operation day 06

================================================

cdn 内容分发网络

**1，首先使用web1主机开启httpd服务**

[root@web1 ~]# ss -ntulp | grep :80 //可以先查询80端口是否被占用

killall nginx //如果nginx占用就杀掉

[root@web1 ~]# systemctl start httpd //仅仅启动httpd

使用proxy测试web1的页面，web1的防火墙需要关闭

[root@proxy ~]# curl 192.168.2.100

[root@proxy ~]# ss -ntulp | grep :80 //检查自身80端口是否被占用,如果有

就把相关服务关掉

yum -y install gcc readline-devel pcre-devel python-docutils

./configure

make

make install

useradd varnish -s /sbin/nologin //创建varnish所需账户

cp etc/example.vcl /usr/local/etc/default.vcl //拷贝配置文件

vim /usr/local/etc/default.vcl //修改配置文件17、18行

.host = "192.168.2.100"; //原始站点服务器的ip

.port = "80"; //原始站点服务器的端口号

varnishd -f /usr/local/etc/default.vcl //指定配置文件路

径并启动varnish服务

[root@proxy varnish-5.2.1]# curl 192.168.2.5 测试访

问proxy的页面，可以看到原始服务器web1的内容

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

程序(文件或者数据)版本管理工具

svn 集中式 用户使用该服务时,需要与服务器保持在线状态, 数据统一

保存在服务端

git 分布式 用户使用该服务时,不需要与服务器保持在线状态, 仅仅传递

数据时需要联网,数据保存在git服务器与git客户端

码云 gitee

linux内核 海量外围软件

test01.sh test02.sh test03.sh

01 02 03

**1，基本操作**

首先在web1与web2安装git工具

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

[root@web1 ~]# mkdir /var/lib/git

[root@web1 ~]# git init /var/lib/git/project --bare //创建空仓库，叫Project

[root@web2 ~]# yum -y install git

[root@web2 ~]#git clone 192.168.2.100:/var/lib/git/project //在客户机

克隆服务器的仓库

[root@web2 ~]#cd project //进入仓库

[root@web2 project]# echo "web2\_01" > web2\_01.txt //创建测试文件

[root@web2 project]# git add . //提交到暂存区

[root@web2 project]# git commit -m "web2\_01.txt" //将文件保存到仓库中，-m后

面的内容是日志提示信息，首次保存会失败，按照下面2条命令输入邮箱和用户

名即可

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

git config --global user.name "Your Name"

[root@web2 project]# git commit -m "web2\_01.txt" //再次提交文件保存到仓库中

[root@web2 project]#git push 将本地仓库中的数据推送到远程服务器，首次推送可以

按ctrl+c终止，按提示输入以下习惯配置

[root@web2 project]# git config --global push.default simple //配置使用习惯

[root@web2 project]#git push //再次推送到远程服务器

所有通过commit提交的记录都可以通过日志查看

git log //查看完整日志

git log --pretty=oneline //查看精简日志

git log --oneline //查看最精简日志

git reflog //查看本机操作记录