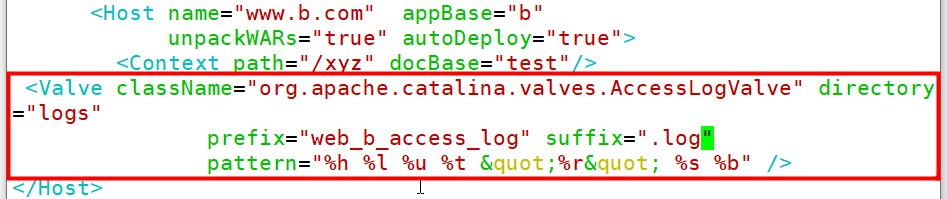
实验一：为tomcat的虚拟主机添加日志功能

1，修改配置文件，在b网站中添加以下内容（可以从最后一页复制），prefix是日志名称，suffix是日志后缀



然后重启tomcat服务

使用浏览器访问[www.b.com:8080](http://www.b.com:8080)

在查看logs目录，b网站日志已经产生

实验二：使用varnish加速网站服务

1. 在proxy 安装varnish

yum -y install gcc pcre-devel readline-devel

yum -y install python-docutils

cd /root/lnmp\_soft

tar -xf varnish-5.2.1.tar.gz

cd varnish-5.2.1

./configure

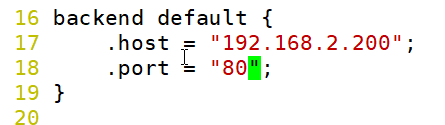
make

make install

useradd -s /sbin/nologin varnish 创建varnish账户

cp etc/example.vcl /usr/local/etc/default.vcl

vim /usr/local/etc/default.vcl



/usr/local/nginx/sbin/nginx -s stop 关闭nginx服务避免80端口冲突

varnishd -f /usr/local/etc/default.vcl 开启varnish服务

然后到web2主机

[root@web2 ~]# /usr/local/nginx/sbin/nginx -s stop

[root@web2 ~]# systemctl restart httpd

把防火墙都关闭

最后使用客户机访问192.168.2.5，可以看到web2的页面

------------------------

svn 集中式数据管理工具，客户机使用时必须与服务器都是在线状态，数据存在服务器

git 分布式数据管理工具，客户机使用时仅仅传输数据才需要与服务器同时在线，数据可以存在每台客户机中

https://github.com/

<https://gitee.com/>

实验三：部署git环境并测试基本操作

1，准备两台虚拟机，安装git

proxy git服务器

web1 git客户端

在proxy主机 （服务器）

[root@proxy ~]# yum -y install git 装服务

2，再服务端创建仓库

[root@proxy ~]# mkdir /var/lib/git 为git创建仓库（数据）的目录

[root@proxy ~]# git init /var/lib/git/project --bare 使用init初始化（创建）一个名为project的空仓库

3，再客户端使用仓库

到web1主机 （客户机）

[root@web1 tomcat]# yum -y install git

[root@web1 tomcat]# cd 回家目录

git clone [root@192.168.2.5:/var/lib/git/project](mailto:root@192.168.2.5:/var/lib/git/project) 克隆git服务器的project仓库

cd project 进入仓库目录

echo abc > web1\_1.txt 创建文件

git add . 提交到暂存区，通知git有哪些新文件产生，或者改动了哪些文件

git config --global user.email "you@example.com" 定义邮箱

git config --global user.name "Your Name" 定义名字

git commit -m "web1\_1" 提交到本地仓库，-m后面写日志记录，可以说明为什么要进行本次提交

git config --global push.default simple 定义新git使用习惯

git push 将本地仓库的最新数据提交到远程服务器

[root@web1 project]# git log 查看完整日志

[root@web1 project]# git log --pretty=oneline 查看精简日志

[root@web1 project]# git log --oneline 最精简日志

[root@web1 project]# git reflog 查看本机操作日志

[root@web1 project]# git status 查看仓库状态

4，测试git pull命令

首先在proxy主机使用克隆命令复制本机仓库

cd 可以克隆到家目录

git clone root@192.168.2.5:/var/lib/git/project

然后在web1，上传新文件，或者新改动的文件

echo wer >> web1\_2\_1.txt 这里是改动了原有的文件

git add . 将这个改动提交给git

git commit -m "web1\_2\_2" 再提交到本地仓库保存

git push 再提交到远程服务器

然后再到proxy主机使用git pull命令，可以将web1上传到服务器的新数据下载到本机

5，测试HEAD指针，此功能可以将数据还原到任意版本，HEAD{0}标识了目前所在的版本

git log --oneline 先查询日志

git reset --hard 87fe 找到还原节点，再hard后面输入

git reflog 查看本机操作日志

实验四：在git中使用分支

[root@web1 project]# git branch 查看所在分支，master是默认分支，\*代表当前所在分支

[root@web1 project]# git branch hotfix 创建hotfix分支

[root@web1 project]# git checkout hotfix 切换到新分支

echo 123 > web1\_3.txt 在新分支中创建文件

git add . 提交到暂存区

git commit -m "web1\_3\_hotfix" 提交到仓库

[root@web1 project]# git checkout master 回到默认分支

ls 查看发现此分支没有新创建的文件，因为文件属于hotfix分支

echo 456 > web1\_3.txt 在master也创建同名文件，但内容不同

git add . 提交

git commit -m "web1\_3\_master" 提交

git merge hotfix 目前在master分支，将hotfix分支的文件合并到master分支，但发生冲突，因为有同名文件，且内容不同

vim web1\_3.txt 使用vim修改文件，文件的最终状态就相当于处理完冲突的结果

实验五：无密码连接git服务器

1，在web1创建密钥，并上传到服务器端

ssh-keygen -f /root/.ssh/id\_rsa -N '' 非交互式创建密钥，注意最后的两个单引号之间是密码，此处为空代表密钥没有密码

ssh-copy-id 192.168.2.5

1. 在web1的project仓库创建新文件

git add . 创建好之后提交

git commit -m "xxxxx"

git push 发现即使是与服务器进行交流也无需密码了