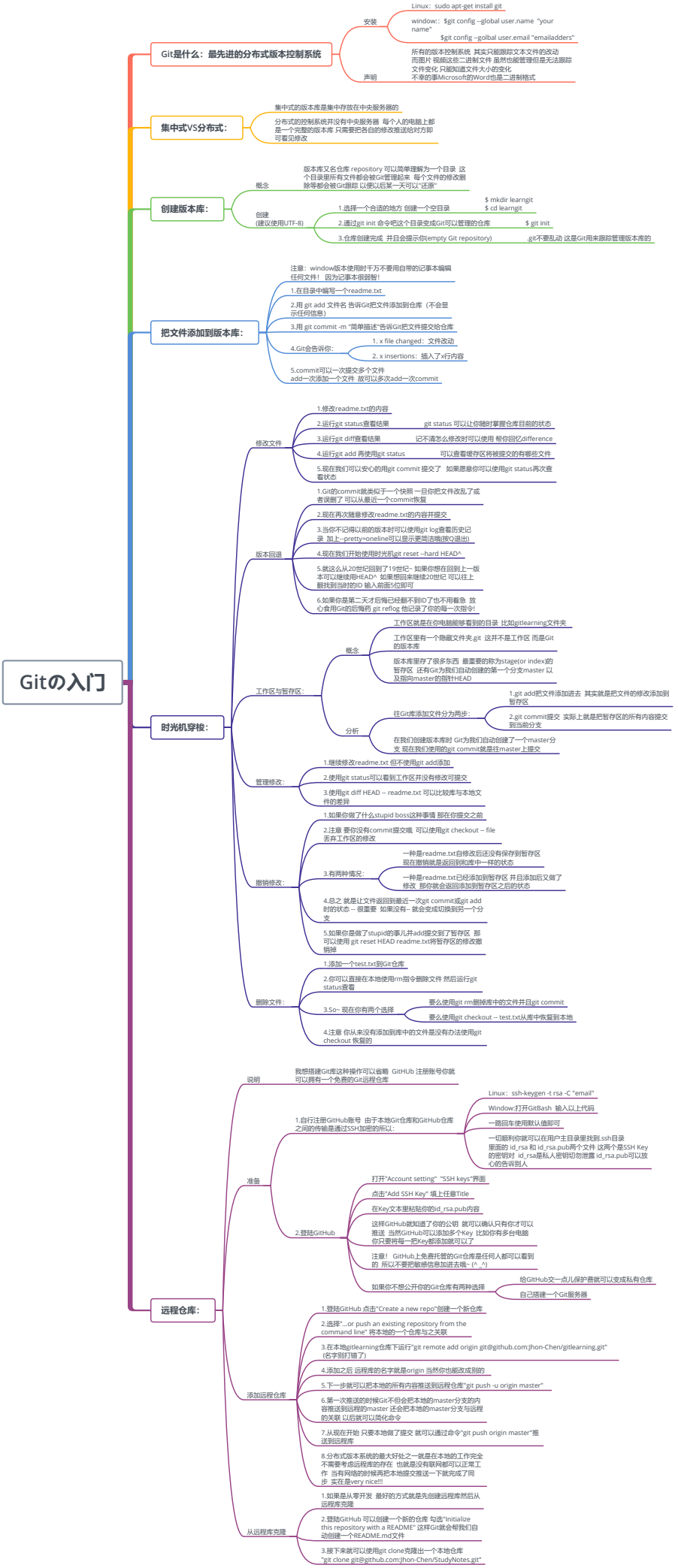
Gitの入门
Git是什么：最先进的分布式版本控制系统
安装
Linux：sudo apt-get install git
window:：$git config --global user.name "your name"
$git config --golbal user.email "emailadders"
声明
所有的版本控制系统 其实只能跟踪文本文件的改动
而图片 视频这些二进制文件 虽然也能管理但是无法跟踪
文件变化 只能知道文件大小的变化
不幸的事Microsoft的Word也是二进制格式
集中式VS分布式：
集中式的版本库是集中存放在中央服务器的
分布式的控制系统并没有中央服务器 每个人的电脑上都
是一个完整的版本库 只需要把各自的修改推送给对方即
可看见修改
创建版本库：
概念
版本库又名仓库 repository 可以简单理解为一个目录 这
个目录里所有文件都会被Git管理起来 每个文件的修改删
除等都会被Git跟踪 以便以后某一天可以"还原"
创建
(建议使用UTF-8)
1.选择一个合适的地方 创建一个空目录
$ mkdir learngit
$ cd learngit
2.通过git init 命令把这个目录变成Git可以管理的仓库
$ git init
3.仓库创建完成 并且会提示你(empty Git repository)
.git不要乱动 这是Git用来跟踪管理版本库的
把文件添加到版本库：
注意：window版本使用时千万不要用自带的记事本编辑
任何文件！ 因为记事本很弱智！
1.在目录中编写一个readme.txt
2.用 git add 文件名 告诉Git把文件添加到仓库（不会显
示任何信息）
3.用 git commit -m "简单描述"告诉Git把文件提交给仓库
4.Git会告诉你：
1. x file changed：文件改动
2. x insertions：插入了x行内容
5.commit可以一次提交多个文件
add一次添加一个文件 故可以多次add一次commit
时光机穿梭：
修改文件：
1.修改readme.txt的内容
2.运行git status查看结果
git status 可以让你随时掌握仓库目前的状态
3.运行git diff查看结果
记不清怎么修改时可以使用 帮你回忆difference
4.运行git add 再使用git status
可以查看缓存区将被提交的有哪些文件
5.现在我们可以安心的用git commit 提交了 如果愿意你可以使用git status再次查
看状态
版本回退
1.Git的commit就类似于一个快照 一旦你把文件改乱了或
者误删了 可以从最近一个commit恢复
2.现在再次随意修改readme.txt的内容并提交
3.当你不记得以前的版本时可以使用git log查看历史记
录 加上--pretty=oneline可以显示更简洁哦(按Q退出)
4.现在我们开始使用时光机git reset --hard HEAD^
5.就这么从20世纪回到了19世纪~ 如果你想在回到上一版
本可以继续用HEAD^ 如果想回来继续20世纪 可以往上
翻找到当时的ID 输入前面5位即可
6.如果你是第二天才后悔已经翻不到D了也不用着急 放
心食用Git的后悔药 git reflog 他记录了你的每一次指令!
工作区与暂存区：
概念
工作区就是在你电脑能够看到的目录 比如gitlearning文件夹
工作区里有一个隐藏文件夹.git 这并不是工作区 而是Git
的版本库
版本库里存了很多东西 最重要的称为stage(or index)的
暂存区 还有Git为我们自动创建的第一个分支master 以
及指向master的指针HEAD
分析
往Git库添加文件分为两步：
1.git add把文件添加进去 其实就是把文件的修改添加到
暂存区
2.git commit提交 实际上就是把暂存区的所有内容提交
到当前分支
在我们创建版本库时 Git为我们自动创建了一个master分
支 现在我们使用的git commit就是往master上提交
管理修改：
1.继续修改readme.txt 但不使用git add添加
2.使用git status可以看到工作区并没有修改可提交
3.使用git diff HEAD -- readme.txt 可以比较库与本地文
件的差异
撤销修改：
1.如果你做了什么stupid boss这种事情 那在你提交之前
2.注意 要你没有commit提交哦 可以使用git checkout -- file
丢弃工作区的修改
3.有两种情况：
一种是readme.txt自修改后还没有保存到暂存区
现在撤销就是返回到和库中一样的状态
一种是readme.txt已经添加到暂存区 并且添加后又做了
修改 那你就会返回添加到暂存区之后的状态
4.总之 就是让文件返回到最近一次git commit或git add
时的状态 -- 很重要 如果没有- 就会变成切换到另一个分
支
5.如果你是做了stupid的事儿并add提交到了暂存区 那
可以使用 git reset HEAD readme.txt将暂存区的修改撤
销掉
删除文件：
1.添加一个test.txt到Git仓库
2.你可以直接在本地使用rm指令删除文件 然后运行git
status查看
3.So~ 现在你有两个选择
要么使用git rm删掉库中的文件并且git commit
要么使用git checkout -- test.txt从库中恢复到本地
4.注意 你从来没有添加到库中的文件是没有办法使用git
checkout 恢复的
远程仓库：
说明
我想搭建Git库这种操作可以省略 GitHub 注册账号你就
可以拥有一个免费的Git远程仓库
准备
1.自行注册GitHub账号 由于本地Git仓库和GitHub仓库
之间的传输是通过SSH加密的所以：
Linux：ssh-keygen -t rsa -C "email"
Window:打开GitBash 输入以上代码
一路回车使用默认值即可
一切顺利你就可以在用户主目录里找到.ssh目录
里面的 id_rsa 和 id_rsa.pub两个文件 这两个是SSH Key
的密钥对 id_rsa是私人密钥切勿泄露 id_rsa.pub可以放
心的告诉别人
2.登陆GitHub
打开"Account setting" "SSH keys"界面
点击"Add SSH Key" 填上任意Title
在Key文本里粘贴你的id_rsa.pub内容
这样GitHub就知道了你的公钥 就可以确认只有你才可以
推送 当然GitHub可以添加多个Key 比如你有多台电脑
你只要将每一把Key都添加就可以了
注意！ GitHub上免费托管的Git仓库是任何人都可以看到
的 所以不要把敏感信息加进去哦~ (^_-)
如果你不想公开你的Git仓库有两种选择
给GitHub交一点儿保护费就可以变成私有仓库
自己搭建一个Git服务器
添加远程仓库
1.登陆GitHub 点击"Create a new repo"创建一个新仓库
2.选择"...or push an existing repository from the
command line" 将本地的一个仓库与之关联
3.在本地gitlearning仓库下运行"git remote add origin git@github.com:Jhon-Chen/gitlearning.git"
(名字别打错了)
4.添加之后 远程库的名字就是origin 当然你也能改成别的
5.下一步就可以把本地的所有内容推送到远程仓库"git push -u origin master"
6.第一次推送的时候Git不但会把本地的master分支的内
容推送到远程的master 还会把本地的master分支与远程
的关联 以后就可以简化命令
7.从现在开始 只要本地做了提交 就可以通过命令"git push origin master"推
送到远程库
8.分布式版本系统的最大好处之一就是在本地的工作完全
不需要考虑远程库的存在 也就是没有联网都可以正常工
作 当有网络的时候再把本地提交推送一下就完成了同
步 实在是very nice!!!
从远程库克隆
1.如果是从零开发 最好的方式就是先创建远程库然后从
远程库克隆
2.登陆GitHub 可以创建一个新的仓库 勾选"Initialize
this repository with a README" 这样Git就会帮我们自
动创建一个README.md文件
3.接下来就可以使用git clone克隆出一个本地仓库
"git clone git@github.com:Jhon-Chen/StudyNotes.git"