

记得用源代码模式看某些地方!!!

Linux目录

没有盘符，根目录是/

Windows: D:\data\work\hello.txt(右斜杠)

Linux:/usr/local/hello.txt(左斜杠)

Linux命令

command [-options] [parameter]

- command:命令本身
- -options:[可选，命令的一些选项]
- parameter: [可选，命令的参数]

eg. ls -l /home

ls

ls [-a -l -h] [Linux路径]

不使用参数：平铺显示当前目录的所有内容

默认当前工作目录：/home/用户名（home目录）

参数

-a:all,列出所有文件，包括隐藏文件和文件夹（隐藏文件最前面有个小点 eg. .config）

-l:以列表形式展示内容，并展示更多内容

drwxr-xr-x (权限) 5 shed shed（用户和用户组） 4096（大小,单位为

b) 3月 12 20:40（创建时间） Documents

-h:默认排列方式

组合使用

ls -l -a; ls -la; ls -al

ls -lh;显示大小单位

cd

cd(change directory)

cd [linux目录]

不写参数：回到home目录

pwd

pwd(print work directory)

打印当前工作目录

语法：pwd

没有选项，没有参数

绝对路径和相对目录

cd /home/homo/Desktop(绝对路径)

cd Desktop (相对路径，无需/)

特殊路径符

.当前目录 cd .

..上一级目录 cd ..

~Home目录 cd ~

mkdir

mkdir (make directory)

mkdir [-p] Linux路径

参数必填，表示要创建的文件夹目录，相对路径和绝对路径都可以

-p:自动创建不存在的父目录，可以连续创建多级目录，相当于java的mkdirs

如果创建多级目录不用-p,报错:eg.

mkdir: 无法创建目录 “./mkdirTest01/test/homo” : 没有那个文件或目录

创建文件夹需要权限很大，出了home目录就会涉及权限问题

eg.mkdir: 无法创建目录 “test” : 权限不够

touch

touch Linux路径

创建文件

touch test.txt

文件夹与文件区分

ls下颜色不一样

ls -l下表示不一样，目录以d开头，如：drwxr-xr-x

文件以-开头，如-re-rw-r--

cat

cat Linux路径

如果文件里面没有换行符是不会自动换行的

tips: ctrl+l 清空控制台屏幕

more

文本较多的时候,more支持翻页查看, cat全部显示出来

more Linux路径

eg.more /etc/services

空格, 下一页, 回车, 一排一排地看, q,退出, 想回看往上翻就是, 超过100%会自己退出

cp

cp(copy)

cp [-r] 参数1 参数2

-r:可选, 复制文件夹时用, 表示递归

(不指定的话会显示略过目录, 不复制)

参数1: linux路径, 表示被复制的文件或文件夹

参数2: linux路径, 表示要复制去的地方

mv

mv 参数1 参数2

参数1: linux路径, 表示被移动的文件或文件夹

参数2: linux路径, 表示要移动到的地方

如果目标不存在, 则有改名的效果

mv test.txt Desktop/

mv test.txt test2.txt test.txt改名为test2.txt

rm

rm(remove)

rm [-r -f] 参数1 参数2.....参数N(空格隔开参数)

-r:递归删除文件夹

-f:force,强力删除 (不会弹出提示信息)

普通用户没提示, root管理员有提示

一个文件不存在, 其他文件如果存在也会删除

通配符

- 任意内容

test* 任何以test开头的内容

*test 任何以test结尾的内容

test 任何含有test的内容

root

su -root: 输入密码, root

exit:返回普通用户

危险命令

rm -rf /

rm -rf /*

which

命令本质上就是程序, 用which可以查找程序文件在哪里

which 要查找的命令

find

find 查找命令

按照文件名查找

find 起始路径 -name "被查找文件名"

进不了root可以

sudo su root

find / -name "test"

可以使用通配符

按照文件大小搜索

find 起始路径 -size +|- n[kMG]

find / size +100M

grep

关键字过滤

grep [-n] 关键字 文件路径

-n:显示匹配的行号

查找文件内文本

wc

wc = word count

统计文件内容

wc [-c -m -l -w] 文件路径

-c:统计bytes数量

-m:统计字符数量

-l:统计行数

-w:统计单词数量

文件内容: 可以作为内容输入端口

一个换行符对应一行

wc --help:获取帮助

管道符

| 将管道左边的输出结果作为右边的输入

eg.cat test.txt | grep test

echo

打印内容，和编程的print语句差不多

echo 指定输出的内容

eg.echo Hello Linux

echo "Hello Linux"

这样误解会少一点

反引号`

echo pwd

echo `pwd`

pwd | echo 输出一行空行

重定向符

将左边命令的结果输出到右边的文件里面

将左边的结果追加到右边的文件

echo "hello linux" > test.txt

echo "I'm linux" >> test.txt

右边的文件不存在会自动创建一个文件

tail

用tail命令，可以查看文件尾部信息，跟踪文件的最新更改

tail [-f -num] Linux路径

-f,持续跟踪

-num,表示查看尾部多少行，默认为10行

eg.tail -f text.txt

CTRL+C 终止文件执行

tail -15 text.txt

tail -f -n 15 text.txt

查看后15行效果并且追踪

vi/vim编辑器

vi/vim是visual interface的简称，是Linux中最经典的文本编辑器

vim是vi的加强版本，兼容vi的所有指令，还具有shell程序编辑功能，可以用不同颜色的字体辨别语法正确性，极大地方便了程序设计和编辑性

命令模式

按键都被理解为命令，完成不同功能

输入模式

按什么输入什么

底线模式

以：开始，常用于文件的保存，退出

vi filename

默认进入命令模式

输入i或a或o进入输入模式

ESC键退出输入模式

： 进入底线模式，回车结束运行

wq退出vi编辑器

vi 文件路径

vim 文件路径

文件不存在，编辑新文件

文件存在，编辑已存在的文件

:wq保存文件

命令

i 在当前光标位置进入输入模式
a 在当前光标位置之后进入输入模式
o 在当前光标的下一行进入输入模式
I 在当前行的开头进入输入模式
A 在当前行的结尾进入输入模式
O 在当前光标的下一行进入输入模式
↑ k 光标向上移动
↓ j 光标下移
← h
键盘右键 l
0 当前行的开头
\$ 当前行的结尾
pageUP 向上翻页
pageDOWN 向下翻页
/进入搜索模式
n 向下搜索
N 向上搜索
dd 删除光标所在行
n dd n是数字，删除从当前光标向下n行

yy 复制当前行
nny 是数字，复制当前行和下面n行
p 粘贴复制的内容
u 撤销更改
ctrl+r 反向撤销
gg 跳转到首行
G 跳到行尾
dG 从当前行开始向下全部删除
dgg 从当前行开始，向上全部删除
d\$ 从当前光标开始，删除到本行结尾
d0 从当前光标开始，删除到本行开头

底线命令模式

:wq 保存并退出

:q 仅退出

: q! 强制退出

:w 仅保存

: set nu 显示行号

: set paste:设置粘贴模式

外部复制是什么样，粘贴模式内部就是什么样，不会产生格式的混乱

Chapter 3

用户

root用户拥有最大的操作权限，但是普通用户很多操作都受限

su/sudo su:登录root

普通用户在home目录不受限制，

出了home目录，大多只有只读和执行权限

su

su [-] [用户名]

-是可选的，表示切换用户之后是否带上环境变量，建议带上

用户名表示切换的用户，不输入就是root

exit

exit/ctrl+d:回退到上一个用户

普通用户切换到root需要密码，root切换到普通用户不需要密码

sudo

不建议长期使用root用户，避免带来系统损坏

sudo命令可以为普通命令授权，临时以root的身份执行

sudo 其他命令

需要为普通用户配置sudo配置

sudo配置

root:

visudo

末尾添加:

shed ALL=(ALL) NOPASSWD:ALL

: wq

exit

mkdir test:无权限

sudo mkdir test:有权限

删除最后一行

没有配置: shed不在sudoers文件中, 此事将被报道

用户与用户组

权限管控: 用户/用户组

用户组

root条件下执行

groupadd 用户组名

groupdel 用户组删除

groupadd userGrp

用户

useradd

useradd [-g -d] 用户名

-g: 指定用户组, 不指定会创建同名的组并且加入, -g指定需要组已经存在, 同名的组存在,必须用-g

-d:指定用户的HOME目录, 不指定的话, HOME目录默认在/home/用户名

ubuntu好像是useradd -m 用户名,deepin应该-g -d都可以

userdel

userdel [-r] 用户名

-r:删除用户的Home目录, 不使用则保留Home目录

id

id [用户名]

查看用户信息, 不加参数查看自己的用户信息, 其他用户在root下查看

usermod

usermod -aG 用户组 用户名

将用户添加到组里面

getent

getent passwd

列出当前操作系统内的用户

信息：用户名:密码(X):用户ID： 组ID： 描述信息（无用）： HOME目录:执行终端（默认bash）

getent group

列出系统所有组

组名称：组认证（默认为x）： 组ID

权限

ls -l

权限信息，文件，文件夹所属用户，文件，文件夹所属用户组

权限信息

第一位：-或d或l(-：文件，d:文件夹，l:软链接)

2-4位:所属用户权限

第二位:r或-（-表示无此权限）

第三位:w或-

第四位:x或-

5-7位：所属用户组权限

第五位:r或-

第六位:w或-

第七位:x或-

8-10位：所属用户组权限

第八位:r或-

第九位:w或-

第十位:x或-

r:Read,读取权限,查看文件夹内容

w:Write,写入权限,针对文件夹进行修改

x:eXecute,执行权限，将工作目录切换到这个文件夹,即cd进入这个目录

eg. /mmkv.default 属于root，其他用户没有root的权限，只能由8-10位来确定

chmod

文件属于用户或root才能修改

chmod修改文件，文件夹的权限信息

chmod [-R] 权限 文件夹或文件

-R,对文件夹内的全部内容进行同样的操作

eg. `chmod u=rwx,g=rx,o=x hello.txt`

user用户：rwx； group组：rx； others x

chmod -R u=rwx,g=rx,o=x test

对test文件夹以及文件夹里面的东西生效

ugo写起来太麻烦了，可以数字简写，如751

r记为4，w记为2，x记为1

0 ---

1 --x

2 -w-
3 -wx
4 r--
5 r-x
6 rw-
7 rwx

chown

chown可以修改文件所属的用户和用户组

chown [-R] [用户][:][用户组] 文件或者文件夹

命令需要在root账户下操作

-R:同chmod

用户:修改所属的用户

用户组: 修改所属的用户组

:用于分隔用户和用户组

chown root hello.txt hello.txt修改用户为root

chown :root hello.txt hello.txt修改用户组为root

chown root:root hello.txt

chown -R root test

Linux实用操作

快捷键

CTRL+C 强中断命令进行,输入新命令

CTRL+D 退出当前账户的登录（可能关闭terminal），会退出某些程序的专属页面（如python）
（Linux自带python）（CTRL+C 会显示KeyboardInterrupt）

history 查看历史命令

! p,从下到上找到第一个以p开始的命令(不太适合太久的命令)

CTRL+R 输入关键字，命中从下向上的第一个匹配命令（按键盘右键get到）

CTRL+a:跳到命令开头

CTRL+e:跳到命令结尾

CTRL+键盘左键，向左跳一个单词

CTRL+键盘右键，向右跳一个单词

CTRL+L 清屏 clear也可以达到相同效果

软件安装

yum(Centos)

yum: RPM的包管理器（RPM: 安装包格式）

yum [-y] [install|remove|search] 软件名称

-y:自动确认

yum需要root权限，yum需要联网

apt (Ubuntu)

deb安装包格式

`apt [-y] [install|remove|search] 软件名称`

ubuntu自带wget哟

systemctl

Linux很多软件都支持使用systemctl控制：启动，停止，开机自启

能被systemctl管理的软件一般称为服务

`systemctl start|stop|status|enable|disable 服务名`

start：启动 stop：关闭 status：查看状态 enable：开启开机自启 disable：关闭开机自启

系统内置的服务比较多，如：

NetworkManager：主网络服务

network：副网络服务

firewalld：防火墙服务

sshd，ssh服务

第三方软件(自己安装)

ntp(时间同步软件)

`systemctl status ntpd`

(注册的服务叫ntpd，在ubuntu下注册的服务好像叫ntp)

要加sudo权限，不然会提醒

Unit ntp.service could not be found.

之所以可以被控制，是因为安装过程中自动集成到systemctl中

ln

创建软链接将文件，文件夹连接到其他位置

`ln -s 参数1 参数2`

-s:创建软链接

参数1：被连接的文件或文件夹

参数2：要链接去的目的地

eg. `ln -s /etc/yum ~/yum`

时间与时区

date

`date [-d] [+格式化字符串]`

-d:按照指定的字符显示日期，一般用于日期的计算

格式化字符串

%Y 年

%y 年份后两位数字 (00-99)

%m 月份 (01-12)

%d 日 (01-31)

%H 小时(00-23)

%M 分钟(00-59)

%S 秒 (00-60)

%s 自1970-01-01 00:00:00 UTC 到现在的秒数

date:2023年 05月 01日 星期一 21:36:11 CST

date +%Y-%m-%d:2023-05-01

date "+%Y-%m-%d %H:%M:%S": 2023-05-01 21:38:49

date -d "+1 month" +%Y-%m-%d:2023-06-01

-d支持的时间表及

year month day minute second

修改时区

su -

rm -f /etc/localtime

ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

UTC成功转为CST时区

ntp

请安装ntp

然后使用systemctl设置服务

记得enable

手动校准(要root权限)

ntpdate -u ntp.aliyun.com

ip地址和主机名

ip地址

IPv4:a.b.c.d(a在0-255间)

ifconfig:查看网络连接信息

127.0.0.1:本机ip

0.0.0.0: 指代本机/端口绑定中确定绑定关系/0.0.0.0在IP地址限制中表示全部ip放行

主机名

eg.Desktop-homo

hostname: 查看主机名

hostnamectl set-hostname deepin:修改主机名为deepin

域名

以字符代替IP地址

eg.baidu.com=110.242.68.66

访问baidu.com:

检查: Windows:C:\Windows\System32\drivers\etc\hosts

Linux:/etc/hosts 中是否有ip记录, 如果没有, 访问DNS查询

网络请求与下载

ping

检验服务器是否连通

`ping [-c num] ip或主机名`

-c:被检查的次数,不使用-c 会无数次检查

eg. ping -c 3 www.baidu.com

wget

非交互式的文件下载器

`wget [-b] url`

-b:后台下载, 不会显示下载进度, 会把日志写到wget-log文件里面

eg. wget <https://archive.apache.org/dist/hadoop/common/hadoop-3.1.3/hadoop-3.1.3.tar.gz>

curl

发起网络请求

`curl [-O] url`

-O:用于下载文件, 当url为下载链接时, 可用此选项保存文件

eg. curl cip.cc(查看主机ip地址的网站)

curl -O <https://archive.apache.org/dist/hadoop/common/hadoop-3.1.3/hadoop-3.1.3.tar.gz>

端口

物理端口--接口; 虚拟端口--计算机内部的端口

ip地址只能锁定计算机, 不能锁定到某一个程序, 可以使用端口访问

Linux支持65535个端口

1-1023: 公认端口,通常用语系统内置或知名程序的预留使用, 如SSH的22端口, HTTPS的443端口, 非特殊需要, 不要占用这个范围的端口

1024-49151: 注册端口, 通常可以随意使用, 用于松散地绑定一些程序和服务

49152-65535:动态端口, 通常不会固定绑定程序, 而是程序进行网络连接时, 用于临时使用

nmap

嗅探端口的占用情况

`nmap 被查看的IP地址`

非系统自带, 记得安装

netstat

查看具体端口的占用情况

`netstat -anp|grep 端口号`

记得安装, ubuntu系好像是net-tools

此处的0.0.0.0是指本机ip

进程管理

任务管理器

为管理程序，每一个运行中的程序，都被系统注册为系统中的一个：进程，并分配进程号（PID）

`ps [-e -f]`

-e:显示出全部的进程

-f:已完全格式化的形式显示进程（展示全部信息）

UID:进程所属的用户ID

PID:进程所属的进程ID

PPID: 进程所属的父ID

C:进程的CPU占用比（百分比）

STIME: 进程启动的时间

TTY: 启动进程的终端序号，若为“？”，则为非终端启动

TIME: 占用CPU的时间

CMD:启动路径或启动命令

进程关闭

`kill [-9] 进程ID`

-9:强制关掉

已终止（不加-9）/已杀死（加了-9）

主机状态管理

`top`

变成任务管理器，5秒钟刷新一次

CTRL+C或键盘Q退出

-top - 21:56:12 up 27 min, 1 user, load average: 3.38, 3.18, 2.45

21:56:12生成的报告，启动了27分钟，1个用户登录了系统，平均负载：1分钟：3.38 5分钟：3.18
15分钟：2.45（1=有一颗cpu 100%繁忙）

Tasks: 299 total, 1 running, 297 sleeping, 0 stopped, 1 zombie

共有299进程，1个正在运行，297进程睡眠，0个停止进程，1个僵尸进程

%Cpu(s): 25.3 us, 5.4 sy, 0.0 ni, 69.3 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st

用户CPU使用率25.3%，系统CPU使用率5.4%，高优先级进程CPU使用时间百分比：0% 空闲CPU
率：69.3% IO等待CPU占用率0.1% cpu硬件中断率：0 cpu软件中断率：0 强制等待占用cpu率：0

MiB Mem: 15911.4 total, 8042.6 free, 2176.0 used, 5692.8 buff/cache

物理内存：total: 总共 free: 空闲 used: 使用 buff/cache:buff/cache占用

MiB Swap: 16384.0 total, 16384.0 free, 0.0 used. 12752.8 avail Mem

虚拟内存（交换空间）

PID 进程id

USER 所属用户

PR 进程优先级，越小越高

NI 负值：高优先级 正值：低优先级

VIRT 使用的虚拟内存，单位KB
RES 使用的实际内存，单位KB
SHR 使用的共享内存，单位KB
S 进程状态（S:休眠 R：运行 Z:僵死状态 N:负优先级 I:空闲状态）
%CPU
%MEM
TIME+
COMMAND 启动命令或文件路径

同时top命令也支持选项

-p:只显示某一个进程的信息
-d:设置刷新时间，默认为5s
-c: 显示产生进程的完整命令
-n:指定刷新次数，如top -n 3 刷新输出3次之后退出
-b:以非交互式的模式运行(一般和重定向符连用)
-i: 不显示任何闲置（idle）或无用（zombie）的进程
-u:找到特定用户启动的进程

交互式快捷键

h:显示帮助画面
c:显示完整命令
f:可以选者需要展示的项目
M:按主流内存（RES）排序
P:按CPU使用百分比大小排序
T:按照时间/累积的时间排序
E:切换顶部内存显示单位
e:切换进程内存显示单位
l:切换显示平均负载和启动时间信息
i:不显示限制或无用进程
t:显示CPU状态信息
m:切换显示内存信息

磁盘信息监控

磁盘剩余空间

df [-h]

-h:用更人性化的命令显示内存

io检测

iostat [-x][num1][num2]

-x:显示更多信息

num1:数字，刷新间隔

num2:刷新几次

ubuntu安装sysstat(内置iostat)

-x:%util:磁盘利用率

rKB/s:每秒发送到设备的请求读取量

wKB/s:每秒发送到设备的请求写入量

网络信息监控

sar命令非常的复杂，只用掌握简单统计网络的固定写法就好

```
sar -n DEV num1 num2
```

-n:查看网络，DEV：网络接口

num1:属性间隔 num2：查看次数（不填无限次数）

IFACE:本地网卡的名称

rxpck/s:每秒钟接受到的数据包

txpck/s:每秒钟发送的数据包

rxKB/s:每秒钟接受的数据包

txKB/s：每秒钟接受的数据包

rxcmp/s:每秒钟接受的压缩数据包

txcmp/s:每秒钟发送的压缩包

rxmcst/s:每秒钟接受的多播数据包

环境变量

操作系统（Windows，MACOS，Linux）在运行时记录的一些关键信息，用于辅助系统运行

env 可以查看

环境变量是KeyValue型结构

```
env | grep PATH
```

会到path下搜索应用程序

\$符

知道键名，可用\$输出值

```
echo $PATH
```

echo \${PATH}ABC 将path的值和ABC拼接在一起

设置环境变量

临时设置

```
export 变量名=变量值
```

临时设置环境变量

永久生效

~/bashrc对当前用户生效

加入export 变量名=变量值

source .bashrc 使之生效

/etc/profile对所有用户生效

其他操作和上面一样

export PATH=\$PATH:自定义路径

不加\$PATH（表示原有环境变量）就只有自定义路径，path的路径之间用冒号隔开

上传和下载

安装lrzsz来安装rz和sz

sz 需要下载的文件

将文件从虚拟机下载下来

rz

弹出文件系统框，自动选择文件（速度比较慢，用FinalShell直接拖拽上传到虚拟机快一点,如果服务器在云端的话速度差不多）

压缩和解压

tar和gz

.tar,tarball归档文件，简单地将文件组装到tar文件，不压缩

.gz 一般.tar.gz比较常见，用gzip算法压缩文件

tar [-c -v -x -f -z -C] 参数1 参数2 ... 参数N

-c: create,创建压缩文件

-v:显示压缩，解压过程，用于查看进度

-x:解压模式

-f:要创建的文件，或者要解压的文件，-f必须位于所有选项位置中的最后一个

-z:zip模式，不使用的話就是tarball格式(一般要用的放在第一位)

-C:选择解压目的地，用于解压模式

eg. tar -cvf text.tar 1.txt 2.txt 3.txt

tar -zxvf text.gz -C /home/shed(一般-C最好单独使用)

tar -zxvf text.gz

zip

zip [-r] 参数1 参数2 ... 参数N

-r:被压缩的文件有文件夹时加-r

unzip [-d] 参数

-d:指定解压位置

解压的时候回替换同名内容

zip zip文件 [-d + 路径]

(解压，Ubuntu系的zip似乎会提示Nothing to do)

软件部署

Mysql

CentOS

Mysql 5.7

1.配置yum仓库

#更新密钥

rpm -import <https://repo.mysql.com/RPM-GPG-KEY-mysql-2022>

#安装Mysql yum库

rpm -Uvh <http://repo.mysql.com//mysql57-community-release-el7-7.noarch.rpm>

2.使用yum安装mysql

```
yum -y install mysql-community-server
```

问题: Unable to find a match: mysql-community-server

解决: yum module disable mysql

##禁用mysql模块

3.启动MYSQL并且配置为开机自启

MYSQL安装完成之后,会自动配置为名称叫做mysqld的服务,可以被systemctl所管理

```
systemctl start mysqld
```

```
systemctl enable mysqld
```

4.检查MYSQL的运行状态

```
systemctl status mysqld
```

配置

主要是配置管理员账户root的密码以及配置允许远程登录的权限

1.获取MYSQL的初始密码

```
grep 'temporary password' /var/log/mysqld.log
```

2.登录MYSQL数据库系统

#执行

```
mysql -uroot -p
```

#-u:要登陆的用户

#-p:用密码登录

3.修改root密码

#在MYSQL控制台内执行

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '密码';
```

#密码需要符合: 大于8位, 有大写字母, 有特殊符号, 不能是连续简单语句如123, abc

#eg.MYSQLNB666^;shedPC6!

4.[扩展],配置简单的密码

我们可以给root设置简单的密码, 如123456

请注意，此配置仅适用于学习或测试环境的MYSQL，正式使用的话，请不要设置简单密码

```
#降低MYSQL的密码安全级别
set global validate_password_policy=LOW;# 密码安全级别为最低
set global validate_password_length=4;#密码长度最低为4位

#然后既可以设置简单密码了
ALTER USER 'root'@'localhost' IDENTIFIED BY '密码';
```

5.[扩展]，配置root远程登陆

```
grant all privileges on *.* to root@"ip地址" identified by '密码' with grant
option;
# %表示任何ip都可以登录
#password:eg.123456
```

6.退出控制台页面

```
exit
```

MYSQL8.0

安装同样需要root权限

1.配置yum仓库

```
#更新密钥
rpm -import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022

#安装Mysql yum库
rpm -Uvh http://repo.mysql.com//mysql80-community-release-el7-2.noarch.rpm
```

2.使用yum安装mysql

```
yum -y install mysql-community-server
```

3.启动MYSQL并且配置为开机自启

MYSQL安装完成之后，会自动配置为名称叫做mysqld的服务，可以被systemctl所管理

```
systemctl start mysqld
systemctl enable mysqld
```

4.检查MYSQL的运行状态

```
systemctl status mysqld
```

配置

主要是配置管理员账户root的密码以及配置允许远程登录的权限

1.获取MYSQL的初始密码

```
grep 'temporary password' /var/log/mysqld.log
```

2.登录MYSQL数据库系统

```
#执行
mysql -uroot -p
#-u:要登陆的用户
#-p:用密码登录
```

无法登陆时可以跳过密码检查

ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)

vim /etc/my.cnf

末尾添加skip-grant-tables

然后重启服务

```
systemctl restart mysqld
```

3.修改root密码

```
#在MYSQL控制台内执行
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '密码';
#密码需要符合：大于8位，有大写字母,有特殊符号，不能是连续简单语句如123, abc
#eg.MYSQLNB666^;shedPC6!
```

4.[扩展],配置简单的密码

我们可以给root设置简单的密码，如123456

请注意，此配置仅适用于学习或测试环境的MYSQL，正式使用的话，请不要设置简单密码

#降低MYSQL的密码安全级别

```
set global validate_password.policy=0;# 密码安全级别为最低
```

```
set global validate_password.length=4;#密码长度最低为4位
```

#然后既可以设置简单密码了

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '密码';
```

报错ERROR 1193 (HY000): Unknown system variable 'validate_password.policy'的解决方法
这个版本的mysql没有安装扩展validate_password所以导致了validate_password.policy这个变量不存在

获取插件路径

```
show variables like "%plugin_dir%";
```

安装插件

```
install plugin validate_password soname 'validate_password.so';
```

检验插件已安装

```
SELECT PLUGIN_NAME, PLUGIN_LIBRARY, PLUGIN_STATUS, LOAD_OPTION FROM  
INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME = 'validate_password';
```

返回值

+	+	+	+	+				
	PLUGIN_NAME		PLUGIN_LIBRARY		PLUGIN_STATUS		LOAD_OPTION	
+	+	+	+	+				
	validate_password		validate_password.so		ACTIVE		ON	
+	+	+	+	+				

确定有这个变量

```
show variables like 'validate_password%';
```

返回值

+	+			
	Variable_name		Value	
+	+			
	validate_password_check_user_name		ON	
	validate_password_dictionary_file			
	validate_password_length		8	
	validate_password_mixed_case_count		1	
	validate_password_number_count		1	
	validate_password_policy		MEDIUM	

```
| validate_password_special_char_count | 1 |
```

```
+-----+-----+
```

注意，变量名字是validate_password_policy而不是validate_password.policy

5.[扩展]，配置root远程登陆

第一次设置root远程登录，并且配置远程密码时使用如下的sql命令

```
create user 'root'@'%' IDENTIFIED WITH mysql_native_password BY '密码';
```

#后续修改远程登录的密码

```
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '密码';
```

ERROR 1396 (HY000): Operation CREATE USER failed for 'root'@'%'的解决办法

应该是因为之前创建的用户信息没有清理干净

```
drop user root@'%';
```

```
flush privileges;
```

Ubuntu

MySQL5.7

由于最新的ubuntu应用商店里mysql是8.0版本的，所以我们需要额外操作来安装5.7版本的MySQL

1.下载apt仓库文件

#ubuntu下软件安装包是deb形式

```
wget https://dev.mysql.com/get/mysql-apt-config_0.8.12-1_all.deb
```

2.手动安装仓库

#使用dpkg命令安装仓库

```
dpkg -i mysql-apt-config_0.8.12-1_all.deb
```

会有一个小小的GUI安装界面

选择ubuntu bionic（这是ubuntu 18的代号）,enter

选择MySQL Server & Cluster(Currently selected:mysql-8.0),enter

选择mysql-5.7,再选ok

3.更新apt仓库的信息

```
#首先导入密钥信息（不然会说这个仓库是不受信任的）
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 467B942D3A79BD29
# 更新仓库信息(把远程仓库的信息缓存到本地，可以节约扫描仓库的大量时间)
apt update
```

4.检查是否成功配置了MySQL5.7的仓库

```
apt-cache policy mysql-server
```

5.安装mysql

```
apt install -f -y mysql-client=5.7.* mysql-community-server=5.7.*
```

安装过程中会要求配置root密码



6..启动mysql

```
#由于MySQL没有注册systemctl的服务，所以不用systemctl管理,但是有启动文件/etc/init.d/mysql
#但是在deepin下好像注册了mysql.service,可用systemctl status mysql查看状态
/etc/init.d/mysql start #启动
/etc/init.d/mysql stop #停止
/etc/init.d/mysql status #查看状态
#执行下面的命令，这是mysql安装后自带的配置程
```

7.初始化mysql

```
#执行下面的命令，这是mysql安装后自带的配置程序
mysql_secure_installation
```

- 1)输入密码
 - 2)询问是否开启密码安全性插件 (VALIDATE PASSWORD plugin) (看自己心情, 想开启就开启)
 - 3)询问是否修改root密码
 - 4)询问是否移除匿名的用户 (移除的话, 就不能以匿名用户登录) (因为匿名用户存在安全隐患, 所以选择n)
 - 5)是否禁止root远程登录 (y/回车跳过: 希望远程登录)
 - 6)是否移除自带的测试数据库 (学习用, 选择不移除, 回车)
 - 7)是否刷新权限 (让刚才做的更改立即生效)
- All done!

Mysql8.0

注意, 如果已经安装了mysql5.7要卸载相关信息

```
#卸载mysql5.7
apt remove -y mysql-client=5.7* mysql-community-server=5.7*
#卸载mysql5.7的仓库信息
dpkg -l | grep mysql | awk '{print $2}' | xargs dpkg -p
#在deepin下用dpkg -l |grep ^rc|awk '{print $2}' |sudo xargs dpkg -P
```

1.更新仓库信息

```
apt update
```

2.安装

```
apt install -y mysql-server
```

3.启动mysql

```
/etc/init.d/mysql start #启动
/etc/init.d/mysql stop  #停止
/etc/init.d/mysql status #查看状态
```

4.登录

```
mysql
#由于deepin版本的已经配置了密码, 所以还是用mysql -uroot -p登录
```

5.设置密码, 初始化操作和centos下一样

Tomcat

Tomcat是由Apache开发的一个Servlet容器, 实现了对Servlet和JSP的支持, 并提供了作为WEB服务器特有的一些功能, 如Tomcat管理和控制平台, 安全与管理和Tomcat阀等

安装JDK环境

本次使用的版本是10.0.27,最低要求的JDK版本是JDK8

可以自己去官网下载安装，或者用yum，好像centos7内置了openjdk1.8，就不演示了

1.创建文件夹,软件就放在里面

```
mkdir -p /export/server/jdk
```

2.切到bin里面后有./bin文件夹

3.配置环境变量（centos已经配置好了）

vim 编辑 /etc/profile

```
export JAVA_HOME=/export/server/jdk
export PATH=$PATH:JAVA_HOME/bin
```

source /etc/profile生效

4.自己配制的环境变量在系统自带的后面，要不配置环境变量，要不删掉系统自带的，要不就用系统自带

Tomcat部署

Tomcat建议以非root用户安装并启动，防止网站被黑了导致安全隐患

1.构建普通用户

```
useradd tomcat
```

2.下载安装包

```
#root操作
wget https://d1cdn.apache.org/tomcat/tomcat-10/v10.0.27/bin/apache-tomcat-10.0.27.tar.gz
#如果出现网络错误
wget --no-check-certificate
https://d1cdn.apache.org/tomcat/tomcat-10/v10.0.27/bin/apache-tomcat-10.0.2
```

3.解压

```
tar -zxvf apache-tomcat-10.0.27.tar.gz
```

4.创建Tomcat并切换

5.启动tomcat

```
/export/server/jdk/apache-tomcat-10.0.27/bin/startup.sh
```

6.tomcat默认在8080端口启动，检验8080是否启动

```
netstat -anp|grep 8080
```

7.放行8080

方法一：关闭防火墙

方法二：配置防火墙，放行端口（太复杂了，选方法1）

#方法一，关闭防火墙

```
systemctl stop firewalld # 关闭防火墙
```

```
systemctl disable firewalld #停止防火墙
```

#方法二，配置防火墙


```
firewall-cmd --add-port=8080/tcp --permanent #--add-port=8080/tcp表示放行8080端口的tcp访问，--permanent表示永久有效
```

```
firewall-cmd --reload
```


这样就可以从外部访问8080端口的网页了，服务器配置成功了

[Home](#) [Documentation](#) [Configuration](#) [Examples](#) [Wiki](#) [Mailing Lists](#) [Find Help](#)

Apache Tomcat/10.0.27



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:

- [Security Considerations How-To](#)
- [Manager Application How-To](#)
- [Clustering/Session Replication How-To](#)

[Server Status](#)

[Manager App](#)

[Host Manager](#)

Developer Quick Start

Tomcat Setup	Realms & AAA	Examples	Servlet Specifications
First Web Application	JDBC DataSources		Tomcat Versions

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 10.0 access to the manager application is split between different users.
[Read more...](#)

[Release Notes](#)

[Changelog](#)

[Migration Guide](#)

[Security Notices](#)

Documentation

[Tomcat 10.0 Documentation](#)

[Tomcat 10.0 Configuration](#)

[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

- [Tomcat 10.0 Bug Database](#)
- [Tomcat 10.0 JavaDocs](#)
- [Tomcat 10.0 Git Repository at GitHub](#)

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

[tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)
User support and discussion

[taglibs-user](#)
User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)
Development mailing list, including commit messages

Other Downloads <ul style="list-style-type: none">Tomcat ConnectorsTomcat NativeTaglibsDeployer	Other Documentation <ul style="list-style-type: none">Tomcat Connectorsmod_jk DocumentationTomcat NativeDeployer	Get Involved <ul style="list-style-type: none">OverviewSource RepositoriesMailing ListsWiki	Miscellaneous <ul style="list-style-type: none">ContactLegalSponsorshipThanks	Apache Software Foundation <ul style="list-style-type: none">Who We AreHeritageApache HomeResources
---	--	---	---	---

Copyright ©1999-2023 Apache Software Foundation. All Rights Reserved

Nginx

nginx(engine x)是一个高性能Http和反向代理的web服务器，同时也提供IMAP/POP3/SMTP服务同tomcat一样，Nginx可以托管用户编写的WEB应用程序成可访问的网页服务，同时也可以作为流量代理服务器，控制流量的中转

安装

同样需要配置额外的yum仓库，root权限

1. 安装yum依赖程序

```
# root执行
yum install -y yum-utils
```

2. 手动添加，nginx的yum仓库

yum程序使用的仓库配置文件，存放在：[/etc/yum.repos.d](#) 内(nginx的安装包在仓库中不存在，要自己倒入网站)

```
# root执行
# 创建文件使用vim编辑
vim /etc/yum.repos.d/nginx.repo
# 填入如下内容并保存退出
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true

[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

通过如上操作，我们手动添加了nginx的yum仓库

3. 通过yum安装最新稳定版的nginx

```
# root执行
yum install -y nginx
```

4. 启动

```
# nginx自动注册了systemctl系统服务
systemctl start nginx      # 启动
systemctl stop nginx       # 停止
systemctl status nginx     # 运行状态
systemctl enable nginx     # 开机自启
systemctl disable nginx    # 关闭开机自启
```

5. 配置防火墙放行

nginx默认绑定80端口，需要关闭防火墙或放行80端口

```
# 方式1（推荐），关闭防火墙
systemctl stop firewalld      # 关闭
systemctl disable firewalld   # 关闭开机自启

# 方式2，放行80端口
firewall-cmd --add-port=80/tcp --permanent  # 放行tcp规则下的80端口，永久生效
firewall-cmd --reload           # 重新加载防火墙规则
```

6. 启动后浏览器输入Linux服务器的IP地址或主机名即可访问

<http://192.168.88.130> 或 <http://centos>

ps：80端口是访问网站的默认端口，所以后面无需跟随端口号

显示的指定端口也是可以的比如：

- <http://192.168.88.130:80>（80端口是默认访问端口）
- <http://centos:80>

至此，Nginx安装配置完成。

RabbitMQ

RabbitMQ一款知名的开源消息队列系统，为企业提供消息的发布、订阅、点对点传输等消息服务。

安装

rabbitmq在yum仓库中的版本比较老，所以我们需要手动构建yum仓库

1. 准备yum仓库

```
# root执行
# 1. 准备gpgkey密钥
```

```
rpm --import https://github.com/rabbitmq/signing-  
keys/releases/download/2.0/rabbitmq-release-signing-key.asc  
rpm --import https://packagecloud.io/rabbitmq/erlang/gpgkey  
rpm --import https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey
```

2. 准备仓库文件

```
vim /etc/yum.repos.d/rabbitmq.repo
```

填入如下内容

##

Zero dependency Erlang

##

```
[rabbitmq_erlang]
```

```
name=rabbitmq_erlang
```

```
baseurl=https://packagecloud.io/rabbitmq/erlang/el/7/$basearch
```

```
repo_gpgcheck=1
```

```
gpgcheck=1
```

```
enabled=1
```

```
# PackageCloud's repository key and RabbitMQ package signing key
```

```
gpgkey=https://packagecloud.io/rabbitmq/erlang/gpgkey
```

```
https://github.com/rabbitmq/signing-
```

```
keys/releases/download/2.0/rabbitmq-release-signing-key.asc
```

```
sslverify=1
```

```
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

```
metadata_expire=300
```

```
[rabbitmq_erlang-source]
```

```
name=rabbitmq_erlang-source
```

```
baseurl=https://packagecloud.io/rabbitmq/erlang/el/7/SRPMS
```

```
repo_gpgcheck=1
```

```
gpgcheck=0
```

```
enabled=1
```

```
# PackageCloud's repository key and RabbitMQ package signing key
```

```
gpgkey=https://packagecloud.io/rabbitmq/erlang/gpgkey
```

```
https://github.com/rabbitmq/signing-
```

```
keys/releases/download/2.0/rabbitmq-release-signing-key.asc
```

```
sslverify=1
```

```
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

```
metadata_expire=300
```

##

RabbitMQ server

##

```
[rabbitmq_server]
```

```
name=rabbitmq_server
```

```
baseurl=https://packagecloud.io/rabbitmq/rabbitmq-server/el/7/$basearch
```

```
repo_gpgcheck=1
```

```
gpgcheck=0
```

```
enabled=1
```

```
# PackageCloud's repository key and RabbitMQ package signing key
gpgkey=https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey
      https://github.com/rabbitmq/signing-
keys/releases/download/2.0/rabbitmq-release-signing-key.asc
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300

[rabbitmq_server-source]
name=rabbitmq_server-source
baseurl=https://packagecloud.io/rabbitmq/rabbitmq-server/el/7/SRPMS
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
```

2. 安装RabbitMQ

```
# root执行
yum install erlang rabbitmq-server -y
#rabbitmq的开发语言是erlang
```

```
Installed:
  erlang.x86_64 0:23.3.4.11-1.el7          rabbitmq-server.noarch
  0:3.10.0-1.el7
```

3. 启动

```
# root执行
# 使用systemctl管控，服务名：rabbitmq-server
systemctl enable rabbitmq-server      # 开机自启
systemctl disable rabbitmq-server     # 关闭开机自启
systemctl start rabbitmq-server       # 启动
systemctl stop rabbitmq-server        # 关闭
systemctl status rabbitmq-server      # 查看状态
```

4. 放行防火墙，RabbitMQ使用5672、15672、25672 3个端口

```
# 方式1（推荐），关闭防火墙
systemctl stop firewalld          # 关闭
systemctl disable firewalld       # 关闭开机自启

# 方式2，放行5672 25672端口
firewall-cmd --add-port=5672/tcp --permanent    # 放行tcp规则下的5672
                                                    # 端口，永久生效
firewall-cmd --add-port=15672/tcp --permanent    # 放行tcp规则下的15672
                                                    # 端口，永久生效
firewall-cmd --add-port=25672/tcp --permanent    # 放行tcp规则下的25672
                                                    # 端口，永久生效
firewall-cmd --reload                # 重新加载防火墙规则
```

5. 启动RabbitMQ的WEB管理控制台

```
rabbitmq-plugins enable rabbitmq_management
```

6. 添加admin用户，并赋予权限

```
rabbitmqctl add_user admin 'shedServer66^'（密码是shedServer66^）
rabbitmqctl set_permissions -p "/" "admin" ".*" ".*" ".*"（给admin用户配置
最大权限）
rabbitmqctl set_user_tags admin administrator（标记admin为管理员）
```

7. 浏览器打开管理控制台

<http://192.168.88.130:15672>

The screenshot shows the RabbitMQ Management UI. At the top, there's a header with the RabbitMQ logo, version (3.10.0), and Erlang version (23.3.4.11). Below the header, there's a navigation bar with tabs: Overview, Connections, Channels, Exchanges, Queues, and Admin. The Overview tab is selected. The main content area shows 'Overview' with a 'Totals' section. It displays various metrics like 'Queued messages', 'Currently idle', 'Message rates', and 'Global counts'. There are also buttons for 'Connections', 'Channels', 'Exchanges', 'Queues', and 'Consumers'. Below this, there's a 'Nodes' section with a table showing details for the 'rabbit@centos' node, including file descriptors, socket descriptors, Erlang processes, memory, disk space, and uptime. At the bottom, there are links for 'Churn statistics', 'Ports and contexts', 'Export definitions', and 'Import definitions'. The footer contains links to 'HTTP API', 'Server Docs', 'Tutorials', 'Community Support', 'Community Slack', 'Commercial Support', 'Plugins', 'GitHub', and 'Changelog'.

至此，RabbitMQ已经安装完成了。

Redis

redis是一个开源的、使用C语言编写的、支持网络交互的、可基于内存也可持久化的Key-Value数据库。

redis的特点就是：**快**，可以基于内存存储数据并提供超低延迟、超快的检索速度

一般用于在系统中提供快速缓存的能力。

1. 配置 EPEL 仓库

EPEL 的全名叫 Extra Packages for Enterprise Linux。EPEL是由 Fedora 社区打造，为 RHEL 及衍生发行版如 CentOS、Scientific Linux 等提供高质量软件包的项目。装上了 EPEL之后，就相当于添加了一个第三方源。EPEL则为服务器版本提供大量的rpm包(yum 程序所使用的程序安装包，类似Windows的exe)，而且大多数rpm包在官方 repository 中 是找不到的。

```
# root执行
yum install -y epel-release
```

2. 安装redis

```
# root执行
yum install -y redis
```

3. 启动redis

```
# root执行
# 使用systemctl管控，服务名：redis
systemctl enable redis      # 开机自启
systemctl disable redis     # 关闭开机自启
systemctl start redis       # 启动
systemctl stop redis        # 关闭
systemctl status redis      # 查看状态
```

4. 放行防火墙，redis使用端口6379

```
# 方式1（推荐），关闭防火墙
systemctl stop firewalld      # 关闭
systemctl disable firewalld   # 关闭开机自启

# 方式2，放行6379端口
firewall-cmd --add-port=6379/tcp --permanent      # 放行tcp规则下的6379
端口，永久生效
firewall-cmd --reload
```

5. 进入redis服务


```
# 执行redis-cli
[root@centos ~]# redis-cli
127.0.0.1:6379> set mykey hello
OK
127.0.0.1:6379> get mykey
"hello"
127.0.0.1:6379>
```

至此，redis安装完成。

ElasticSearch

全文搜索属于最常见的需求，开源的 **Elasticsearch**（以下简称 es）是目前全文搜索引擎的首选。

它可以快速地储存、搜索和分析海量数据。维基百科、Stack Overflow、Github 都采用它。

Elasticsearch简称es，在企业内同样是一款应用非常广泛的搜索引擎服务。

很多服务中的搜索功能，都是基于es来实现的。

1. 添加yum仓库

```
# root执行
# 导入仓库密钥
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

# 添加yum源
# 编辑文件
vim /etc/yum.repos.d/elasticsearch.repo

[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md

# 更新yum缓存
yum makecache
```

2. 安装es

```
yum install -y elasticsearch
```

3. 配置es

```
vim /etc/elasticsearch/elasticsearch.yml

# 17行, 设置集群名称
cluster.name: my-cluster

# 23行, 设置节点名称
node.name: node-1

# 56行, 允许外网访问
network.host: 0.0.0.0 (绑定ip, 意思是这个ip可以访问, 这里是指任意ip都可以访问)

# 74行, 配置集群master节点
cluster.initial_master_nodes: ["node-1"]
```

4. 启动es

```
systemctl start | stop | status | enable | disable elasticsearch
```

5. 关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

6. 测试

浏览器打开: <http://ip:9200/?pretty>

← → ↻ 🏠 ⚠ 不安全 | 192.168.88.130:9200/?pretty

```
{
  "name" : "node-1",
  "cluster_name" : "my-application",
  "cluster_uuid" : "DRbuWXEBQlOVrmLYiRpGAg",
  "version" : {
    "number" : "7.17.6",
    "build_flavor" : "default",
    "build_type" : "rpm",
    "build_hash" : "f65e9d338dc1d07b642e14a27f338990148ee5b6",
    "build_date" : "2022-08-23T11:08:48.893373482Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

集群化虚拟机

介绍

在前面，我们所学习安装的软件，都是以单机模式运行的。

后续，我们将要学习大数据相关的软件部署，所以后续我们所安装的软件服务，大多数都是以集群化（多台服务器共同工作）模式运行的。

所以，在当前小节，我们需要完成集群化环境的前置准备，包括创建多台虚拟机，配置主机名映射，SSH免密登录等等。

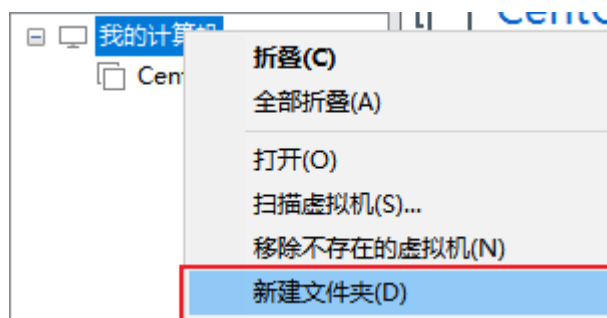
部署

配置多台Linux虚拟机

安装集群化软件，首要条件就是要有多台Linux服务器可用。

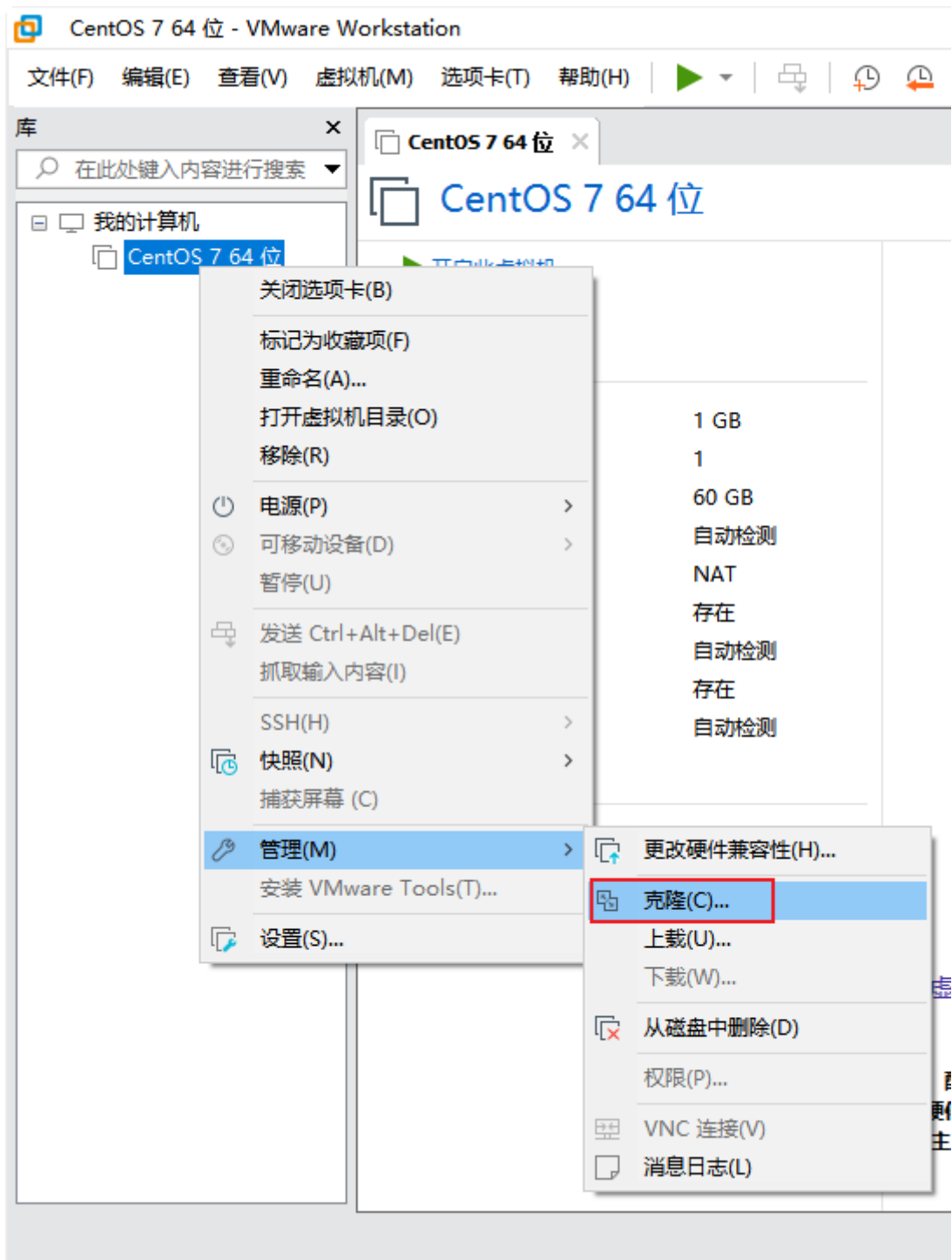
我们可以使用VMware提供的克隆功能，将我们的虚拟机额外克隆出3台来使用。

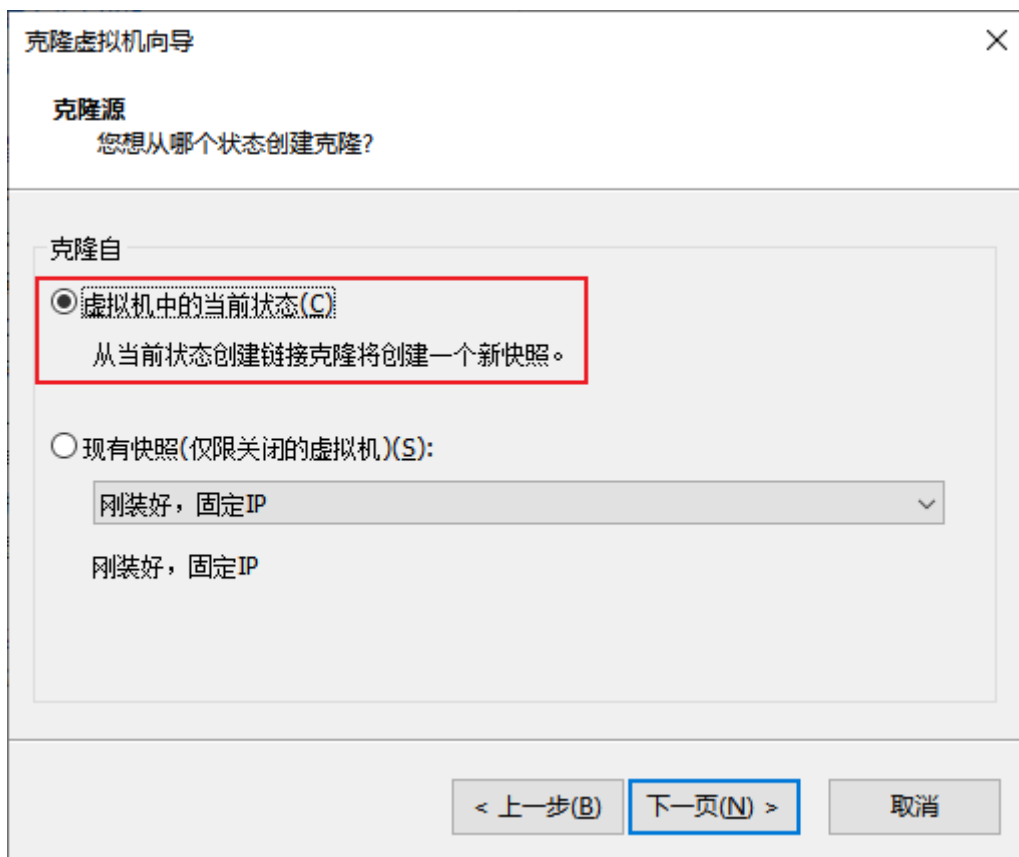
1. 首先，关机当前CentOS系统虚拟机（可以使用root用户执行 `init 0` 来快速关机）
2. 新建文件夹



文件夹起名为：虚拟机集群

3. 克隆





克隆虚拟机向导

×

克隆类型

您希望如何克隆此虚拟机?

克隆方法

☐ 创建链接克隆(L)

链接克隆是对原始虚拟机的引用，所需的存储磁盘空间较少。但是，必须能够访问原始虚拟机才能运行。

☒ 创建完整克隆(F)

完整克隆是原始虚拟机当前状态的完整副本。此副本虚拟机完全独立，但需要较多的存储磁盘空间。

< 上一步(B)

下一页(N) >

取消

克隆虚拟机向导

×

新虚拟机名称

您希望该虚拟机使用什么名称?

虚拟机名称(V)

node1

位置(L)

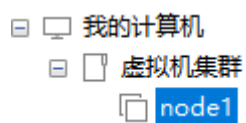
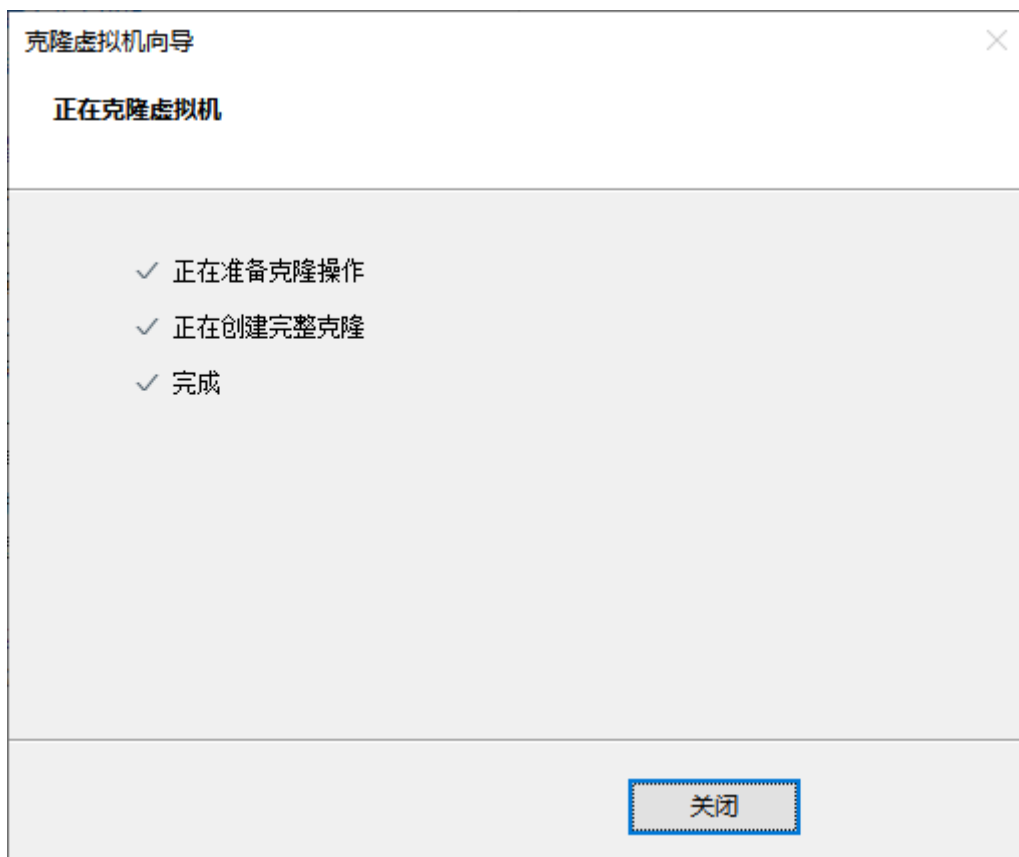
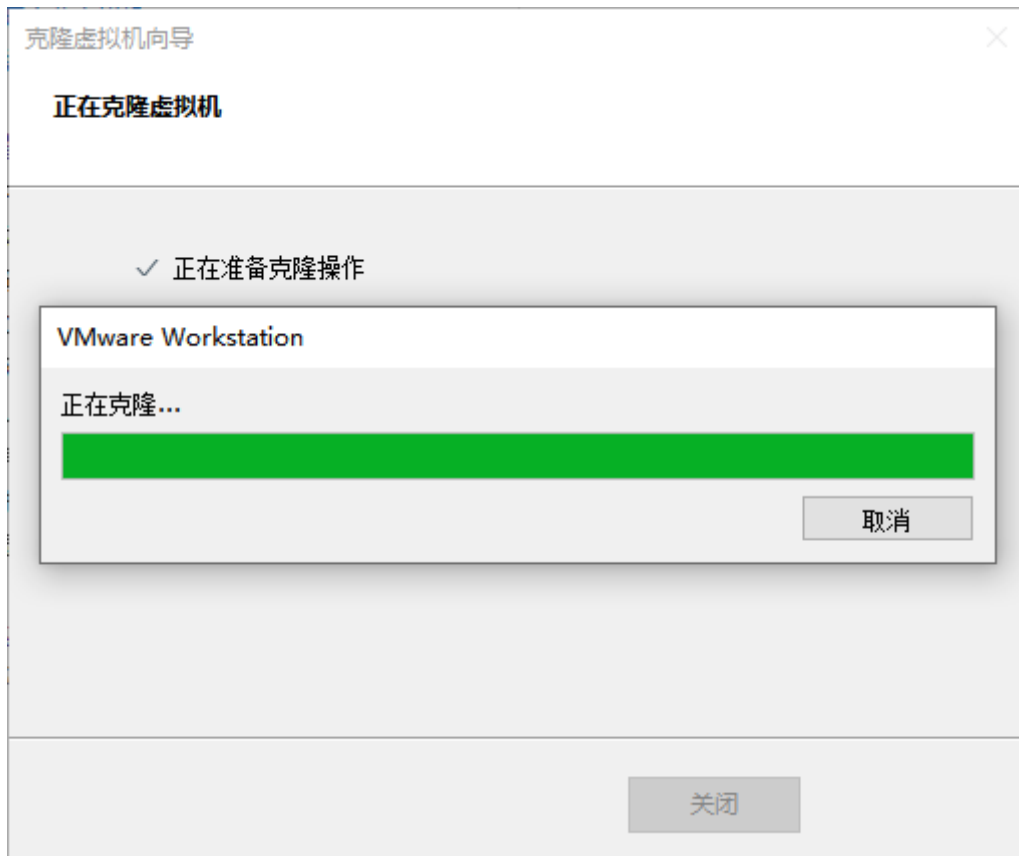
E:\vm\cluster\node1

浏览(R)...

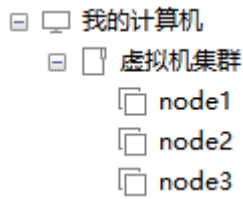
< 上一步(B)

完成

取消



4. 同样的操作克隆出：node2和node3



5. 开启node1，修改主机名为node1，并修改固定ip为：192.168.88.131

```
# 修改主机名
hostnamectl set-hostname node1

# 修改IP地址
vim /etc/sysconfig/network-scripts/ifcfg-ens33
IPADDR="192.168.88.131"

# 重启网卡
systemctl stop network
systemctl start network

# 或者直接
systemctl restart network
```

6. 同样的操作启动node2和node3,

修改node2主机名为node2，设置ip为192.168.88.132

修改node2主机名为node3，设置ip为192.168.88.133

7. 配置FinalShell，配置连接到node1、node2、node3的连接

为了简单起见，建议配置root用户登录

准备主机名映射

1. 在Windows系统中修改hosts文件，填入如下内容：

如果同学们使用MacOS系统，请：

1. sudo su -, 切换到root
2. 修改/etc/hosts文件

```
192.168.67.5 node1
192.168.67.3 node2
192.168.67.4 node3
192.168.67.2 gateway
```

2. 在3台Linux的/etc/hosts文件中，填入如下内容（==3台都要添加==）


```
192.168.88.131 node1
192.168.88.132 node2
192.168.88.133 node3
```

SSH免密配置

后续安装的集群化软件，多数需要远程登录以及远程执行命令，我们可以简单起见，配置三台Linux服务器之间的免密码互相SSH登陆

1. 在每一台机器都执行：`ssh-keygen -t rsa -b 4096`，一路回车到底即可（为当前服务器生成远程登录的密钥）
2. 在每一台机器都执行：

```
ssh-copy-id node1 #把ssh密鑰給node1
ssh-copy-id node2 #把ssh密鑰給node2
ssh-copy-id node3 #把ssh密鑰給node3
```

3. 执行完毕后，node1、node2、node3之间将完成root用户之间的免密互通

配置JDK环境

后续的大数据集群软件，多数是需要Java运行环境的，所以我们为==每一台==机器都配置JDK环境。

JDK配置参阅：[Tomcat](#) 安装部署环节。

关闭防火墙和SELinux

集群化软件之间需要通过端口互相通讯，为了避免出现网络不通的问题，我们可以简单的在集群内部关闭防火墙。

==在每一台机器都执行==

```
systemctl stop firewalld
systemctl disable firewalld
```

Linux有一个安全模块：SELinux，用以限制用户和程序的相关权限，来确保系统的安全稳定。

SELinux的配置同防火墙一样，非常复杂

在当前，我们只需要关闭SELinux功能，避免导致后面的软件运行出现问题即可，

==在每一台机器都执行==

```
vim /etc/sysconfig/selinux
```

```
# 将第七行，SELINUX=enforcing 改为
```

```
SELINUX=disabled
```

```
# 保存退出后，重启虚拟机即可，千万要注意disabled单词不要写错，不然无法启动系统
```

```
init 0 服务器关机
```

```
init 6 服务器重启
```

添加快照

为了避免后续出现问题，在完成上述设置后，为+=每一台虚拟机==都制作快照，留待使用。

scp命令

后续的安装部署操作，我们将会频繁的在多台服务器之间相互传输数据。

为了更加方便的互相传输，我们补充一个命令：scp

scp命令是cp命令的升级版，即：ssh cp，通过SSH协议完成文件的复制。

其主要功能就是：在不同的Linux服务器之间，通过 SSH 协议互相传输文件。

只要知晓服务器的账户和密码（或密钥），即可通过SCP互传文件。

语法：

```
scp [-r] 参数1 参数2
```

```
- -r选项用于复制文件夹使用，如果复制文件夹，必须使用-r
```

```
- 参数1: 本机路径 或 远程目标路径
```

```
- 参数2: 远程目标路径 或 本机路径
```

```
把参数1的内容复制到参数2
```

如：

```
scp -r /export/server/jdk root@node2:/export/server/
```

将本机上的jdk文件夹，以root的身份复制到node2的/export/server/内
同SSH登陆一样，账户名可以省略（使用本机当前的同名账户登陆）

如：

```
scp -r node2:/export/server/jdk /export/server/
```

将远程node2的jdk文件夹，复制到本机的/export/server/内

```
# scp命令的高级用法
```

```
cd /export/server
```

```
scp -r jdk node2:`pwd`/      # 将本机当前路径的jdk文件夹，复制到node2服务器的同名路径下，``表示执行里面的程序
scp -r jdk node2:$PWD        # 将本机当前路径的jdk文件夹，复制到node2服务器的同名路径下，$表示取环境变量中的pwd，和pwd的返回值一样
```

Zookeeper集群安装部署

简介

ZooKeeper是一个**分布式的**，开放源码的**分布式应用程序**协调服务，是Hadoop和Hbase的重要组成部分。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

除了为Hadoop和HBase提供协调服务外，Zookeeper也被其它许多软件采用作为其分布式状态一致性的依赖，比如Kafka，又或者一些软件项目中，也经常能见到Zookeeper作为一致性协调服务存在。

Zookeeper不论是大数据领域亦或是其它服务器开发领域，涉及到分布式状态一致性的场景，总有它的身影存在。

安装

Zookeeper是一款分布式的集群化软件，可以在多台服务器上部署，并协同组成分布式集群一起工作。

1. 首先，要确保已经完成了 **集群化环境前置准备** 环节的全部内容
2. 【node1上操作】下载Zookeeper安装包，并解压

```
# 下载
wget http://archive.apache.org/dist/zookeeper/zookeeper-3.5.9/apache-zookeeper-3.5.9-bin.tar.gz

# 确保如下目录存在，不存在就创建
mkdir -p /export/server

# 解压
tar -zxvf apache-zookeeper-3.5.9-bin.tar.gz -C /export/server
```

3. 【node1上操作】创建软链接

```
ln -s /export/server/apache-zookeeper-3.5.9 /export/server/zookeeper
```

4. 【node1上操作】修改配置文件

```
vim /export/server/zookeeper/conf/zoo.cfg
```

```
tickTime=2000
# zookeeper数据存储目录
dataDir=/export/server/zookeeper/data
clientPort=2181
initLimit=5
syncLimit=2
server.1=node1:2888:3888
server.2=node2:2888:3888
server.3=node3:2888:3888
```

5. 【node1上操作】 配置 myid

```
# 1. 创建Zookeeper的数据目录
mkdir /export/server/zookeeper/data

# 2. 创建文件，并填入1
vim /export/server/zookeeper/data/myid
# 在文件内填入1即可（标识设备）
```

6. 【在node2和node3上操作】，创建文件夹

```
mkdir -p /export/server
```

7. 【node1上操作】将Zookeeper 复制到node2和node3

```
cd /export/server

scp -r apache-zookeeper-3.5.9 node2:`pwd`/
scp -r apache-zookeeper-3.5.9 node3:`pwd`/
```

8. 【在node2上操作】

```
# 1. 创建软链接
ln -s /export/server/apache-zookeeper-3.5.9 /export/server/zookeeper

# 2. 修改myid文件
vim /export/server/zookeeper/data/myid
# 修改内容为2
```

9. 【在node3上操作】

```
# 1. 创建软链接
ln -s /export/server/apache-zookeeper-3.5.9 /export/server/zookeeper

# 2. 修改myid文件
vim /export/server/zookeeper/data/myid
# 修改内容为3
```

10. 【在node1、node2、node3上分别执行】 启动Zookeeper

```
# 启动命令
/export/server/zookeeper/bin/zkServer.sh start      # 启动Zookeeper
```

11. 【在node1、node2、node3上分别执行】 检查Zookeeper进程是否启动

```
jps

# 结果中找到有：QuorumPeerMain 进程即可（系统自带的openjdk好像没有jps，不能执行）
```

12. 【node1上操作】 验证Zookeeper

```
/export/server/zookeeper/zkCli.sh

# 进入到Zookeeper控制台中后，执行
ls /

# 如无报错即配置成功
```

至此Zookeeper安装完成

补充

没有jps

```
yum install -y java-1.8.0-openjdk-devel
```

安装jps插件

CentOS ping不同外网

- 1.确保当前虚拟机为nat模式
- 2.设置静态ip
- 3.编辑DNS

```
vim /etc/resolv.conf

nameserver 114.114.114.114
```

4.重启网络服务

zookeeper启动失败

1.确定失败原因

```
./zkServer.sh start-foreground
```

Caused by: java.lang.IllegalArgumentException: myid file is missing

at

org.apache.zookeeper.server.quorum.QuorumPeerConfig.checkValidity(QuorumPeerConfig.java:738)

at

org.apache.zookeeper.server.quorum.QuorumPeerConfig.setupQuorumPeerConfig(QuorumPeerConfig.java:609)

at

org.apache.zookeeper.server.quorum.QuorumPeerConfig.parseProperties(QuorumPeerConfig.java:423)

at

org.apache.zookeeper.server.quorum.QuorumPeerConfig.parse(QuorumPeerConfig.java:153)

... 2 more

2.myid要放在data下

Kafka集群安装部署

简介

Kafka是一款 分布式的、去中心化的、高吞吐低延迟、订阅模式 的消息队列系统。

同RabbitMQ一样，Kafka也是消息队列。不过RabbitMQ多用于后端系统，因其更加专注于消息的延迟和容错。

Kafka多用于大数据体系，因其更加专注于数据的吞吐能力。

Kafka多数都是运行在分布式（集群化）模式下，所以课程将以3台服务器，来完成Kafka集群的安装部署。

安装

1. 确保已经跟随前面的视频，安装并部署了JDK和Zookeeper服务

Kafka的运行依赖JDK环境和Zookeeper请确保已经有了JDK环境和Zookeeper

2. 【在node1操作】下载并上传Kafka的安装包

```
# 下载安装包
wget http://archive.apache.org/dist/kafka/2.4.1/kafka_2.12-2.4.1.tgz
```

3. 【在node1操作】解压

```
mkdir -p /export/server          # 此文件夹如果不存在需先创建

# 解压
tar -zxvf kafka_2.12-2.4.1.tgz -C /export/server/

# 创建软链接
ln -s /export/server/kafka_2.12-2.4.1 /export/server/kafka
```

4. 【在node1操作】修改Kafka目录内的config目录内的 `server.properties` 文件

```
cd /export/server/kafka/config
# 指定broker的id
broker.id=1
# 指定 kafka的绑定监听的地址
listeners=PLAINTEXT://node1:9092
# 指定Kafka数据的位置
log.dirs=/export/server/kafka/data
# 指定Zookeeper的三个节点
zookeeper.connect=node1:2181,node2:2181,node3:2181
```

5. 【在node1操作】将node1的kafka复制到node2和node3

```
cd /export/server

# 复制到node2同名文件夹
scp -r kafka_2.12-2.4.1 node2:`pwd`/
# 复制到node3同名文件夹
scp -r kafka_2.12-2.4.1 node3:$PWD
```

6. 【在node2操作】

```
# 创建软链接
ln -s /export/server/kafka_2.12-2.4.1 /export/server/kafka

cd /export/server/kafka/config
# 指定broker的id
broker.id=2
# 指定 kafka的绑定监听的地址
listeners=PLAINTEXT://node2:9092
# 指定Kafka数据的位置
log.dirs=/export/server/kafka/data
# 指定Zookeeper的三个节点
zookeeper.connect=node1:2181,node2:2181,node3:2181
```

7. 【在node3操作】

```
# 创建软链接
ln -s /export/server/kafka_2.12-2.4.1 /export/server/kafka

cd /export/server/kafka/config
# 指定broker的id
broker.id=3
# 指定 kafka的绑定监听的地址
listeners=PLAINTEXT://node3:9092
# 指定Kafka数据的位置
log.dirs=/export/server/kafka/data
# 指定Zookeeper的三个节点
zookeeper.connect=node1:2181,node2:2181,node3:2181
```

8. 启动kafka

```
# 请先确保Zookeeper已经启动了

# 方式1: 【前台启动】分别在node1、2、3上执行如下语句（关闭finalshell窗口软件就没有了）
/export/server/kafka/bin/kafka-server-start.sh
/export/server/kafka/config/server.properties

# 方式2: 【后台启动】分别在node1、2、3上执行如下语句
nohup /export/server/kafka/bin/kafka-server-start.sh
/export/server/kafka/config/server.properties 2>&1 >>
/export/server/kafka/kafka-server.log &
```

9. 验证Kafka启动

```
# 在每一台服务器执行
jps
```



```
[root@node1 server]# jps
12930 NodeManager
14082 QuorumPeerMain
11843 DataNode
11652 NameNode
14340 HMaster
14599 ThriftServer
13705 JobHistoryServer
12746 ResourceManager
12237 SecondaryNameNode
19247 Jps
13428 WebAppProxyServer
14460 HRegionServer
14783 Kafka
```

测试Kafka能否正常使用

1. 创建测试主题

```
# 在node1执行，创建一个主题
/export/server/kafka_2.12-2.4.1/bin/kafka-topics.sh --create --zookeeper
node1:2181 --replication-factor 1 --partitions 3 --topic test
```

2. 运行测试，请在FinalShell中打开2个node1的终端页面

```
# 打开一个终端页面，启动一个模拟的数据生产者
/export/server/kafka_2.12-2.4.1/bin/kafka-console-producer.sh --broker-list
node1:9092 --topic test
# 再打开一个新的终端页面，启动一个模拟的数据消费者
/export/server/kafka_2.12-2.4.1/bin/kafka-console-consumer.sh --bootstrap-
server node1:9092 --topic test --from-beginning
```

大数据集群（Hadoop生态）安装部署

简介

- 1) Hadoop是一个由Apache基金会所开发的分布式系统基础架构。
- 2) 主要解决，海量数据的存储和海量数据的分析计算问题。

Hadoop HDFS 提供分布式海量数据存储能力

Hadoop YARN 提供分布式集群资源管理能力

Hadoop MapReduce 提供分布式海量数据计算能力

前置要求

- 请确保完成了集群化环境前置准备章节的内容
- 即：JDK、SSH免密、关闭防火墙、配置主机名映射等前置操作

Hadoop集群角色

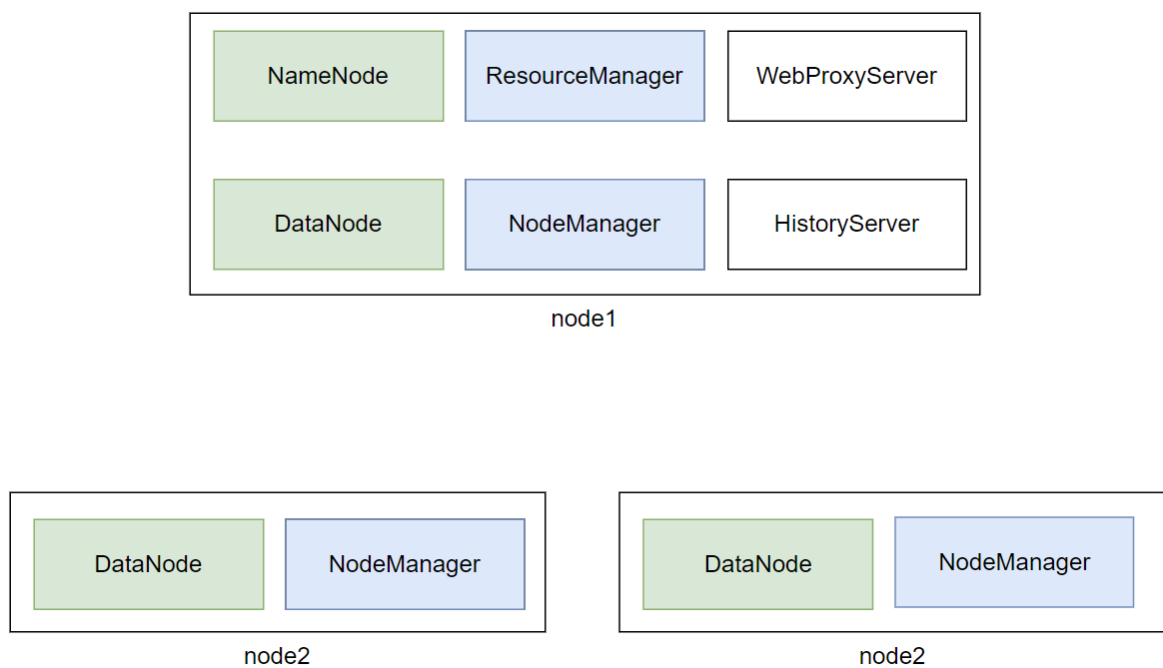
Hadoop生态体系中总共会出现如下进程角色：

1. Hadoop HDFS的管理角色：Namenode进程（ 仅需1个即可（管理者一个就够） ）
2. Hadoop HDFS的工作角色：Datanode进程（ 需要多个（工人，越多越好，一个机器启动一个） ）
3. Hadoop YARN的管理角色：ResourceManager进程（ 仅需1个即可（管理者一个就够） ）
4. Hadoop YARN的工作角色：NodeManager进程（ 需要多个（工人，越多越好，一个机器启动一个） ）
5. Hadoop 历史记录服务器角色：HistoryServer进程（ 仅需1个即可（功能进程无需太多1个足够） ）
6. Hadoop 代理服务器角色：WebProxyServer进程（ 仅需1个即可（功能进程无需太多1个足够） ）
7. Zookeeper的进程：QuorumPeerMain进程（ 仅需1个即可（Zookeeper的工作者，越多越好） ）

角色和节点分配

角色分配如下：

1. node1: Namenode、Datanode、ResourceManager、NodeManager、HistoryServer、WebProxyServer、QuorumPeerMain
2. node2: Datanode、NodeManager、QuorumPeerMain
3. node3: Datanode、NodeManager、QuorumPeerMain



安装

调整虚拟机内存

如上图，可以看出node1承载了太多的压力。同时node2和node3也同时运行了不少程序

为了确保集群的稳定，需要对虚拟机进行内存设置。

请在VMware中，对：

1. node1设置4GB或以上内存
2. node2和node3设置2GB或以上内存

大数据的软件本身就是集群化（一堆服务器）一起运行的。

现在我们在电脑中以多台虚拟机来模拟集群，确实会有很大的内存压力哦。

Zookeeper集群部署

略

Hadoop集群部署

1. 下载Hadoop安装包、解压、配置软链接

```
# 1. 下载
wget http://archive.apache.org/dist/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz

# 2. 解压
# 请确保目录/export/server存在
tar -zxvf hadoop-3.3.0.tar.gz -C /export/server/

# 3. 构建软链接
ln -s /export/server/hadoop-3.3.0 /export/server/hadoop
```

2. 修改配置文件： `hadoop-env.sh`

Hadoop的配置文件要修改的地方很多，请细心

cd 进入到/export/server/hadoop/etc/hadoop，文件夹中，配置文件都在这里

修改hadoop-env.sh文件

此文件是配置一些Hadoop用到的环境变量

这些是临时变量，在Hadoop运行时有用

如果要永久生效，需要写到/etc/profile中

```
# 在文件开头加入：
# 配置Java安装路径
export JAVA_HOME=/export/server/jdk
# 配置Hadoop安装路径
export HADOOP_HOME=/export/server/hadoop
# Hadoop hdfs配置文件路径
```

```
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
# Hadoop YARN配置文件路径
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
# Hadoop YARN 日志文件夹
export YARN_LOG_DIR=$HADOOP_HOME/logs/yarn
# Hadoop hdfs 日志文件夹
export HADOOP_LOG_DIR=$HADOOP_HOME/logs/hdfs

# Hadoop的使用启动用户配置
export HDFS_NAMENODE_USER=root
export HDFS_DATANODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
export YARN_PROXYSERVER_USER=root
```

3. 修改配置文件： `core-site.xml`

如下，清空文件，填入如下内容

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
  implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://node1:8020</value>
    <description></description>
  </property>

  <property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
    <description></description>
  </property>
```

```
</configuration>
```

4. 配置: `hdfs-site.xml` 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
  implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.datanode.data.dir.perm</name>
    <value>700</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/data/nn</value>
    <description>Path on the local filesystem where the NameNode stores
the namespace and transactions logs persistently.</description>
  </property>

  <property>
    <name>dfs.namenode.hosts</name>
    <value>node1,node2,node3</value>
    <description>List of permitted DataNodes.</description>
  </property>

  <property>
    <name>dfs.blocksize</name>
    <value>268435456</value>
    <description></description>
  </property>

  <property>
```

```

    <name>dfs.namenode.handler.count</name>
    <value>100</value>
    <description></description>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/data/dn</value>
  </property>
</configuration>

```

5. 配置: `mapred-env.sh` 文件

```

# 在文件的开头加入如下环境变量设置
export JAVA_HOME=/export/server/jdk
export HADOOP_JOB_HISTORYSERVER_HEAPSIZE=1000
export HADOOP_MAPRED_ROOT_LOGGER=INFO,RFA

```

6. 配置: `mapred-site.xml` 文件

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
  implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property><?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

```

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

<description></description>

</property>

<property>

<name>mapreduce.jobhistory.address</name>

<value>node1:10020</value>

<description></description>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>node1:19888</value>

<description></description>

</property>

<property>

<name>mapreduce.jobhistory.intermediate-done-dir</name>

<value>/data/mr-history/tmp</value>

<description></description>

</property>

<property>

<name>mapreduce.jobhistory.done-dir</name>

<value>/data/mr-history/done</value>

<description></description>

</property>

<property>

<name>yarn.app.mapreduce.am.env</name>

<value>HADOOP_MAPRED_HOME=\$HADOOP_HOME</value>

</property>

<property>

<name>mapreduce.map.env</name>

<value>HADOOP_MAPRED_HOME=\$HADOOP_HOME</value>


```
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
</configuration>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
  <description></description>
</property>

  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>node1:10020</value>
    <description></description>
  </property>

  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>node1:19888</value>
    <description></description>
  </property>

  <property>
    <name>mapreduce.jobhistory.intermediate-done-dir</name>
    <value>/data/mr-history/tmp</value>
    <description></description>
  </property>

  <property>
    <name>mapreduce.jobhistory.done-dir</name>
    <value>/data/mr-history/done</value>
    <description></description>
  </property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
</configuration>
```

7. 配置: `yarn-env.sh` 文件

```
# 在文件的开头加入如下环境变量设置
export JAVA_HOME=/export/server/jdk
export HADOOP_HOME=/export/server/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_LOG_DIR=$HADOOP_HOME/logs/yarn
export HADOOP_LOG_DIR=$HADOOP_HOME/logs/hdfs
```

8. 配置: `yarn-site.xml` 文件

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
  implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.log.server.url</name>
    <value>http://node1:19888/jobhistory/logs</value>
    <description></description>
  </property>

  <property>
    <name>yarn.web-proxy.address</name>
    <value>node1:8089</value>
    <description>proxy server hostname and port</description>
  </property>

  <property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
    <description>Configuration to enable or disable log
aggregation</description>
  </property>
```

```

    <property>
      <name>yarn.nodemanager.remote-app-log-dir</name>
      <value>/tmp/logs</value>
      <description>Configuration to enable or disable log
aggregation</description>
    </property>

<!-- Site specific YARN configuration properties -->
    <property>
      <name>yarn.resourcemanager.hostname</name>
      <value>node1</value>
      <description></description>
    </property>

    <property>
      <name>yarn.resourcemanager.scheduler.class</name>

      <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.Fair
Scheduler</value>
      <description></description>
    </property>

    <property>
      <name>yarn.nodemanager.local-dirs</name>
      <value>/data/nm-local</value>
      <description>Comma-separated list of paths on the local filesystem
where intermediate data is written.</description>
    </property>

    <property>
      <name>yarn.nodemanager.log-dirs</name>
      <value>/data/nm-log</value>
      <description>Comma-separated list of paths on the local filesystem
where logs are written.</description>
    </property>

    <property>
      <name>yarn.nodemanager.log.retain-seconds</name>
      <value>10800</value>
      <description>Default time (in seconds) to retain log files on the
NodeManager Only applicable if log-aggregation is disabled.</description>
    </property>

    <property>

```

```
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
<description>Shuffle service that needs to be set for Map Reduce
applications.</description>
</property>
</configuration>
```

9. 修改workers文件

```
# 全部内容如下
node1
node2
node3
```

10. 分发hadoop到其它机器

```
# 在node1执行
cd /export/server

scp -r hadoop-3.3.0 node2:`pwd`/
scp -r hadoop-3.3.0 node3:`pwd`/
```

11. 在node2、node3执行

```
# 创建软链接
ln -s /export/server/hadoop-3.3.0 /export/server/hadoop
```

12. 创建所需目录

- 在node1执行：

```
mkdir -p /data/nn
mkdir -p /data/dn
mkdir -p /data/nm-log
mkdir -p /data/nm-local
```

- 在node2执行：

```
mkdir -p /data/dn
mkdir -p /data/nm-log
mkdir -p /data/nm-local
```

- 在node3执行：

```
mkdir -p /data/dn
mkdir -p /data/nm-log
mkdir -p /data/nm-local
```

13. 配置环境变量

在node1、node2、node3修改/etc/profile

```
export HADOOP_HOME=/export/server/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

执行 `source /etc/profile` 生效

14. 格式化NameNode，在node1执行

```
hadoop namenode -format
```

hadoop这个命令来自于：\$HADOOP_HOME/bin中的程序

由于配置了环境变量PATH，所以可以在任意位置执行hadoop命令哦

15. 启动hadoop的hdfs集群，在node1执行即可

```
start-dfs.sh
```

```
# 如需停止可以执行
stop-dfs.sh
```

start-dfs.sh这个命令来自于：\$HADOOP_HOME/sbin中的程序

由于配置了环境变量PATH，所以可以在任意位置执行start-dfs.sh命令哦

16. 启动hadoop的yarn集群，在node1执行即可

```
start-yarn.sh
```

```
# 如需停止可以执行
stop-yarn.sh
```

17. 启动历史服务器

```
mapred --daemon start historyserver
```

```
# 如需停止将start更换为stop
```

18. 启动web代理服务器

```
yarn-daemon.sh start proxyserver
```

如需停止将start更换为stop

验证Hadoop集群运行情况

1. 在node1、node2、node3上通过jps验证进程是否都启动成功
2. 验证HDFS，浏览器打开：<http://node1:9870>

创建文件test.txt，随意填入内容，并执行：

```
hadoop fs -put test.txt /test.txt
```

```
hadoop fs -cat /test.txt
```

3. 验证YARN，浏览器打开：<http://node1:8088>

执行：

创建文件words.txt，填入如下内容

```
itheima itcast hadoop
```

```
itheima hadoop hadoop
```

```
itheima itcast
```

将文件上传到HDFS中

```
hadoop fs -put words.txt /words.txt
```

执行如下命令验证YARN是否正常

```
hadoop jar /export/server/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount -Dmapred.job.queue.name=root.root /words.txt /output
```

大数据NoSQL数据库HBase集群部署

简介

HBase 是一种**分布式**、可扩展、支持海量数据存储的 NoSQL 数据库。

和Redis一样，HBase是一款KeyValue型存储的数据库。

不过和Redis设计方向不同

- Redis设计为少量数据，超快检索
- HBase设计为海量数据，快速检索

HBase在大数据领域应用十分广泛，现在我们来在node1、node2、node3上部署HBase集群。

安装

1. HBase依赖Zookeeper、JDK、Hadoop（HDFS），请确保已经完成前面

- 集群化软件前置准备（JDK）
- Zookeeper
- Hadoop
- 这些环节的软件安装

2. 【node1执行】下载HBase安装包

```
# 下载
wget http://archive.apache.org/dist/hbase/2.1.0/hbase-2.1.0-bin.tar.gz

# 解压
tar -zxvf hbase-2.1.0-bin.tar.gz -C /export/server

# 配置软链接
ln -s /export/server/hbase-2.1.0 /export/server/hbase
```

3. 【node1执行】，修改配置文件，修改 `conf/hbase-env.sh` 文件

```
# 在28行配置JAVA_HOME
export JAVA_HOME=/export/server/jdk
# 在126行配置：
# 意思表示，不使用HBase自带的Zookeeper，而是用独立Zookeeper
export HBASE_MANAGES_ZK=false
# 在任意行，比如26行，添加如下内容：
export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"
```

4. 【node1执行】，修改配置文件，修改 `conf/hbase-site.xml` 文件

```
# 将文件的全部内容替换成如下内容：
<configuration>
  <!-- HBase数据在HDFS中的存放的路径 -->
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://node1:8020/hbase</value>
  </property>
  <!-- hbase的运行模式。false是单机模式，true是分布式模式。若为
false,hbase和Zookeeper会运行在同一个JVM里面 -->
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
```

```

<!-- ZooKeeper的地址 -->
<property>
    <name>hbase.zookeeper.quorum</name>
    <value>node1,node2,node3</value>
</property>
<!-- ZooKeeper快照的存储位置 -->
<property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/export/server/apache-zookeeper-3.6.0-bin/data</value>
</property>
<!-- v2.1版本，在分布式情况下，设置为false -->
<property>
    <name>hbase.unsafe.stream.capability.enforce</name>
    <value>false</value>
</property>
</configuration>

```

5. 【node1执行】，修改配置文件，修改 `conf/regionserver` 文件

```

# 填入如下内容
node1
node2
node3

```

6. 【node1执行】，分发hbase到其它机器

```

scp -r /export/server/hbase-2.1.0 node2:/export/server/
scp -r /export/server/hbase-2.1.0 node3:/export/server/

```

7. 【node2、node3执行】，配置软链接

```

ln -s /export/server/hbase-2.1.0 /export/server/hbase

```

8. 【node1、node2、node3执行】，配置环境变量

```

# 配置在/etc/profile内，追加如下两行
export HBASE_HOME=/export/server/hbase
export PATH=$HBASE_HOME/bin:$PATH

source /etc/profile

```

9. 【node1执行】启动HBase

请确保：Hadoop HDFS、Zookeeper是已经启动了的


```
start-hbase.sh
```

```
# 如需停止可使用  
stop-hbase.sh
```

由于我们配置了环境变量`export PATH=$PATH:$HBASE_HOME/bin`
`start-hbase.sh`即在`$HBASE_HOME/bin`内，所以可以无论当前目录在哪，均可直接执行

10. 验证HBase

浏览器打开：<http://node1:16010>，即可看到HBase的WEB UI页面

11. 简单测试使用HBase

【node1执行】

```
hbase shell  
  
# 创建表  
create 'test', 'cf'  
  
# 插入数据  
put 'test', 'rk001', 'cf:info', 'itheima'  
  
# 查询数据  
get 'test', 'rk001'  
  
# 扫描表数据  
scan 'test'
```

分布式内存计算Spark环境部署

注意

本小节的操作，基于：[大数据集群（Hadoop生态）安装部署](#) 环节中所构建的Hadoop集群

如果没有Hadoop集群，请参阅前置内容，部署好环境。

简介

Spark是一款分布式内存计算引擎，可以支撑海量数据的分布式计算。

Spark在大数据体系是明星产品，作为最新一代的综合计算引擎，支持离线计算和实时计算。

在大数据领域广泛应用，是目前世界上使用最多的大数据分布式计算引擎。

我们将基于前面构建的Hadoop集群，部署Spark Standalone集群。

安装

1. 【node1执行】 下载并解压

```
wget https://archive.apache.org/dist/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz

# 解压
tar -zxvf spark-2.4.5-bin-hadoop2.7.tgz -C /export/server/

# 软链接
ln -s /export/server/spark-2.4.5-bin-hadoop2.7 /export/server/spark
```

2. 【node1执行】 修改配置文件名称

```
# 改名
cd /export/server/spark/conf
mv spark-env.sh.template spark-env.sh
mv slaves.template slaves
```

3. 【node1执行】 修改配置文件， `spark-env.sh`

```
## 设置JAVA安装目录
JAVA_HOME=/export/server/jdk

## HADOOP软件配置文件目录，读取HDFS上文件和运行YARN集群
HADOOP_CONF_DIR=/export/server/hadoop/etc/hadoop
YARN_CONF_DIR=/export/server/hadoop/etc/hadoop

## 指定spark老大Master的IP和提交任务的通信端口
export SPARK_MASTER_HOST=node1
export SPARK_MASTER_PORT=7077

SPARK_MASTER_WEBUI_PORT=8080
SPARK_WORKER_CORES=1
SPARK_WORKER_MEMORY=1g
```

4. 【node1执行】 修改配置文件， `slaves`

```
node1
node2
node3
```

5. 【node1执行】 分发

```
scp -r spark-2.4.5-bin-hadoop2.7 node2:$PWD
scp -r spark-2.4.5-bin-hadoop2.7 node3:$PWD
```

6. 【node2、node3执行】 设置软链接

```
ln -s /export/server/spark-2.4.5-bin-hadoop2.7 /export/server/spark
```

7. 【node1执行】 启动Spark集群

```
/export/server/spark/sbin/start-all.sh

# 如需停止，可以
/export/server/spark/sbin/stop-all.sh
```

8. 打开Spark监控页面，浏览器打开：<http://node1:8081>

9. 【node1执行】 提交测试任务(算圆周率)

```
/export/server/spark/bin/spark-submit --master spark://node1:7077 --class
org.apache.spark.examples.SparkPi
/export/server/spark/examples/jars/spark-examples_2.11-2.4.5.jar
```

分布式内存计算Flink环境部署

注意

本小节的操作，基于：[大数据集群（Hadoop生态）安装部署](#) 环节中所构建的Hadoop集群

如果没有Hadoop集群，请参阅前置内容，部署好环境。

简介

Flink同Spark一样，是一款分布式内存计算引擎，可以支撑海量数据的分布式计算。

Flink在大数据体系同样是明星产品，作为最新一代的综合计算引擎，支持离线计算和实时计算。

在大数据领域广泛应用，是目前世界上除去Spark以外，应用最为广泛的分布式计算引擎。

我们将基于前面构建的Hadoop集群，部署Flink Standalone集群

Spark更加偏向于离线计算而Flink更加偏向于实时计算。

安装

1. 【node1操作】 下载安装包

```
wget https://archive.apache.org/dist/flink/flink-1.10.0/flink-1.10.0-bin-scala_2.11.tgz

# 解压
tar -zxvf flink-1.10.0-bin-scala_2.11.tgz -C /export/server/

# 软链接
ln -s /export/server/flink-1.10.0 /export/server/flink
```

2. 【node1操作】 修改配置文件， `conf/flink-conf.yaml`

```
# jobManager 的IP地址
jobmanager.rpc.address: node1
# JobManager 的端口号
jobmanager.rpc.port: 6123
# JobManager JVM heap 内存大小
jobmanager.heap.size: 1024m
# TaskManager JVM heap 内存大小
taskmanager.heap.size: 1024m
# 每个 TaskManager 提供的任务 slots 数量大小
taskmanager.numberOfTaskSlots: 2
#是否进行预分配内存，默认不进行预分配，这样在我们不使用flink集群时候不会占用集群资源
taskmanager.memory.preallocate: false
# 程序默认并行计算的个数
parallelism.default: 1
#JobManager的web界面的端口（默认：8081）
jobmanager.web.port: 8081
```

3. 【node1操作】， 修改配置文件， `conf/slaves`

```
node1
node2
node3
```

4. 【node1操作】 分发Flink安装包到其它机器

```
cd /export/server
scp -r flink-1.10.0 node2:\pwd\
scp -r flink-1.10.0 node3:\pwd\
```

5. 【node2、node3操作】

```
# 配置软链接
ln -s /export/server/flink-1.10.0 /export/server/flink
```

6. 【node1操作】，启动Flink

```
/export/server/flink/bin/start-cluster.sh
```

7. 验证Flink启动

```
# 浏览器打开
http://node1:8081
```

8. 提交测试任务

【node1执行】

```
/export/server/flink/bin/flink run /export/server/flink-1.10.0/examples/batch/wordCount.jar
```

