Week1

综述

- Linux部分，是所有的基础。这次讲座分享的Linux是我们后面在服务器上部署所面对的，因为服务器一般都在运行Linux，相对更加稳定。更加重要的是，命令行的使用，会在后面前后端的编程中经常使用到，同时也是程序员必备的技能。大家现在所用的鼠标啥的图形化操作，最后还是要转成命令传给电脑，命令行能让你接触到更多底层的东西。最后，要是大家尝试用命令行连接服务器，就没有图形化操作了。
- Docker部分，是部署网站所需要的。大家可能现在还没有遇到环境配置的问题，因为IDE已经帮你解决了这些烦恼，如VS和Xcode会自动安装好C/C++开发所需要的所有工具。当大家需要把自己编写好的程序放在其他电脑上跑，难免会因为环境问题而报错。而Docker保证了环境的一致，因为容器里面就是一个完整的环境，包含了运行程序所需要的一切。另一方面，Docker也更加安全，里面乱搞也不影响外面的系统，外面系统重启了内部也能继续运行。
- Git部分，是代码管理的选择。告别另存为->版本2的工作模式，Git能帮你更好地管理你写的东西。通过每次的commit输入的信息，你能够精确地回到过去的某个版本，不用担心乱搞而回不去；通过diff查看不同，你能够知道对文件变更，不同担心忘记之前写了啥。在以后大家的多人协作中，也推荐大家使用Git，你也不想和队友一行一行对代码来复制粘贴吧。

后面的练习是帮助大家巩固讲座的内容，加上拓展的部分，就能满足后面开发所需的所有知识了。

Linux

练习

回忆linux的目录结构

1. 查看当前位置绝对路径，和当前目录下的所有文件(包括隐藏文件)
   pwd ls -a
2. 创建一个ASTA文件夹，并在其下创建一个a.txt文件
   mkdire ASTA
   cd ASTA
   touch a.txt
3. 将历史命令中包含cd的命令导入到a.txt(提示:history | grep)
   history | grep cd > a.txt

4. 使用vim对a.txt进行编辑，编辑内容自定
   vim a.txt
   i # 进入编辑模式
   #@#@(内容随意编辑)
   esc # 退出编辑模式
   :wq
5. 使用命令查看a.txt的内容
   cat a.txt
6. 使用命令查看a.txt的详细信息(如权限)
   ls -l a.txt
7. 使用命令修改对a.txt的权限，使得所有人均不可写，再次使用vim尝试修改a.txt的内容
   chmod a-w a.txt
   vim a.txt
   :wq会失败, read only
   只能:q
8. 再次使用命令恢复对a.txt的权限
   chmod a+w a.txt
9. 将a.txt移动或拷贝到与ASTA平级的目录下
   cp a.txt ../
10. 退回到上一层文件夹
    cd ..
11. Linux如何安装软件包，可以尝试安装g++
    sudo apt install g++
12. 如何提升权限以及原理是?
    在原有指令前加上sudo，原理是系统首先会通过/etc/sudoers文件验证用户权限，确认权限后需要输入用户密码，确认密码正确，执行sudo命令
13. 补全命令的操作
    tab
14. 选择上一条命令的操作
    上方向键

拓展

1. kernel、shell、terminal三者的关系

command -> terminal -> shell -> kernel -> hardware
终端是提供一个供命令输入输出的设备，命令输入进来后，会被shell解释。而shell本身就是linux内核的一个外壳，可以把接收到的指令解释为内核可以理解的命令。内核执行是驱动了底层硬件。最终结果被层层返回，输出到终端上。

（默认的shell，当然可以自己装喜欢的。大家装了docker之后直接在docker里面开容器就好，因为内核就是Linux）
linux：bash，我们后面都以这个为准。
win：powershell，大部分命令相同吧，但是大家装了git bash的话就用这个，不仅省事还帮你美化了一点点界面。
mac：zsh，和bash基本一样，不方便在linux测试的命令可以在这里写也行。）
2. 如何查看当前电脑运行的所有/特定进程
请参考自强学堂的进程分析
3. 请查看~/.ssh/config文件（如果没有可以自己新建一个）并尝试解读是什么意思
Host tencent
HostName 1.2.3.4
User root
Port 22
IdentityFile ~/.ssh/id_rsa

使用~/.ssh/id_rsa私钥以用户root通过22号端口连接IP地址为1.2.3.4的服务器。
有了这个文件之后，你以后连接服务器就可以直接ssh tencent了。
4. 更多ssh，请参考ssh教程

Docker

练习

1. docker ps和docker ps -a的区别
   docker ps：列出正在运行的容器
   docker ps -a：显示所有的容器，包括未运行的。
2. -i -t -d分别代表什么
   -i 参数是keep STDIN open even if not attached，意思就是会把交互界面保留着
   -t ： allocate a pseudo-TTY.作用是分配一个虚拟的终端
   -d：run container in background and print container ID,就是在后台运行容器
3. 如何帮容器起名字：
   ```
   docker run --name=<容器的别名> -it <imageName>
   ```
4. 如何从零制作镜像
   利用dockerfile写好镜像文件所需要的依赖以及需要打包的文件，通过docker build命令创建镜像，就可以得到镜像文件。
5. 如何打包容器成镜像（例如你对容器进行了很大的修改，然后后面的部署想以这个为准，让其他人的容器都从你这个出发。这时候你就要把你的容器打包成镜像给其他人）
   利用 `docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]` 命令将容器打包成镜像。

拓展

1. docker start、docker stop和docker restart
   举一个例子。例如如果我要对运行中的网站进行维护，我需要终止网站的服务，防止在维护期间还有用户的请求。这时候我可以docker stop ，维护完成后再docker start 。至于docker restart还有这三个命令对挂载文件夹的影响，大家可以自行尝试。
2. docker容器的状态有哪些
   状态有7种：
   created（已创建）
   restarting（重启中）
   running 或 Up（运行中）
   removing（迁移中）
   paused（暂停）
   exited（停止）
   dead（死亡）
3. 容器编号的哈希值
   对该容器的操作只需要输入哈希值的前几位就好，如docker exec -it de3 /bin/bash，其中de3就是那个容器的哈希值的前三位，因为docker会自动帮你匹配的。这个和git reset到某次commit有异曲同工之妙。
4. docker attach和docker exec的区别？为什么建议用docker exec
   都是进入在后台运行的容器，但是docker exec进入容器后再退出，不会停止容器，而docker attach会停止容器。为了防止有意料不到的情况出现，大家就用docker exec，如果需要停止容器，就在外面使用docker stop。
5. 解读命令docker run -it -p 443:443 --restart=always -v asta:/home/asta demo:v1.1.0 /bin/bash
   用标签为v1.1.0的镜像demo启动容器，保留标准输入并且分配虚拟终端，systemctl restart docker之后容器也能自动重启，外部443端口映射到容器内部443端口，外部的asta（相对路径）文件夹挂载到容器的/home/asta，启动容器后运行/bin/bash
6. 启动容器忘记添加某些参数，但是容器又运行起来了，不想关掉重新开容器咋办
   参考docker container update
7. 端口是啥
   简而言之，就是电脑与外界通信的门。例如，你访问一个http的网页，你就是访问那个服务器的80端口，然后服务器通过80端口将网页的内容返回给你，浏览器再渲染，你就能看到网页啦。
8. 想了解更多docker，请戳菜鸟docker

Git

练习

1. 完成讲座布置的pull request
   （提示：fork->clone->write a file->add->commit->push->pull request)

拓展

1. Git能管理文本文件吗？能管理二进制文件吗?
   文本文件指的是ASCII编码、Unicode编码的纯字符文件，我们常用的.md .cpp .py .txt等都属于文本文件。二进制文件内部内容均为01，需要特殊解码才能得到有效内容。可执行程序、图像、音乐等一般都为二进制文件。
   Git作为代码版本管理程序，有强大的管理文本文件地能力。但是二进制文件管理并不是很方便，没有特殊处理的话只能告诉你文件大小的变化，并不能告诉你里面发生了什么变化。
2. 集中式的版本控制你了解多少？和分布式相比，优点和缺点?
   集中式的版本控制系统，所有的版本库是放在中央服务器中的，也就是说我们每一次的修改上传都是保存在中央服务器中的。中央服务器就是个大仓库，大家把产品都堆里面，每一次需要改进和完善的时候，需要去仓库里面把文件给提出来，然后再操作。

常用的集中式版本控制系统包括CVS，SVN等。

集中式版本控制与分布式相比，优点有：

1. 逻辑简单，上手快
2. 代码一致性高
3. 目录权限管理方便
   同时其缺点有：
4. 对服务器性能要求高
5. 必须全程联网
6. 不适合开源开发（开源开发中开发者可能遍布全球）
7. 分支管控不灵活