates=true

core.ignorecase=true

core.precomposeunicode=true

编辑 git 配置文件:

$ git config -e # 针对当前仓库

或者：

$ git config -e --global # 针对系统上所有仓库

设置提交代码时的用户信息：

$ git config --global user.name "runoob"

$ git config --global user.email test@runoob.com

如果去掉 **--global** 参数只对当前仓库有效。

# 6、Git 基本操作

Git 的工作就是创建和保存你项目的快照及与之后的快照进行对比。

本章将对有关创建与提交你的项目快照的命令作介绍。

Git 常用的是以下 6 个命令：**git clone**、**git push**、**git add** 、**git commit**、**git checkout**、**git pull**，后面我们会详细介绍。



**说明：**

* workspace：工作区
* staging area：暂存区/缓存区
* local repository：版本库或本地仓库
* remote repository：远程仓库

一个简单的操作步骤：

$ git init

$ git add .

$ git commit

* git init - 初始化仓库。
* git add . - 添加文件到暂存区。
* git commit - 将暂存区内容添加到仓库中。 /6.创建仓库命令

下表列出了 git 创建仓库的命令：

|  |  |
| --- | --- |
| **命令** | **说明** |
| [git init](https://www.runoob.com/git/git-init.html) | 初始化仓库 |
| [git clone](https://www.runoob.com/git/git-clone.html) | 拷贝一份远程仓库，也就是下载一个项目。 |

## 6.1 提交与修改

Git 的工作就是创建和保存你的项目的快照及与之后的快照进行对比。

下表列出了有关创建与提交你的项目的快照的命令：

|  |  |
| --- | --- |
| **命令** | **说明** |
| [git add](https://www.runoob.com/git/git-add.html) | 添加文件到暂存区 |
| [git status](https://www.runoob.com/git/git-status.html) | 查看仓库当前的状态，显示有变更的文件。 |
| [git diff](https://www.runoob.com/git/git-diff.html) | 比较文件的不同，即暂存区和工作区的差异。 |
| [git commit](https://www.runoob.com/git/git-commit.html) | 提交暂存区到本地仓库。 |
| [git reset](https://www.runoob.com/git/git-reset.html) | 回退版本。 |
| [git rm](https://www.runoob.com/git/git-rm.html) | 将文件从暂存区和工作区中删除。 |
| [git mv](https://www.runoob.com/git/git-mv.html) | 移动或重命名工作区文件。 |

### 6.1.1提交日志

|  |  |
| --- | --- |
| **命令** | **说明** |
| [git log](https://www.runoob.com/git/git-commit-history.html#git-log) | 查看历史提交记录 |
| [git blame <file>](https://www.runoob.com/git/git-commit-history.html#git-blame) | 以列表形式查看指定文件的历史修改记录 |

### 6.1.2远程操作

|  |  |
| --- | --- |
| **命令** | **说明** |
| [git remote](https://www.runoob.com/git/git-remote.html) | 远程仓库操作 |
| [git fetch](https://www.runoob.com/git/git-fetch.html) | 从远程获取代码库 |
| [git pull](https://www.runoob.com/git/git-pull.html) | 下载远程代码并合并 |
| [git push](https://www.runoob.com/git/git-push.html) | 上传远程代码并合并 |

# Git 分支管理

几乎每一种版本控制系统都以某种形式支持分支，一个分支代表一条独立的开发线。

使用分支意味着你可以从开发主线上分离开来，然后在不影响主线的同时继续工作。

Git 分支实际上是指向更改快照的指针。

有人把 Git 的分支模型称为**必杀技特性**，而正是因为它，将 **Git** 从版本控制系统家族里区分出来。

创建分支命令：

git branch (branchname)

切换分支命令:

git checkout (branchname)

当你切换分支的时候，Git 会用该分支的最后提交的快照替换你的工作目录的内容， 所以多个分支不需要多个目录。

合并分支命令:

git merge

你可以多次合并到统一分支， 也可以选择在合并之后直接删除被并入的分支。

开始前我们先创建一个测试目录：

$ mkdir gitdemo

$ cd gitdemo/

$ git init

Initialized empty Git repository...

$ touch README

$ git add README

$ git commit -m '第一次版本提交'

[master (root-commit) 3b58100] 第一次版本提交

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

create mode 100644 README

## 7、Git 分支管理

### 7.1 列出分支

列出分支基本命令：

git branch

没有参数时，**git branch** 会列出你在本地的分支。

$ git branch

\* master

此例的意思就是，我们有一个叫做 **master** 的分支，并且该分支是当前分支。

当你执行 **git init** 的时候，默认情况下 Git 就会为你创建 **master** 分支。

如果我们要手动创建一个分支。执行 git branch (branchname) 即可。

$ git branch testing

$ git branch

\* master

testing

现在我们可以看到，有了一个新分支 **testing**。

当你以此方式在上次提交更新之后创建了新分支，如果后来又有更新提交， 然后又切换到了 **testing** 分支，Git 将还原你的工作目录到你创建分支时候的样子。

接下来我们将演示如何切换分支，我们用 git checkout (branch) 切换到我们要修改的分支。

$ ls

README

$ echo 'runoob.com' > test.txt

$ git add .

$ git commit -m 'add test.txt'

[master 3e92c19] add test.txt

1 file changed, 1 insertion(+)

create mode 100644 test.txt

$ ls

README        test.txt

$ git checkout testing

Switched to branch 'testing'

$ ls

README

当我们切换到 **testing** 分支的时候，我们添加的新文件 test.txt 被移除了。切换回 **master** 分支的时候，它们又重新出现了。

$ git checkout master

Switched to branch 'master'

$ ls

README        test.txt

我们也可以使用 git checkout -b (branchname) 命令来创建新分支并立即切换到该分支下，从而在该分支中操作。

$ git checkout -b newtest

Switched to a new branch 'newtest'

$ git rm test.txt

rm 'test.txt'

$ ls

README

$ touch runoob.php

$ git add .

$ git commit -am 'removed test.txt、add runoob.php'

[newtest c1501a2] removed test.txt、add runoob.php

2 files changed, 1 deletion(-)

create mode 100644 runoob.php

delete mode 100644 test.txt

$ ls

README        runoob.php

$ git checkout master

Switched to branch 'master'

$ ls

README        test.txt

如你所见，我们创建了一个分支，在该分支上移除了一些文件 test.txt，并添加了 runoob.php 文件，然后切换回我们的主分支，删除的 test.txt 文件又回来了，且新增加的 runoob.php 不存在主分支中。

使用分支将工作切分开来，从而让我们能够在不同开发环境中做事，并来回切换。

### 7.2 删除分支

删除分支命令：

git branch -d (branchname)

例如我们要删除 testing 分支：

$ git branch

\* master

testing

$ git branch -d testing

Deleted branch testing (was 85fc7e7).

$ git branch

\* master

### 7.3 分支合并

一旦某分支有了独立内容，你终究会希望将它合并回到你的主分支。 你可以使用以下命令将任何分支合并到当前分支中去：

git merge

$ git branch

\* master

newtest

$ ls

README        test.txt

$ git merge newtest

Updating 3e92c19..c1501a2

Fast-forward

runoob.php | 0

test.txt | 1 -

2 files changed, 1 deletion(-)

create mode 100644 runoob.php

delete mode 100644 test.txt

$ ls

README        runoob.php

以上实例中我们将 newtest 分支合并到主分支去，test.txt 文件被删除。

合并完后就可以删除分支:

$ git branch -d newtest

Deleted branch newtest (was c1501a2).

删除后， 就只剩下 master 分支了：

$ git branch

\* master

### 7.4 合并冲突

合并并不仅仅是简单的文件添加、移除的操作，Git 也会合并修改。

$ git branch

\* master

$ cat runoob.php

首先，我们创建一个叫做 change\_site 的分支，切换过去，我们将 runoob.php 内容改为:

<?php

echo 'runoob';

?>

创建 change\_site 分支：

$ git checkout -b change\_site

Switched to a new branch 'change\_site'

$ vim runoob.php

$ head -3 runoob.php

<?php

echo 'runoob';

?>

$ git commit -am 'changed the runoob.php'

[change\_site 7774248] changed the runoob.php

1 file changed, 3 insertions(+)

将修改的内容提交到 change\_site 分支中。 现在，假如切换回 master 分支我们可以看内容恢复到我们修改前的(空文件，没有代码)，我们再次修改 runoob.php 文件。

$ git checkout master

Switched to branch 'master'

$ cat runoob.php

$ vim runoob.php # 修改内容如下

$ cat runoob.php

<?php

echo 1;

?>

$ git diff

diff --git a/runoob.php b/runoob.php

index e69de29..ac60739 100644

--- a/runoob.php

+++ b/runoob.php

@@ -0,0 +1,3 @@

+<?php

+echo 1;

+?>

$ git commit -am '修改代码'

[master c68142b] 修改代码

1 file changed, 3 insertions(+)

现在这些改变已经记录到我的 "master" 分支了。接下来我们将 "change\_site" 分支合并过来。

$ git merge change\_site

Auto-merging runoob.php

CONFLICT (content): Merge conflict in runoob.php

Automatic merge failed; fix conflicts and then commit the result.

$ cat runoob.php # 打开文件，看到冲突内容

<?php

<<<<<<< HEAD

echo 1;

=======

echo 'runoob';

>>>>>>> change\_site

?>

我们将前一个分支合并到 master 分支，一个合并冲突就出现了，接下来我们需要手动去修改它。

$ vim runoob.php

$ cat runoob.php

<?php

echo 1;

echo 'runoob';

?>

$ git diff

diff --cc runoob.php

index ac60739,b63d7d7..0000000

--- a/runoob.php

+++ b/runoob.php

@@@ -1,3 -1,3 +1,4 @@@

<?php

+echo 1;

+ echo 'runoob';

?>

在 Git 中，我们可以用 git add 要告诉 Git 文件冲突已经解决

$ git status -s

UU runoob.php

$ git add runoob.php

$ git status -s

M runoob.php

$ git commit

[master 88afe0e] Merge branch 'change\_site'

现在我们成功解决了合并中的冲突，并提交了结果。

# 8、Git 查看提交历史

Git 提交历史一般常用两个命令：

* **git log** - 查看历史提交记录。
* **git blame <file>** - 以列表形式查看指定文件的历史修改记录。

### 8.1 git log

在使用 Git 提交了若干更新之后，又或者克隆了某个项目，想回顾下提交历史，我们可以使用 git log 命令查看。

针对我们前一章节的操作，使用 git log 命令列出历史提交记录如下：

$ git log

commit d5e9fc2c811e0ca2b2d28506ef7dc14171a207d9 (HEAD -> master)

Merge: c68142b 7774248

Author: runoob <test@runoob.com>

Date: Fri May 3 15:55:58 2019 +0800

Merge branch 'change\_site'

commit c68142b562c260c3071754623b08e2657b4c6d5b

Author: runoob <test@runoob.com>

Date: Fri May 3 15:52:12 2019 +0800

修改代码

commit 777424832e714cf65d3be79b50a4717aea51ab69 (change\_site)

Author: runoob <test@runoob.com>

Date: Fri May 3 15:49:26 2019 +0800

changed the runoob.php

commit c1501a244676ff55e7cccac1ecac0e18cbf6cb00

Author: runoob <test@runoob.com>

Date: Fri May 3 15:35:32 2019 +0800

我们可以用 --oneline 选项来查看历史记录的简洁的版本。

$ git log --oneline

$ git log --oneline

d5e9fc2 (HEAD -> master) Merge branch 'change\_site'

c68142b 修改代码

7774248 (change\_site) changed the runoob.php

c1501a2 removed test.txt、add runoob.php

3e92c19 add test.txt

3b58100 第一次版本提交

这告诉我们的是，此项目的开发历史。

我们还可以用 --graph 选项，查看历史中什么时候出现了分支、合并。以下为相同的命令，开启了拓扑图选项：

\* d5e9fc2 (HEAD -> master) Merge branch 'change\_site'

|\

| \* 7774248 (change\_site) changed the runoob.php

\* | c68142b 修改代码

|/

\* c1501a2 removed test.txt、add runoob.php

\* 3e92c19 add test.txt

\* 3b58100 第一次版本提交

现在我们可以更清楚明了地看到何时工作分叉、又何时归并。

你也可以用 **--reverse** 参数来逆向显示所有日志。

$ git log --reverse --oneline

3b58100 第一次版本提交

3e92c19 add test.txt

c1501a2 removed test.txt、add runoob.php

7774248 (change\_site) changed the runoob.php

c68142b 修改代码

d5e9fc2 (HEAD -> master) Merge branch 'change\_site'

如果只想查找指定用户的提交日志可以使用命令：git log --author , 例如，比方说我们要找 Git 源码中 Linus 提交的部分：

$ git log --author=Linus --oneline -5

81b50f3 Move 'builtin-\*' into a 'builtin/' subdirectory

3bb7256 make "index-pack" a built-in

377d027 make "git pack-redundant" a built-in

b532581 make "git unpack-file" a built-in

112dd51 make "mktag" a built-in

如果你要指定日期，可以执行几个选项：--since 和 --before，但是你也可以用 --until 和 --after。

例如，如果我要看 Git 项目中三周前且在四月十八日之后的所有提交，我可以执行这个（我还用了 --no-merges 选项以隐藏合并提交）：

$ git log --oneline --before={3.weeks.ago} --after={2010-04-18} --no-merges

5469e2d Git 1.7.1-rc2

d43427d Documentation/remote-helpers: Fix typos and improve language

272a36b Fixup: Second argument may be any arbitrary string

b6c8d2d Documentation/remote-helpers: Add invocation section

5ce4f4e Documentation/urls: Rewrite to accomodate transport::address

00b84e9 Documentation/remote-helpers: Rewrite description

03aa87e Documentation: Describe other situations where -z affects git diff

77bc694 rebase-interactive: silence warning when no commits rewritten

636db2c t3301: add tests to use --format="%N"

更多 git log 命令可查看：<http://git-scm.com/docs/git-log>

### 8.2 git blame

如果要查看指定文件的修改记录可以使用 git blame 命令，格式如下：

git blame <file>

git blame 命令是以列表形式显示修改记录，如下实例：

$ git blame README

^d2097aa (tianqixin 2020-08-25 14:59:25 +0800 1) # Runoob Git 测试

db9315b0 (runoob 2020-08-25 16:00:23 +0800 2) # 菜鸟教程

# 9、Git 标签

如果你达到一个重要的阶段，并希望永远记住那个特别的提交快照，你可以使用 git tag 给它打上标签。

比如说，我们想为我们的 runoob 项目发布一个"1.0"版本。 我们可以用 git tag -a v1.0 命令给最新一次提交打上（HEAD）"v1.0"的标签。

-a 选项意为"创建一个带注解的标签"。 不用 -a 选项也可以执行的，但它不会记录这标签是啥时候打的，谁打的，也不会让你添加个标签的注解。 我推荐一直创建带注解的标签。

$ git tag -a v1.0

当你执行 git tag -a 命令时，Git 会打开你的编辑器，让你写一句标签注解，就像你给提交写注解一样。

现在，注意当我们执行 git log --decorate 时，我们可以看到我们的标签了：

\* d5e9fc2 (HEAD -> master) Merge branch 'change\_site'

|\

| \* 7774248 (change\_site) changed the runoob.php

\* | c68142b 修改代码

|/

\* c1501a2 removed test.txt、add runoob.php

\* 3e92c19 add test.txt

\* 3b58100 第一次版本提交

如果我们忘了给某个提交打标签，又将它发布了，我们可以给它追加标签。

例如，假设我们发布了提交 85fc7e7(上面实例最后一行)，但是那时候忘了给它打标签。 我们现在也可以：

$ git tag -a v0.9 85fc7e7

$ git log --oneline --decorate --graph

\* d5e9fc2 (HEAD -> master) Merge branch 'change\_site'

|\

| \* 7774248 (change\_site) changed the runoob.php

\* | c68142b 修改代码

|/

\* c1501a2 removed test.txt、add runoob.php

\* 3e92c19 add test.txt

\* 3b58100 (tag: v0.9) 第一次版本提交

如果我们要查看所有标签可以使用以下命令：

$ git tag

v0.9

v1.0

指定标签信息命令：

git tag -a <tagname> -m "runoob.com标签"

PGP签名标签命令：

git tag -s <tagname> -m "runoob.com标签"

# 10、Git 远程仓库(Github)

Git 并不像 SVN 那样有个中心服务器。

目前我们使用到的 Git 命令都是在本地执行，如果你想通过 Git 分享你的代码或者与其他开发人员合作。 你就需要将数据放到一台其他开发人员能够连接的服务器上。

本例使用了 Github 作为远程仓库，你可以先阅读我们的 [Github 简明教程。](https://www.runoob.com/w3cnote/git-guide.html)

![手机屏幕的截图

描述已自动生成](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDuRXhpZgAATU0AKgAAAAgABAE7AAIAAAAMAAAISodpAAQAAAABAAAIVpydAAEAAAAYAAAQzuocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEVzaG5hIFZlcm1hAAAFkAMAAgAAABQAABCkkAQAAgAAABQAABC4kpEAAgAAAAMxOQAAkpIAAgAAAAMxOQAA6hwABwAACAwAAAiYAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAyMDowNToyNiAyMDo0MDozMQAyMDIwOjA1OjI2IDIwOjQwOjMxAAAARQBzAGgAbgBhACAAVgBlAHIAbQBhAAAA/+ELHmh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMjAtMDUtMjZUMjA6NDA6MzEuMTg1PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkVzaG5hIFZlcm1hPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIARACFgMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APpGiisXxfqsui+E7++t/wDWxx4Q+hJwD+tNauwbFu81zStOkEd9qNtbuf4ZJQp/Kq//AAlnh/8A6DNj/wB/1/xr5vJudRu2dmeeeQlmZjksfWpP7Kvf+fdv0rrjhZSV0mzlliIxdm0j6M/4Szw//wBBmx/7/r/jR/wlnh//AKDNj/3/AF/xr5z/ALKvf+fdv0o/sq9/592/Sq+p1P5X9wvrUO6+8+jP+Es8P/8AQZsf+/6/40f8JZ4f/wCgzY/9/wBf8a+c/wCyr3/n3b9KP7Kvf+fdv0o+p1P5X9wfWod1959Gf8JZ4f8A+gzY/wDf9f8AGj/hLPD/AP0GbH/v+v8AjXzn/ZV7/wA+7fpR/ZV7/wA+7fpR9Tqfyv7g+tQ7r7z6M/4Szw//ANBmx/7/AK/40f8ACWeH/wDoM2P/AH/X/GvnP+yr3/n3b9KP7Kvf+fdv0o+p1P5X9wfWod1959Fnxb4eAydasf8Av+tLB4s8P3Uyw2+tWMkjHCqs65P6184S2F1CoMsLKDwM1kajAYZEfbtLZoeDqKPM07egLFQbsmr+p9e0V5z8F/EF3rHhae1v5GmewlEccjHJKEZAJ9uf0r0auKS5XY6k7q4UUUUhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUdOtADJZBEmW/AVnyStK2WP4U6eUyyE9h0qKgAooooAKKKKAFBIORwau29xv+V/vdj61RpQcHI60AatFRwSebGD36GpM0AFFGc0UAFFFFABRRRQAVy3xJ/5J/qP+6v8A6EK6muW+JP8AyT/Uf91f/QhVR+JClseFaH/yFE+h/lXT1zGh/wDIUT6H+VdPX1+W/wAF+p8xj/4vyCiinJG8pIjRnOOijNd9SSjBybsccIuUkkQmQhj0pVkGPmroNK+xW1mVvdMnnlLElggIx6cirv2nSf8AoC3H/ftf8K+DwtTMY2qqre62cr/g2fY4iGBlem6drdUrfocn5i+tHmL611n2nSf+gLcf9+1/wqO4l0uW2kjj0e4RmUhWEa8H16V6Dx2Z2+KP4f5nEsJl/wDLL8f8jmQc9KKe0EsKgyRsozgEjGaZX0uDrOvh4VG7tpX9TwcVSVGvOmtk2UNW/wBQn+9/SuT13pB+P9K6zVv9Qn+9/SuT13pB+P8ASujG/wC4y+X5owwn++R/roerfAP/AJBesf8AXaP/ANBNeuV5H8A/+QXrH/XaP/0E165XwVT42fZU/hQUUUVmWFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABUVy+yA46nipaq3p+VB7mgCnRRRQAUUUUAFFcz4y+IPh/wLZefrt4FlYZitYvmlk+i+nucCvn7xV+0l4l1OR4vDcEOj22flcqJZiPcsNo/AfjQB9T0V8G6l408TauzHU9e1G5DHJWS5cr+WcCsyO/vIZN8V1Mj/wB5ZCDQB+hCSOgIQ4zTSxPUk18O6P8AFLxrocgaw8R3xA48ueTzk/75fIr1/wAG/tMJLLHa+NrBYgxx9tswcD3ZDz+IP4UAfQYJHQ4qaO6dPvfMPes3S9Vsda06K+0q6iu7WYZSWJsg1boA1I5FkXKmnVmxSGJwR07itFWDKCOhoAWiiigArlviT/yT/Uf91f8A0IV1Nct8Sf8Akn+o/wC6v/oQqo/EhS2PCtD/AOQon0P8q6euY0P/AJCifQ/yrp6+vy3+C/U+Yx/8X5BV7SH26go/vAiqNTWj+XeRP2DDNa5hS9thKtPvF/kZ4Op7LE059mvzOoooor8OP1oKKKKQGTrj/LCnuSax60dafdfBf7iAf1rOr9kyGl7LLaS7q/3u5+X5xU9pjqj7O33aFDVv9Qn+9/SuT13pB+P9K6zVv9Qn+9/SuT13pB+P9K9nG/7jL5fmjzMJ/vkf66Hq3wD/AOQXrH/XaP8A9BNeuV5H8A/+QXrH/XaP/wBBNeuV8FU+Nn2VP4UFFFFZlhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRXlnizxBr83xmtfCem+JodBsbjRjdGRreOR2lEjD5N/G7C9DkYDcZwRhaT8TfEM9v4Q+26jbtDPrt1pt7fxxqsV9HGMI4PRQx9MZI44oA9rubu3s0V7ueOBXdY1aRwoLMcKoz3JOAKSG7t7mSaO3njleB9kqo4JjbAOGHY4IOPevn/AMQ+J7/xNpKPf3a3UVl8Qo7W0dFUAQrnaMjr169a7n4U/wDI4fEP/sN/+yUAem0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVa9HyKfQ1I9zGnGcn2qvNciWMrs/HNAFaiiigArzj4t/FW2+H2lC2s9k+tXSEwQnkRL08xvb0Hcj2rsfFPiK08KeF77WtQP7m0iL7c4Lt0VR7kkD8a+G/EWv33ijxBd6xqsnmXN1IXb0UdlHsBgD2FAHaeFvh74r+MMuo642oxGSOUK8967fvHIztXAOMDH5isXxf8ADLxT4I/ea3pzC1Jwt1Cd8R/EdPocV9Bfs1zQv8MZY4iPMjvpBKB1yQCP0xXrN3aW9/Zy2t7BHPbzKUkikUMrg9QQaAPz3or7g0z4WeB9Ij2WnhmwbnO64i85vzfJrSfwV4WdNr+GtIK+n2GL/wCJoA+DlVncKilmY4AAyTXpvhv4A+NfEFqtzNbw6VC4ypvXKsR/uAEj8cV9MWfw38H6frsesWPh+zt72IYR40wq+4T7oPvjNdPQB8d6T4h8S/A7x9c6VNIJ4IpALm1DExToQCHX0OCCD+Br6u8MeJdO8W6Bb6vo8wlt516fxI3dSOxFfMv7SctvJ8Uo1g2+ZHYRLNj+9liM/wDAStU/gX8QJPCXjCPTL2YjStUcRyBjxFIeFf29D7fSgD68q3b3CpFtc9DxVTrRQBofa4vU/lRWfRQBrVy3xJ/5J/qP+6v/AKEK6muW+JP/ACT/AFH/AHV/9CFVH4kKWx4Vof8AyFE+h/lXT1zGh/8AIUT6H+VdPX1+W/wX6nzGP/i/IKUHBBrP124ltfD2oXFu5SWK2kdGHYhSQa868Na1fy3ejnT9X1HUp7hh9vt7hC0cSd2DEcY9jXRWxEac1Brf/hjCnQlUi5p7HtP9uSd4UP4mj+3JP+eCfma81g8dXf8AZ+qald6aiWNhK8CukuWlkDhVGOw5GTVyLxJqdvqSafr+nw20l1byS2728xYfIMlTxwcdxXiLKcof/Lr/ANK/rU9X+0czX/Lz8jvv7ck/54J+Zo/tyT/ngn5mvJ7Px09roWkQW0atdXULzM97OzBFDEctgkkkHHpWsPGk91pemtYadu1DUZWiSCV9qoV+8xOOQOtTHK8nkv4a/Hy/zKlmGZx+3+X9dDsrmdrm4aVhgt2Haoq4jxFrPia0s9G2wW1pdT6ktu6rKWSXI+XtkKec9xgV2sRcwoZgok2jcFOQD3xXv0JU0vZU1ZRSX4aHj1ozb9pN3crlLVv9Qn+9/SuT13pB+P8ASus1b/UJ/vf0rk9d6Qfj/Stsb/uMvl+aMsJ/vkf66Hq3wD/5Besf9do//QTXrleR/AP/AJBesf8AXaP/ANBNeuV8FU+Nn2VP4UFFFFZlhRRTZZBDC8jZ2opY4GT+VAEV7e22nWUt3eyrDBEu53Y9KraFq8eu6PDqMEbRxzF9it1wGK5P5Zrxbxp4yufE18Y03Q2ELfuoTwWP95vf+VY9tr+r2VutvaaldQQpnbHHKQo5z0rw55tGNWyV4/qe9DKJSpJt2k/wR9I0V86f8JTr3/QYvv8Av+3+NH/CU69/0GL7/v8At/jT/tin/Kxf2LU/mR9F0V86f8JTr3/QYvv+/wC3+NH/AAlOvf8AQYvv+/7f40f2xT/lYf2LU/mR9F0V86f8JTr3/QYvv+/7f40f8JTr3/QYvv8Av+3+NH9sU/5WH9i1P5kfRdFfOn/CU69/0GL7/v8At/jR/wAJTr3/AEGL7/v+3+NH9sU/5WH9i1P5ket658NNB8S+Mv7e12Bb5TpxsGs54w0ePM3hx3DA5GR61q3ngzw3f6BBod3oljJpluQ0NoYQI4yO4HY8n8zXh/8AwlOvf9Bi+/7/ALf40f8ACU69/wBBi+/7/t/jR/bFP+Vh/YtT+ZHt8Pgrw1b2Is7fQ7GK2W6W8WKOEKomXGJAB0PArQstI0/Tbm7uLCzht5b2Xzrl41wZXxjcfU4FeAf8JTr3/QYvv+/7f40f8JTr3/QYvv8Av+3+NH9sU/5WH9i1P5kfRdFfOn/CU69/0GL7/v8At/jR/wAJTr3/AEGL7/v+3+NH9sU/5WH9i1P5kfRdFfOn/CU69/0GL7/v+3+NH/CU69/0GL7/AL/t/jR/bFP+Vh/YtT+ZH0XRXzp/wlOvf9Bi+/7/ALf40f8ACU69/wBBi+/7/t/jR/bFP+Vh/YtT+ZH0XRXzp/wlOvf9Bi+/7/t/jR/wlOvf9Bi+/wC/7f40f2xT/lYf2LU/mR9F0V86f8JTr3/QYvv+/wC3+NWdO8T65Jqlqj6vesrTKCDO2CM/Wms3pt25WJ5NUSvzI+gqKKK9o8MKKKKAAnAyelUZ7kuSqcL/ADqS8lx+7H41ToAKKKKACiiigD5//ag8SPFaaT4cgcgTE3c4B6gfKg/PcfwFfOaqzuFQEsxwAO5r0/8AaGvGuvi9eRMcra28MSj0yu7/ANmrlvhtYJqfxM0C1lGUa9jZge4U7sfpQB9VfCPwDF4E8GxRSDOo3gWa8fP8WOFHsBxXeV5p4l8Sa7qHjHVNF0XV00Gw0OzS5vbz7KLiSQsMhVQ9gBzjmqg8cNceD/Ds512TU55/EEOnzXdhH9l3EsTtkR1JxjGQMZ7UAerUV5GPHnit9Q+I0LwQx22hwSPaTh1Jt2ERKDbj592N3PTpWe3xgvb3R/B0Fit/BqN7e2kV/cT2OyKZWGHCsRjknIx26UAe2UVweq/E+LTI/F7HTHk/4RryQw80Dz/MH0+XH41hT+N/EqfFXULXRNMm1iA6Nb3MdibhYkhJOS2SPvHOPfj0oA5L9pLwFEixeMtPQh2ZYL9R34wj+3Taf+A+9fPAJVgQcEHINfY/iTWLP4gfAPVNUt4niiuLGSXypOqPGckfgyHmvjigD7h+GHiNvFPw30jU5mLTtD5UxPUuh2sfxIz+NdZXi/7Md483gDUbZ2JFvfnaPQMin+YNe0UAFFFFAGtXLfEn/kn+o/7q/wDoQrqazvEGkprug3enSNsE8ZUN/dPUH86cXZ3E9UfOWh/8hRPof5V0+RXO6p4Z1fR7x7e7spSVOA8allb3BFUv7Pvf+fO4/wC/Tf4V72Fx3sIcvLf5njYjCe2nzXt8jqL21ivrGe0nJ8ueNo3wcHBGDUemadBpWlwWFsWMUEYRSx5IHrXN/wBn3v8Az53H/fpv8KP7Pvf+fO4/79N/hXR/aa5ubk19f+AY/UHa3Pp6f8E1IvCumR6Ne6W4eW2vZnmkDtyGYg8EdMYGKqL4TWyWa6t7q41C/W3aC3a/nysQIwQMDj68mq39n3v/AD53H/fpv8KP7Pvf+fO4/wC/Tf4Vk8dTdv3e3n/wDRYSp/P+H/BH6d4Iig0PS4J7yWC/sY2UXNo204YkleQcjnuO1aN34WtLzTbW1kubsS2j+ZDdiX98rdzuPXr06Vl/2fe/8+dx/wB+m/wo/s+9/wCfO4/79N/hRHG0krKn+P8AwAeEqN35/wAC/ceEbW60qGznv753huRdJctMDIsg7gkYA9gK3o18uJULl9oA3MeT7muS/s+9/wCfO4/79N/hR/Z97/z53H/fpv8ACrjmMIO8Yfj/AMAiWBlJWc/wOg1Y/uE/3v6Vyeu9IP8AgX9KtmwvAObS4/79N/hVKTTdS1C7SC00+5kboAImGTV1s09rQdHltfrfz9CaWX+zrKrzbeX/AAT1j4B/8gvWP+u0f/oJr1yuL+GHhGbwn4ZK3pU3l2/myqpyE4wFz3x/Wu0r5qo05No9+CtEKKKKgoKKKKAPOPH/AMP/ALYJNX0OLFwPmnt1H+s/2lH9727/AF64fhz4ZDXtAt9SOpm3M27MfkbtuGK9cj0r2OmRRRwpshRUXJbCjAyTkn8zmvOnl9CdX2jW/Tz7npQzKvCl7NPVdfLseZ/8KaH/AEGj/wCA/wD9lR/wpof9Bo/+A/8A9lXp9FV/Z2F/l/F/5i/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyo/4U0P8AoNH/AMB//sq9Poo/s7C/y/i/8w/tPF/z/gv8jzD/AIU0P+g0f/Af/wCyqW1+EItruGf+2C3luHx9nxnBz/er0qihZdhU78v4v/MTzLFNW5vwX+QUUUV3nnhRRRQBmzNumY+9R0rcMc+tJQAUUUUAFFFFAHx58f4Gh+MepswwJYoXX3HlqP6VyPgvV10Hxxo+pyttjtruN5D6JnDfoTXrv7UGgtDrmka9Gn7u4hNrIwHRlO5c/UMfyrwagD7c8QeBl1zV4/EOga3c6HqrwCJrq2RZFmj6gMjcH2NVovhVpkGk2Fml/dvJa6ymsy3EuGe4nHXd0AB9q85+Cnxps10638MeLrlbeSECOzvZThHXsjnsR0BPB/n7+rK6hkIZSMgg5BoA4q4+G8UuseJ7uLV7iG28SWpgubQRqVV9mzzA3Xgdvc+2LN78P7O98P8AhzSWvJli0C4t54n2gmXyVwA3pn2rraKAPO/Enwjh1/VdZuYtevbC31qNBeWsMaMsjpjY2SM4GOQOvrU2p/DCWfxBLrWieJb7SL2Swisd0UaOuxOuQepIxjpgjPtXfVR1jW9N8P6ZLqGtXsNnaxDLSStgfQep9hzQB554/t9O+G/wFvdJsJGEZh+yQmU5aV5GJYn3ILmvkOvRfi98UJfiHriR2SvDo9mSLaNuGkJ6yN7nsOw/GvOqAPp/9l+Ar4J1eYjiS/2j8EX/ABr26uA+COgtoPwo0tJkCzXYN3JjvvOVz/wHbXf0AFFFFAGtWJ4hv3hVbaFtpYZcj0radgiFmOAoyTXD31ybq8lmP8R4HtVRWpMnoZ1zqljaS+Xd3cMT4ztdwDiof7d0n/oIW3/fwVwusaJrup6tPdNYSYdvlGRwo6d6pf8ACKa3/wA+En5j/GujlXcw5mej/wBu6T/0ELb/AL+Cj+3dJ/6CFt/38Fecf8Iprf8Az4SfmP8AGj/hFNb/AOfCT8x/jT5V3DmfY9H/ALd0n/oIW3/fwUf27pP/AEELb/v4K84/4RTW/wDnwk/Mf40f8Iprf/PhJ+Y/xo5V3DmfY9H/ALd0n/oIW3/fwUf27pP/AEELb/v4K84/4RTW/wDnwk/Mf40f8Iprf/PhJ+Y/xo5V3DmfY9H/ALd0n/oIW3/fwUf27pP/AEELb/v4K84/4RTW/wDnwk/Mf40f8Iprf/PhJ+Y/xo5V3DmfY9H/ALd0n/oIW3/fwVXudT0u4A8m+t2lyAAsgy3tXAf8Iprf/PhJ+Y/xpV8La4rBlsZAQcghhx+tHKu4cz7HrXhXV5LXUEtZZCYJjgAn7rdq7yvILVrhIYmuYzDOFBZT2NepaRfDUdLhuAfmK4f2YdawmuptF9C7RRRWZYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBnXCbJ29+RUVX7mHzEyv3hVGgBKKKKACiiigDk/iZ4OXxx4EvtJUKLoDzrRj/DKvK/nyPoa+RPCOux+CvFpl1rQ7XUokJgurO9gDFRn5toYfKwx/MV9z14N8dfhBJqrS+K/DFvvuwM31rGOZQP+Wij19R369c5AMP4n/CnStT8L2/jT4cWpNtMglns4BkbDzvVe2OhUfpg15r4Z+J/jDwgqwaRrEy26H/j2nAkjHsFbO38MV3vwX+Mdt4RtT4d8U7107eWguApbyCeqsOu3PPHT+Xskq/CrXWOoz/8ACN3TN8zSs0WT9f8A69AG38P9a1DxH4C0rWNYEK3d5D5riBSqgEnGASe2K6OqumNYvpdudIMBsfLHkG3x5ezttxxj6VaoA8g+O3xD8R+BI9MXw9NbwpfrIGkeHe6MuORk45DenavmbXvE+teJ7z7Vr+pXF9Lzt81yQnso6KPYV7r+1P8A8e/hv/fn/klfOtABXX/DHwXN458b2mnBG+xxsJrtx0WMHkZ9T0H1rA0PQ9R8R6xBpmj2z3N1O2FRR09yewHrX2T8Mfh5Z/D3w0tpHtmv58Pd3AH329B/sjt/9egDsoo0hhSKNQqIoVQOwFOoooAKKKKAF8QXfkWPlKcPKcfh3rlKpfFiPW7SW11PTJ51s1j8qYRdEbJIY/XOM+1eZf8ACTaz/wBBGf8A76reEbrQxlKzPWqK8l/4SbWf+gjP/wB9Uf8ACTaz/wBBGf8A76q+Rk8yPWqK8l/4SbWf+gjP/wB9Uf8ACTaz/wBBGf8A76o5GHMj1qiuI8K6vf3iXRuruSXaV27j0610H2qb/no1S1YadzXorI+1Tf8APRqzdf1G7ttFmlguHjkUrhlPI+YUWHc6mivJf+Em1n/oIz/99Uf8JNrP/QRn/wC+qrkZPMj1qivJf+Em1n/oIz/99Uf8JNrP/QRn/wC+qORhzI9N1CLKrIO3BrZ8F6h5dzJZSH5ZBuT6jr/n2rxG88VauluynUpstwBure+GQ8Ra/wCL7WQXdybK1cSXEhPy4HRc+/TH1olD3dQUtdD6EooorlOgKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKrz2wf5k4b09asUUAZbIyHDDFNrVIDDDAH60w28R/gFAGbRWj9mi/ufrR9mi/ufrQBnUVo/Zov7n60fZov7n60AeKfEj4D6T4vkl1PQnTStWfLPhf3M5/wBoD7p9x+Rr528RfDTxd4Yvfs+p6JdEM22OaBDLHIe2GXI59OvtX3p9mi/ufrR9mi/ufrQBxfw+0660j4daFYX8ZiuYLKNZYz1Rscg+4ro60fs0X9z9aPs0X9z9aAPnv9prQ9S1LR9EvbCzluYbWWVZjEhYpuC7SQO3ynmvL/BnwL8W+KpI5bu1bRrBuTcXilWI/wBmPqfxwPevtT7NF/c/Wj7NF/c/WgDhPA3w60HwBpv2fR4N9xIP395Lgyyn69h7Dj8ea6qtH7NF/c/Wj7NF/c/WgDOorR+zRf3P1o+zRf3P1oAzqK0fs0X9z9aKAJGUOpVgCD1BFZjeGdCZizaPYkk5JNuvP6VqUUXAyv8AhF9B/wCgNYf+A6f4Uf8ACL6D/wBAaw/8B0/wrVop3YrIyv8AhF9B/wCgNYf+A6f4Uf8ACL6D/wBAaw/8B0/wrVoouwsihBoOk22fs+mWkW7rshUZ/Spf7LsP+fKD/v2KtUUXYWRV/suw/wCfKD/v2KZLo2mTxmObT7aRD1VolIP6VdoouxmV/wAIvoP/AEBrD/wHT/Cj/hF9B/6A1h/4Dp/hWrRRdisjK/4RfQf+gNYf+A6f4Uf8IvoP/QGsP/AdP8K1aKLsLIyD4S8OscnQ9PJ97ZP8K0ra0t7KBYLOCOCJfupGgVR+AqWii7CyCiiikMKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoqK4uoLSMSXU0cKFlQNIwUFicAZPckgVLkHoaACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOU8d+Lk8NaV5VswOoXAIiX+4O7n+nvXD+EfiXdabItprzyXVoTxMTmSP/AOKH6/yrrvG/gFPETNf2Ehj1BVA2ux2SAdvY/wCfevGr2xudOu3tb6F4ZozhkcYIr53HVsTRr8+y6dvmfS4Chha2H5N317/I+lLS8t7+1S5s5kmhkGVdDkGpq+e/DPi3UPDF1utX8y3Y5lt3Pyt/gfevbPD3ibT/ABLZCewkxIo/eQMfnjPuPT3r0sJjoYhW2l2/yPLxmAqYZ33j3/zNiiiivQPOCiiigAooooAKKKKACiiigAooooAKKKKACvJtH8ReJfFHiHWbxfF1joNnpOsGwXS5rSN/ORCBl3ZgwL54xj8a9Zrn73wH4W1HxBHrd9oVjPqUZBFw8ILEjoT2JGBgnkUAeXX/AI78V3Oh+JvFlp4msdLi0K+kt4tDntUYSiMgYkcneGfsBjn9F1/xn411HXfFQ0XWV0O20XRbbU1tpLFJpNzQ+Y0ZZumeQTz0GMc59PvPAPhTUNeTWr3QLCbUVO7z3hBJPZj2JGOCeRV6Xw5o895f3U2nQPPqUAtrtynM8YBAVvUYYj/9VAHlnhzxh4wi8V+FBrms2+oWfijSpbsWsdmsIs3WPzBtbJZvTk+tc/pPjP4i32leDtRPiu2A8S3ktiYW0uMi3wzKJMggswx04HTIPOfco/DOixS6dLHptuH0yA29k2zmCMgKVX04AFRQeEdAtrbTreDSbVIdLmM9kgj4gc5yy+/zGgDxq78X6/ceDZLTXbm11W60/wAcxaQbqeyj/exKQd2zG1W9xyB+ddJ4Mg1Jvj/44MmtzG2g+ylrRo02yhoiVGeq7OnHXvXoD+D/AA/JHMj6TbFZ75dRkBT71yMYl/3vlH159TUh8L6IfEw8Q/2Zb/2sI/KF3s+fbjGM/TjPpxQBrUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFIzqiFnYKqjJJOABXl3jT4llvM07w5JgfdkvB39Qn+P5etc+IxFPDx5ps6cPhqmInywRW+IOt3Gk+IsaHr1zuYZntllLLC3t259O1YVv8RfFEbKv9pbxnGHhQ/rjNcuSWYliSScknvWzY+H9QjutKuLu0kjtby4RI3YcN8w/L29a+WeIrVKjlTbXo3ofWLDUKVNRqJO3dLU+hISxgjMnLbRu+uKfQBgAUV9ifFBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFYniTwpp3iaz8u8TZMo/dzoPmQ/1HtW3RUThGceWSui4TlTkpQdmfPHiPwtqPhq88q9j3Qsf3c6j5X/wPtWdp+o3el3qXdhO0M0ZyGU/p7ivpC+sLXUrN7W+hSaGQYZGFePeMfh1daIZL3Sg1zYDll6vD9fUe/5+tfN4vL50H7SlqvxR9Pg8yhXXs62j/BnaeD/iHaa8qWepFLW/6DnCS/T0PtXa18vgkEEHBHQivRvB3xMktNlj4idpYeAl11ZP971Hv1+tdODzNP3K33/5nLjcravOh93+R61RTIZormFJreRZI3GVdDkEfWn17x8/sFFFFABRRRQAUUUUAFFFFABRRRQAUVzfi/x9oPgm1R9auv8ASJuLeygG+ec9gqDk88Z6e9cgn/CzfH+HDR+BdGcZUFRNfSqfUHiPIPswPrQB6NqWs6Zo9u0+rahbWUS9XnlVAPzNcfe/G34f2T7D4hiuW9LSN5v1UEVHpnwU8HWdwLvVba41+/zlrvV7hp2b6qflx+FdnYaHpOlx7NM0yzs19Le3VB+goA4P/hfngXd/x9X4H97+z5cf+g1fsfjV8P79ti+I4LdumLpHh/8AQgK7ny0xjYuPpVDUPD+jasu3VNJsbwf9PFuj4/MUAT2GqWGq26z6Ze293EwyrwShwfxFWq871H4I+FJbhrzQBeeG9QJ3C60m5aI/Tbkrj2AFUGv/AIlfD/LarDH430VOWuLSMQ3sK8ZJj6PjngcnuRQB6nRWB4T8baF41037Z4fvkn24EsLfLLCf7roeQf0PbNb9ABRRRQAUUUUAFFFFABRRRQAUUUUAFV7+/tdMs5Lq+mWGGMZZ2NUvEHiOw8N2BudQk5PEcS/ekPoB/WvD/E3iu/8AE94ZLpvLt1P7q3U/Kg/qfevPxeOhh1Zay7f5no4PATxLu9I9/wDI1/GXxAuvEDvaWBa308HGM4aX3b29q46ON5ZFjiVndjhVUZJPpVjTtNu9WvktNPgaaZzwqjp7n0HvXtPg7wFaeG41ubrbc6iRzJj5Y/Zf8a8KlRr46pzSenf/ACPoKtehgKfLFa9v8zD8FfDVbfy9R8Qxh5fvR2p5C+7ep9q9FntYLmNY54ldUZXUEdCpyD+YqWivpaOHp0IckEfLV8TUrz55v/gBRRRXQc4UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRXMaz8RfC/h7Wl0rWNTFrdHbnfE+xN33dzgbVz7mgDp6OvWuU8TfEnw14XumsdQ1OJL/yTMsO1mAHYuQCEB9TisvRfizoz+A9B17xPcRadcaujbLaJXkLMrFWKqAWxxn2z1oAi8Y/DSK/8y/8PqsFz957bokn+7/dP6fSvJri3mtbh4LmJopYzhkcYIP0r274d+N/+E607V71I4VgstVnsoJIWJWaNNpV+fUNV3xV4M0/xPb5lHkXiDEdwg5+h9RXjYzLY1PfpaPt3PbwWZyp2hW1XfseS+FPG2oeGJgik3Fix+e3Y9PdT2Ne1aJr1h4gsFutNmDr/Gh4aM+jDtXgmu+HtQ8O3xttRiK5+5IvKyD1BqHSdYvdEv1u9NnaGVeuOjD0I7ivOw2NqYWXs6i07dUelisDSxcfaU3r36M+k6K5Pwj48svEkawT7bXUAOYieH91P9OtdZX01OrCrHmg7o+Xq0p0pck1ZhRRRWhkFFFFABRRRQAV5/428f3lrrCeEfA1supeKLhNzbuYdPjP/LWY9uvC/T1ANr4keNLjw3p9tpegRC68R6u/kafbjnae8jf7K9f8mrfgDwNbeCtFdGkN5q163n6lqEnMlzKeSSTztBJwP6k0AU/Bnw0svDl0+s6zcPrviW45uNUuxllP92MfwKOmB/8AWrt6KKACiiigAooooAKKKKAOC8YfDKLVNSHiLwldnQPE8Iyl5AuEuP8AYmTowPr1+vSpPAnj+XXLyfw94otP7J8U2K5uLQ/cnX/nrEf4lP6V3NcZ8Q/AzeKbKDUNGuP7P8SaWTNpt8nBVv8Anm3qjdCP58ggHZ0VyPw78a/8JloLm9h+x6zYSG21OyIwYZhwcD+6cZH5dq66gAooooAKKKKACiijp1oAK5Xxf45s/DMJhi23OoMPlhB4T3b0Ht1P61i+NPiRHYb9P0B1lueklwOVj9h6n9K8lmmkuJnlndpJHJZnY5LH1JrxcbmSp3hS1ffse5gcsdS1Stou3cs6rq17rV+95qM7TTP3PRR6AdhV3w54Y1DxNfeRYptjU/vZ2Hyxj+p9q1fB3gO78SSLc3O6208HmTHzSey/417Rp2m2ekWKWthCsMEY4VR+p9T71w4TATxD9pV2/FnfjMwhhl7Klv8Agij4b8L6f4ZsfJsk3SsP3s7D5pD/AEHtWzXG23xb8EXd9b2tvr0LyXEphQ7HCbwSNpbGASRwCeeo4NaV3468OWMGszXWppHHobRpqB2MfIL42jpznI6Zr6aEIwioxVkfLznKpJyk7s6CiuX1P4j+FdH1caZf6oEutqM6pC7iMP8Ad3soIXPbJFc8vxl0XT/G2vaH4muIdOjsLiKK1lw7eaHQElsAhQD3OByKog9JopFYOoZTlSMgjvS0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAV4r8RfAHjjxReeIoYWF5Z3nktpw/tDyY4FQgsjRYw7Ejgk4HqOle1UUAeS6n4M8Xad4o1q50HT9M1O28Q2EVtNJeT7PsbpGUyRgl1PoO/p1rP0X4e+LvCdn4P1PTdPsdT1DS7CayvLCe52BBJIXDo+CARnBPccDNe1UUAcJ8J/Cmr+E9D1iHxBHax3N9rFxeqlo2Ywr7cY9BwcD0xXd0UUAU9U0qy1mxe01GBZom7HqD6g9jXjPi/wAA3vh13ubXddafniQD5o/Zh/XpXuVIyq6FXUMrDBBGQRXHisHTxK10fc7cLjamGlpqux8wxyPFIrxsUdTlWU4INeoeDvibny7DxG+D91Lv+j/4/n60vjH4Yh99/wCG0AblpLPsfdP8Py9K8ukjeKRo5VZHU4ZWGCD6V87/ALRgKn9WZ9L/ALPmFL+ro+nkdZEV42DKwyGByCKXOeleCeH/AB5q/h6zktIXE8DIRGkvPlMe6/j26V6h8OL6S/8ABsMk8jSSrLIHZjkkli3P5172Gx8MRJRS1sfP4rL6mGi5t3Vzq6KKK9A80KiurqGys5rq6kWKGFDJI7HAVQMkk1LXnPxov7h/DFj4Y01yl74kvY7BWU8pGTmRsdxtGD7GgCp8MbObxbr+o/EfV0bN4WttIicf6i1U43Y7Fjz+frXqNVtN0+30nS7XT7KMR29rEsUaAcBVGBVmgDzX4nXOqX3i/wAI+FNO1a60m31mW5a6ubNtsu2KMMFVu2c0f8Kcb/of/GP/AIMv/saT4nw6lp3jLwf4rsdGv9YtdHluVurfTovNnAljCgqnfkUv/C5j/wBE48ff+CT/AOzoAP8AhTjf9D/4x/8ABl/9jR/wpxv+h/8AGP8A4Mv/ALGj/hcx/wCicePv/BJ/9nR/wuY/9E48ff8Agk/+zoAP+FON/wBD/wCMf/Bl/wDY0f8ACnG/6H/xj/4Mv/saP+FzH/onHj7/AMEn/wBnR/wuY/8AROPH3/gk/wDs6AD/AIU43/Q/+Mf/AAZf/Y0f8Kcb/of/ABj/AODL/wCxo/4XMf8AonHj7/wSf/Z0f8LmP/ROPH3/AIJP/s6AMa40rVPh58SPCVvZ+Kta1W01q4lt7q31S4Ey4CZBXgYOa9jrxu51nUviH8SfCE9j4Q8R6RbaPcy3F1caxY/Z02lMAKdxyc9q9koA8q8ewt4B8eaf8QLDK2F46WGuxr0KHiOY+6nAJ+g7mvVEdZEV0IKsMgjuKzfEuiQeJPDOoaPdgGK9t2iJI+6SOD+Bwfwrlfg3rVxqnw9hstRz/aGjTPpt1n+9EcD/AMd20Ad7RRRQAUUVR1fWbHQ9Pe81KYRRr0Hdz6AdzSlJRV3sVGLk7R3LU88VtA81xIsUUY3M7nAUepNeReNPiPLqnmWGhs0Nn0ebo0v09B+tYvi3xte+J7gx/Nb2CH5LcHr7t6n+Vc9a2s97cx29pE000hwiIMkmvm8ZmMqr9nR2/M+mwWWxpL2lbft0RFyT6mvSfBfw1e58vUfEUZSLho7U8F/dvQe1bvgv4dQaMI7/AFhUnv8A7yx9Uh/xPv8Al613db4LLbfvK33f5nPjs0venQfz/wAhscaRRrHEqoijCqowAPSh+Y2+hp1Fe8fPnzf4O8I+L/EPwwh8NR6Rp9tplzqsk02oyyFJ7cRz/MDGRkvlSAQenBx1rf8AGPw38ZXd144sNDtLG50/xP8AZp1uprrY8TRbcptxySR14GO9e40UAeNeJ/AXi2fxgNQ8MWUdhdS+Urarb6oVRlQKP39uynecAgY9qk174Y67qNr8RRElq8uvtbGxLSAbvLALZ/u88D/Jr2GigCtp0D2ul2tvKQXihRGI7kKAas0UUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFcP8RfDejXOkT6rdOLO6iX5ZUH+tPZSO+ema7ioLqxtb5VW8topwh3KJUDAH15rGvSVWm4NG9Cq6NRTT+4+Z0jeRgI0ZyegUZr2P4TxXdtoN3Bd200AE+9PNjK7gVHTP0ruI7aCFQIYY0A6BVAqSvPwuXfV6ntOa/yPQxeZfWKfs+W3zCiiivWPICvMtUH9s/tIaNaSH91oejy3qgd5JW8vn8MGvTa808Nj7T+0L4xmbk2thaQKT2DDdj9KAPS6KKKACivL/inBJrvjjwX4UuLu5t9L1aW6e8S2lMbS+VGGVSw5xkmn/8ACgPBH93Vf/BlL/jQB6bRXmX/AAoDwR/d1X/wZS/40f8ACgPBH93Vf/BlL/jQB6bRXmX/AAoDwR/d1X/wZS/40f8ACgPBH93Vf/BlL/jQB6bRXmX/AAoDwR/d1X/wZS/40f8ACgPBH93Vf/BlL/jQB6bRXil54Rsfht8TvBg8LXWoQxatdS295DNdvKkqBMjIY+pr2ugArzLwYBo3xu8a6MPlivkg1OFO2SNsh/FsV6bXmjjyv2nY+3neGT+OJ6APS6KKKACvEviZaatB4jMmpTNPayZNq2MKq/3ceo/XrXttZ2uaJaeINLksb5co3KsPvI3Yj3rjxmHeIpcqep24LErDVeZrQ8A0XQ77X9QW006Eu5+854WMerHsK9u8KeDbDwvbZjAnvHH7y4Ycn2HoK0tF0Ox0DT1tNOhCIOWY/ec+pPetCsMHgI0FzS1l+R0Y3MJ4h8kNI/n6hRRRXpnlBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAV5poZ+x/tG+JrduDfaTbXK+4Q7P616XXmXignQfj34V1cnbBq9nNpUzN0DA+Yg+pPFAHptFFFAHn/xG0DxBceIvDXifwpYQ6pd6HJPusJbgQ+csqBSQ54BGO9Vf+Ex+Kn/AESyD/woIP8ACu71jXdK8PWP2zXNQtrC33bRJcSBAT6DPU1z/wDwtjwF/wBDZpf/AH/FAGJ/wmPxU/6JZB/4UEH+FH/CY/FT/olkH/hQQf4Vt/8AC2PAX/Q2aX/3/FH/AAtjwF/0Nml/9/xQBif8Jj8VP+iWQf8AhQQf4Uf8Jj8VP+iWQf8AhQQf4Vt/8LY8Bf8AQ2aX/wB/xR/wtjwF/wBDZpf/AH/FAGJ/wmPxU/6JZB/4UEH+FH/CY/FT/olkH/hQQf4Vt/8AC2PAX/Q2aX/3/FH/AAtjwF/0Nml/9/xQByo0/wAd+M/H3hrUfEfhW38O2OhzyXDSf2nHctMWXAUBBx+Nes1gaN468LeIr37Homv2F9c7S3kwzgsQOpA6mt+gArzRD9o/adkK8/ZvDQVvYtPmvS68x+HBOt/Ezx14kA3Qfa49Nt29oVw+PYkigD06iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK4T4waDc6z4BlutLUnU9HmTUbMr1DxnJwO+V3DHriu7pCAykEZBGCKAMnwr4gtvFPhXT9asjmK8hV8Z+638Sn3ByPwrXryjwxKfhp8SLnwjeNs0LXZGu9GlY4WKUn54M/qB9O5r1egDy74jWVtqnxe+HlhqMEd1ZyvfNJbzKGRysSkEqeDg12n/CDeFP8AoWtJ/wDAKP8AwrM8deB7nxVd6TqWkay+javpEkjW10sIlADrtYFTwcgCsX/hCviZ/wBFPX/wTRf40Adb/wAIN4U/6FrSf/AKP/Cj/hBvCn/QtaT/AOAUf+Fcl/whXxM/6Kev/gmi/wAaP+EK+Jn/AEU9f/BNF/jQB1v/AAg3hT/oWtJ/8Ao/8KP+EG8Kf9C1pP8A4BR/4VyX/CFfEz/op6/+CaL/ABo/4Qr4mf8ART1/8E0X+NAHW/8ACDeFP+ha0n/wCj/wo/4Qbwp/0LWk/wDgFH/hXJf8IV8TP+inr/4Jov8AGj/hCviZ/wBFPX/wTRf40AZvjbQtJ0T4pfDyTRtMtLB5dQmWRrWBYy48vodoGa9crzfTvhr4hn8W6TrXjDxk2trpLvLawJYJAA7DGSVPP0r0igDnPH/iePwh4G1LV2I86OIpbr3eZuEAH1Iqp8LvDL+FPh7p1jcj/TZENxdserTP8zZPcjOPwrlbxz8UPitFYQHf4a8JzCW5cfdur0dEHYhO/vn1FesUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAc5458G2fjfwzLpl2xhmUiW1ulHz28w+64+n6jNYPw/8c3l1fTeD/GqrZ+KtPXB5wmoRDpNGe+QOQPfpyB6DXLeOPAWneNrCLz3kstTtG8yw1O34mtZOxB7j1Xv7HBAB1NFeW6V8RtU8H6hFoHxYhW1kYhLXXoUP2W79N5/5Zv6546ngYz6fDNFcwJNbyJLFIoZHRgVYHoQR1FAD6KKKACiiigAooqK6u7extJLq9njt7eJS0ksrhVQDqSTwBQBLXm3j3xhf6lqw8B+ApBJr12n+mXiH5NLgP3nYjo+DwOvI9s07/wAd658QbyTRvhYhislby7zxJcRkRRDuIQfvt7/4g12XgvwRpfgfRzZ6YrSzzN5l3ezHdNdSd2du/U8dqALHhHwrp3gzwza6LpKEQwL80jfelc/edj3JP+HStuiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAq6jptlq9hLZapaw3dtKMPFMgZWH0Necv8LdZ8KzNcfC/xHLpsRO5tJ1DM9q3+7n5k+ozXqFFAHmC/Ejxf4f8A3fjXwHesqfevtFYXERHrt+8B9TVy1+OngKc7bjVZbCTul7ayREfmMV6HVe5sLO8Xbd2kE49JYw386AOSPxi+HwTd/wAJXp+PTec/lis+6+OvgWFilpqFxqUnZLK0kkJ/HGP1rsP+EX8Phtw0PTd2c5+xx5/lV63srW0ULa20MCjoI4wo/SgDzc/EDxz4j/d+DfAs9nG33b7XX8lAPXyx8x/Olt/hPfeIrlL34n+IJ9dZGDJp1vmCzjP+6OW+pr06igCGzs7bT7OK1sYI7e3iULHFEoVVA6AAdKmoooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA//9k=)

## 10.1 添加远程库

要添加一个新的远程仓库，可以指定一个简单的名字，以便将来引用,命令格式如下：

git remote add [shortname] [url]

本例以 Github 为例作为远程仓库，如果你没有 Github 可以在官网 <https://github.com/>注册。

由于你的本地 Git 仓库和 GitHub 仓库之间的传输是通过SSH加密的，所以我们需要配置验证信息：

使用以下命令生成 SSH Key：

$ ssh-keygen -t rsa -C "youremail@example.com"

后面的 **your\_email@youremail.com** 改为你在 Github 上注册的邮箱，之后会要求确认路径和输入密码，我们这使用默认的一路回车就行。

成功的话会在 **~/** 下生成 **.ssh** 文件夹，进去，打开 **id\_rsa.pub**，复制里面的 **key**。

$ ssh-keygen -t rsa -C "429240967@qq.com"

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/tianqixin/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase): # 直接回车

Enter same passphrase again: # 直接回车

Your identification has been saved in /Users/tianqixin/.ssh/id\_rsa.

Your public key has been saved in /Users/tianqixin/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:MDKVidPTDXIQoJwoqUmI4LBAsg5XByBlrOEzkxrwARI 429240967@qq.com

The key's randomart image is:

+---[RSA 3072]----+

|E\*+.+=\*\*oo |

|%Oo+oo=o. . |

|%\*\*.o.o. |

|OO. o o |

|+o+ S |

|. |

| |

| |

| |

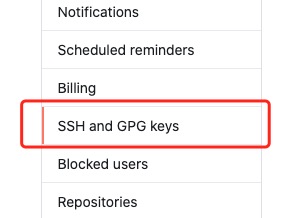
+----[SHA256]-----+

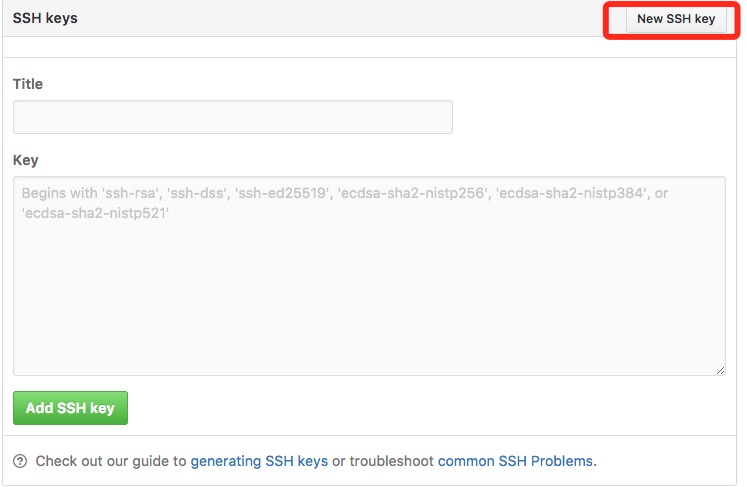
回到 github 上，进入 Account => Settings（账户配置）。

图形用户界面, 应用程序

描述已自动生成

左边选择 **SSH and GPG keys**，然后点击 **New SSH key** 按钮,title 设置标题，可以随便填，粘贴在你电脑上生成的 key。





添加成功后界面如下所示

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

为了验证是否成功，输入以下命令：

$ ssh -T git@github.com

The authenticity of host 'github.com (52.74.223.119)' can't be established.

RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

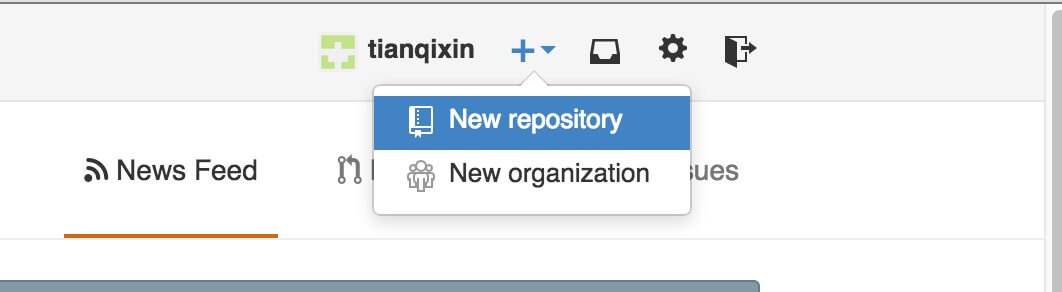
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes # 输入 yes

Warning: Permanently added 'github.com,52.74.223.119' (RSA) to the list of known hosts.

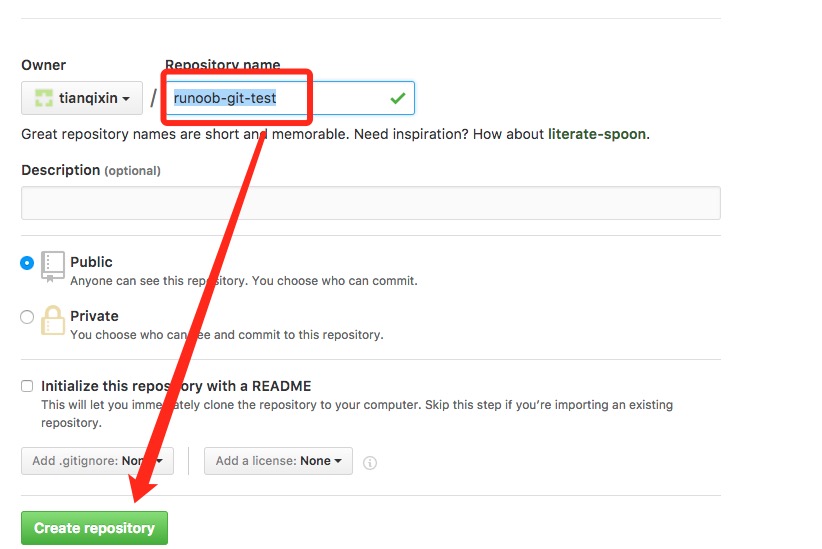
Hi tianqixin! You've successfully authenticated, but GitHub does not provide shell access. # 成功信息

以下命令说明我们已成功连上 Github。

之后登录后点击" New repository " 如下图所示：



之后在在Repository name 填入 runoob-git-test(远程仓库名) ，其他保持默认设置，点击"Create repository"按钮，就成功地创建了一个新的Git仓库：



创建成功后，显示如下信息：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

以上信息告诉我们可以从这个仓库克隆出新的仓库，也可以把本地仓库的内容推送到GitHub仓库。

现在，我们根据 GitHub 的提示，在本地的仓库下运行命令：

$ mkdir runoob-git-test # 创建测试目录

$ cd runoob-git-test/ # 进入测试目录

$ echo "# 菜鸟教程 Git 测试" >> README.md # 创建 README.md 文件并写入内容

$ ls # 查看目录下的文件

README

$ git init # 初始化

$ git add README.md # 添加文件

$ git commit -m "添加 README.md 文件" # 提交并备注信息

[master (root-commit) 0205aab] 添加 README.md 文件

1 file changed, 1 insertion(+)

create mode 100644 README.md

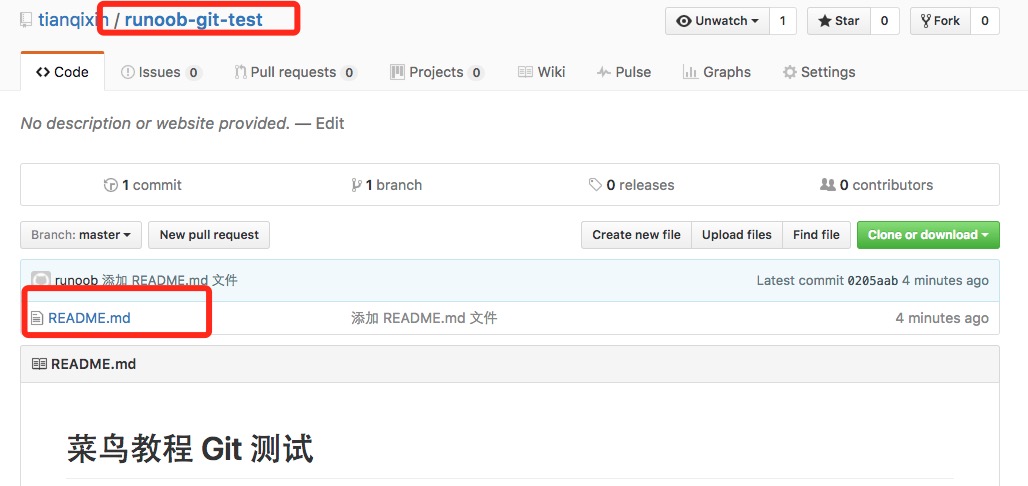
# 提交到 Github

$ git remote add origin git@github.com:tianqixin/runoob-git-test.git

$ git push -u origin master

以下命令请根据你在Github成功创建新仓库的地方复制，而不是根据我提供的命令，因为我们的Github用户名不一样，仓库名也不一样。

接下来我们返回 Github 创建的仓库，就可以看到文件已上传到 Github上：



## 10.2 查看当前的远程库

要查看当前配置有哪些远程仓库，可以用命令：

git remote

### 10.2.1实例

$ git remote

origin

$ git remote -v

origin    git@github.com:tianqixin/runoob-git-test.git (fetch)

origin    git@github.com:tianqixin/runoob-git-test.git (push)

执行时加上 -v 参数，你还可以看到每个别名的实际链接地址。

## 10.3 提取远程仓库

Git 有两个命令用来提取远程仓库的更新。

1、从远程仓库下载新分支与数据：

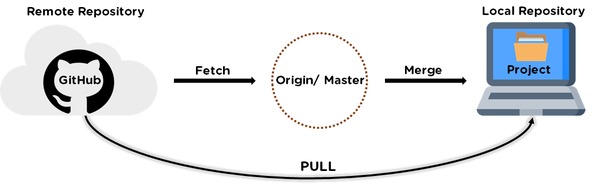
git fetch

该命令执行完后需要执行 git merge 远程分支到你所在的分支。

2、从远端仓库提取数据并尝试合并到当前分支：

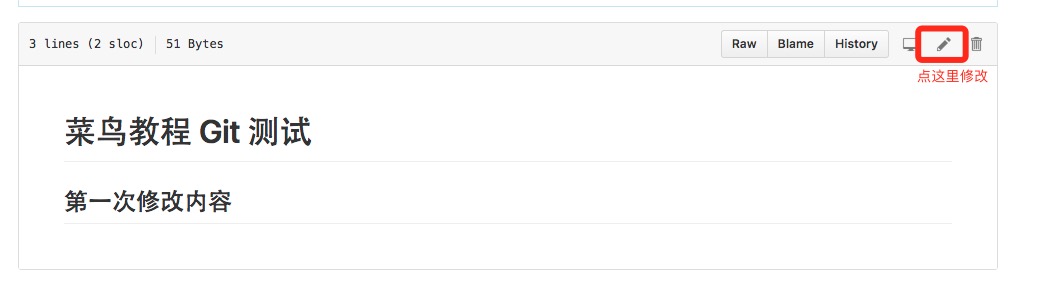
git merge

该命令就是在执行 **git fetch** 之后紧接着执行 **git merge** 远程分支到你所在的任意分支。



假设你配置好了一个远程仓库，并且你想要提取更新的数据，你可以首先执行 **git fetch [alias]** 告诉 Git 去获取它有你没有的数据，然后你可以执行 **git merge [alias]/[branch]** 以将服务器上的任何更新（假设有人这时候推送到服务器了）合并到你的当前分支。

接下来我们在 Github 上点击" README.md" 并在线修改它:



然后我们在本地更新修改。

$ git fetch origin

remote: Counting objects: 3, done.

remote: Compressing objects: 100% (2/2), done.

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

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

From github.com:tianqixin/runoob-git-test

0205aab..febd8ed master -> origin/master

以上信息"0205aab..febd8ed master -> origin/master" 说明 master 分支已被更新，我们可以使用以下命令将更新同步到本地：

$ git merge origin/master

Updating 0205aab..febd8ed

Fast-forward

README.md | 1 +

1 file changed, 1 insertion(+)

查看 README.md 文件内容：

$ cat README.md

# 菜鸟教程 Git 测试

## 第一次修改内容

## 10.4 推送到远程仓库

推送你的新分支与数据到某个远端仓库命令:

git push [alias] [branch]

以上命令将你的 [branch] 分支推送成为 [alias] 远程仓库上的 [branch] 分支，实例如下。

$ touch runoob-test.txt # 添加文件

$ git add runoob-test.txt

$ git commit -m "添加到远程"

master 69e702d] 添加到远程

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

create mode 100644 runoob-test.txt

$ git push origin master # 推送到 Github

重新回到我们的 Github 仓库，可以看到文件已经提交上来了：

应用程序

低可信度描述已自动生成

## 10.5 删除远程仓库

删除远程仓库你可以使用命令：

git remote rm [别名]

### 10.5.1 实例

$ git remote -v

origin    git@github.com:tianqixin/runoob-git-test.git (fetch)

origin    git@github.com:tianqixin/runoob-git-test.git (push)

# 添加仓库 origin2

$ git remote add origin2 git@github.com:tianqixin/runoob-git-test.git

$ git remote -v

origin    git@github.com:tianqixin/runoob-git-test.git (fetch)

origin    git@github.com:tianqixin/runoob-git-test.git (push)

origin2    git@github.com:tianqixin/runoob-git-test.git (fetch)

origin2    git@github.com:tianqixin/runoob-git-test.git (push)

# 删除仓库 origin2

$ git remote rm origin2

$ git remote -v

origin    git@github.com:tianqixin/runoob-git-test.git (fetch)

origin    git@github.com:tianqixin/runoob-git-test.git (push)

# 11、Git Gitee

大家都知道国内访问 Github 速度比较慢，很影响我们的使用。

如果你希望体验到 Git 飞一般的速度，可以使用国内的 Git 托管服务——[Gitee（gitee.com）](https://gitee.com/?utm_source=remote_blog_cnjc)。

Gitee 提供免费的 Git 仓库，还集成了代码质量检测、项目演示等功能。对于团队协作开发，Gitee 还提供了项目管理、代码托管、文档管理的服务，5 人以下小团队免费。

接下来我们学习一下如何使用 Gitee。

由于我们的本地 Git 仓库和 Gitee 仓库之间的传输是通过SSH加密的，所以我们需要配置验证信息。

**1、我们先在** [**Gitee**](https://gitee.com/?utm_source=remote_blog_cnjc) **上注册账号并登录后，然后上传自己的 SSH 公钥。**

我们在 Git Github 章节已经生成了自己的 SSH 公钥，所以我们只需要将用户主目录下的 ~/.ssh/id\_rsa.pub 文件的内容粘贴 Gitee 上。

选择右上角用户头像 -> 设置，然后选择 "SSH公钥"，填写一个便于识别的标题，然后把用户主目录下的 .ssh/id\_rsa.pub 文件的内容粘贴进去：





成功添加后如下图所示：



接下来我们创建一个项目。

点击右上角的 + 号，新建仓库：



然后添加仓库信息：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

创建成功后看到如下信息：



接下来我们看下连接信息：



项目名称最好与本地库保持一致。

然后，我们在本地库上使用命令 git remote add 把它和 Gitee 的远程库关联：

git remote add origin git@gitee.com:imnoob/runoob-test.git

之后，就可以正常地用 git push 和 git pull 推送了！

如果在使用命令 git remote add 时报错：

git remote add origin git@gitee.com:imnoob/runoob-test.git

fatal: remote origin already exists.

这说明本地库已经关联了一个名叫 origin 的远程库，此时，可以先用 git remote -v 查看远程库信息：

git remote -v

origin    git@github.com:tianqixin/runoob.git (fetch)

origin    git@github.com:tianqixin/runoob.git (push)

可以看到，本地库已经关联了 origin 的远程库，并且，该远程库指向 GitHub。

我们可以删除已有的 GitHub 远程库：

git remote rm origin

再关联 Gitee 的远程库（注意路径中需要填写正确的用户名）：

git remote add origin git@gitee.com:imnoob/runoob-test.git

此时，我们再查看远程库信息：

git remote -v

origin    git@gitee.com:imnoob/runoob-test.git (fetch)

origin    git@gitee.com:imnoob/runoob-test.git (push)

现在可以看到，origin 已经被关联到 Gitee 的远程库了。

通过 git push 命令就可以把本地库推送到 Gitee 上。

有的小伙伴又要问了，一个本地库能不能既关联 GitHub，又关联 Gitee 呢？

答案是肯定的，因为 git 本身是分布式版本控制系统，可以同步到另外一个远程库，当然也可以同步到另外两个远程库。

使用多个远程库时，我们要注意，git 给远程库起的默认名称是 origin，如果有多个远程库，我们需要用不同的名称来标识不同的远程库。

仍然以 runoob-test 本地库为例，我们先删除已关联的名为 origin 的远程库：

git remote rm origin

然后，先关联 GitHub 的远程库：

git remote add github git@github.com:tianqixin/runoob-git-test.git

注意，远程库的名称叫 github，不叫 origin 了。

接着，再关联 Gitee 的远程库：

git remote add gitee git@gitee.com:imnoob/runoob-test.git

同样注意，远程库的名称叫 gitee，不叫 origin。

现在，我们用 git remote -v 查看远程库信息，可以看到两个远程库：

git remote -v

gitee    git@gitee.com:imnoob/runoob-test.git (fetch)

gitee    git@gitee.com:imnoob/runoob-test.git (push)

github    git@github.com:tianqixin/runoob.git (fetch)

github    git@github.com:tianqixin/runoob.git (push)

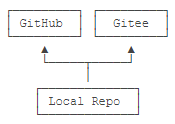
如果要推送到 GitHub，使用命令：

git push github master

如果要推送到 Gitee，使用命令：

git push gitee master

这样一来，我们的本地库就可以同时与多个远程库互相同步：



# 12、Git 服务器搭建

上一章节中我们远程仓库使用了 Github，Github 公开的项目是免费的，2019 年开始 Github 私有存储库也可以无限制使用。

这当然我们也可以自己搭建一台 Git 服务器作为私有仓库使用。

接下来我们将以 Centos 为例搭建 Git 服务器。

### 12.1 安装Git

$ yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel perl-devel

$ yum install git

接下来我们 创建一个git用户组和用户，用来运行git服务：

$ groupadd git

$ useradd git -g git

### 12.2 创建证书登录

收集所有需要登录的用户的公钥，公钥位于id\_rsa.pub文件中，把我们的公钥导入到/home/git/.ssh/authorized\_keys文件里，一行一个。

如果没有该文件创建它：

$ cd /home/git/

$ mkdir .ssh

$ chmod 755 .ssh

$ touch .ssh/authorized\_keys

$ chmod 644 .ssh/authorized\_keys

### 12.3 初始化Git仓库

首先我们选定一个目录作为Git仓库，假定是/home/gitrepo/runoob.git，在/home/gitrepo目录下输入命令：

$ cd /home

$ mkdir gitrepo

$ chown git:git gitrepo/

$ cd gitrepo

$ git init --bare runoob.git

Initialized empty Git repository in /home/gitrepo/runoob.git/

以上命令Git创建一个空仓库，服务器上的Git仓库通常都以.git结尾。然后，把仓库所属用户改为git：

$ chown -R git:git runoob.git

### 12.4 克隆仓库

$ git clone git@192.168.45.4:/home/gitrepo/runoob.git

Cloning into 'runoob'...

warning: You appear to have cloned an empty repository.

Checking connectivity... done.

192.168.45.4 为 Git 所在服务器 ip ，你需要将其修改为你自己的 Git 服务 ip。

这样我们的 Git 服务器安装就完成。

# 13、Sourcetree 使用教程



Git 有很多图形界面工具 ( GUI )，比如 SourceTree、Github Desktop、TortoiseGit 等。

SourceTree 是一个 Git 客户端管理工具，适用于 Windows 和 Mac 系统。

SourceTree 简化了开发者与代码仓库之间的 Git 操作方式，我们可以通过界面菜单很方便的处理 Git 操作，而不需要通过命令。

通过 SourceTree，我们可以管理所有的 Git 库，无论是远程还是本地的。SourceTree 支持 Bitbucket、GitHub 以及 Gitlab 等远程仓库。

## 13.1 Sourcetree 安装

我们可以在 Sourcetree 官网 <https://www.sourcetreeapp.com/> 直接点击下载按钮来下载：



其他版本可以查看：<https://www.sourcetreeapp.com/download-archives>

下载完成后，解压 ZIP 文件，双击应用程序文件（Windows 为 EXE 或 Mac 为 DMG）进行安装。

安装过程中，我们只需要同意 Atlassian 客户协议并点击 Continue 即可。

图形用户界面, 文本

描述已自动生成

创建 Bitbucket 账户可跳过初始设置，加载 SSH 密钥，如果有会默认加载进来，也可以后面再设置。

图形用户界面, 应用程序, Word

描述已自动生成

按照提示操作，直到您完成设置。如果您还没有准备好克隆存储库，请单击 Skip Setup。

## 13.2 连接 Github 账户

如果你还没有 Github 账户需要创建一个，可以参考 [Git 远程仓库(Github)](https://www.runoob.com/git/git-remote-repo.html)。

接下来我们将介绍使用 Sourcetree 管理我们的 Github 账户。

我们可以将多个帐户添加到 Sourcetree。

单击右上角的按钮并选择**帐户**。



接下来我们通过添加账户按钮添加 Github 账户。

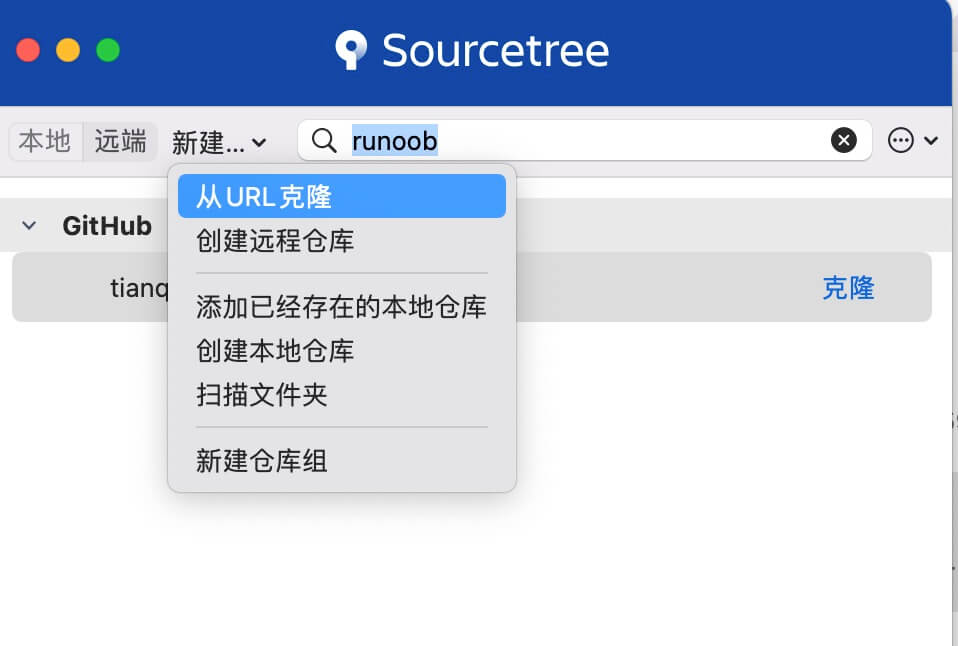


Github 账户添加完成后，就可以直接克隆我们自己的账户：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

也可以克隆其他远程仓库：



克隆完成后，我们就可以进行提交、拉取、合并等操作：



## 13.3 创建本地仓库

单击**新建**，然后选择**创建本地仓库**。

图形用户界面, 应用程序

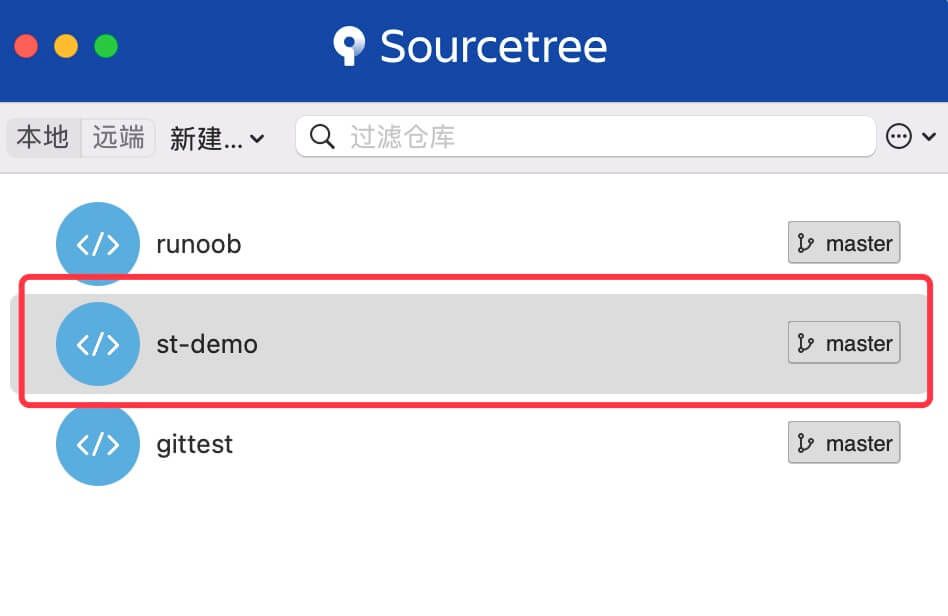
描述已自动生成

输入您的目标路径和存储库名称，然后点击**创建**。

图形用户界面, 文本, 应用程序, 电子邮件

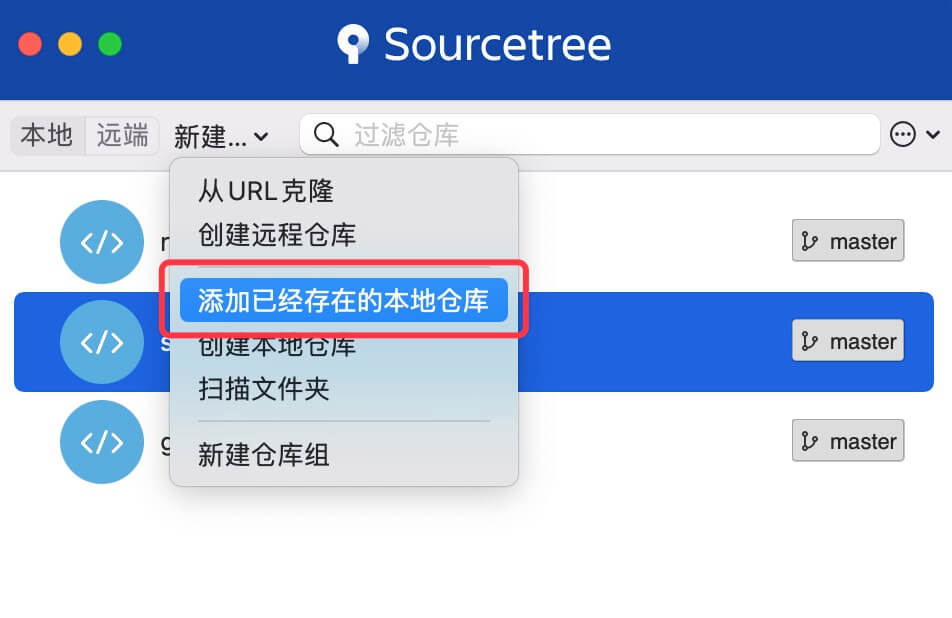
描述已自动生成

创建完成后就会显示在本地的列表中：



如果我们本地已经存在一个仓库，可以直接添加它。

点击**新建**，然后 点击**添加已经存在的本地仓库**：



# 14、Git 测验

### 14.1 [开始](https://www.runoob.com/git/git-quiz.html) https://www.runoob.com/git/git-quiz.html#

### 14.2 其他相关测试

* [Git 测验一](https://c.runoob.com/quiz/7106) https://c.runoob.com/quiz/7106
* [Git 测验二](https://c.runoob.com/quiz/7107) https://c.runoob.com/quiz/7107
* [Git 测验三](https://c.runoob.com/quiz/7108) https://c.runoob.com/quiz/7108