敏捷开发和最佳实践

lass-->DevOps -->practice

Testing（测试环境）

Staging（模拟环境）

createinstallmedia -->gitlab（用于仓库管理系统，用git作为代码管理工具）+Jira（项目与事务跟踪工具）+Jenkins（自动测试环境）+... --->openstack+（K8S管理容器）Docker

GitLab是可以部署到自己的服务器上，数据库等一切信息都掌握在自己手上，适合团队内部协作开发，你总不可能把团队内部的智慧总放在别人的服务器上吧？简单来说可把GitLab看作个人版的GitHub

cwd(current work directory)

(change work directory)

volume 指磁盘卷标，还有音量

GPT直接作为信息源直接给createinstallmedia，不要在其之间用parted的输出字符界面（CLI），因为它没有标准输出，不够健壮。

分区检测 parted检测分区表，由于版本问题或者中文问题导致检测不准确

盲模式启动 grub load linux内核因缺乏video module而导致一段时间内屏幕没有log输出

分区映射 卷标（磁盘）在分区前被mount，导致分区表不能和设备结点同步

自动分区 自动分区支持，不需要依赖之前的分区操作

默认ISO路径 使用当前目录（CWD）作为默认ISO路径

{var%pattern},${var%%pattern} 从右边开始匹配

${var#pattern},${var##pattern} 从左边开始匹配

${var%pattern} ,${var#pattern} 表示最短匹配,匹配到就停止,非贪婪

${var%%pattern},${var##pattern} 是最长匹配

${file#\*/}：删掉第一个 / 及其左边的字符串：dir1/dir2/dir3/my.file.txt

${file##\*/}：删掉最后一个 / 及其左边的字符串：my.file.txt

${file#\*.}：删掉第一个 . 及其左边的字符串：file.txt

${file##\*.}：删掉最后一个 . 及其左边的字符串：txt

${file%/\*}：删掉最后一个 / 及其右边的字符串：/dir1/dir2/dir3

${file%%/\*}：删掉第一个 / 及其右边的字符串：(空值)

${file%.\*}：删掉最后一个 . 及其右边的字符串：/dir1/dir2/dir3/my.file

${file%%.\*}：删掉第一个 . 及其右边的字符串：/dir1/dir2/dir3/my

记忆的方法为：

# 是 去掉左边（键盘上#在 $ 的左边）

%是去掉右边（键盘上% 在$ 的右边）

单一符号是最小匹配；两个符号是最大匹配

${file:0:5}：提取最左边的 5 个字节：/dir1

${file:5:5}：提取第 5 个字节右边的连续5个字节：/dir2

git使用： 粘贴是insert而不是ctrl v

1. mkdir demo
2. cd demo
3. git init
4. echo.>a.txt
5. git add .
6. git commit -m “成功”
7. git remote add origin <http://github/Fateeeee/demo.git>
8. git push origin master

如何使用git查看commit修改了哪些内容，以便后续的代码工作。与之相关的命令就是:

git log 查看所有的commit提交记录

git show 查看提交的详情

git remote -v 查看远程仓库地址

git remote rm origin删除远程仓库地址

git reset --soft HEAD^ 这样就成功的撤销了你的commit

--mixed

意思是：不删除工作空间改动代码，撤销commit，并且撤销git add . 操作

这个为默认参数,git reset --mixed HEAD^ 和 git reset HEAD^ 效果是一样的。

--soft

不删除工作空间改动代码，撤销commit，不撤销git add .

--hard

删除工作空间改动代码，撤销commit，撤销git add .

注意完成这个操作后，就恢复到了上一次的commit状态。

但是一般是创建一个文件，进入这个文件夹，然后在从github上clone下来，修改完后再push上去，如果github上有更新可以使用git pull更新同步到本地

ctrl+F可以查找当前页面字符

要使系统和虚拟机能够交换文件，可以建立共享文件夹（用于在windows上编码，在utunbu上运行，一般在mnt下的hgfs下有该文件夹），或者使用openssh，即可用桌面上的MobaXterm

Mobax使用：创建一个session，填入目标ip地址和用户名

连接虚拟机时，可在虚拟机中先ip -a获得虚拟机的ip，然后ssh 自身ip（ ssh 192.168.17.128

），看看自己是否ping通，自己都ping不通需要在虚拟机中下一个ssh的server（sudo apt install -y openssh-server），然后应该可以了

然后连接上后可以把windows中文件拖进左边列表文件夹中，即可实现文件共享

windows 下回到根目录是cd /,而linux是cd ~

git创建ssh密钥：

1. ssh-keygen
2. 然后一直回车即可
3. 在上面的输出信息中找到ssh密钥存储位置 (C:\Users\john/.ssh/id\_rsa):
4. 然后cd /进入根目录，找到.ssh文件，使用clip<id\_rsa.pub将密钥复制进剪切板
5. 进入github的setting中找个SSH，然后贴进ssh密钥

git push -u origin master 把本地库的所有内容推送到远程库上。把本地库的内容推送到远程，用git push命令，实际上是把当前分支master推送到远程。由于远程库是空的，我们第一次推送master分支时，加上了-u参数(推送和关联)，Git不但会把本地的master分支内容推送到远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

创建和合并分支  
git checkout -b dev创建一个新的分支：dev，并且会切换到dev分支。所以这条命令有两个作用。git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：git branch dev和git checkout dev  
git branch dev，新建分支是新建指针,指向当前commit  
git checkout dev切换到dev分支  
git checkout masterdev分支的工作完成，我们就可以切换回master分支(此时在dev分支的修改在master上是看不到的)  
git merge dev 这是在master分支上执行的命令，作用是：把dev分支上的工作成果合并到master分支上  
git branch -d dev 删除已合并的分支。删除分支就是删除指针  
git branch -D devGit友情提醒，dev分支还没有被合并，如果删除，将丢失掉修改，如果要强行删除，需要使用git branch -D dev命令  
git rebase master变基。在当前分支(非master)下执行该命令，则相当于把当前分支和mater分支合并，和merge操作类似，但提交历史不同，rebase操作的log更干净。具体可参考Git 分支 - 变基

虚拟机转变为字符界面

sudo systemctl set-default multi-user.target

后面把multi-user变为graphical即可回到图形界面

git使用

创建新仓库

创建新文件夹，打开，然后执行

git init

以创建新的 git 仓库。

检出仓库

执行如下命令以创建一个本地仓库的克隆版本：

git clone /path/to/repository

如果是远端服务器上的仓库，你的命令会是这个样子：

git clone username@host:/path/to/repository

工作流

你的本地仓库由 git 维护的三棵“树”组成。第一个是你的 工作目录，它持有实际文件；第二个是 缓存区（Index），它像个缓存区域，临时保存你的改动；最后是 HEAD，指向你最近一次提交后的结果。

添加与提交

你可以计划改动（把它们添加到缓存区），使用如下命令：

git add <filename>

git add \*

这是 git 基本工作流程的第一步；使用如下命令以实际提交改动：

git commit -m "代码提交信息"

现在，你的改动已经提交到了 HEAD，但是还没到你的远端仓库。

推送改动

你的改动现在已经在本地仓库的 HEAD 中了。执行如下命令以将这些改动提交到远端仓库：

git push origin master

可以把 master 换成你想要推送的任何分支。

如果你还没有克隆现有仓库，并欲将你的仓库连接到某个远程服务器，你可以使用如下命令添加：

git remote add origin <server>

如此你就能够将你的改动推送到所添加的服务器上去了。

分支

分支是用来将特性开发绝缘开来的。在你创建仓库的时候，master 是“默认的”。在其他分支上进行开发，完成后再将它们合并到主分支上。

创建一个叫做“feature\_x”的分支，并切换过去：

git checkout -b feature\_x

切换回主分支：

git checkout master

再把新建的分支删掉：

git branch -d feature\_x

除非你将分支推送到远端仓库，不然该分支就是 不为他人所见的：

git push origin <branch>

更新与合并

要更新你的本地仓库至最新改动，执行：

git pull

以在你的工作目录中 获取（fetch） 并 合并（merge） 远端的改动。

要合并其他分支到你的当前分支（例如 master），执行：

git merge <branch>

两种情况下，git 都会尝试去自动合并改动。不幸的是，自动合并并非次次都能成功，并可能导致 冲突（conflicts）。 这时候就需要你修改这些文件来人肉合并这些 冲突（conflicts） 了。改完之后，你需要执行如下命令以将它们标记为合并成功：

git add <filename>

在合并改动之前，也可以使用如下命令查看：

git diff <source\_branch> <target\_branch>

标签

在软件发布时创建标签，是被推荐的。这是个旧有概念，在 SVN 中也有。可以执行如下命令以创建一个叫做 1.0.0 的标签：

git tag 1.0.0 1b2e1d63ff

1b2e1d63ff 是你想要标记的提交 ID 的前 10 位字符。使用如下命令获取提交 ID：

git log

你也可以用该提交 ID 的少一些的前几位，只要它是唯一的。

替换本地改动

假如你做错事（自然，这是不可能的），你可以使用如下命令替换掉本地改动：

git checkout -- <filename>

此命令会使用 HEAD 中的最新内容替换掉你的工作目录中的文件。已添加到缓存区的改动，以及新文件，都不受影响。

假如你想要丢弃你所有的本地改动与提交，可以到服务器上获取最新的版本并将你本地主分支指向到它：

git fetch origin

git reset --hard origin/master

有用的贴士

内建的图形化 git：

gitk

彩色的 git 输出：

git config color.ui true

显示历史记录时，只显示一行注释信息：

git config format.pretty oneline

交互地添加文件至缓存区：

git add -i

先设置文件系统类型和大小进行分区，下建立iso、usb1、usb2三个目录（这个命令就不用说了吧），分别对应iso文件和新建的U盘上的两个分区

然后创建一个文件夹把U盘下的挂载进来

然后把iso镜像文件和createinstallmedia文件，进行安装grub 引导

主要的是createinstallmedia的编写，老师教我们使用python进行了重写，进行理解，但是我可能记不清了，当时也不是学的很清楚，只顾着完成整个流程没注意细节了。

事实上你没有做什么是不？

老师要求我们主要是对这个过程和代码进行理解吧，毕竟是学习嘛。学习了一些linux下的操作和python，还有后面的gitlab一些的

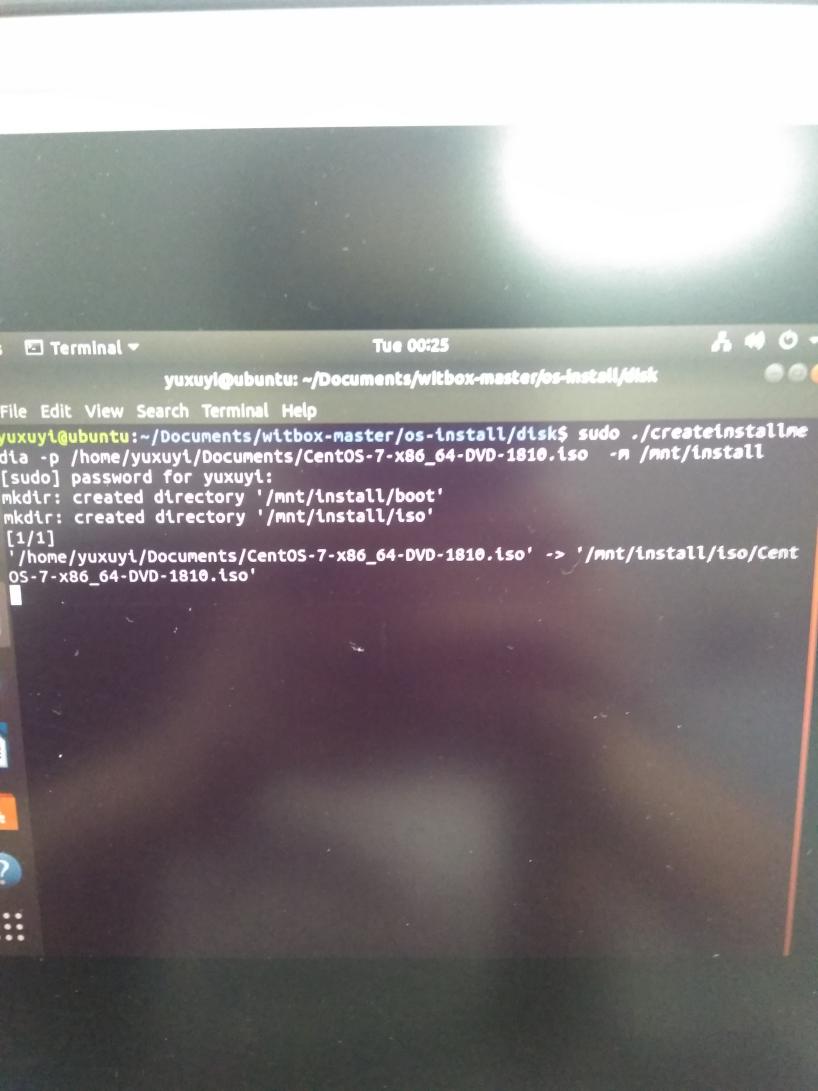
我记得操作不是很复杂，就是经常碰见问题，需要再重新来一次或者使用其他命令

Createinstallmedia 和 一个iso镜像文件进行操作

EFI系统分区（ESP）是一个使用FAT32格式化的小分区，通常为100MB，其中存储已安装系统的EFI引导加载程序以及启动时固件使用的应用程序。

虚拟机创建启动盘

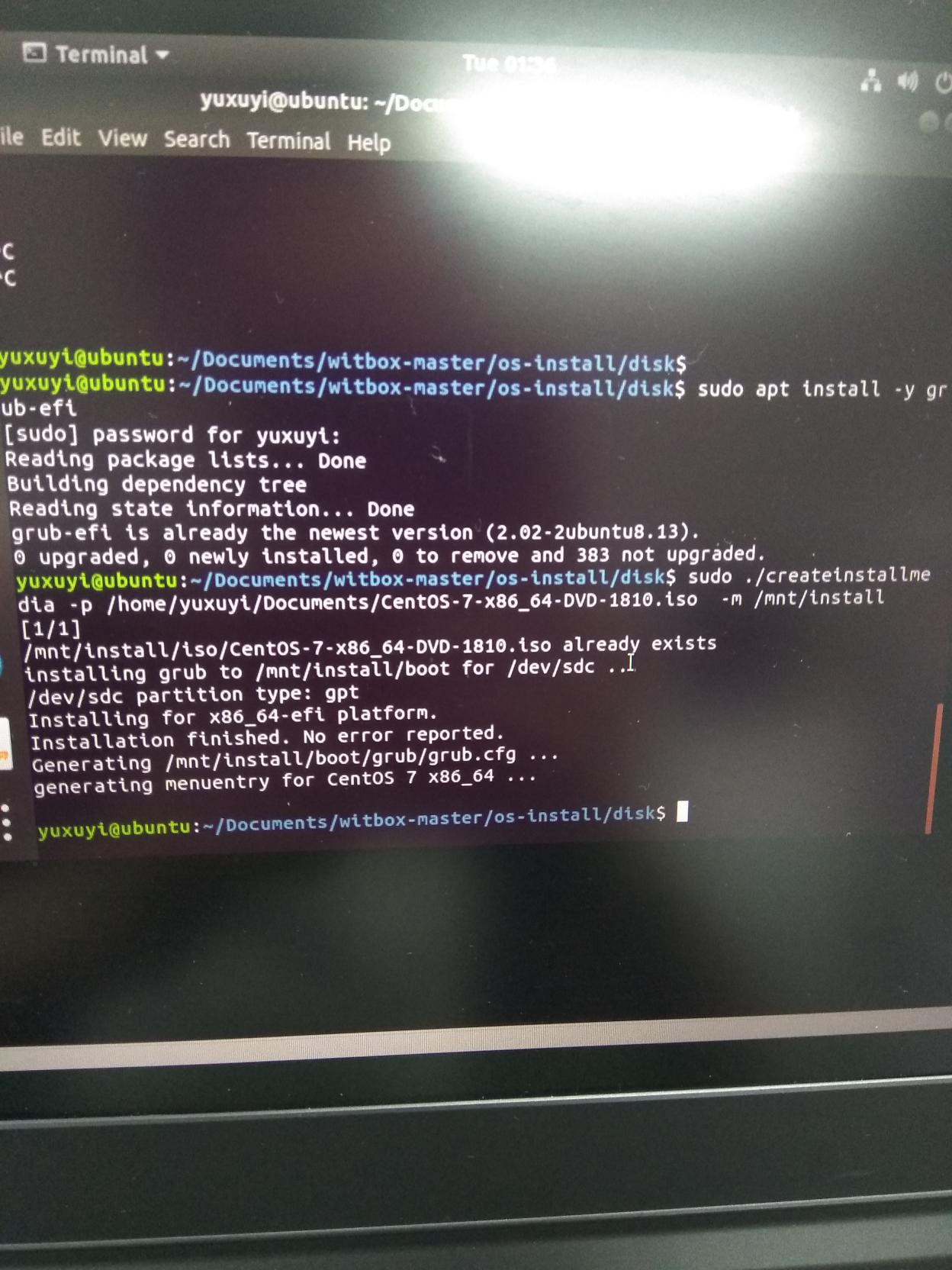
1.sudo fdisk -l //查看硬盘及分区信息

1. sudo cfdisk -z /dev/sdc //进入分区界面（-z表示从零开始）
2. sudo mkfs.vfat -F32 /dev/sdc1 //设置sdc1分区的文件系统类型为vfat
3. sudo mkfs.ext4 /dev/sdc2 //设置sdc2分区的文件类型为ext4
4. 在/mnt下创建一个install文件夹用于将u盘下的sdc2（用于存储）mount进来(sudo mkdir /mnt/install)
5. sudo mount /dev/sdc2 /mnt/install //将u盘的sdc2分区挂载进来
6. sudo parted /dev/sdc print //输出sdc分区信息
7. 

把iso文件和createinstallmedia文件导入进来，然后转到存createinstallmedia文件的文件夹下进入命令行，然后再输入上面代码 （ ISO文件其实就是光盘的镜像文件,）

可能会遇到错误，再输入下面代码

sudo apt install -y grub-efi

1. 

ifconfig -a可以查虚拟机ip 可简写为 ip-a

安装gitlab

1.

CentOS上运行：curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash

1. sudo yum install gitlab-ce -y
2. 配置gitlab

sudo vi gitlab.rb

sudo gitlab-cl reconfigure

sudo gitlab-ctl restart

linux命令 ：

1. 查看用户当前处于系统什么位置 ：pwd
2. ls -l，ls -lh 可以输出当前目录文件详细信息（包括一些权限）
3. ls -a可以输出当前目录下文件，包括隐藏文件和目录（前面有一个点代表该文件为隐藏文件）
4. 上面两个组合即为ls -lha，即可输出详细信息，也可输出隐藏文件
5. cd /表示进入根目录，cd ~表示进入用户在该系统的home目录，即root/user
6. cd ..表示回到上级目录，cd进不去的表示这是个文件
7. vim 命令可以创建文件，如vim .a 表示创建一个名为a的隐藏文件，而vim a表示创建一个普通文件，进入vim后可以进入编辑界面，刚进入是无法编辑的，需要按i进入编辑，按ESC可以退出编辑，然后在按shift+冒号，输入wq回车保存
8. cat可以查看文件，查看隐藏文件在前面加个点即可，cat a可以查看，但不能编辑，more可以更高级地一页一页查看文件，按回车一行一行往下翻，按空格是一页一页往下翻，如more a
9. mkdir创建目录，mkdir -p 可以递归地创建目录，如mkdir -p mul/mul1/mul2表示创建mul文件夹，并在mul下创建子文件夹mul1，以此类推
10. tree 可以以树形结构查看多级目录，如tree mul，需要安装（yum -y install tree）
11. mv用于移动目录，而不是删除，mv mul1 mul2表示把mul1移到mul2里，注意不能把文件或目录移动到文件，不然就是替换文件了
12. 可以用mv把文件向下移动到多级目录下，如mv lin mul/mul1/mul2 即把lin移到mul2下去
13. 也可用mv把文件向上移动 如mv lin ../../../
14. 移动时注意，要移动该文件，必须cd到其当前目录下去
15. rm 删除文件，rm a会询问你是否删除，输入y/n
16. rm -rf 表示删除目录（包括其下子目录）或文件，不进行询问，（-r表示向下递归，不管有多少目录，一并删除，-f表示直接删除，不进行任何确认）
17. ifconfig 查看服务器的ip地址 网关，子网掩码等信息
18. man 查询及帮助命令 如man ls
19. man查询不到的用help，如ls --help
20. find 查找文件 如find / -name \*.mp3 可以用来查找某一类型的全部文件，前面 /表示根目录下找，.表当前目录下找
21. touch 创建普通文件
22. grep 可以查找一个文件内的字符，如grep “b” lin 表示在lin文件中查找b字符
23. tar 表示文件打包命令 ，tar zcvf tmp.tar.gz tmp./ 表示将tmp目录及以下所有文件打包为tmp.tar.gz .表示当前目录下，/表示所有文件
24. du -sh 查看文件大小 du -sh a ，查看a文件大小
25. ll -h 表示查看当前目录下所有文件大小
26. free -m查看内存情况
27. chmod 更改文件权限，chmod 755 a表示更改文件权限为rwx rx，rx
28. 更改目录权限chown
29. date查看时间

Linux scp命令用于Linux之间复制文件和目录。

scp是 secure copy的缩写, scp是linux系统下基于ssh登陆进行安全的远程文件拷贝命令

查看与某台机器的连接情况：ping

查看当前系统的端口使用：netstat -an 该命令在windows下为nestat -aov