Git 指南

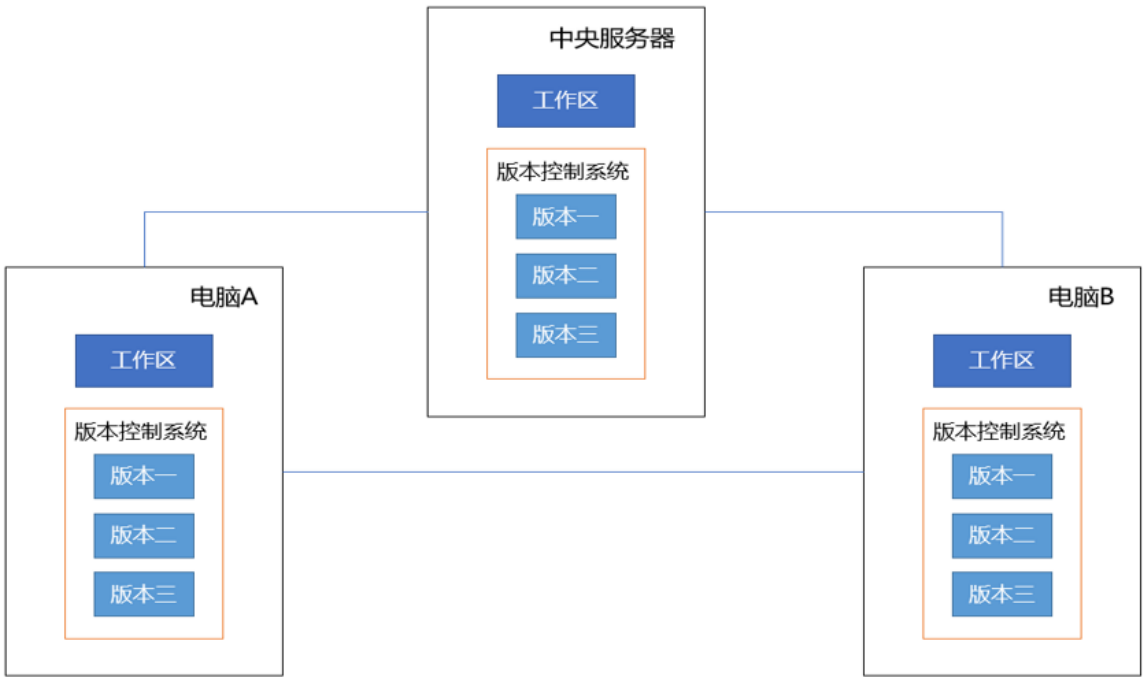
> 本教程涉及命令行指令说明时，[] 表示该参数为可选，<> 仅说明参数意义，需根据实际来定义

Git 简介

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或大或小的项目。利用 git，可以在本地很好地控制项目版本，让项目有条不紊地进行，编写代码错误还能通过回退进行纠正；通过提交、拉取、推送等操作，与队友或同事参与远程共同开发。

分布式版本控制系统

在分布式版本控制系统中，系统保存的不是文件变化的差量，而是文件的快照，即把文件的整体复制下来保存，而不关心具体的变化内容。其次，最重要的是该控制系统是分布式的，开发者从中央服务器拷贝下来代码时，拷贝的是一个完整的版本库，包括历史纪录，提交记录等，这样即使某一台机器宕机也能找到文件的完整备份。



Git 安装

Windows

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/

完成安装之后，就可以在 cmd 或 powershell 等命令行工具使用 git 工具了。一般情况下，在某个文件夹点击右键，你可以看到 Git Bash Here，通过这个也可以打开 Git 工具。

Mac

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/

Linux

尝试在终端输入 `git`，看看系统有没有安装；没有的话输入以下命令安装：

```
sudo apt-get install git
```

Git 配置

Git 提供了一个叫做 git config 的工具，用于配置或读取相应的工作环境变量。这些环境变量决定了 Git 在各个环节的管理员信息和使用方式，存放于 /etc/gitconfig (所有用户生效 –system) 或 ~/.gitconfig (当前用户配置 –global)。

需要设置一下用户信息才能使用 Git 提交 commit，在终端输入：

```
git config --global user.name "yourname"
git config --global user.email "youremail"
```

Git 基本流程

创建仓库

首先在 Github/Gitee 平台创建仓库 (New a repositry)，克隆到本地：

```
git clone <remote_url> [local_dir_name]
```

或者在本地初始化仓库，并添加远程仓库链接：

```
git init    # 初始化仓库
git remote add origin <remote_url>  # 添加远程仓库，并命名为 origin（默认）
```

如果采用 clone 的方式，一定要 `cd` 进入到克隆下来的文件夹里，才能执行后续 Git 操作

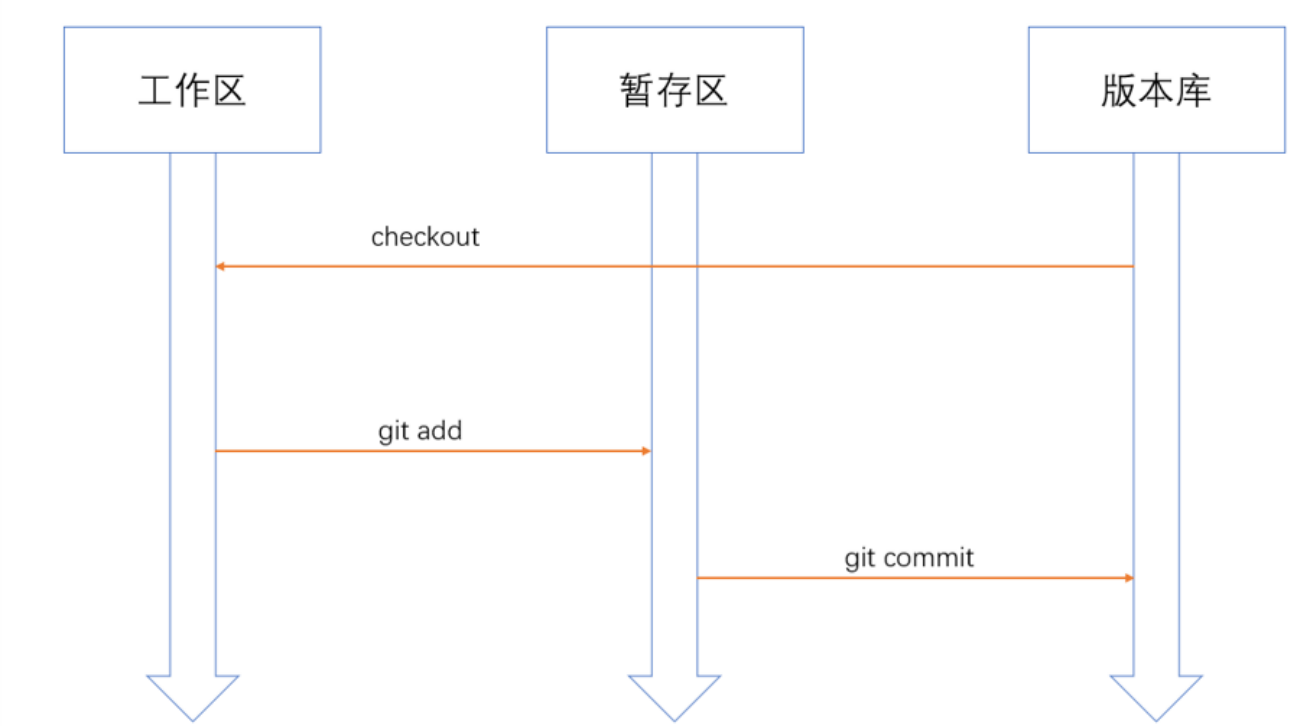
修改提交

```
git add <file_name>         # 将指定文件加入暂存区
git add .                   # 或使用.将项目所有文件加入暂存区（除.gitignore指定的文件
外）
git commit -m "<message>"   # 将暂存区内容打包成commit，提交到本地仓库
git push                    # 将本地仓库内容提交到远程仓库
```

这样就实现了一次修改的提交和推送。

工作区、暂存区和本地仓库

- 工作区：即当前进行工作的文件目录，文件修改但未提交，处于已修改状态（modified）
- 暂存区：运行git add命令后文件保存的区域，也就是下次提交要保存的文件，文件处于已暂存状态（staged）
- 本地仓库：即版本库，记录了提交的完整状态和内容，该区域文件处于已提交状态（committed）



同步远程仓库

在多人合作开发的情况下，需要定期 pull 仓库保持进度同步：

```
git pull [<remote_name> <remote_branch>[:<local_branch>]]
```

分支工作

在项目开发中，常常会建立两个分支：

- master/main：存储生产环境，可部署版本
- dev：存储开发环境

当然，在多人开发中，也可能会为每个人建立分支，大家在各自分支上完成自己的开发任务。

命令	作用
git branch -l	查看所有分支
git checkout <branch_name>	切换到指定分支

命令	作用
git checkout -b <branch_name>	根据当前分支，新建一个分支并切换到该分支上
git merge [<branch1>] <branch2>	将branch2合并到branch1，省略branch1参数时表示合并到当前分支

当合并分支出现冲突时，也就是两个人分别在各自分支上修改了同一处，Git 无法自动合并，则会报错冲突。此时需要手动修改冲突文件，可选择保留哪一分支的内容，之后再 `add/commit/push`。

错误回退

Git 多工作区和版本库机制，允许你回头是岸。

如果发现提交了错误文件：

- 若文件未添加至暂存区：使用 `git checkout <filename>` 将文件替换成暂存区版本，或 `git checkout .` 将工作区内的所有文件替换成暂存区内的文件（谨慎使用）
- 若文件已添加到暂存区但未提交到本地仓库：使用 `git reset HEAD .` 撤销暂存区的所有修改至工作区，接下来就回到了上一步
- 若已提交到本地仓库：使用 `git reset --hard HRAD^` 将所有文件回退至上一版本

本文仅介绍 Git 工作流程，具体指令见 Git 指令全集