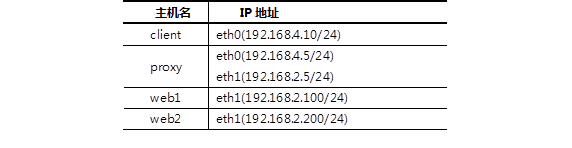
# 2019.11.13 day1

## 搭建Nginx服务器

克隆4台机器:



unix和linux平台下的web服务器:

apache,nginx,tengine,lighttpd 用于静态网站,php网站

tomcat,IBM WebSphere,Jboss 只能用于java的网站

windows平台下的web服务器:

iis(internet information server)

传统的权限都是针对用户的

程序没有权限,程序的权限是登陆的用户赋予的.

linux安装软件的时候会新建对应服务的用户

在编译安装源码包的时候自定义对应的用户和管理组

启动软件的时候会以某一个用户的身份来启动,如果没有定义的时候会以root身份来启用

SUID 普通用户执行命令时获取root权限

## \*\*使用源码包安装nginx软件包步骤:

yum -y install gcc pcre-devel openssl-devel        #安装nginx依赖包

useradd -s /sbin/nologin nginx

tar -xf nginx-1.12.2.tar.gz

cd nginx-1.12.2

./configure \ \换行的意思

--prefix=/usr/local/nginx \ #指定安装路径

--user=nginx \ #指定用户

--group=nginx \ #指定组

--with-http\_ssl\_module #开启SSL加密功能模块

make && make install #编译并安装

安装成功/usr/local/nginx目录里面有四个目录 logs sbin conf html

注释:

--with-http\_ssl\_module 启用模块功能

--without-http\_ssl\_module 关闭模块功能

指定一个没有登录服务器的用户

## nginx命令的用法:

/usr/local/nginx/sbin/nginx #启动服务

/usr/local/nginx/sbin/nginx -s stop #关闭服务

/usr/local/nginx/sbin/nginx -s reload #重新加载配置文件(不关闭服务)

/usr/local/nginx/sbin/nginx -V #查看软件信息

使用netstat -ntulp |　grep nginx查看服务是否开启,如果开启的话端口会有服务使用

## 使用linux内部浏览器打开:curl

curl http://192.168.4.5

-i information 显示request数据包头返回的信息

nginx的配置文件路径

/usr/local/nginx/conf/nginx.conf (路径是相对于编译安装时候自定义的路径)

## 用户访问web登录账号密码认证:

修改配置文件添加内容

auth\_basic "Input Passwd:"; 指在访问网站的时候输出的指引信息

auth\_basic\_user\_file "/usr/local/nginx/passwd"; 注意账号秘密存放目录

存储用户密码的目录,不用手动创建,安装httpd-tools软件包

主虚拟主机配置修改如下:

...

server {

listen 80;

server\_name www.example.com;

root html;

auth\_basic "Input Passwd:";

auth\_basic\_user\_file "/usr/local/nginx/passwd";

}

...

安装httpd-tools软件包,用于nginx创建新的用户和密码

安装完成后会生成htpasswd命令

## \*\*\*创建一个叫tom的用户\*\*\*

htpasswd -c /usr/local/nginx/passwd tom 创建密码文件

注释:c 指创建一个用户,如果指定目录相同的情况下,只能使用一次,如果再次出入会覆盖原用户信息

注意:配置文件里的存放的账号和密码的目录,要和创建用户时候指定的目录一致

创建完用户后重新加载配置文件

/usr/local/nginx/sbin/nginx -s reload //重新加载配置文件

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

## 基于域名的虚拟主机(一个server一台虚拟主机)

修改配置文件,创建两个虚拟web主机

server {

listen 80;

server\_name www.a.com;

auth\_basic "Input Passwd:";

auth\_basic\_user\_file "/usr/local/nginx/passwd";

location / {

root html;

index index.html index.htm;

}

}

server {

listen 80;

server\_name www.b.com;

location / {

root www;

index index.html index.htm;

}

}

/usr/local/nginx/sbin/nginx -s reload 重启配置文件,如果重启没有报错的话证明修改配置文件没有错误

mkdir /usr/local/nginx/www 创建第二台虚拟web主机网站的家目录

修改客户机的hosts文件,里面添加 192.168.4.5 www.a.com www.b.com

然后可以在客户机上测试网页是否显示成功.

http协议是明文协议

在网络上发送的所有数据包,都是可以被中间人抓包获得明文信息.

加密算法一般分为对称算法、非对称算法、信息摘要。

对称算法有：AES、DES，主要应用在单机数据加密。

非对称算法有：RSA、DSA，主要应用在网络数据加密。

信息摘要：MD5、sha256，主要应用在数据完整性校验。

md5只对数据内容做校验(内容不变md5不变),修改文件名\修改时间\所有者等md5数字不会变化

## md5用法/格式:

md5sum 文件名

案例:

]# md5sum passwd

md5sum passwd

3bf9c8c59ccc63c6bf88bfd9663e085c passwd

]# vim passwd 修改文件后

]# md5sum passwd

f959411f71f9a8c5f9e61230acfd61f3 passwd 对比前面的md5

## sha256sum 文件名:

案例:

]# sha256sum passwd

7ee129e68b66d62da5cf3250f6f90c2098c557ede19e40429ec2f577530f3c5e passwd

]# vim passwd 修改文件后

]# sha256sum passwd

c921d0f08f05267e8cbcb465d52e53cd3cd0ff6575a4855ecbb0e45334d5d63e passwd

## 配置https虚拟主机

cd /usr/local/nginx/conf 进入配置文件目录

openssl genrsa > cert.key 生成私钥

openssl req -new -x509 -key cert.key > cert.pem 生成证书(公钥)

注释:

req request请求

-new 创建新的

-x509 证书的型号

-key 制定私钥是谁

修改配置文件:

server {

listen 443 ssl;

server\_name www.c.com;

ssl\_certificate cert.pem; 这里是证书文件(公钥)

ssl\_certificate\_key cert.key; 这里是私钥文件

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m; 超时时间

ssl\_ciphers HIGH:!aNULL:!MD5; 网站加密的时候不能使用空密码,不能使用md5

ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

/usr/local/nginx/sbin/ngxin -s reload 重启配置服务

在客户端测试访问:

在客户端hosts中添加 192.168.4.5 www.c.com

firefox https://www.c.com 测试登录访问

由于是自己生成的密钥文件,所有浏览器会显示不安全的报错信息,把他添加到信用站点就可以正常访问网页.

# 2019.11.14 day2

## 部署动态网站

LNMP linux nginx mariadb php

LAMP linux apache mariadb php

## LNMP常见问题日志分析位置:

Nginx的默认访问日志文件为/usr/local/nginx/logs/access.log

Nginx的默认错误日志文件为/usr/local/nginx/logs/error.log

PHP默认错误日志文件为/var/log/php-fpm/www-error.log

如果动态网站访问失败，可用参考错误日志，查找错误信息

实验环境准备工作如下:

安装mariadb软件包依赖

yum -y install mariadb (数据库客户端软件) mariadb-server (数据库服务器软件) mariadb-devel (其他客户端软件的依赖包)

安装php软件包依赖

yum -y install php (解释器) php-mysql (PHP的数据库扩展包) php-fpm (进程管理器服务) php和nginx通讯用的软件包

nginx php-fpm 启动服务:

/usr/local/nginx/sbin/nginx 启动nginx服务

systemctl restart mariadb 启动mariadb服务

systemctl restart php-fpm 启动php和数据库通信的服务

查看软件服务启动的状态

netstat -nutlp | grep 80 nginx服务

netstat -nutlp | grep 3306 mariadb服务

netstat -nutlp | grep 9000 php-fpm服务

## php-fpm配置文件路径: /etc/php-fpm.d/www.conf

配置文件内容注释:

listen = 127.0.0.1:9000 //PHP端口号

pm.max\_children = 32 //最大进程数量

pm.start\_servers = 15 //最小进程数量

## location 匹配用户的地址栏(从域名后面开始匹配)

案例:

...

server\_name www.baidu.com

location /abc { allow 192.168.1.1; deny all; }

location /test { deny all; } 用户地址栏输入/test,拒绝所有访问

location ~ /nb { allow all; } 与用户输入地址栏做对比,包含/nb的允许所有访问

(\*\*支持正则表达式匹配)

location / { allow all; } 网站根目录的优先级最低,于放置的位置无关.(可以匹配所有页面)

...

测试:

firefox http://www.baidu.com/abc

firefox http://www.baidu.com/test

firefox http://www.baidu.com/nb

firefox http://www.baidu.com/xxyy

编写php脚本案例:

vim /usr/local/nginx/html/test.php

php格式:

<?php

$i=33; #在php脚本里,指定变量前边都要加$

echo $i; #在php脚本里,每行后面都要加;

?> #<? ?>指定是php脚本的固定格式

如果不做php编译的直接访问结果是,访问网站后浏览器提示是否下载php脚本.

修改nginx主配置文件/usr/local/nginx/conf/nginx.conf

...

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; 请求转发到本机9000端口php解析器

fastcgi\_index index.php;

include fastcgi.conf; 加载其他配置文件

}

...

修改完后重新加载配置文件,然后测试网站内容.

## 创建php网页,测试LNMP架构是否能解析php页面

架构: client-->nginx-->php-->mariadb

复制教学环境准备好的php到网站根目录

cp lnmp\_soft/php\_scripts/mysql.php /usr/local/nginx/html/

总结:

1. 安装nginx , php , php-fpm , php-mysql

启动服务nginx mariadb-server mariadb-devel

1. 修改nginx配置文件nginx-->php

动静分离:(location,类似于if语句的吧..)

location / { root html; }

location ~\.$php { fastcgi\_pass :9000; }

重启配置文件

1. 测试:
   1. test.php 网页 mysql.php 网页

## 地址重写

格式:

rewrite 旧地址 新地址 [选项]

选项:

last 不再读其他rewrite(在同一个location下的情况)

break 不再读其他语句，结束请求(在不同的location下的情况)

redirect 临时重定向(给网络蜘蛛看的标示,蜘蛛指百度,谷歌)

permanent 永久重定向(给网络蜘蛛看的标示,蜘蛛指百度,谷歌)

## 本地子页面跳转,地址栏不变案例:

server {

listen 80;

server\_name localhost;

rewrite /a.html /b.html;            #修改配置地方;本地子页面跳转,地址栏不变

location / {

    root html;

index index.html index.htm;

}

}

## 本地子页面跳转,地址栏改变案例:

server {

listen 80;

server\_name localhost;

rewrite /a.html /b.html redirect;            #修改配置地方;本地子页面跳转,地址栏改变

location / {

    root html;

index index.html index.htm;

}

}

## 域名重定向到百度案例:

server {

listen 80;

server\_name localhost;

rewrite ^/ https://www.baidu.com;            #修改配置地方;请求重定向到百度,使用到正则表达式;^从域名后面开始匹配

location / {

    root html;

index index.html index.htm;

}

}

## 实现重定向到百度的子页面相同案例:

server {

listen 80;

server\_name localhost;

rewrite ^/(.\*) https://www.baidu.com/$1;            #修改配置地方;请求重定向到百度,使用正则表达式,使本网站跳转到百度网站上的子页面相同;^从域名后面开始匹配

location / {

    root html;

index index.html index.htm;

}

}

## 识别浏览器标识的重定向网页案例:

准备两个页面,在不同的目录下:

echo "i am normal page" > /usr/local/nginx/html/test.html

mkdir /usr/local/nginx/html/firefox/

echo "firefox page" > /usr/local/nginx/html/firefox/test.html

添加到配置文件

...

if ($http\_user\_agent ~\* firefox) {  #识别客户端firefox浏览器

rewrite ^/(.\*) /firefox/$1; # $1指匹配复制前面正则表达式()里内容

}

...

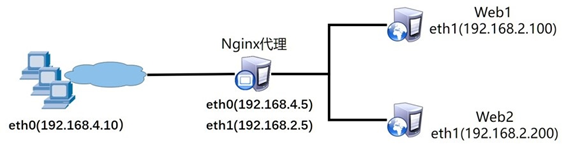
注释:~符号代表正则匹配，\*符号代表不区分大小写

# 2019.11.15 day3

## **nginx**代理(帮助访问网站)

## 负载均衡

## 健康检查



实验环境搭建:

修改web1和web2的yum仓库的地址,测试是否能使用仓库.

给web1和web2服务器上搭建apache服务,在nginx代理服务器上测试web1和web2网页能够正常访问.

## 步骤一:配置nginx服务器,添加服务器池,实现反向代理功能(轮巡算法)

修改nginx配置文件:

...

upstream webserver { #在http{}里面任何位置都可以,webserver集群名称(可以随意写)

server 192.168.2.100:80; 代理地址1

server 192.168.2.200:80; 代理地址2

}

...

server {

listen        80;

server\_name localhost;

location / {

...

proxy\_pass http://webserver; #通过proxy\_pass将用户访问web的请求转发给webserver集群

#调用代理地址,使用轮巡算法

#proxy(指令)优先级比root(web家目录)优先级高,添加在location里面任何位置都可以.

...

}

}

## 步骤二:配置upstream服务器集群池属性(修改网站的权重值)(加权轮巡)

## 添加权重案例:

修改nginx配置文件

...

upstream webserver {

server 192.168.2.100:80 weight=2; 修改这台服务器的权重(数值越大,权重越高)

server 192.168.2.200:80; 默认情况下权重为1

}

...

使用环境:当服务器的性能不一致的时候,可以使用加权轮训算法来实现负载均衡.

## 添加健康检查超时时间案例:

修改nginx配置文件:

...

upstream webserver {

server 192.168.2.100:80 max\_fails=1 fail\_timeout=30; #max\_fails设置最大失败次数，测试服务器几次才确认服务器失败; fail\_timeout设置失败超时时间，单位为秒

server 192.168.2.200:80;

}

...

使用环境:按照以上修改配置为例,当访问web1服务失败次数到达nignx服务器定义的最大次数的时候,nginx服务器会在30秒内不会对web服务器健康检查.

## 步骤三:配置upstream服务器集群的调度算法

设置相同客户端访问相同Web服务器案例1:

...

upstream webserver {

ip\_hash; #通过ip\_hash设置调度规则为：相同客户端访问相同服务器

server 192.168.2.100:80 max\_fails=1 fail\_timeout=30;

server 192.168.2.200:80;

}

...

## nginx 软件包扩展功能:

7层代理(http mysql ftp dhcp dns),默认安装

4层代理(tcp udp) 支持4层代理功能版本在nginx v1.9以上版本,默认不安装

## nginx的tcp/udp调度器

实现如下功能:

后端SSH服务器两台

Nginx编译安装时需要使用--with-stream，开启ngx\_stream\_core\_module模块

Nginx采用轮询的方式调用后端SSH服务器

环境准备:

步骤一：部署支持4层TCP/UDP代理的Nginx服务器

安装依赖包...

./configure \

> --with-http\_ssl\_module        \                 //开启SSL加密功能

> --with-stream                                 //开启4层反向代理功能

编译安装...

步骤二：配置Nginx服务器，添加服务器池，实现TCP/UDP反向代理功能

网络原则:ip地址唯一对应一台主机,端口唯一对应一个服务.

计算机最大端口号:65535

修改配置文件:添加4层代理的代码时,不要放在http{}的里面!

...

stream {

        upstream backend {

    server 192.168.2.100:22; //后端SSH服务器的IP和端口

    server 192.168.2.200:22;

}

         server {

         listen 12345; //Nginx监听的端口

         proxy\_pass backend; //调用上面定义的集群

             }

}

http {

.. ..

}

重启nginx服务

测试访问代理服务器轮训效果:

ssh 192.168.4.5 -p 12345            //使用该命令多次访问查看效果

## 步骤一：自定义报错页面

修改配置文件

.. ..

charset utf-8; //仅在需要中文时修改该选项

error\_page 404 /404.html;     //自定义错误页面

.. ..

创建一个在网站根目录下的404.html网页,然后重启配置服务后测试.

常见http状态码:



200类正常,300类重定向问题,400类用户输入错误,500类nginx服务错误.

## 步骤二：如何查看服务器状态信息（非常重要的功能）

1.编译安装时使用--with-http\_stub\_status\_module开启状态页面模块

...

./configure \

> --with-http\_ssl\_module                        //开启SSL加密功能

> --with-stream                                 //开启TCP/UDP代理模块

> --with-http\_stub\_status\_module          //开启status状态页面

...

修改配置文件,在server虚拟网站下面添加下面内容:

...

location /status {

stub\_status on;

#allow IP地址;

#deny IP地址;

}

...

重启配置文件后,查看状态页面信息:

curl http://192.168.4.5/status

网页输出内容:

Active connections: 1

server accepts handled requests

10 10 3

Reading: 0 Writing: 1 Waiting: 0

注释:

Active connections：当前活动的连接数量。

Accepts：已经接受客户端的连接总数量。

Handled：已经处理客户端的连接总数量。（一般与accepts一致，除非服务器限制了连接数量）。

Requests：客户端发送的请求数量。

Reading：当前服务器正在读取客户端请求头的数量。

Writing：当前服务器正在写响应信息的数量。

Waiting：当前多少客户端在等待服务器的响应。

## \*\*\*压力测试nginx服务器:\*\*\*

安装httpd-tools才有ab命令

ab -c 100 -n 100 http://192.168.4.5/ !#最后一个/号记得输入,不然会测试报错

注释:

-c 模拟人的数量

-n 模拟总的点击量(请求量) !#-n的数值一定要比-c大,小于会报错

# ---nginx服务器优化(3种优化)---

优化Nginx并发量

(apache实验数据并发量3万,nginx实验室并发量5万)

修改nginx配置文件

...

worker\_processes 2; #与CPU核心数量一致

events {

worker\_connections 65535; #每个worker最大并发连接数

}

...

重启配置文件

## 优化linux内核参数(最大文件数量)

方法一:

ulimit -a 查看控制shell进程的资源

ulimit -Hn 100000                //设置硬限制（临时规则）不能超过这个值,超过这个值就报错

ulimit -Sn 100000                //设置软限制（临时规则）警告值(可以超过这个值)

方法二:

vim /etc/security/limits.conf

...

\* soft nofile 100000

\* hard nofile 100000

#该配置文件分4列，分别如下：

#用户或组 硬限制或软限制 需要限制的项目 限制的值

## 优化Nginx数据包头缓存(缓存优化):

修改配置文件,增加数据包头部缓存大小

...

http {

...

client\_header\_buffer\_size 1k; //默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k; //大请求包头部信息的缓存个数与容量

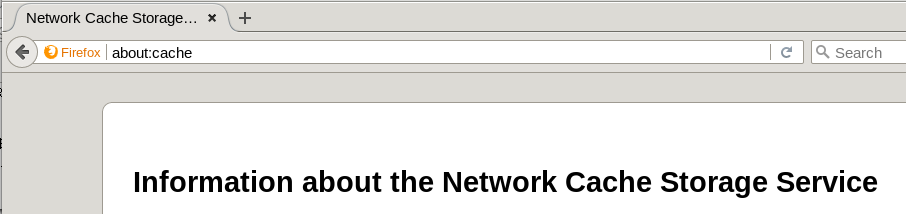
...

}

## 浏览器本地缓存静态数据

1）使用Firefox浏览器查看缓存

以Firefox浏览器为例，在Firefox地址栏内输入about:cache将显示Firefox浏览器的缓存信息，点击List Cache Entries可以查看详细信息。



浏览器缓存的东西:多媒体资料,图片,视频

修改nginx配置文件:

...

server {

...

location ~ \.(jpg|png)$ { #正则匹配.jpg .png后缀的图片匹配条件

expires 30d; #定义客户端缓存时间为30天

}

...

}

保存重启配置文档

测试...

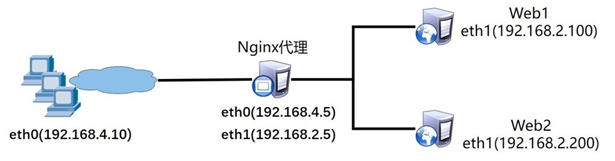
# 2019.11.16 day4

## session和cookies

Session：存储在服务器端(保存用户名、登陆状态、浏览历史记录等信息。)

Cookies：由服务器下发给客户端，保存在客户端的一个文件里。(保存的内容主要包括：SessionID)

准备实验拓扑环境:



......

cd /var/lib/php/session/            //查看服务器本地的Session信息

ls

sess\_qqek1tmel07br8f63d6v9ch401                        //注意这里的ID是随机的

安装memcached软件包

(数据只存在内存中,处理速度快,重启后所有数据会消失.)

yum -y install memcached telnet

systemctl start memcached

............................补笔记

# 2019.11.18 day5 IMG_256

部署tomcat服务器软件(在web1服务器上安装)

安装JDK软件包 java-1.8.0-openjdk

yum -y install java-1.8.0-openjdk #安装JDK软件包

java -version #查看java版本

\*安装JDK软件包,里面包含JVM(java虚拟管理工具)和java编辑器

java目录里的内容:

· bin/                                            //主程序目录

· lib/                                            //库文件目录

· logs/                                         //日志目录

· temp/                                         //临时目录

· work/                                        //自动编译目录jsp代码转换servlet

· conf/                                        //配置文件目录

· webapps/                                        //页面目录

启动服务:

/usr/local/tomcat/bin/startup.sh

使用netstat -nutlp 查看java启动的端口

tcp6 0 0 127.0.0.1:8005 :::\* LISTEN 4113/java

tcp6 0 0 :::8009 :::\* LISTEN 4113/java

tcp6 0 0 :::8080 :::\* LISTEN 4113/java

firefox http://192.168.2.100:8080

strings /dev/random #等系统自动生成乱码文件,速度取决于内存

strings /dev/urandom #注释:strings类似于cat命令;人为生成随机乱码文件

提示：如果检查端口时，8005端口启动非常慢，默认tomcat启动需要从/dev/random读取大量的随机数据，默认该设备生成随机数据的速度很慢，可用使用下面的命令用urandom替换random（非必须操作）。

[root@web1 ~]# mv /dev/random /dev/random.bak

[root@web1 ~]# ln -s /dev/urandom /dev/random

在/usr/local/tomcat/webapps/ROOT目录下新建一个test.jsp的页面,内容如下:

· <html>

· <body>

· <center>

· Now time is: <%=new java.util.Date()%>            //显示服务器当前时间,调用类库函数变量

· </center>

· </body>

· </html>

java命令格式: <%执行的命令%>

测试页面显示内容: firefox http://192.168.2.100:8080/test.jsp

tomcat的web目录在: /usr/local/tomcat/webapps

java的压缩包格式:jar war

注释: cat /usr/local/tomcat/conf/server.xml

· <Server>

· <Service>

· <Connector port=8080 />

· <Connector port=8009 />

· <Engine name="Catalina" defaultHost="localhost"> 当输入ip的时候自动找到localhost

· <Host name(域名)="www.a.com" appBase(网站根目录)="a" unpackWARS(自动解压)="true" autoDeploy(自动部署,开发用)="true">

· </Host>

· <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

· </Host>

修改配置文件:

/usr/local/tomcat/conf/server.xml

· <Server> 虚拟web主机

· <Service> 服务

· <Connector port=8080 />

· <Connector port=8009 />

· <Engine name="Catalina" defaultHost="localhost">

· <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

· </Host>

· <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

· </Host>

创建网页家目录

mkdir -p /usr/local/tomcat/a/ROOT

mkdir -p /usr/local/tomcat/b/ROOT

创建网站主页面

echo aaa > /usr/local/tomcat/a/ROOT/index.html

echo bbb > /usr/local/tomcat/a/ROOT/index.html

重启tomcat服务器

/usr/local/tomcat/bin/shutdown.sh

/usr/local/tomcat/bin/startup.sh

修改测试机的hosts文件,添加192.168.2.100 www.a.com www.b.com

firefox http://www.a.com:8080

firefox http://www.b.com:8080

步骤二:

修改配置文件,使www.b.com网站的首页目录为base

<Host name="www.b.com" appBase="b"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="base" /> #使用docBase参数可以修改默认网站的首页路径

</Host>

创建文件夹

mkdir /usr/local/tomcat/b/base

echo “BASE” > /usr/local/tomcat/b/base/index.html

重启服务

/usr/local/tomcat/bin/shutdown.sh

/usr/local/tomcat/bin/startup.sh

测试网站首页显示

firefox http://www.a.com:8080

步骤三:

跳转,类似于nignx的地址重写:

修改配置文件

<Host name="www.a.com" appBase="a"

unpackWARs="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html"/>

</Host>

注释:

#Contest path 地址跳转 docBase 跳转到那个目录下

步骤四:配置tomcat支持ssl加密网站

创建加密用的私钥和证书文件

keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore

注释:

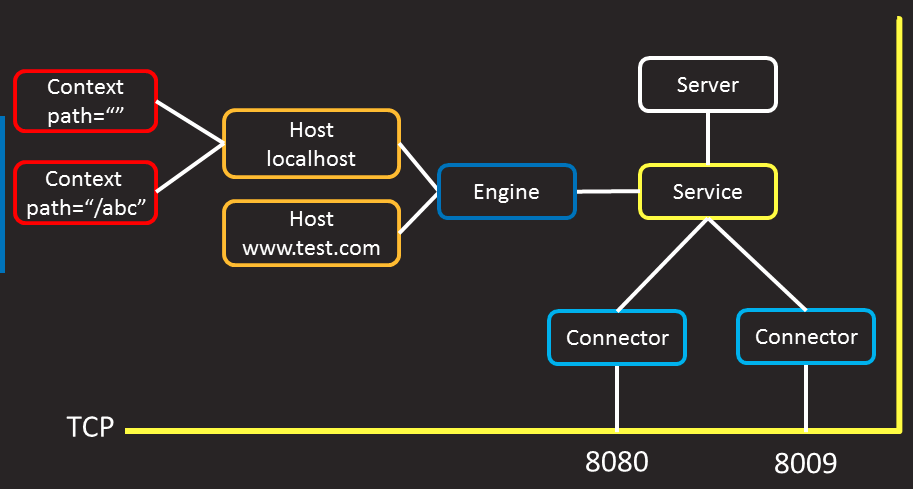
· //-genkeypair     生成密钥对

· //-alias tomcat     密钥别名

· //-keyalg RSA     定义密钥算法为RSA算法

· //-keystore         定义密钥文件存储在:/usr/local/tomcat/keystore

## tomcat工作原理:



配置tomcat支持ssl网站加密:

修改配置文件:

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS"

keystoreFile="/usr/local/tomcat/keystore" keystorePass="123456"/>

重启tomcat服务

tomcat日志目录: /usr/local/tomcat/logs/

CDN全称是Content Delivery Network,即内容分发网络.(优点,打开网站加速,更加的稳定)

## 部署varnish缓存服务器(代理服务器)

安装软件包:

readline-devel ncurses-devel python-docutils pcre-devel

创建用户:(用普通用户来调用软件较安全)

usradd -s /sbin/nologin varnish

varnish是源码包,需要解压源码包安装

复制配置文件

[root@proxy varnish-5.2.1]# cp etc/example.vcl /usr/local/etc/default.vcl

修改配置文件:

[root@proxy ~]# vim /usr/local/etc/default.vcl

backend default {

.host = "192.168.2.100"; 指定源网站地址

.port = "80"; 和端口

}

启动服务

varnishd -f /usr/local/etc/default.vcl #启动varnish的时候,会占用80端口,如有占用的话请关闭占用端口服务

注释:

//varnishd命令的其他选项说明如下：

//varnishd -s malloc,128M        定义varnish使用内存作为缓存，空间为128M

//varnishd -s file,/var/lib/varnish\_storage.bin,1G 定义varnish使用文件作为缓存

客户端测试

curl <http://192.168.4.5>

查看varnish日志,是一条命令,有人访问的时候会出现日志信息

varnishlog #详细版

varnishncsa #精简版

缓存服务器更新时间大概2分钟左右,到更新时间会自动更新.

手动更新缓存服务器:

[root@proxy ~]# varnishadm

varnish> ban req.url ~ .\* #在生产环境中,不要使用.\* ,这样会导致雪崩效应.(生产环境中,一个个的清除)

//清空缓存数据，支持正则表达式

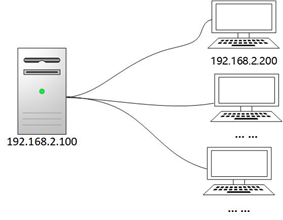
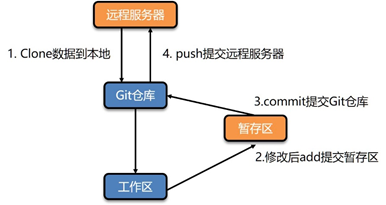
# 2019.11.19 day6

## git基本操作

集中式版本控制svn(subversion) (客户端下载最新版, 缺点:断网的话无法正常备份)

分布式版本控制git (客户端克隆下载,修改一次,在断网的情况下,先保存到本地仓库,然后有网络的情下在上传到服务器上,相对集中式比较安全.)(作者是linux) github

实验拓扑图:

搭建git服务器的操作流程:

安装Git软件

创建版本库

客户端克隆版本仓库到本地

本地工作目录修改数据

提交本地修改到服务器

## 步骤一:

搭建部署git服务器(在web1上做)

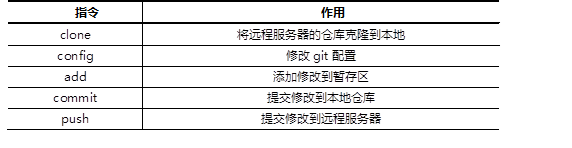
yum -y install git

mkdir /var/git 创建一个仓库目录

git init /var/git/project --bare 创建一个空的目录,不能手动创建,里面会生成文件

步骤二:

使用git常用指令列表



1.在客户端上操作(web2),clone克隆服务器仓库到本地

yum -y install git

git clone root@192.168.2.100:/var/git/project/ 默认下载到当前目录

ls /root/project #仓库工作区

ls /root/project/.git #历史仓库

2.修改git配置

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

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

3.在本地仓库对数据进行增删改查

echo "init date" > init.txt

mkdir demo

cp /etc/hosts demo

4.查看仓库中数据状态:

git status #会显示有什么文件没有上传备份

5.将工作区修改添加到暂存区

git add . 添加当前目录所有文件到暂存区

6.将暂存区修改提交到本地仓库

git commit -m “注释” 注释可以为任意字符

\*如果没有配置git配置,输入上面命令会报错提示输入.

7.将本地仓库(wbe2)中的数据推送到远程服务器(web1)上

git config --global push.default simple

git push #如果没有输入上面命令,会提示报错上面命令信息.

1. 将服务器上的数据更新到本地(web1的数据更新到wbe2)

git pull #只更新修改的数据, clone是把库里面的所有数据都更新下来

root@192.168.2.100's password:

Already up-to-date.

1. 查看版本日志

git log #查看详细的版本信息

git log --pretty=oneline #查看一行的详细信息

git log --oneline #查看一行的简要信息

git reflog #显示..指针

\*在windows客户端上安装git和tortoiseGit就可以和git服务器通信.

## HEAD指针基本操作

1. 准备工作（多对数据仓库进行修改、提交操作，以产生多个版本）。

2) 查看Git版本信息。

3）移动HEAD指针，将数据还原到任意版本。

提示：当前HEAD指针为HEAD@{0}。

[root@web2 project]# git reset --hard 301c0 #移动指针回到以前的版本

[root@web2 project]# git reflog #查看日志指针

301c090 HEAD@{0}: reset: moving to 301c0

04ddc0f HEAD@{1}: commit: num.txt:789

7bba57b HEAD@{2}: commit: num.txt:456

301c090 HEAD@{3}: commit: num.txt:123

b427164 HEAD@{5}: commit: new.txt:third

0584949 HEAD@{6}: commit: new.txt:second

ece2dfd HEAD@{7}: commit: new.txt:first line

e1112ac HEAD@{8}: commit: add new.txt

1a0d908 HEAD@{9}: commit (initial): 初始化

[root@web2 project]# cat num.txt                 #查看文件是否为123

123

[root@web2 project]# git reset --hard 7bba57b

[root@web2 project]# cat num.txt                 #查看文件是否为123，456

123

456

[root@web2 project]# git reflog                 #查看指针移动历史

7bba57b HEAD@{0}: reset: moving to 7bba57b

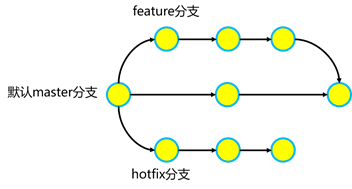
301c090 HEAD@{1}: reset: moving to 301c0

… …

[root@web2 project]# git reset --hard 04ddc0f     #如果找回以前修改版本的历史内容,找回后复制到别的目录下,最后一次要回到最新的版本后才能上传服务器,不然会报错.(最后一行reset的下面就是最新的commit)

## git分支操作

Git支持按功能模块、时间、版本等标准创建分支，分支可以让开发分多条主线同时进行，每条主线互不影响



常见的分支规范如下：

MASTER分支（MASTER是主分支，是代码的核心）。

DEVELOP分支（DEVELOP最新开发成果的分支）。

RELEASE分支（为发布新产品设置的分支）。

HOTFIX分支（为了修复软件BUG缺陷的分支）。

FEATURE分支（为开发新功能设置的分支）。

案例:

## 查看当前分支:

[root@web2 project]# git status #查看当前状态

# 位于分支 master

无文件要提交，干净的工作区

[root@web2 project]# git branch -v #查看当前分支

\* master 5b79ef7 pig zhu

## 创建分支和切换分支:

[root@web2 project]# git branch hotfix #创建分支

[root@web2 project]# git branch feature #创建分支

[root@web2 project]# git branch -v #查看分支

feature 5b79ef7 pig zhu

hotfix 5b79ef7 pig zhu

\* master 5b79ef7 pig zhu

[root@web2 project]# git checkout hotfix #切换分支

切换到分支 'hotfix'

## 合并分支:

将hotfix修改的数据合并到master分支。

注意，合并前必须要先切换到master分支，然后再执行merge命令。

git merge hotfix #在master分支里面,把hotfix分支合并进来

## 案例环境搭建:(创建两个相同文件)

[root@web2 project]# git checkout hotfix

切换到分支 'hotfix'

[root@web2 project]# echo AAA > a.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "hotfix a.txt"

[hotfix 1fc98e2] hotfix a.txt

1 file changed, 1 insertion(+)

create mode 100644 a.txt

[root@web2 project]# git checkout master

切换到分支 'master'

您的分支领先 'origin/master' 共 1 个提交。

（使用 "git push" 来发布您的本地提交）

[root@web2 project]# echo BBB > a.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "master a.txt"

[master 3904fdd] master a.txt

1 file changed, 1 insertion(+)

create mode 100644 a.txt

[root@web2 project]# git merge hotfix

自动合并 a.txt

冲突（添加/添加）：合并冲突于 a.txt

自动合并失败，修正冲突然后提交修正的结果。

当显示冲突的时候, 查看a.txt文件,然后把里面多余的东西删掉后就可以上交服务器作为一个新的版本.

## \*\*非交互式的生成ssh密钥文件\*\*

ssh-keygen -f /root/.ssh/id\_rsa -N ‘’ #生成秘钥,设置秘钥密码为空

ssh-copy-id 192.168.2.100 #把公共秘钥发送给192.168.2.100

## git协议服务器

类型:

创建SSH协议服务器 (安全,加密传输)

创建Git协议服务器

创建HTTP协议服务器

在web1虚拟机上做操作:

部署git协议服务器

安装软件包

yum -y install git-daemon

创建一个仓库

git init --bare /var/git/lanceyan

修改git仓库配置文件

vim /usr/lib/systemd/system/git@.service

[Unit]

Description=Git Repositories Server Daemon

Documentation=man:git-daemon(1)

[Service]

User=nobody

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/git --export-all --user-path=public\_git --syslog --inetd --verbose 注释:base-path 指定哪个目录做共享目录

StandardInput=socket

重启服务

systemctl restart git.socket

在web2上做操作

git clone git://192.168.2.100/lanceyan

## 创建http协议 git服务器

在web1上做操作

安装软件包 httpd gitweb

yum -y install httpd gitweb

修改配置文件:

vim +11 /etc/gitweb.conf

...

$projectroot = "/var/git";

...

重启httpd服务

systemctl restart httpd

在web2上做测试:

firefox http://192.168.2.100/git

sshd--->git clone root@ip:/目录

git-daemon--->git clone git://ip/mulu

httpd --->http://ip/目录

## 制作nginx的RPM包

RPM(自定义,旧版)

在web1上安装软件包

yum -y install rpm-build

[root@web1 ~]# rpmbuild -ba xxx.rpm #-ba后面可以乱输入,出现错误后,会在/root/rpmbuild/下自动生成一下目录

错误：stat /root/xxx.azsdfa 失败：没有那个文件或目录

[root@web1 ~]# ls rpmbuild/

BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

复制nginx源码包到rpmbuild/SOURCES目录下

进入rpmbuild/SPECS目录

vim nginx.spec

Name:nginx #源码包软件名字

Version:1.12.2 #源码包软件的版本号

Release:6 #制作rpm包版本号

Summary:This is a web server. #rpm软件描述

#Group: #定义软件组

License:GPL #软件的协议

URL:www.lanceyan.com #网址

Source0:nginx-1.12.2.tar.gz 源码包文件的全称

#BuildRequires:

#Requires:

%description #软件的详细描述

This is a web server toooooooooooooo.

%prep

%setup -q #自动解压源码包(nginx-1.12.2.tar.gz)并cd进入目录

%post #添加命令或脚本(非必须项)

useradd lanceyan

touch /tmp/lanceyan.txt

%build

./configure

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc

/usr/local/nginx/\* #对哪些文件与目录打包

%changelog

使用配置文件创建rpm包

安装依赖软件包

yum -y install gcc pcre-devel openssl-devel

rpmbuild创建rpm软件包

rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

ls /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

安装软件

yum install /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

# 2019.11.20 day7

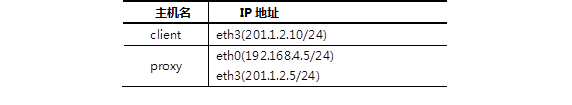
配置GRE VPN服务器

在公网网络上搭建专用私有的网络,进行加密通信

连接完成后,各个地区的子公司可以想局域网一样通信

在搭建vpn的前提下,首先公共的光缆必须得互相通信

准备实验环境:



GRE > PPTP > XL2TP+IPsec

GRE 搭建简单,装包就可以用,隧道不加密

PPTP 搭建一般,部分加密,修改一个配置文件

XL2TP+IPsec 搭建难度大,隧道全部加密,修改4个配置文件

linux默认支持GRE模块,默认是不启动的

windows不支持GRE

lsmod #显示linux里面激活的模块

modprobe 模块名 #激活模块

modinfo 模块名 #查看内核模块的信息

rmmod 模块名 #关掉激活的模块