Git命令

1. 全局
   1. ssh-keygen -t rsa -C “mail.address@domain.com” #生成密钥
   2. git init # 初始化本地git仓库（创建新仓库）
   3. git clone git+ssh://git@192.168.53.168/VT.git # clone远程仓库
   4. git config --global user.name "xxx" # 配置用户名
   5. git config --global user.email "xxx@xxx.com" # 配置邮件
   6. git config --global color.ui true # git status等命令自动着色
   7. git config --global color.status auto
   8. git config --global color.diff auto
   9. git config --global color.branch auto
   10. git config --global color.interactive auto
   11. git remote add origin git+ssh://git@192.168.53.168/VT.git # 增加远程定义（用于push/pull/fetch）
   12. git whatchanged # 显示提交历史对应的文件修改
   13. git revert dfb02e6e4f2f7b573337763e5c0013802e392818 # 撤销提交dfb02e6e4f2f7b573337763e5c0013802e392818
   14. git ls-tree HEAD # 内部命令：显示某个git对象
   15. git rev-parse v2.0 # 内部命令：显示某个ref对于的SHA1 HASH
   16. git reflog # 显示所有提交，包括孤立节点
   17. git show HEAD@{5}
   18. git show master@{yesterday} # 显示master分支昨天的状态
   19. git log --pretty=format:'%h %s' --graph # 图示提交日志
   20. git show HEAD~3
   21. git show -s --pretty=raw 2be7fcb476
   22. git stash # 暂存当前修改，将所有至为HEAD状态
   23. git stash list # 查看所有暂存
   24. git stash show -p stash@{0} # 参考第一次暂存
   25. git stash apply stash@{0} # 应用第一次暂存
   26. git grep "delete from" # 文件中搜索文本“delete from”
   27. git grep -e '#define' --and -e SORT\_DIRENT
   28. git gc
   29. git fsck
   30. git remote add upstream git://github.com/pjhyett/github-services.git 添加远端repo
   31. git://github.com/pjhyett/github-services.git为“upstream” 重命名远端repo
   32. git remote –v 获取远端git仓库地址
2. 分支
   1. Index区域操作
      1. git add xyz # 添加xyz文件至index
      2. git add . # 增加当前子目录下所有更改过的文件至index
      3. git commit -m 'xxx' # 提交
      4. git commit --amend -m 'xxx' # 合并上一次提交（用于反复修改）
      5. git commit -am 'xxx' # 将add和commit合为一步
      6. git rm --cached <path/filename> #移除缓存区中的文件
      7. git rm -r \* # 递归删除
   2. 差异比较
      1. git status # 查看当前版本状态（是否修改）
      2. git diff # 显示所有未添加至index的变更
      3. git diff --cached # 显示所有已添加index但还未commit的变更
      4. git diff HEAD^ # 比较与上一个版本的差异
      5. git diff HEAD -- ./lib # 比较与HEAD版本lib目录的差异
      6. git diff origin/master..master # 比较远程分支master上有本地分支master上没有的
      7. git diff origin/master..master --stat # 只显示差异的文件，不显示具体内容
   3. 分支操作
      1. git branch # 显示本地分支
      2. git branch --contains 50089 # 显示包含提交50089的分支
      3. git branch -a # 显示所有分支
      4. git branch -r # 显示所有远程分支
      5. git branch --merged # 显示所有已合并到当前分支的分支
      6. git branch --no-merged # 显示所有未合并到当前分支的分支
      7. git branch -m master master\_copy # 本地分支改名
      8. git fetch --all #这样先将同步远程服务器，再可以通过git branch --all 来查看所有分支，包括远端服务器上的分支
      9. git branch -d hotfixes/BJVEP933 # 删除分支hotfixes/BJVEP933（本分支修改已合并到其他分支）
      10. git branch -D hotfixes/BJVEP933 # 强制删除分支hotfixes/BJVEP933
      11. git push --delete origin dev\_cms #删除远端分支dev\_cms
      12. 如何在远端创建一个分支？先在本地创建一个分支，如git branch zhaohuagang ，然后再执行 git push origin 这样就将创建分支事件推送到了远端
      13. git checkout -b master\_copy # 从当前分支创建新分支master\_copy并检出
      14. git checkout -b master master\_copy # 上面的完整版
      15. git checkout features/performance # 检出已存在的features/performance分支
      16. git checkout --track hotfixes/BJVEP933 # 检出远程分支hotfixes/BJVEP933并创建本地跟踪分支
      17. git checkout v2.0 # 检出版本v2.0
      18. git checkout -b devel origin/develop # 从远程分支develop创建新本地分支devel并检出
      19. git checkout -- README # 检出head版本的README文件（可用于修改错误回退）
      20. git merge origin/master # 合并远程master分支至当前分支
      21. git cherry-pick ff44785404a8e # 合并提交ff44785404a8e的修改
      22. git push origin master # 将当前分支push到远程master分支
      23. git push origin :hotfixes/BJVEP933 # 删除远程仓库的hotfixes/BJVEP933分支
      24. git push --tags # 把所有tag推送到远程仓库
      25. git fetch # 获取所有远程分支（不更新本地分支，另需merge）
      26. git fetch --prune # 获取所有原创分支并清除服务器上已删掉的分支
      27. git pull origin master # 获取远程分支master并merge到当前分支
      28. git mv README README2 # 重命名文件README为README2
      29. git reset --hard HEAD # 将当前版本重置为HEAD（通常用于merge失败回退）
      30. git rebase
      31. git ls-files # 列出git index包含的文件
      32. git show-branch # 图示当前分支历史
      33. git show-branch --all # 图示所有分支历史
3. log及tag
   1. git log # 显示提交日志
   2. git log -1 # 显示1行日志 -n为n行
   3. git log -5
   4. git log --stat # 显示提交日志及相关变动文件
   5. git log -p -m
   6. git show dfb02e6e4f2f7b573337763e5c0013802e392818 # 显示某个提交的详细内容
   7. git show dfb02 # 可只用commitid的前几位
   8. git show HEAD # 显示HEAD提交日志
   9. git show HEAD^ # 显示HEAD的父（上一个版本）的提交日志 ^^为上两个版本 ^5为上5个版本
   10. git tag # 显示已存在的tag
   11. git tag -a v2.0 -m 'xxx' # 增加v2.0的tag
   12. git show v2.0 # 显示v2.0的日志及详细内容
   13. git log v2.0 # 显示v2.0的日志
4. git status //查看更改或添加的文件。
5. git add –A // 将更改或添加的文件加入缓存区，-A表示所有的更改和新增
6. git commit –m “注释” // 将缓存区的内容提交到本地的git仓库，这个仓库在git根目录下的.git文件夹下。
7. git push origin dev // origin是远程仓库的别名，可以通过git remote add 别名 仓库路径进行添加.dev是远程仓库的分支，可以通过git branch查看分支，通过git checkout切换分支。

Git 合并分支代码

1. 首先把本地的代码提交到dev分支
2. 然后进行切换分支到test
3. 在test上执行同样的拉代码 git pull
4. 把dev上的代码合并到test缓存区 git merge dev –no-ff

* 把缓存区代码提交到test分支，同dev分支提交代码一样