

# Git Talk 1

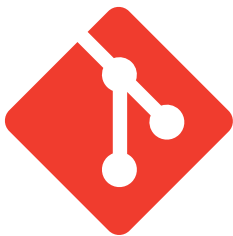


这个指引文档在 [知识共享署名-相同方式共享 3.0 协议](#) 之条款下提供

Available under [Creative Commons Attribution-ShareAlike License](#)

# Git 是什么

- ▶ Git 是目前世界上最先进的分布式版本控制系统
  - ▶ 来自[廖雪峰的官方网站](#)教程
  - ▶ HCC 目前的所有项目，包括幻灯片都在用 Git 管理



git



# Git 设计理念

## ► Git 的项目管理方式

- 工作区
- 暂存区
- 版本库

## ► 例

```
shiyiquan
├─.git
│   └─暂存区
│       └─版本库
│           └─98d22f3
│               └─118532c
├─shiyiquan
└─README.md
```

```
# 项目目录
# 隐藏目录
# 临时存储
# 永久存储
# 一个版本
# 另一个版本
# 工作区文件
# 可随意更改
```



# 基础操作

## ► 创建版本库

```
mkdir hcc      # 创建项目目录
cd hcc         # cd 进去
git init       # 创建 git 目录结构
```

## ► 开始工作

```
cat > jk1      # 假设 jk1 是我们正在编写的文件
```

## ► 提交代码

```
git add jk1     # 工作区 → 暂存区
git add -A      # 添加全部文件
git commit      # 暂存区 → 版本库
                # 第一次提交需设置用户
                # 和 vi 操作一样
```



# 状态查看

## ► 命令

```
git status      # 检查状态
git diff        # 对比工作区和暂存区
git log         # 查看版本库日志
```

## ► 尝试

```
cat >> tmp      # 输入一些内容
git add -A
cat >> tmp      # 再输入一些内容
git status
git diff
git log          # 按 q 退出
```



# 高级选项

```
git log
```

```
--oneline    # 摘要模式（左侧是版本号的前几位）
```

```
--graph      # 显示版本关系图（在复杂时后很有用）
```

```
git commit
```

```
-m "提交说明"    # 不进入 vi
```



# 回溯

## ▶ 命令

```
git reset --hard 版本号
```

```
HEAD^          # 当前版本的前一个版本
```

```
HEAD~10        # 前 10 个版本
```

## ▶ 回溯后将很难看到更靠后的版本

## ▶ 查看操作记录

- ▶ `git reflog`

- ▶ 可以看到以后的版本并跳转回去



# 连接远程版本库

- ▶ 通过连接提供 git 服务的服务器和开发团队同步代码
  - ▶ `git remote add origin 链接`
- ▶ 如何获得链接
  - ▶ 服务商一般会提供链接，复制粘贴即可
  - ▶ 一般链接是有规律的，例如
    - ▶ `https://github.com/用户/项目.git`
    - ▶ `git@github.com: 用户/项目.git`
  - ▶ 链接甚至可以是一个本地的 git 目录
  - ▶ “origin” 是链接的名称，用于快速访问链接





# 推送和接收

## ▶ 推送

- ▶ `git push origin master`

- ▶ 将当前所在分支推送到 origin 的 master 分支

## ▶ 接收

- ▶ `git pull origin master`

- ▶ 将 origin 的 master 分支拉到当前分支

- ▶ “master” 是默认分支名称



# 分支

- ▶ 分支可以代表不同阶段的代码，可以互相合并
- ▶ 创建
  - ▶ `git branch dev`
  - ▶ 从当前分支创建 dev 分支
- ▶ 跳转
  - ▶ `git checkout dev`
  - ▶ 切换到 dev 分支
- ▶ 合并
  - ▶ `git merge master`
  - ▶ 将 master 分支合并到当前分支
  - ▶ 从网络拉取 (`git pull`) 也可能会合并分支



## 分支练习

- ▶ 从 master 开始执行

```
git branch dev
```

```
echo "master ver" >> jkl
```

```
git commit -am "Master" # -am 等于 add 后提交
```

```
git checkout dev # jkl 变了
```

```
echo "dev version" >> jkl
```

```
git commit -am "Dev"
```

```
git merge master # 出现冲突
```



# 冲突

- ▶ 冲突后 git 会提示冲突的文件，只要按照标记手动合并然后重新提交即可

```
<<<<<< HEAD
```

# 当前分支

```
dev version
```

```
=====
```

# 分割线

```
master ver
```

```
>>>>>> master
```

# 合并分支



## 其它功能

<code>git tag</code>	# 标记代码版本
<code>git stash</code>	# 隐藏版本
<code>git config</code>	# 自定义 git
<code>.gitignore</code>	# 忽略文件
<code>git grep</code>	# 全文搜索



# 感谢参加此次活动

