Linux命令学习

1. set -e -u -x

设置shell的属性

-e 命令错误立即退出脚本而不是继续执行后边命令

-u shell遇到未设置变量抛出错误而不是忽略

-x 命令执行前打印该命令，方便检查脚本

语法：set -eux

1. /bin/sh -c

设置/bin/sh的工作模式为非交互模式

语法：/bin/sh -c

shell脚本常用 /bin/sh与/bin/bash

编写shell脚本的声明可以是 #!/bin/sh 也可以是后者

执行shell脚本的命令 ./xx.sh

1. apt
   1. apt-get update/install/remove/clean packagename
   2. apt-cache show/madison packagename
2. 环境变量设置

export 查询当前环境变量

export PTAH=$PATH:/...

export PATH=/...:$PATH

1. cp命令 cp 要复制文件 复制文件地址 -r-f
2. C语言编译执行以及GO语言编译执行

gcc xxx.c -o xxx(目标文件名)

go run xxx.go

go build xxx.go #默认生成二进制文件xxx go build -o xxx xxx.go

./xxx #执行二进制文件

1. echo XXX 输出XXX
2. 创建新文件 touch 文件名
3. 解压tar.gz

tar -zxvf project.tar.gz 解压命令

tar -zcvf project.tar.gz project 压缩命令

1. 解压jar
   1. jar -xvf xxx.jar 解压
   2. jar -cvf xxx.jar\*/. 压缩
   3. jar tf xxx.jar查看当前jar包内容
   4. jar uf 添加文件到该jar包 jar uf xxx.jar xxx.file
   5. jar cf 创建指定名字jar包并添加文件 jar cf xxx.jar xxx.file
2. 查看当前目录下的jar包是否包含.so文件

for file in $(find ./ -name "\*.jar"); do jar tf $file | grep ".so\$" && echo $file; done

或

#!/bin/bash

for file in $(find ./ -name "\*.jar")

do

#echo "search"

jar tf $file | grep "\.so\$" && echo $file

done

1. mvn 命令

install 安装

clean

<goal> -rf :重构项目名 直接从该项目开始构建，避免重新构建之前项目

1. 修改当前目录或文件的权限（所有者 所属组）
   1. chown -R updateowner:updategrp path or path/\*
2. 查看当前用户
   1. id
   2. whoami
3. rpm打包
   1. 安装rpm-build和rpmdevtools
4. scp传输文件或目录（-r）分别是传文件到远程和接受远程的文件
   1. scp -r bitshuffle loongson@10.130.0.7:~
   2. scp -r username@remote\_host:remote\_directory local\_directory

Git命令学习

/etc/gitconfig 对应--system 对系统中所有用户适用的配置

~./gitconfig 对应--global 对当前文件目录对应的用户适用

配置用户名邮箱等信息

git config --global user.name/email/... “XX”

查看当前配置信息

git config --list

查看某个属性的信息

git config XXX

git clone -b --depth url

git add .

git commit -m

git合并远程更新到本地 fetch+merge 不创建临时仓库

git fetch origin 分支名 如 gie fetch origin main

git log -p main ..origin/main 查看本地和远程的差异

git merge origin/main 合并

如果提示Merge made by the 'recursive' strategy.

也可以使用pull git pull origin main 这样无法处理冲突

git撤销add与commit

撤销add rm命令

撤销commit git reset --hard commit\_id

git撤销commit到上一次，暂存区也撤销到上一次 git reset HEAD^

回退到第前三次提交 git reset --hard HEAD~3

回退到某次 git reset --hard commit\_Id

git 撤销commit到上一次，暂存区不撤销 git reset --soft HEAD^

git反悔撤销commit该如何操作

git reflog 查看历史记录

git cherry -pick HEAD@{数字} 反悔

需要注意反悔时当前版本不能修改，也就是说git status显示当前系统文件没有任何修改

git diff <commit1> <commit2>

查看某次提交与某次提交之间的详细修改记录

git diff staged 查看暂存区和上次提交之间的修改记录

git 创建和提交补丁

1. 将已修改的文件/代码添加到暂存区 git add .
2. 查看提交历史 git log
3. 提交暂存区文件并再次查看历史 git commit -m ‘port to loong64’ git log
4. 创建补丁 git format-patch -1 (-1代表打包最近一次提交信息)

提交补丁时和项目一起提交

1. 将git项目进行同步 git status
2. 执行./generate-new.sh 脚本，自动生成目录及Makefile

提供三个变量 大项目名 具体项目名 版本号

1. 将patach复制到生成目录下
2. 修改当前目录的Makefile 主要关注SOURCE\_URL make -C字段
3. 测试make命令能否成功生成对应镜像，成功后提交

git项目修改后提交流程

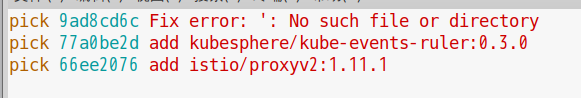
1. git diff 或git status查看修改情况
2. git add . 放置暂存区
3. git commit -m ‘XXX’

git修改某次提交的commit message

git commit --amend 修改最近一次

git rebase -i HEAD~n 修改前某次

进入vim界面



将想修改的message前面的’pick’改为’edit’ 保存退出

根据提示信息 输入git commit --amend 进行修改

最后git rebase --continue保存

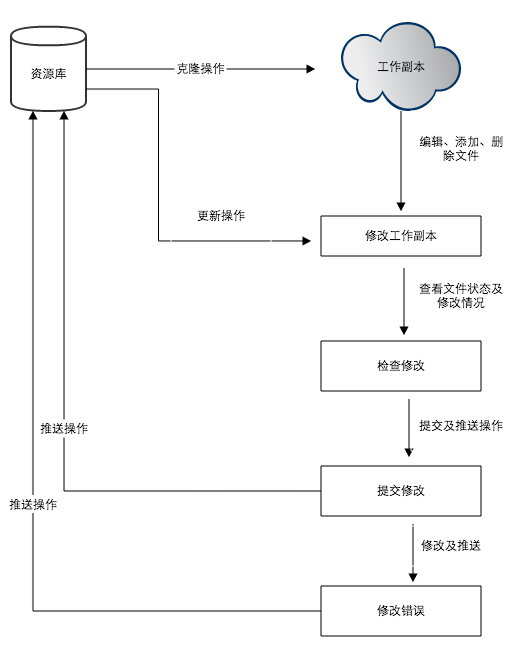
ok

git连接远程仓库，并上传的过程与命令

1. git init 进入项目文件夹，使用该命令将当前目录变成git可以管理的仓库
2. git add . 将文件添加到暂存库 也可以使用git add 指定文件
3. git commit -m ‘’ 提交
4. git remote add origin [git@github.com:lowerpierman/Loongson.git关联远程仓库地址（必须是ssh命令）](mailto:git@github.com:lowerpierman/Loongson.git关联远程仓库地址（必须是ssh命令）)
5. remote重置方法:
   1. git remote -v
   2. git remote rm origin
   3. git remote add ...
6. git pull --rebase origin master 与远程仓库合并
7. git push/git push -u origin master -f
8. git fetch 下载远程仓库所有变化
9. git merge origin/master 将下载的变化与本地合并

Linux配置公钥流程（通过公钥开启SSH连接）

1. 本机创建公钥文件 ssh-keygen -t rsa -C “email”
2. 查看公钥是否生成 cd ~/.ssh ls -l
3. id\_rsa为私钥文件 id\_rsa.pub为公钥文件
4. 将id\_rsa.pub中除邮箱外字符复制到github的SSH中，完成设置



git操作流程：

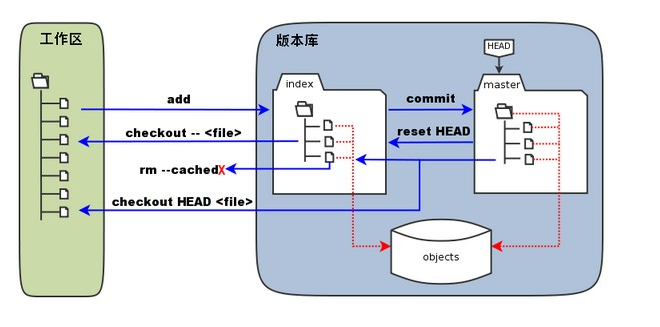
本机从资源库clone对应资源

-》对资源进行修改等操作

-》在修改过程中可以检查上游资源库是否更新，并同步更新

-》在提交本机资源前检查对应上游资源是否存在修改

-》提交修改-》若当前修改错误可以撤回



git主要分为工作区、暂存区以及版本库

在

设置Java和Go的环境变量

Java：

先确定java安装路径

1. which java

显示在/usr/bin/java

ls -l /usr/bin/java 发现不是目录而是link

指向/etc/alternatives/java

确定是否是真正java ls -l /etc/alternatives/java 发现也是link

[root@39ed9065abd2 /]# ls -lrt /etc/alternatives/java

lrwxrwxrwx 1 root root 82 Jul 17 07:07 /etc/alternatives/java -> /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-8.1.10.lns8.loongarch64/jre/bin/java

确定真正java目录

1. find / -name java

/usr/bin/java

/usr/include/google/protobuf/compiler/java

/usr/include/compiler/java

/usr/lib/java

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-8.1.10.lns8.loongarch64/jre/bin/java

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-8.1.10.lns8.loongarch64/bin/java

/usr/lib/jvm/java

/usr/local/include/google/compiler/java

/usr/local/include/google/protobuf/compiler/java

/usr/share/java

接下来配置java路径，可以使用直接在export、编辑/etc/profile或编辑~/.bashrc三种方式，推荐最后一种

profile针对所有用户 bashrc只针对特定用户

export JAVA\_HOME=路径

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

最终source /etc/profile or ~/.bashrc

echo $JAVA\_HOME 判断是否成功

Go：

1.

[root@39ed9065abd2 rocksdb]# which go

/usr/bin/go

[root@39ed9065abd2 rocksdb]# ls -l /usr/bin/go

lrwxrwxrwx 1 root root 20 Mar 29 09:52 /usr/bin/go -> /etc/alternatives/go

[root@39ed9065abd2 rocksdb]# ls -l /etc/alternatives/go

lrwxrwxrwx 1 root root 27 Jul 17 11:39 /etc/alternatives/go -> /usr/lib/golang-1.20/bin/go

2.

[root@39ed9065abd2 rocksdb]# find /usr -name go

/usr/bin/go

/usr/lib/golang-1.20/src/cmd/go

/usr/lib/golang-1.20/src/cmd/vendor/golang.org/x/tools/go

/usr/lib/golang-1.20/src/go

/usr/lib/golang-1.20/bin/go

配置环境变量

vim /etc/profile or ~/.bashrc

export $GOROOT=路径

//此处GOPATH可配置也可不配置

export $GOPATH=$GOROOT/...

export $PATH=$PATH:$GOROOT/bin

source ~/.bashrc

echo $GOROOT