# 概述

SCM （Software Configuration Management）:

软件配置管理是指通过执行版本控制、变更控制的规程，以及使用合适的配置管理软件，来保证所有配置项的完整性和可跟踪性。配置管理是对工作成果的一种有效保护。如git、svn、p4 等版本控制软件。

Git

Git是一个开源的分布式版本控制系统。

# 安装 & 配置

1. 下载安装http://git-scm.com/download/windows安装完后在开始菜单里会存在“Git”-->“Git Bash”
2. 查看版本号$ git --version
3. 配置名称和邮箱$ git config --global user.name zhangxl$ git config --global user.email zhangxl@butel.com全局配置保存在：$Home/.gitconfig本地仓库配置保存在：.git/config

# Git命令

## 1. 常用命令

1. $ git remote add origin git@10.134.101.126:root/HelloWordDemo.git # 配置远程git版本库
2. $ git pull origin master # 下载代码及快速合并
3. $ git push origin master # 上传代码及快速合并
4. $ git fetch origin # 从远程库获取代码
5. $ git branch # 显示所有分支
6. $ git checkout master # 切换到master分支
7. $ git checkout -b dev # 创建并切换到dev分支
8. $ git commit -m "change log" # 提交
9. $ git status # 查看状态
10. $ git log # 查看提交历史

## 2. 创建远程库

1. $ git clone <url> # 克隆远程版本库
2. $ git init # 初始化本地版本库

## 3. 修改和提交

1. $ git status # 查看状态
2. $ git diff # 查看变更内容
3. $ git add . # 跟踪所有改动过的文件
4. $ git add <file> # 跟踪指定的文件
5. $ git mv <old> <new> # 文件改名
6. $ git rm <file> # 删除文件
7. $ git rm --cached <file> # 停止跟踪文件但不删除
8. $ git commit -m “commit message” # 提交所有更新过的文件
9. $ git commit --amend # 修改最后一次提交

## 4. 查看提交历史

1. $ git log # 查看提交历史
2. $ git log -p <file> # 查看指定文件的提交历史
3. $ git blame <file> # 以列表方式查看指定文件的提交历史
4. $ git log -a # 查看所有提交历史

## 5. 撤销

1. $ git reset --hard HEAD # 撤消工作目录中所有未提交文件的修改内容
2. $ git reset --hard <version> # 撤销到某个特定版本
3. $ git checkout HEAD <file> # 撤消指定的未提交文件的修改内容
4. $ git checkout -- <file> # 同上一个命令
5. $ git revert <commit> # 撤消指定的提交

## 6. 分支与标签

1. $ git branch # 显示所有本地分支
2. $ git checkout <branch/tag> # 切换到指定分支或标签
3. $ git branch <new-branch> # 创建新分支
4. $ git branch -d <branch> # 删除本地分支
5. $ git tag # 列出所有本地标签
6. $ git tag <tagname> # 基于最新提交创建标签
7. $ git tag -a "v1.0" -m "一些说明" # -a指定标签名称，-m指定标签说明
8. $ git tag -d <tagname> # 删除标签
9. $ git checkout dev # 合并特定的commit到dev分支上
10. $ git cherry-pick 62ecb3

## 7. 合并

1. $ git merge <branch> # 合并指定分支到当前分支
2. $ git merge --abort # 取消当前合并，重建合并前状态
3. $ git merge dev -Xtheirs # 以合并dev分支到当前分支，有冲突则以dev分支为准
4. $ git rebase <branch> # 衍合指定分支到当前分支

## 8. 远程操作

1. $ git remote -v # 查看远程版本库信息
2. $ git remote show <remote> # 查看指定远程版本库信息
3. $ git remote add <remote> <url> # 添加远程版本库
4. $ git remote remove <remote> # 删除指定的远程版本库
5. $ git fetch <remote> # 从远程库获取代码
6. $ git pull <remote> <branch> # 下载代码及快速合并
7. $ git push <remote> <branch> # 上传代码及快速合并
8. $ git push <remote> :<branch/tag-name> # 删除远程分支或标签
9. $ git push --tags # 上传所有标签

## 9. 打包

git打包命令会自动忽略.gitignore中指定的目录和文件，以及.git目录。

1. $ git archive --format=zip --output ../file.zip master # 将master分支打包成file.zip文件，保存在上一级目录
2. $ git archive --format=zip --output ../v1.2.zip v1.2 # 打包v1.2标签的文件，保存在上一级目录v1.2.zip文件中
3. $ git archive --format=zip v1.2 > ../v1.2.zip # 作用同上一条命令

## 10. 远程与本地合并

1. $ git init # 初始化本地代码仓
2. $ git add . # 添加本地代码
3. $ git commit -m "add local source" # 提交本地代码
4. $ git pull origin master # 下载远程代码
5. $ git merge master # 合并master分支
6. $ git push -u origin master # 上传代码

## 11. 创建命令别名

1. $ git config --global alias.br="branch" # 创建/查看本地分支
2. $ git config --global alias.co="checkout" # 切换分支
3. $ git config --global alias.cb="checkout -b" # 创建并切换到新分支
4. $ git config --global alias.cm="commit -m" # 提交
5. $ git config --global alias.st="status" # 查看状态
6. $ git config --global alias.pullm="pull origin master" # 拉取分支
7. $ git config --global alias.pushm="push origin master" # 提交分支
8. $ git config --global alias.log="git log --oneline --graph --decorate --color=always" # 单行、分颜色显示记录
9. $ git config --global alias.logg="git log --graph --all --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(bold white)— %an%C(reset)%C(bold yellow)%d%C(reset)' --abbrev-commit --date=relative" # 复杂显示