Git 基本指令手册

> 指令中 `[]` 包裹的内容表示可选参数

一、克隆项目到本地

```
git clone 仓库地址 [-b 分支名] [相对路径]
```

例，将下列仓库 `dev` 分支的代码克隆到相对于命令执行上下文的 `./field-workorder` 目录中：

```
git clone http://10.200.1.145/local-outwork.git -b dev ./field-workorder
```

二、基本操作

```
# 切换分支
git checkout 分支名

# 查看分支状态
git status

# 拉取同步
git pull

# 提交
git commit [-m 提交说明]

# 将提交推送到远程仓库
git push
```

三、暂存本地修改

> git stash 指令是跨分支的。

了解更多：https://git-scm.com/docs

一) 指令说明

```
# 暂存，可选参数为暂存注释
# 可执行多次暂存，以形成暂存列表
# 可选参数 save massage 表示可指定暂存备注信息
git stash [save massage]

# 查看暂存列表
git stash list

# 将暂存进度恢复到当前分支
# 注意，暂存堆栈是先进后出的
# 注意，该命令会将恢复的进度从暂存堆栈中删除
```

```
# 可选参数 stashID 表示可应用暂存列表中指定的暂存记录，否则应用最后暂存的记录
# stashID 形式为 stash@{序号}
git stash pop [stashID]

# 将暂存进度恢复到当前分支
# 不同于 git stash pop，该指令不会将内容从堆栈中删除，即该指令能够将堆栈的内容多次应用到多个分支。
# 可选参数 stashID 表示可应用暂存列表中指定的暂存记录，否则应用最后暂存的记录
git stash apply [stashID]

# 删除暂存列表中 ID 为 stashID 的记录
git stash drop stashID

# 清空暂存区
git stash clear

# 查看暂存记录与当前分支差异
# 可选参数 stashID 表示可应用暂存列表中指定的暂存记录，否则应用最后暂存的记录
# 可选参数 -p 表示可查看详细差异
git stash show [stashID] [-p]
```

二) 应用场景

1、开发过程中切换分支紧急修复 bug

```
git stash
git checkout bug修复分支

# 紧急修复 bug 开始
# …
# 紧急修复 bug 结束

git checkout 原开发分支
git stash pop
```

2、切错分支，将开发进度迁移到正确分支

```
git stash
git checkout 正确分支
git stash pop
```

3、提交特定文件

如果对多个文件做了修改，但只想提交几个文件。

```
# 将一个或多个文件添加到待提交中
git add file1 [file2, file3…]
# 暂存其他修改，使其在提交中忽略
# -k 参数指示仓库保持文件的完整
# -u 参数指示仓库包括无路径的文件，比如新的和未添加到git版本管理中的文件
git stash -k -u
# 提交文件
git commit [-m 提交说明]
# 拉取同步
```

```
git pull
# 将提交推送到远端仓库
git push
# 恢复忽略的未提交的文件
git stash pop
```

4、开发过程中临时同步远程代码

```
git stash
git pull
git stash pop
```

5、暂停当前开发任务，先处理其他需求

```
git stash

# 其他内容开发开始
# …
# 其他内容开发结束

git stash pop
```

四、分支管理

一) 新建分支并推送到远端

```
# 假设基于 dev 分支进行分支创建
git checkout dev
git pull
# 新建分支
git checkout -b 新分支名
# 推送至远端
git push origin 分支名
# 或重命名远端分支名
git push origin 分支名:远程分支名
```

二) 删除本地和远程分支

```
# 查看分支列表
git branch
# -d 参数只能删除参与了合并的分支，未合并的分支不能删除；强制删除使用 -D 参数
git branch -d 分支名

# 查看远程分支列表
> git branch -r
> git push origin -d 分支名
```

三) 合并分支

```
# 假如将 iteration1 分支合并到 dev 分支

# 确保要合并的分支代码最新
git pull
# 切换到目标分支
git checkout dev
# 确保目标分支代码最新
git pull
# 合并分支
git merge iteration1
# 提交合并
git push
```

四) 将指定提交记录合并到其他分支

```
# 假设将分支 A 的某次提交合并至分支 B
# 获取某次提交的 commitId
git log
# 复制 commitId 后输入 q 退出 log 视图
q
# 切换至分支 B
git checkout B
# 提取提交记录
git cherry-pick commitId

# 忽略该指令，除非遇到冲突，在冲突解决后执行
git cherry-pick --continue

# 忽略该指令，除非遇到冲突，想放弃合并，回到操作前的样子
git cherry-pick --abort

# 忽略该指令，除非遇到冲突，想退出 cherry-pick 但不回到操作前的样子
git cherry-pick --quit

# 提交合并
git push
```

五) 拉取并切换到远程分支

当需要切换到远程新建分支时，使用下列操作方法。

```
Git common command manual# 方案 1
git checkout -b 新建本地分支名 origin/远程分支名

# 方案 2，同方案 1，默认创建同名本地分支
git checkout --track origin/远程分支名

# 方案 3，方案 2 简写
git checkout -t origin/远程分支名
```

```
# 方案 4，本地分支名不写则默认与远程分支名同名
git fetch origin 远程分支名:[本地分支名]
git checkout 上一步本地分支名

# 方案 5，直接拉取远程所有分支
git fetch
# 当不确定分支名时可查看以确认分支名
git branch -r
git checkout 目标分支名
```

六) 清理分支提交历史记录，且保留最新版本状态

假设需要清理 dev 分支的提交历史记录。

此处需要使用 orphan 参数来创建不包含就提交记录的分支。

了解更多：https://git-scm.com/docs

```
# 使用 --orphan 参数新建不包含历史记录的临时分支
git checkout --orphan temp_branch
# 添加文件
git add -A
# 提交
git commit -am "commit message"
# 删除原来分支
git branch -D dev
# 重命名分支
git branch -m dev
# 提交到远程
git push -f origin dev
```

五、Tag 相关操作

```
# 查看
git tag
# 带筛选查看
git tag -l 'v1.2.4.*'

# 创建
git tag tag名称
# 带注释信息创建
git tag -a tag名称 -m 注释信息

# 删除
git tag -d tag名称
# 提交删除
git push origin :refs/tags/tag名称

# 查看远程
git tag -r

# 推送至远端
git push origin tag名称
```

六、其他常用指令

一) 撤销单个未提交文件的修改

```
git checkout -- 文件名
```

二) 舍去本地修改和提交，对齐远程分支版本

```
git reset --hard origin/分支名
```

三) 放弃更改，直接 git pull 强制覆盖本地

```
# 从远程拉取所有内容
git fetch --all
# reset 本地代码
git reset --hard origin/分支名
# 重新拉取对齐
git pull
```

四) 忽略并不再追踪不想提交的文件

```
git update-index --assume-unchanged file

# 撤销忽略追踪
git update-index --no-assume-unchanged file
```

五) 更换仓库地址

```
git remote set-url origin 新仓库地址
```

更多：Git 官方文档