系统工程师实战

<wangpeng20199@gmail.com>

目录

前言

排坑日记

===

按时间先后

Chapter 1. Zabbix

基于 CentOS 7

1.1. zabbix

1.1.1. 安装

增加 zabbix [source,console]

```
yum[source,console]
```

```
源
```

```
rpm -Uvh https://repo.zabbix.com/zabbix/4.4/rhel/7/x86_64/zabbix-release-4.4-1.el7.noarch.rpm
yum -y install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-agent
```

查看安装的软件包

```
[root@redas_core ~]# rpm -aq |grep zabbix
zabbix-web-mysql-4.4.4-1.el7.noarch
zabbix-get-4.4.4-1.el7.x86_64
zabbix-server-mysql-4.4.4-1.el7.x86_64
zabbix-agent-4.4.4-1.el7.x86_64
zabbix-release-4.4-1.el7.noarch
zabbix-web-4.4.4-1.el7.noarch
```

增加开机启动

```
systemctl enable zabbix-server[source,console]
```

1.1.2. 配置 mysql数据库

创建数据库

```
CREATE DATABASE zabbix DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_unicode_ci;[source,console]
```

查看数据库

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| redas              |
| sys                |
| zabbix             |
+--------------------+
```

新建用户

```
mysql -uroot -pgeek -e "CREATE USER zabbix@localhost identified with
mysql_native_password by 'geek';"[source,console]

mysql -uroot -pgeek -e "GRANT ALL ON zabbix.* TO 'zabbix'@'localhost' WITH
GRANT OPTION;"[source,console]
```

查看用户

```
[root@redas_core ~]# mysql -uzabbix -pgeek zabbix -e "show tables"
+----------------------------+
| Tables_in_zabbix           |
+----------------------------+
| acknowledges               |
| actions                    |
| alerts                     |
...
```

导入数据

```
gzip -d /usr/share/doc/zabbix-server-mysql-
4.4.4/create.sql.gz[source,console]

mysql -uzabbix -pgeek zabbix < /usr/share/doc/zabbix-server-mysql-
4.4.4/create.sql[source,console]
```

查看导入数据

```
mysql> show tables;
+----------------------------+
| Tables_in_zabbix           |
+----------------------------+
| acknowledges               |
| actions                    |
| alerts                     |
...
```

1.1.3. 配置服务器

编辑 /etc/zabbix/zabbix_server.conf 文件

```
sed -i '124s/# DBPassword=/DBPassword=geek/'
/etc/zabbix/zabbix_server.conf[source,console]
```

添加时区

```
sed -i '20s@        # php_value date.timezone Europe/Riga@php_value
date.timezone Asia/Shanghai@'
/etc/httpd/conf.d/zabbix.conf[source,console]
```

1.1.4. 添加客户端

增加 zabbix yum 源

```
rpm -Uvh https://repo.zabbix.com/zabbix/4.4/rhel/7/x86_64/zabbix-release-
4.4-1.el7.noarch.rpm[source,console]

yum -y install  zabbix-agent[source,console]
```

增加开机启动

```
systemctl enable zabbix-agent[source,console]
```

1.1.5. 配置客户端

设置主机ip和客户端名称

```
sed -i '98s/Server=/Server=192.168.2.8/' /etc/zabbix/zabbix_agentd.conf

sed -i '139s/ServerActive=/ServerActive=192.168.2.8/'
/etc/zabbix/zabbix_agentd.conf

sed -i '150s/Hostname=/Hostname=yinxin/'
/etc/zabbix/zabbix_agentd.conf[source,console]
```

1.1.6. 启动服务

服务端

```
systemctl start zabbix-server[source,console]
```

客户端

```
systemctl start zabbix-server[source,console]
```

查看服务状态

```
systemctl status zabbix-server[source,console]
```

浏览器访问

```
http://192.168.2.8/zabbix/setup.php[source,console]
```

解决中文乱码
将windows /Windows/fonts/simkai.ttf 拷贝到服务器 /usr/share/zabbix/assets/fonts目录下
将/usr/share/zabbix/include/defines.inc.php文件下的graphfont替换为 simkai

企业微信

```
企业ID
ww496d47cc707b7239
AgentId
1000003
Secret
ebxky-GfkJKXl_oOeKQ2zdw_P2lKLUYoRZ5g6FM0l24


{ITEM.NAME}故障
IP:{HOST.HOST}
故障信息:{TRIGGER.NAME}
```

```
监控取值:{ITEM.LASTVALUE}
告警时间:{EVENT.DATE} {EVENT.TIME}


主机:{HOST.NAME}故障
地址:{HOSTNAME1}
功能模块:{ITEM.NAME}
监控取值:{ITEM.LASTVALUE}
告警信息:{TRIGGER.NAME}
告警时间:{EVENT.DATE} {EVENT.TIME}


#!/bin/bash
CorpID="ww496d47cc707b7239"                    #企业下面的CorpID
Secret="ebxky-GfkJKXl_oOeKQ2zdw_P2lKLUYoRZ5g6FM0l24"
#创建的应用那有Secret
GURL="https://qyapi.weixin.qq.com/cgi-
bin/gettoken?corpid=$CorpID&corpsecret=$Secret"
Token=$(/usr/bin/curl -s -G $GURL |awk -F\": '{print $4}'|awk -F\" '{print
$2}')
#echo $Token
PURL="https://qyapi.weixin.qq.com/cgi-
bin/message/send?access_token=$Token"

function body(){
        local int agentid=1000003   #改为AgentId 在创建的应用那里看
        local UserID=$1             #发送的用户位于$1的字符串
        local PartyID=2            #第一步看的通讯录中的部门ID
        local Msg=$(echo "$@" | cut -d" " -f3-)
        printf '{\n'
        printf '\t"touser": "'"$UserID"\"",\n"
        printf '\t"toparty": "'"$PartyID"\"",\n"
        printf '\t"msgtype": "text",\n'
        printf '\t"agentid": "'"$agentid"\"",\n"
        printf '\t"text": {\n'
        printf '\t\t"content": "'"$Msg"\""\n"
        printf '\t},\n'
        printf '\t"safe":"0"\n'
        printf '}\n'
}
/usr/bin/curl --data-ascii "$(body $1 $2 $3)" $PURL
企业ID
ww496d47cc707b7239
AgentId
1000003
Secret
ebxky-GfkJKXl_oOeKQ2zdw_P2lKLUYoRZ5g6FM0l24
```

```
{ITEM.NAME}故障
IP:{HOST.HOST}
故障信息:{TRIGGER.NAME}
监控取值:{ITEM.LASTVALUE}
告警时间:{EVENT.DATE} {EVENT.TIME}


主机:{HOST.NAME}故障
地址:{HOSTNAME1}
功能模块:{ITEM.NAME}
监控取值:{ITEM.LASTVALUE}
告警信息:{TRIGGER.NAME}
告警时间:{EVENT.DATE} {EVENT.TIME}


#!/bin/bash
CorpID="ww496d47cc707b7239"                    #企业下面的CorpID
Secret="ebxky-GfkJKXl_oOeKQ2zdw_P2lKLUYoRZ5g6FM0l24"
#创建的应用那有Secret
GURL="https://qyapi.weixin.qq.com/cgi-
bin/gettoken?corpid=$CorpID&corpsecret=$Secret"
Token=$(/usr/bin/curl -s -G $GURL |awk -F\": '{print $4}'|awk -F\" '{print
$2}')
#echo $Token
PURL="https://qyapi.weixin.qq.com/cgi-
bin/message/send?access_token=$Token"

function body(){
        local int agentid=1000003   #改为AgentId 在创建的应用那里看
        local UserID=$1             #发送的用户位于$1的字符串
        local PartyID=2            #第一步看的通讯录中的部门ID
        local Msg=$(echo "$@" | cut -d" " -f3-)
        printf '{\n'
        printf '\t"touser": "'"$UserID"\"",\n"
        printf '\t"toparty": "'"$PartyID"\"",\n"
        printf '\t"msgtype": "text",\n'
        printf '\t"agentid": "'"$agentid"\"",\n"
        printf '\t"text": {\n'
        printf '\t\t"content": "'"$Msg"\""\n"
        printf '\t},\n'
        printf '\t"safe":"0"\n'
        printf '}\n'
}
/usr/bin/curl --data-ascii "$(body $1 $2 $3)" $PURL
```

用阿里云搭建zabbix用25端口发邮件超时问题
刚才并不清楚原因，首先服务器Telnet smtp.exmail.qq.com 25端口，发现25端口不通，肯定会发生连接超时。
查询原因是阿里云管控垃圾邮件，封了25端口服务，可以使用ssl方式的465端口进行邮件的发送。

Chapter 2. Gitlab

基于 CentOS 7

2.1. Gitlab

2.1.1. 下载安装

官方下载RPM包

```
wget https://packages.gitlab.com/gitlab/gitlab-ce/packages/el/7/gitlab-ce-12.8.7-ce.0.el7.x86_64.rpm
```

yum 安装

```
yum localinstall gitlab-ce-12.8.7-ce.0.el7.x86_64.rpm
```

2.1.2. 初始化

nginx 端口冲突，关闭本地 nginx 服务

```
systemctl stop nginx

systemctl disable nginx
```

初始化 gitlab

```
gitlab-ctl reconfigure
```

启动

```
gitlab-ctl start
```

停止

```
gitlab-ctl stop
```

2.1.3. 修改配置

git 克隆地址改为本机IP地址

```
sed -i '13s/    host: gitlab.example.com/    host: 192.168.2.7/'
/opt/gitlab/embedded/service/gitlab-rails/config/gitlab.yml
```

客户端访问

```
192.168.2.7
```

初始用户名：root
密码自定义

2.1.4. 用户设置

设置 [source,console]

```
Access Tokens
```

```
用户 --> settings --> Access Tokens

填写名称:

勾选: api
      read_user

确定: Create personal access token

记录access token
PhY2RK4spkepP9g_3yf9
```

添加公钥

```
用户 --> settings --> SSH Keys

cd ~/.ssh/ 查看自己的公钥添加到gitlab,通常是以.pub结尾的文件
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDKhTFrjT3qhHAKhQDaU1ZKkIYANiByqFYXA+3L+thfgh
kwt3+hrCNZ17LU+/vIl0lIn4pbgjtkYuI7wVVUW9Z7PeNRVMSy6Ql70RD6ZF9BdfCBnPoNs3Ad
3S2IVQ3WkqIrXKfKDcypLibSECR8+7yQagnxAOXl2qjEeywv06M3Z2TUdUJ9VgSagZOvTvQB3h
4SEuJ3RxinhkRNiuWwUcx3Q1t2Bf2VTvBkCfZHTADXc0+uKVwHGuTHZzP+Y1hK9Pqpb3pv4qUc
q4Pu0dcLS8HElOe2xo8DXu7yXvDGOJrDbNQytk/+bzc3wmIGlkr8SKvuEVZ+nD32D2byCyNnP9
1d root@vir19

最后 Add key

服务器添加公钥之后在本机添加公钥钥匙
cat << EOF > ~/.ssh/config
Host 192.168.2.7
    IdentityFile ~/.ssh/mk
EOF
```

创建项目

```
Create new project --> 填写项目名称 --> Create project --> clone
```

创建 [source,console]

```
webhook[source,console]
```

```
Projects --> redas_lib --> settings --> Integrations
 (Jenkins --> 项目 --> Build when a change is pushed to GitLab. GitLab
webhook URL:)
url: http://192.168.2.19:8080/project/redas_lib
勾选: Push events
      Merge request events
确认添加
```

允许本地网络请求

```
Admin Area --> settings --> Network --> Outbound requests --> Expand
勾选: Allow requests to the local network from web hooks and services
```

Chapter 3. Jenkins

基于 CentOS 7

3.1. Jenkins

3.1.1. 下载安装

官方下载RPM包

```
wget https://pkg.jenkins.io/redhat-stable/jenkins-2.204.5-1.1.noarch.rpm
```

yum 安装

```
yum localinstall jenkins-2.204.5-1.1.noarch.rpm
```

3.1.2. 启动

检查 java 版本

```
java -version
```

设置 java版本为1.8

```
sudo update-alternatives --config java
```

java 1.8安装

```
yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel java-1.8.0-
openjdk-headless[source,console]
```

正常启动

```
java -jar /usr/lib/jenkins/jenkins.war
```

代理方式启动

```
java -DsocksProxyHost=192.168.2.5 -DsocksProxyPort=1080 -jar
jenkins.war[source,console]
```

先用代理启动，Jenkins初始化需要安装许多外网插件，插件安装完后去掉代理，正常启动

3.1.3. 初始化

安装插件

```
gitlab
gitlab hook
```

配置 SSH 密钥

```
凭据 --> 系统 --> 全局凭据 -->添加凭据

kind: SSH Username with private key
description:jenkins_ssh_key
Enter directly（勾选）
Private Key:(~/.ssh/目录查看自己的私钥，添加)

保存
```

配置 API token

```
凭据 --> 系统 --> 全局凭据 -->添加凭据

kind: Gitlab api token
API token: (gitlab服务器创建的API token)
描述: gitlab_api_token

确定
```

修改配置

```
系统管理 -> 系统配置 ->
Enable authentication for '/project' end-point  (web hook 403
权限问题去掉打勾)
Gitlab
    Connection name :gitlab7

    Gitlab host URL :http://192.168.2.7 (gitlab服务器地址)
    Credentials : API Token for accessing Gitlab （选配置好的API token）

    Test Connection(点击测试，返回Success为正确)
    Success
```

Chapter 4. Mysql

基于 CentOS 7

4.1. web 备份迁移

4.1.1. mysql备份还原

备份

```
mysqldump -uroot -p'geek' --add-drop-table dedecms_foo_com >
/data/db/dedecms_foo_com.sql
```

备份的每条数据增加 insert 语句

```
mysqldump -uroot -p'geek' --extended-insert=FALSE
dedecms_foo_com>dedecms_foo_com.sql
```

还原

```
mysql -uroot -p'geek' dedecms_foo_com < /data/db/dedecms_foo_com.sql
```

打包压缩

```
tar -zcvf /data/db/dedecms_foo_com.sql.tar.gz -C
/data/db/dedecms_foo_com.sql
```

备份到服务器

```
rsync -a --log-file=/tmp/rsync.log /data/db/
root@192.168.2.8:/data/backup/db/
```

4.1.2. web文件备份

备份到服务器

```
rsync -a --log-file=/tmp/rsync.log /data/web/
root@192.168.2.8:/data/backup/web/
```

4.1.3. 命令参数解释

mysqldump

```
```

```
--add-drop-database 在每个 CREATE DATABASE 语句之前添加 DROP DATABASE 语句
--add-drop-table    在每个 CREATE TABLE 语句之前添加 DROP TABLE 语句
--add-drop-trigger  在每个 CREATE TRIGGER 语句之前添加 DROP TRIGGER 语句
--add-locks 用 LOCK TABLES 和 UNLOCK TABLES 语句包围每个 table 转储
--all-databases 转储所有数据库中的所有 table
--allow-keywords    允许创建作为关键字的列名
--apply-slave-statements    在 CHANGE MASTER 语句之前包含 STOP
SLAVE，在输出结束时包含 START SLAVE
--bind-address  使用指定的网络接口连接到 MySQL Server
--character-sets-dir    字符集的安装目录
--comments  添加 Comments 到转储文件
--compact   产生更紧凑的输出
--compatible    产生与其他数据库系统或更旧的 MySQL 服务器更兼容的输出
--complete-insert   使用包含列名称的完整 INSERT 语句
--compress  压缩 Client 端和服务器之间发送的所有信息
--create-options    在 CREATE TABLE 语句中包括所有特定于 MySQL 的 table 选项
--databases 将所有名称参数解释为数据库名称
--debug 编写调试日志
--debug-check   程序退出时打印调试信息
--debug-info    程序退出时打印调试信息，内存和 CPU 统计信息
--default-auth  身份验证插件使用
--default-character-set 指定默认字符集
--defaults-extra-file   除常规选项文件外，还读取命名的选项文件
--defaults-file 只读命名的选项文件
--defaults-group-suffix 选项组后缀值
--delete-master-logs    在主复制服务器上，执行转储操作后删除二进制日志
--disable-keys  对于每个 table，在 INSERT 语句周围加上用于禁用和启用键的语句
--dump-date 如果给出--comments，则将转储日期包括为“转储完成于”Comments
--dump-slave    包含 CHANGE MASTER 语句，该语句列出了从属主机的二进制日志坐标
--enable-cleartext-plugin   启用明文身份验证插件  5.7.10
--events    从转储的数据库中转储事件
--extended-insert   使用多行 INSERT 语法
--fields-enclosed-by    该选项与--tab 选项一起使用，其含义与 LOAD DATA
的相应子句相同
--fields-escaped-by 该选项与--tab 选项一起使用，其含义与 LOAD DATA 的相应子句相同
--fields-optionally-enclosed-by 该选项与--tab 选项一起使用，其含义与 LOAD DATA
的相应子句相同
--fields-terminated-by  该选项与--tab 选项一起使用，其含义与 LOAD DATA
的相应子句相同
--flush-logs    开始转储之前刷新 MySQL 服务器日志文件
--flush-privileges  转储 mysql 数据库后发出 FLUSH PRIVILEGES 语句
--force 即使在 table 转储期间发生 SQL 错误，也要 continue
--get-server-public-key 从服务器请求 RSA 公钥   5.7.23
--help  显示帮助信息并退出
--hex-blob  使用十六进制 table 示法转储二进制列
--host  MySQL 服务器所在的主机
--ignore-error  忽略指定的错误
--ignore-table  不要转储给定的 table
--include-master-host-port  在由--dump-slave 生成的 CHANGE MASTER 语句中包括
```

```
MASTER_HOST/MASTER_PORT 选项
--insert-ignore 编写 INSERT IGNORE 而不是 INSERT 语句
--lines-terminated-by   该选项与--tab 选项一起使用，其含义与 LOAD DATA
的相应子句相同
--lock-all-tables   锁定所有数据库中的所有 table
--lock-tables   转储之前锁定所有 table
--log-error 将警告和错误附加到命名文件
--login-path    从.mylogin.cnf 中读取登录路径选项
--master-data   将二进制日志文件的名称和位置写入输出
--max-allowed-packet    发送到服务器或从服务器接收的最大数据包长度
--net-buffer-length TCP/IP 和套接字通信的缓冲区大小
--no-autocommit 将每个转储 table 的 INSERT 语句包含在 SET autocommit = 0 和
COMMIT 语句内
--no-create-db  不要写 CREATE DATABASE 语句
--no-create-info    不要编写重新创建每个转储 table 的 CREATE TABLE 语句
--no-data   不要转储 table 内容
--no-defaults   不读取选项文件
--no-set-names  与--skip-set-charset 相同
--no-tablespaces    不要在输出中写入任何 CREATE LOGFILE GROUP 或 CREATE
TABLESPACE 语句
--opt   --add-drop-table --add-locks --create-options --disable-keys
--extended-insert --lock-tables --quick --set-charset 的简写
--order-by-primary  转储按主键或第一个唯一索引排序的每个 table 的行
--password  连接服务器时使用的密码
--pipe  使用命名管道连接到服务器(仅 Windows)
--plugin-dir    安装插件的目录
--port  用于连接的 TCP/IP 端口号
--print-defaults    打印默认选项
--protocol  使用的传输协议
--quick 一次从服务器检索 table 的行
--quote-names   反引号字符内的引号标识符
--replace   编写 REPLACE 语句而不是 INSERT 语句
--result-file   直接输出到给定文件
--routines  从转储的数据库中转储存储的例程(过程和函数)
--secure-auth   不要以旧(4.1 之前)格式向服务器发送密码  Yes
--server-public-key-path    包含 RSA 公钥的文件的路径名    5.7.23
--set-charset   将 SET NAMES default_character_set 添加到输出
--set-gtid-purged   是否将 SET @@ GLOBAL.GTID_PURGED 添加到输出
--shared-memory-base-name   共享内存连接的共享内存名称(仅 Windows)
--single-transaction    从服务器转储数据之前发出 BEGIN SQL 语句
--skip-add-drop-table   不要在每个 CREATE TABLE 语句之前添加 DROP TABLE 语句
--skip-add-locks    不添加锁
--skip-comments 不要添加 Comments 到转储文件
--skip-compact  不要产生更紧凑的输出
--skip-disable-keys 不要禁用按键
--skip-extended-insert  关闭扩展插入
--skip-opt  关闭--opt 设置的选项
--skip-quick    不要一次从服务器检索 table 的行
--skip-quote-names  不引用标识符
```

```
--skip-set-charset  不要写 SET NAMES 语句
--skip-triggers 不要转储触发器
--skip-tz-utc   关闭 tz-utc
--socket    Unix 套接字文件或 Windows 命名管道使用
--ssl   启用连接加密
--ssl-ca    包含受信任的 SSL 证书颁发机构列 table 的文件
--ssl-capath    包含受信任的 SSL 证书颁发机构证书文件的目录
--ssl-cert  包含 X.509 证书的文件
--ssl-cipher    连接加密的允许密码
--ssl-crl   包含证书吊销列 table 的文件
--ssl-crlpath   包含证书吊销列 table 文件的目录
--ssl-key   包含 X.509 密钥的文件
--ssl-mode  与服务器连接的所需安全状态   5.7.11
--ssl-verify-server-cert    根据服务器证书的通用名身份验证主机名
--tab   产生制 table 符分隔的数据文件
--tables    覆盖-数据库或-B 选项
--tls-version   允许的 TLS 协议进行加密连接    5.7.10
--triggers  每个转储 table 的转储触发器
--tz-utc    添加 SET TIME_ZONE ='00:00'来转储文件
--user  连接服务器时使用的 MySQL 用户名
--verbose   Verbose mode
--version   显示版本信息并退出
--where 仅转储给定 WHERE 条件选择的行
--xml   产生 XML 输出
```

mysql

示例:

```
mysqldump -uroot -pgeek --add-drop-table db_name > db_name.sql
```

参数解释:
-h：mysql服务器的ip地址或主机名
-u：连接mysql服务器的用户名
-e：执行mysql内部命令
-p：连接mysql服务器的密码
-D：使用哪个数据库

4.1.4. tar

示例:
打包

```
tar -zcvf test.tar.gz -C /root/document/test
```

列出文档

```
tar -tf test.tar
```

提取文档

```
tar -xf test.tar
```

参数解释：
-c：创建归档
-f：指定归档文件
-v：显示指令执行过程
-z：通过gzip指令压缩/解压缩文件，文件名最好为*.tar.gz
-C：改变工作目录
-t：列出归档文件的内容
-x：从归档文件中提取文件

4.1.5. crontab

```
crontab
```

命令被用来提交和管理用户的需要周期性执行的任务

示例：

```
[root@cky ~]# crontab  -l
*/1 * * * * /bin/sh /usr/bin/vnstat_dump.sh
#1 2 * * * /bin/sh /bak/local_bak/script/tar_mysql_bin.sh
1 3 * * * /bin/sh /bak/local_bak/script/rsync_from_local.sh
1 4 * * * /bin/sh /bak/local_bak/script/full_db_bak.sh

#*/1 * * * * /bin/sh /bak/script/kill_query_timeout.sh
#*/10 * * * * /etc/init.d/php-fpm_www  restart

1 */1 * * * /bin/sh /opt/git_pull_new_ck.sh
59 23 * * 0,6 /bin/sh /bak/script/renew.sh
```

参数解释：
-e：编辑该用户的计时器设置
-l：列出该用户的计时器设置

crontab文件的含义：用户所建立的crontab文件中，每一行都代表一项任务，每行的每个字段代表一项设置，它的格式共分为六个字段，前五段是时间设定段，第六段是要执行的命令段，格式如下：

minute hour day month week command 顺序：分 时 日 月 周 命令段

minute： 表示分钟，可以是从0到59之间的任何整数

hour：表示小时，可以是从0到23之间的任何整数

day：表示日期，可以是从1到31之间的任何整数

month：表示月份，可以是从1到12之间的任何整数

week：表示星期几，可以是从0到7之间的任何整数，这里的0或7代表星期日

command：要执行的命令，可以是系统命令，也可以是自己编写的脚本文件

在以上各个字段中，还可以使用以下特殊字符：

*：代表所有可能的值，例如month字段如果是星号，则表示在满足其它字段的制约条件后每月都执行该命令操作

, ：可以用逗号隔开的值指定一个列表范围，例如， “1,2,5,7,8,9”

-：可以用整数之间的中杠表示一个整数范围，例如“2-6”表示“2,3,4,5,6”

/ ：可以用正斜线指定时间的间隔频率，例如“0-23/2”表示每两小时执行一次。同时正斜线可以和星号一起使用，例如*/10，如果用在minute字段，表示每十分钟执行一次

4.2. Mysql 主从

mysql Ver 8.0.19 for Linux on x86_64 (MySQL Community Server - GPL)

master 地址：192.168.43.3 slave 地址：192.168.43.114

4.2.1. GTID主从配置

配置master

```
vim /etc/my.cnf
    [mysqld]
    default-authentication-plugin=mysql_native_password
    datadir=/var/lib/mysql
    socket=/var/lib/mysql/mysql.sock
    log-error=/var/log/mysqld.log
    pid-file=/var/run/mysqld/mysqld.pid
    user=mysql
    port=3306
    server-id=1
    gtid-mode=ON
    enforce-gtid-consistency=ON
    binlog_format=row
```

重启 mysql 服务，关闭防火墙

systemctl restart mysqld systemctl stop firewalld systemctl mask firewalld.service

配置slave

```
vim /etc/my.cnf
[mysqld]
    default-authentication-plugin=mysql_native_password
    datadir=/var/lib/mysql
    socket=/var/lib/mysql/mysql.sock
    log-error=/var/log/mysqld.log
    pid-file=/var/run/mysqld/mysqld.pid
    user=mysql
    port=3306
    server-id=2
    gtid-mode=ON
    enforce-gtid-consistency=ON
    binlog_format=row
```

重启 mysql 服务，关闭防火墙

```
systemctl restart mysqld
systemctl  stop  firewalld
systemctl mask firewalld.service
```

master创建用户授权

```
mysql -uroot -pgeek
set global validate_password_policy=0;
set global validate_password_length=1;
flush privileges;
CREATE USER 'slave123'@'%' IDENTIFIED BY 'geek';
GRANT REPLICATION SLAVE ON *.* TO 'slave123'@'%';
flush privileges;
```

查看master状态

```
mysql> show master status;
```

slave同步设置

mysql -uroot -pgeek

```
mysql -uroot -pgeek
CREATE USER 'slave123'@'%' IDENTIFIED BY 'geek';
GRANT REPLICATION SLAVE ON *.* TO 'slave123'@'%';
flush privileges;
stop slave;
change master to master_host='192.168.2.17',
master_user='slave123',master_password='geek',master_port=3306,master_auto
_position=1;
start slave;

#查看slave状态
show slave status\G
```

mysql8卸载安全策略

```
UNINSTALL COMPONENT 'file://component_validate_password';
```

验证主从

```
master:
    create database test1;
    show master status;

slave:
    show databases;
    show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.43.3
                  Master_User: slave123
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: binlog.000035
          Read_Master_Log_Pos: 1687
               Relay_Log_File: dbslave-relay-bin.000005
                Relay_Log_Pos: 1895
        Relay_Master_Log_File: binlog.000035
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
                  Master_UUID: 76475db1-5150-11ea-8853-0800275e2671
             Master_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for
more updates
           Master_Retry_Count: 86400
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set: 76475db1-5150-11ea-8853-0800275e2671:88-99
            Executed_Gtid_Set: 76475db1-5150-11ea-8853-0800275e2671:1-99,
93a238d7-5150-11ea-9fb7-080027da57dd:1-66
```

解决Last_IO_Error：1236报错

```
RESET MASTER;
set global gtid_purged = 'xxxx';                    -- 这里xxxx是Master
的Executed_Gtid_Set
start slave;
show slave status\G;
```

4.3. Mycat

4.3.1. 安装 mycat

install

```
wget -c wget -c http://dl.mycat.io/1.6.7.3/Mycat-server-1.6.7.3-release-
20190828135747-linux.tar.gz
tar xf Mycat-server-1.6.7.3-release-20190828135747-linux.tar.gz
mv mycat /usr/local/
rm -rf mycat
```

安装 java 1.8

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

修改配置文件/usr/local/mycat/conf/schema.xml

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

    <schema name="db1" checkSQLschema="true" sqlMaxLimit="1000"
dataNode="data_node1" />

    <dataNode name="data_node1" dataHost="db_host1" database="db1"/>

    <dataHost name="db_host1" maxCon="1000" minCon="10" balance="1"
              writeType="0" dbType="mysql" dbDriver="native" switchType="-
1" slaveThreshold="100">

        <heartbeat>select user()</heartbeat>

        <writeHost host="host_write" url="192.168.43.3:3306"
user="slave123" password="geek">
            <readHost host="host_read" url="192.168.43.114:3306"
user="slave123" password="geek"/>
        </writeHost>

    </dataHost>
</mycat:schema>
```

修改配置 /usr/local/mycat/conf/server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mycat:server SYSTEM "server.dtd">
<mycat:server xmlns:mycat="http://io.mycat/">

        <system>
                <property name="bindIp">0.0.0.0</property>
        </system>

        <firewall>
           <whitehost>
              <host host="127.0.0.1" user="slave123"/>
              <host host="192.168.43.*" user="slave123"/>
           </whitehost>
           <blacklist check="false">
           </blacklist>
       </firewall>

        <user name="slave123" defaultAccount="true">
                <property name="password">geek</property>
                <property name="schemas">db1</property>
        </user>
</mycat:server>
```

必须将主机名加入HOSTS文件

```
echo "127.0.0.1 dbmaster" >> /etc/hosts

.mycat 启动
[source,txt]
```

/usr/local/mycat/bin/mycat start

重启

```
/usr/local/mycat/bin/mycat restart
```

停止

```
/usr/local/mycat/bin/mycat stop
```

查看状态

```
/usr/local/mycat/bin/mycat status
```

连接数据库

```
mysql -h127.0.0.1 -P9066 -uslave123 -pgeek  db1
```

查看读写分离

```
show @@datasource;
```

检测心跳线

```
show @@heartbeat;
```

Chapter 5. Centos

5.1. Centos7初始化

```
#查看主机名
hostnamectl status

#修改主机名
hostnamectl set-hostname 主机名

#禁用SELINUX，必须重启才能生效
echo SELINUX=disabled>/etc/selinux/config
echo SELINUXTYPE=targeted>>/etc/selinux/config

#临时关闭SElinux，不需要重启
setenforce 0

#如果你想使用自己的 iptables 静态防火墙规则，那么请安装 iptables-services 并且禁用
firewalld ，启用 iptables
systemctl  stop  firewalld
systemctl mask firewalld.service


systemctl mask NetworkManager

#最大可以打开的文件
echo "*               soft   nofile            65535" >>
/etc/security/limits.conf
echo "*               hard   nofile            65535" >>
/etc/security/limits.conf

# ssh登录时，登录ip被会反向解析为域名，导致ssh登录缓慢
sed -i "s/#UseDNS yes/UseDNS no/" /etc/ssh/sshd_config
sed -i "s/GSSAPIAuthentication yes/GSSAPIAuthentication no/"
/etc/ssh/sshd_config
sed -i "s/GSSAPICleanupCredentials yes/GSSAPICleanupCredentials no/"
/etc/ssh/sshd_config
sed -i "s/#MaxAuthTries 6/MaxAuthTries 10/" /etc/ssh/sshd_config
# server每隔30秒发送一次请求给client，然后client响应，从而保持连接
sed -i "s/#ClientAliveInterval 0/ClientAliveInterval 30/"
/etc/ssh/sshd_config
# server发出请求后，客户端没有响应得次数达到3，就自动断开连接，正常情况下，
client不会不响应
sed -i "s/#ClientAliveCountMax 3/ClientAliveCountMax 10/"
/etc/ssh/sshd_config

#支持gbk文件显示
```

```
echo "set fencs=utf-8,gbk" >> /etc/vimrc

#设定系统时区
yes|cp /usr/share/zoneinfo/Asia/Chongqing /etc/localtime

#如果是x86_64系统，排除32位包
echo "exclude=*.i386 *.i586 *.i686" >> /etc/yum.conf
```

查看当前开了哪些端口

```
firewall-cmd --list-services
```

查看还有哪些服务可以打开

```
firewall-cmd --get-services
```

添加一个服务到firewalld

```
firewall-cmd --add-service=http
```

5.2. CentOS7 Install LNMP

5.2.1. 安装 Nginx

增加 Nginx 官方源

```
cat << EOF > /etc/yum.repos.d/nginx.repo
[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
EOF
```

EPEL 源中的 ----nginx.service ---- 由于 ----KILL ---- 参数问题，启动后无法停止，不建议使用。

安装Nginx

```
yum install -y nginx
```

备份Nginx配置文件

```
echo y|cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.default
```

修改 nginx.conf

```
cat << EOF > /etc/nginx/nginx.conf
# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

worker_rlimit_nofile 65535;

events {
    worker_connections 65535;
}

http {
    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    log_format  main  '\$host \$server_port \$remote_addr - \$remote_user
[\$time_local] "\$request" '
                      '\$status \$request_time \$body_bytes_sent
"\$http_referer" '
                      '"\$http_user_agent" "\$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;
```

```
    server_names_hash_bucket_size 128;
    server_name_in_redirect off;
    client_header_buffer_size 32k;
    large_client_header_buffers 4 32k;

    client_header_timeout  3m;
    client_body_timeout    3m;
    client_max_body_size 50m;
    client_body_buffer_size 256k;
    send_timeout           3m;

    gzip  on;
    gzip_min_length  1k;
    gzip_buffers     4 16k;
    gzip_http_version 1.0;
    gzip_comp_level 2;
    gzip_types text/plain application/x-javascript text/css
application/xml;
    gzip_vary on;

    proxy_redirect off;
    proxy_set_header Host \$host;
    proxy_set_header X-Real-IP \$remote_addr;
    proxy_set_header REMOTE-HOST \$remote_addr;
    proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
    proxy_connect_timeout 60;
    proxy_send_timeout 60;
    proxy_read_timeout 60;
    proxy_buffer_size 256k;
    proxy_buffers 4 256k;
    proxy_busy_buffers_size 256k;
    proxy_temp_file_write_size 256k;
    proxy_next_upstream error timeout invalid_header http_500 http_503
http_404;
    proxy_max_temp_file_size 128m;
    #让代理服务端不要主动关闭客户端的连接，协助处理499返回代码问题
    proxy_ignore_client_abort on;

    fastcgi_buffer_size 64k;
    fastcgi_buffers 4 64k;
    fastcgi_busy_buffers_size 128k;

    index index.html index.htm index.php default.html default.htm
default.php;

    # Load modular configuration files from the /etc/nginx/conf.d
directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
```

```
    include /etc/nginx/conf.d/*.conf;
}
EOF
```

增加默认Host

```
mkdir /etc/nginx/conf.d

cat << EOF > /etc/nginx/conf.d/default.conf
server {
    listen       80 default_server;
    listen       [::]:80 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
EOF
```

检测配置

```
nginx -t && rm -f /var/run/nginx.pid
```

nginx -t /var/run/nginx.pid 空文件会一直被保留，而nginx.service并不能处理 PIDFile为空的情况，导致启动失败。需要手动删除 /var/run/nginx.pid

from nginx/1.16.1

启动Nginx

```
systemctl start nginx
```

查看Nginx状态

```
# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor
preset: disabled)
   Active: active (running) since Fri 2019-11-29 14:02:31 CST; 1h 18min
ago
 Main PID: 15759 (nginx)
   CGroup: /system.slice/nginx.service
           ├─15759 nginx: master process /usr/sbin/nginx
           └─17285 nginx: worker process

Nov 29 14:02:31 iZ6weebcmroarpx8rrxscrZ systemd[1]: Starting The nginx
HTTP and reverse proxy server...
Nov 29 14:02:31 iZ6weebcmroarpx8rrxscrZ nginx[15753]: nginx: the
configuration file /etc/nginx/nginx.conf syntax is ok
Nov 29 14:02:31 iZ6weebcmroarpx8rrxscrZ nginx[15753]: nginx: configuration
file /etc/nginx/nginx.conf test is successful
Nov 29 14:02:31 iZ6weebcmroarpx8rrxscrZ systemd[1]: Failed to parse PID
from file /run/nginx.pid: Invalid argument
Nov 29 14:02:31 iZ6weebcmroarpx8rrxscrZ systemd[1]: Started The nginx HTTP
and reverse proxy server.


# ss -antpl|grep nginx
LISTEN     0      128          *:80                       *:*
users:(("nginx",pid=17285,fd=6),("nginx",pid=15759,fd=6))
LISTEN     0      128         :::80                      :::*
users:(("nginx",pid=17285,fd=7),("nginx",pid=15759,fd=7))
```

增加开机启动

```
systemctl enable nginx
```

5.2.2. 安装 MySQL

安装 MySQL

```
yum install -y mariadb-server
```

备份 my.cnf

```
cp /etc/my.cnf /etc/my.cnf.default
```

修改

my.cnf

```
cat << EOF > /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security
risks
symbolic-links=0
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd

max_allowed_packet=20M
max_heap_table_size = 100M
read_buffer_size = 2M
read_rnd_buffer_size = 16M
sort_buffer_size = 8M
join_buffer_size = 8M
tmp_table_size = 100M

# 查询缓存
#query_cache_limit=4M
#query_cache_type=on
#query_cache_size=2G

bind-address = 127.0.0.1
# 跳过主机名解析，比如localhost，foo.com之类，加速访问
skip-name-resolve

# SQL执行日志
general_log=off
general_log_file=/var/log/mariadb/general.log

# SQL慢查询日志
slow_query_log=off
slow_query_log_file=/var/log/mariadb/slowquery.log
long_query_time = 5

max_connections = 1000

# 兼容老MySQL代码，比如使用空字符串代替NULL插入数据
sql_mode = ""

[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
pid-file=/var/run/mariadb/mariadb.pid
```

```
#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
EOF
```

配置 mysqldump

命令参数

```
sed -i '16 aquick\nquote-names\nmax_allowed_packet = 100M'
/etc/my.cnf.d/mysql-clients.cnf
```

创建日志文件

```
touch /var/log/mariadb/general.log /var/log/mariadb/slowquery.log
chown mysql:mysql /var/log/mariadb/general.log
/var/log/mariadb/slowquery.log
```

增加开机启动

```
systemctl enable mariadb
```

启动 MySQL 服务

```
systemctl start mariadb
```

查看 MySQL 服务状态

```
# systemctl status mariadb
● mariadb.service - MariaDB database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled;
vendor preset: disabled)
   Active: active (running) since Fri 2019-11-29 14:18:12 CST; 1h 7min ago
  Process: 16688 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID
(code=exited, status=0/SUCCESS)
  Process: 16653 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir %n
(code=exited, status=0/SUCCESS)
 Main PID: 16687 (mysqld_safe)
   CGroup: /system.slice/mariadb.service
           ├─16687 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
           └─17043 /usr/libexec/mysqld --basedir=/usr
--datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/plugin --log
-error=/var/log/mariadb/mariadb.lo...

Nov 29 14:18:10 iZ6weebcmroarpx8rrxscrZ systemd[1]: Starting MariaDB
database server...
Nov 29 14:18:10 iZ6weebcmroarpx8rrxscrZ mariadb-prepare-db-dir[16653]:
Database MariaDB is probably initialized in /var/lib/mysql already,
nothing is done.
Nov 29 14:18:11 iZ6weebcmroarpx8rrxscrZ mysqld_safe[16687]: 191129
14:18:11 mysqld_safe Logging to '/var/log/mariadb/mariadb.log'.
Nov 29 14:18:11 iZ6weebcmroarpx8rrxscrZ mysqld_safe[16687]: 191129
14:18:11 mysqld_safe Starting mysqld daemon with databases from
/var/lib/mysql
Nov 29 14:18:12 iZ6weebcmroarpx8rrxscrZ systemd[1]: Started MariaDB
database server.

# ss -antpl|grep mysql
LISTEN     0      50     127.0.0.1:3306                     *:*
users:(("mysqld",pid=17043,fd=14))
```

修改root密码

```
mysqladmin -uroot password "geek"
```

删除测试数据库和空密码用户

```
mysql -uroot -pgeek -e 'show databases;'
mysql -uroot -pgeek -e 'drop database test;'
mysql -uroot -pgeek mysql -e 'delete from db;'
mysql -uroot -pgeek mysql -e 'delete from user where Password="";'
mysql -uroot -pgeek -e 'flush privileges;'
```

5.2.3. 安装 PHP7

增加SCL源

```
yum install -y centos-release-scl
```

安装PHP7.2

```
yum install -y rh-php72 \
    rh-php72-php  \
    rh-php72-php-bcmath \
    rh-php72-php-fpm \
    rh-php72-php-gd \
    rh-php72-php-intl \
    rh-php72-php-mbstring \
    rh-php72-php-mysqlnd \
    rh-php72-php-opcache \
    rh-php72-php-pdo \
    rh-php72-php-pecl-apcu \
    rh-php72-php-xmlrpc \
    rh-php72-php-devel
```

进入 rh-php72 环境

```
scl enable rh-php72 bash
```

确认PHP状态

```
# php -v
PHP 7.2.24 (cli) (built: Nov  4 2019 10:23:08) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.2.24, Copyright (c) 1999-2018, by Zend
Technologies
```

备份php.ini

```
cp /etc/opt/rh/rh-php72/php.ini /etc/opt/rh/rh-php72/php.ini.default
```

修改php.ini

```
# 启用 '<? ... ?>' 代码风格
sed -i '197s/short_open_tag = Off/short_open_tag = On/' /etc/opt/rh/rh-
php72/php.ini

# 禁止一些危险性高的函数
sed -i '314s/disable_functions =/disable_functions =
system,exec,shell_exec,passthru,set_time_limit,ini_alter,dl,openlog,syslog
,readlink,symlink,link,leak,popen,escapeshellcmd,virtual,socket_create,mai
l,eval/' /etc/opt/rh/rh-php72/php.ini

# 配置中国时区
sed -i '902s#;date.timezone =#date.timezone = Asia/Shanghai#'
/etc/opt/rh/rh-php72/php.ini
```

增加开机启动

```
systemctl enable rh-php72-php-fpm
```

启动 PHP-FPM 服务

```
systemctl start rh-php72-php-fpm
```

查看 PHP-FPM 服务状态

```
# systemctl status rh-php72-php-fpm
● rh-php72-php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/rh-php72-php-fpm.service;
enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-11-29 13:36:03 CST; 1h 56min
ago
 Main PID: 15360 (php-fpm)
   Status: "Processes active: 0, idle: 6, Requests: 56, slow: 0, Traffic:
0req/sec"
   CGroup: /system.slice/rh-php72-php-fpm.service
           ├─15360 php-fpm: master process (/etc/opt/rh/rh-php72/php-
fpm.conf)
           ├─15361 php-fpm: pool www
           ├─15362 php-fpm: pool www
           ├─15363 php-fpm: pool www
           ├─15364 php-fpm: pool www
           ├─15365 php-fpm: pool www
           └─17211 php-fpm: pool www

Nov 29 13:36:03 iZ6weebcmroarpx8rrxscrZ systemd[1]: Starting The PHP
FastCGI Process Manager...
Nov 29 13:36:03 iZ6weebcmroarpx8rrxscrZ systemd[1]: Started The PHP
FastCGI Process Manager.

# ss -antpl|grep php-fpm
LISTEN     0      128    127.0.0.1:9000                     *:*
users:(("php-fpm",pid=17211,fd=9),("php-fpm",pid=15365,fd=9),("php-
fpm",pid=15364,fd=9),("php-fpm",pid=15363,fd=9),("php-
fpm",pid=15362,fd=9),("php-fpm",pid=15361,fd=9),("php-
fpm",pid=15360,fd=7))
```

5.2.4. LNMP 环境测试

增加数据库

```
mysql -uroot -pgeek -e 'create database drupal;grant all privileges on
drupal.*  to drupal@"localhost" identified by "drupal_password";flush
privileges;'
```

增加Nginx Host设置

```
cat << EOF > /etc/nginx/conf.d/drupal.foo.com.conf
server {
    listen       80;

    server_name  drupal.foo.com;
    root         /data/web/drupal.foo.com;
    error_log /var/log/nginx/drupal.foo.com_error.log;
    access_log /var/log/nginx/drupal.foo.com_access.log  main;

    location / {
        try_files \$uri /index.php\$is_args\$query_string;
    }

    location ~ \.php\$ {
        try_files \$uri \$uri/ 404;

        fastcgi_pass 127.0.0.1:9000;
        include fastcgi.conf;
    }
}
EOF

# 重载Nginx配置
nginx -t && nginx -s reload
```

准备 Drupal

```
mkdir -p /data/web/drupal.foo.com

# 使用 -O 参数指定保存文件名，会强制覆盖已经存在的文件
wget https://ftp.drupal.org/files/projects/drupal-8.7.10.tar.gz -O drupal-
8.7.10.tar.gz
tar xf drupal-8.7.10.tar.gz

mv drupal-8.7.10/* /data/web/drupal.foo.com
rm -rf drupal-8.7.10
chown -R apache:nginx /data/web/drupal.foo.com
chmod -R 755 /data/web/drupal.foo.com
```

drupal-8.7.10/core/INSTALL.txt

```
Drupal requires:
```

- A web server with PHP support, for example:
- Apache 2.0 (or greater) (http://httpd.apache.org/).
- Nginx 1.1 (or greater) (http://nginx.com/).
- PHP 5.5.9 (or greater) (http://php.net/). For better security support it is recommended to update to at least 5.5.21 or 5.6.5.
- One of the following databases:
- MySQL 5.5.3 (or greater) (http://www.mysql.com/).
- MariaDB 5.5.20 (or greater) (https://mariadb.org/). MariaDB is a fully compatible drop-in replacement for MySQL.
- Percona Server 5.5.8 (or greater) (http://www.percona.com/). Percona Server is a backwards-compatible replacement for MySQL.
- PostgreSQL 9.1.2 (or greater) (http://www.postgresql.org/).
- SQLite 3.7.11 (or greater) (http://www.sqlite.org/).

设置本地解析

```
echo '47.74.60.161 drupal.foo.com' >> /etc/hosts
```

/etc/hosts (Linux)

C:\Windows\System32\drivers\etc\hosts （Windows）

最后，访问 http://drupal.foo.com 完成安装。

Drupal数据库相关信息：

- 数据库服务器：localhost
- 数据库端口：3306
- 数据库名称：drupal
- 数据库用户名：drupal
- 数据库密码：drupal_password

5.3. Centos8LNMP

5.3.1. 安装Nginx

增加EPEL源

```
dnf install -y epel-release
```

备份Nginx配置文件

```
echo y|cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.defaultv
```

修改 nginx.conf

```
cat << EOF > /etc/nginx/nginx.conf
    # For more information on configuration, see:
    #   * Official English Documentation: http://nginx.org/en/docs/
    #   * Official Russian Documentation: http://nginx.org/ru/docs/

    user nginx;
    worker_processes auto;
    error_log /var/log/nginx/error.log;
    pid /run/nginx.pid;

    # Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
    include /usr/share/nginx/modules/*.conf;

    worker_rlimit_nofile 65535;

    events {
        worker_connections 65535;
    }

    http {
        include             /etc/nginx/mime.types;
        default_type        application/octet-stream;

        log_format  main  '\$host \$server_port \$remote_addr -
\$remote_user [\$time_local] "\$request" '
                          '\$status \$request_time \$body_bytes_sent
"\$http_referer" '
                          '"\$http_user_agent" "\$http_x_forwarded_for"';

        access_log  /var/log/nginx/access.log  main;

        sendfile            on;
        tcp_nopush          on;
```

```
        tcp_nodelay         on;
        keepalive_timeout   65;
        types_hash_max_size 2048;

        server_names_hash_bucket_size 128;
        server_name_in_redirect off;
        client_header_buffer_size 32k;
        large_client_header_buffers 4 32k;

        client_header_timeout  3m;
        client_body_timeout    3m;
        client_max_body_size 50m;
        client_body_buffer_size 256k;
        send_timeout           3m;

        gzip  on;
        gzip_min_length  1k;
        gzip_buffers     4 16k;
        gzip_http_version 1.0;
        gzip_comp_level 2;
        gzip_types text/plain application/x-javascript text/css
application/xml;
        gzip_vary on;

        proxy_redirect off;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header REMOTE-HOST \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_connect_timeout 60;
        proxy_send_timeout 60;
        proxy_read_timeout 60;
        proxy_buffer_size 256k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
        proxy_temp_file_write_size 256k;
        proxy_next_upstream error timeout invalid_header http_500 http_503
http_404;
        proxy_max_temp_file_size 128m;
        #让代理服务端不要主动关闭客户端的连接，协助处理499返回代码问题
        proxy_ignore_client_abort on;

        fastcgi_buffer_size 64k;
        fastcgi_buffers 4 64k;
        fastcgi_busy_buffers_size 128k;

        index index.html index.htm index.php default.html default.htm
default.php;
```

```
        # Load modular configuration files from the /etc/nginx/conf.d
directory.
        # See http://nginx.org/en/docs/ngx_core_module.html#include
        # for more information.
        include /etc/nginx/conf.d/*.conf;
    }
EOF
```

增加默认Host

```
mkdir /etc/nginx/conf.d

cat << EOF > /etc/nginx/conf.d/default.conf
    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
            location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }
EOF
```

检查配置

```
nginx -t
```

启动Nginx

```
systemctl start nginx
```

查看Nginx状态

```
systemctl status nginx
```

增加开机启动

```
systemctl enable nginx
```

查看版本号

```
nginx -v
```

卸载nginx

```
dnf remove nginx
```

5.3.2. 安装 MySQL

安装 MySQL

```
yum install -y mariadb-server
```

备份 my.cnf

```
cp /etc/my.cnf /etc/my.cnf.default
```

修改 my.cnf

```
cat << EOF > /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security
risks
symbolic-links=0
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd

max_allowed_packet=20M
max_heap_table_size = 100M
read_buffer_size = 2M
read_rnd_buffer_size = 16M
sort_buffer_size = 8M
join_buffer_size = 8M
tmp_table_size = 100M

# 查询缓存
```

```
#query_cache_limit=4M
#query_cache_type=on
#query_cache_size=2G

bind-address = 127.0.0.1
# 跳过主机名解析，比如localhost，foo.com之类，加速访问
skip-name-resolve

# SQL执行日志
general_log=off
general_log_file=/var/log/mariadb/general.log

# SQL慢查询日志
slow_query_log=off
slow_query_log_file=/var/log/mariadb/slowquery.log
long_query_time = 5

max_connections = 1000

# 兼容老MySQL代码，比如使用空字符串代替NULL插入数据
sql_mode = ""

[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
pid-file=/var/run/mariadb/mariadb.pid

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
EOF
```

增加开机启动

```
systemctl enable mariadb
```

启动MySQL

```
systemctl start mariadb
```

查看 MySQL 服务状态

```
systemctl status mariadb
```

修改root密码

```
mysqladmin -uroot password "geek"
```

删除测试数据库和空密码用户

```
mysql -uroot -pgeek -e 'show databases;'
mysql -uroot -pgeek -e 'drop database test;'
mysql -uroot -pgeek mysql -e 'delete from db;'
mysql -uroot -pgeek mysql -e 'delete from user where Password="";'
mysql -uroot -pgeek -e 'flush privileges;'
```

5.3.3. 安装PHP7

增加SCL源

```
yum install -y centos-release-scl
```

安装PHP7.2

```
dnf install -y \
        php  \
        php-bcmath \
        php-fpm \
        php-gd \
        php-intl \
        php-mbstring \
        php-mysqlnd \
        php-opcache \
        php-pdo \
        php-pecl-apcu \
        php-xmlrpc \
        php-devel
```

安装mlocate

```
dnf install -y mlocate
```

更新

```
updatedb
```

查找配置文件路径

```
locate php.ini
```

修改php.ini

```
sed -i '197s/short_open_tag = Off/short_open_tag = On/' /etc/opt/rh/rh-
php72/php.ini

# 禁止一些危险性高的函数
sed -i '314s/disable_functions =/disable_functions =
system,exec,shell_exec,passthru,set_time_limit,ini_alter,dl,openlog,syslog
,readlink,symlink,link,leak,popen,escapeshellcmd,virtual,socket_create,mai
l,eval/' /etc/opt/rh/rh-php72/php.ini

# 配置中国时区
sed -i '902s#;date.timezone =#date.timezone = Asia/Shanghai#'
/etc/opt/rh/rh-php72/php.ini
```

增加开机启动

```
systemctl enable php-fpm
```

启动 PHP-FPM 服务

```
systemctl start php-fpm
```

查看 PHP-FPM 服务状态

```
systemctl status php-fpm
```

5.3.4. LNMP环境测试

增加数据库

```
mysql -uroot -pgeek -e 'create database drupal;grant all privileges on
drupal.* to drupal@"localhost" identified by "drupal_password";flush
privileges;'
```

增加Nginx Host设置

```
cat << EOF > /etc/nginx/conf.d/drupal.foo.com.conf
    server {
        listen       80;

        server_name  drupal.foo.com;
        root         /data/web/drupal.foo.com;
        error_log /var/log/nginx/drupal.foo.com_error.log;
        access_log /var/log/nginx/drupal.foo.com_access.log  main;

        location / {
            try_files \$uri /index.php\$is_args$query_string;
        }

        include /etc/nginx/default.d/php.conf;
    }
EOF
```

重载Nginx配置

```
nginx -s reload
```

准备 Drupal

```
mkdir -p /data/web/drupal.foo.com
```

使用 -O 参数指定保存文件名，会强制覆盖已经存在的文件

```
wget https://ftp.drupal.org/files/projects/drupal-8.7.10.tar.gz -O drupal-
8.7.10.tar.gz
tar xf drupal-8.7.10.tar.gz
mv drupal-8.7.10/* /data/web/drupal.foo.com
rm -rf drupal-8.7.10
chown -R apache:nginx /data/web/drupal.foo.com
chmod -R 755 /data/web/drupal.foo.com
```

设置本地解析

```
echo '47.74.60.161 drupal.foo.com' >> /etc/hosts
```

Chapter 6. XXL-JOB

6.1. 安装配置

6.1.1. 安装依赖

Git

```
yum install -y git
```

Maven && Java 8

Maven 包默认依赖 Java 8。以下 Java 将被安装：

- java-1.8.0-openjdk
- java-1.8.0-openjdk-devel
- java-1.8.0-openjdk-headless

安装 Maven

```
yum install -y maven
```

查看 Maven 版本

```
[root@geekcamp ~]# mvn --version
Apache Maven 3.0.5 (Red Hat 3.0.5-17)
Maven home: /usr/share/maven
Java version: 1.8.0_232, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.232.b09-
0.el7_7.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-1062.el7.x86_64", arch: "amd64",
family: "unix"
```

查看 Java 版本

```
[root@geekcamp ~]# java -version
openjdk version "1.8.0_232"
OpenJDK Runtime Environment (build 1.8.0_232-b09)
OpenJDK 64-Bit Server VM (build 25.232-b09, mixed mode)
```

设置 Maven 全局镜像源

```
mkdir ~/.m2

cat << EOF > ~/.m2/settings.xml
<settings>
  <mirrors>
    <mirror>
      <id>aliyun</id>
      <name>Aliyun Central</name>
        <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
</settings>
EOF
```

Java 11（可选）

安装 Java 11：

```
yum install -y java-11-openjdk java-11-openjdk-devel java-11-openjdk-
headless
```

查看当前默认 Java 版本：

```
[root@geekcamp ~]# java -version
openjdk version "1.8.0_232"
OpenJDK Runtime Environment (build 1.8.0_232-b09)
OpenJDK 64-Bit Server VM (build 25.232-b09, mixed mode)

[root@geekcamp ~]# alternatives --display java
java - status is auto.
 link currently points to /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.232.b09-
0.el7_7.x86_64/jre/bin/java
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.232.b09-0.el7_7.x86_64/jre/bin/java
- family java-1.8.0-openjdk.x86_64 priority 1800232
 slave jre: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.232.b09-
0.el7_7.x86_64/jre
.......
Current

best version is /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.232.b09-
0.el7_7.x86_64/jre/bin/java.
```

设置Java11为默认版本：

```
alternatives --set java
ls /usr/lib/jvm/java-11-openjdk-*/bin/java


alternatives --set jre_openjdk
ls -d /usr/lib/jvm/java-11-openjdk-*


alternatives --set java_sdk_openjdk
ls -d /usr/lib/jvm/java-11-openjdk-*
```

查看当前默认 Java 版本：

```
[root@geekcamp ~]# java -version
openjdk version "11.0.6" 2020-01-14 LTS
OpenJDK Runtime Environment 18.9 (build 11.0.6+10-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.6+10-LTS, mixed mode, sharing)
```

MySQL

安装 MySQL：

```
yum install -y mariadb-server mariadb
```

设置开机启动：

```
systemctl enable mariadb
```

启动 MySQL 服务：

```
systemctl start mariadb
```

设置密码：

```
mysqladmin -uroot password "geek"
```

默认 Mysql 初始化并启动后，可以使用空密码登录。

测试 Mysql：

```
[root@geekcamp ~]# mysql -uroot -pgeek -e "show databases;"
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

6.1.2. XXL-JOB 安装

构建：

```
[ ! -d "~/downloads/" ] && mkdir ~/downloads/
cd ~/downloads/
git clone https://github.com/xuxueli/xxl-job.git
cd xxl-job
mvn -T 4 package
```

安装：

```
[ ! -d "/data/xxl-job/admin" ] && mkdir -p /data/xxl-job/admin
[ ! -d "/data/xxl-job/executor" ] && mkdir -p /data/xxl-job/executor
[ ! -d "/data/applogs" ] && mkdir -p /data/applogs

echo y | cp xxl-job-admin/target/xxl-job-admin-2.2.0-SNAPSHOT.jar
/data/xxl-job/admin
echo y | cp xxl-job-admin/src/main/resources/application.properties
/data/xxl-job/admin

echo y | cp xxl-job-executor-samples/xxl-job-executor-sample-
springboot/target/xxl-job-executor-sample-springboot-2.2.0-SNAPSHOT.jar
/data/xxl-job/executor
echo y | cp xxl-job-executor-samples/xxl-job-executor-sample-
springboot/src/main/resources/application.properties /data/xxl-
job/executor
```

确认安装：

```
[root@geekcamp xxl-job]# find /data/xxl-job/*
/data/xxl-job/admin
/data/xxl-job/admin/xxl-job-admin-2.2.0-SNAPSHOT.jar
/data/xxl-job/admin/application.properties
/data/xxl-job/executor
/data/xxl-job/executor/xxl-job-executor-sample-springboot-2.2.0-
SNAPSHOT.jar
/data/xxl-job/executor/application.properties
```

6.1.3. XXL-JOB 运行前

XXL-JOB 服务端

MySQL配置

创建 xxl_job 数据库：

```
mysql -uroot -pgeek -e "CREATE DATABASE xxl_job DEFAULT CHARACTER SET
utf8mb4 DEFAULT COLLATE utf8mb4_unicode_ci;"
```

创建 xxl_job 用户：

```
mysql -uroot -pgeek -e "CREATE USER 'xxl_job'@'localhost' IDENTIFIED BY
'geek';"
mysql -uroot -pgeek -e "GRANT ALL ON xxl_job.* TO 'xxl_job'@'localhost'
WITH GRANT OPTION;"
```

导入 SQL 文件：

```
mysql -uxxl_job -pgeek xxl_job < doc/db/tables_xxl_job.sql
```

查看 xxl_job 表：

```
[root@geekcamp xxl-job]# mysql -uxxl_job -pgeek xxl_job -e "show tables;"
+--------------------+
| Tables_in_xxl_job  |
+--------------------+
| xxl_job_group      |
| xxl_job_info       |
| xxl_job_lock       |
| xxl_job_log        |
| xxl_job_log_report |
| xxl_job_logglue    |
| xxl_job_registry   |
| xxl_job_user       |
+--------------------+
```

设置配置文件

```
sed -i
'26s/spring.datasource.username=root/spring.datasource.username=xxl_job/'
/data/xxl-job/admin/application.properties

sed -i
'27s/spring.datasource.password=root_pwd/spring.datasource.password=geek/'
/data/xxl-job/admin/application.properties
```

创建系统用户

```
useradd --user-group --no-create-home --shell /sbin/nologin --comment
"XXL-JOB" xxl_job
chown -R xxl_job:xxl_job /data/applogs /data/xxl-job
```

增加 systemd 文件

```
cat << EOF > /usr/lib/systemd/system/xxl_job_admin.service
[Unit]
Description=XXL-JOB Admin Service
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=simple
Environment="PORT=8080"
Environment="CONFIG_FILE=/data/xxl-job/admin/application.properties"
Environment="JAR_FILE=/data/xxl-job/admin/xxl-job-admin-2.2.0-
SNAPSHOT.jar"

User=xxl_job
Group=xxl_job
WorkingDirectory=/data/xxl-job/admin
ExecStart=/usr/bin/java -Dserver.port=\${PORT}
-Dspring.config.location=\${CONFIG_FILE} -jar \${JAR_FILE}
SuccessExitStatus=143
StandardOutput=null
StandardError=journal

[Install]
WantedBy=multi-user.target
EOF
```

XXL-JOB 客户端

创建目录

```
[ ! -d "/data/xxl-job/executor" ] && mkdir -p /data/xxl-job/executor
[ ! -d "/data/applogs" ] && mkdir -p /data/applogs
```

设置配置文件

```
sed -i '9s#xxl.job.admin.addresses=http://127.0.0.1:8080/xxl-job-
admin#xxl.job.admin.addresses=http://192.168.2.11:8080/xxl-job-admin#' \
/data/xxl-job/executor/application.properties

appname=xxl-job-executor-s01

sed -i "12s/xxl.job.executor.appname=xxl-job-executor-
sample/xxl.job.executor.appname=${appname}/" \
/data/xxl-job/executor/application.properties
```

按需更改参数

xxl.job.admin.addresses=http://127.0.0.1:8080/xxl-job-admin

创建系统用户

```
useradd --user-group --no-create-home --shell /sbin/nologin --comment
"XXL-JOB" xxl_job
chown -R xxl_job:xxl_job /data/applogs /data/xxl-job/executor
```

增加 systemd 文件

```
cat << EOF > /usr/lib/systemd/system/xxl_job_executor.service
[Unit]
Description=XXL-JOB Executor Service
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=simple
Environment="PORT=8081"
Environment="CONFIG_FILE=/data/xxl-job/executor/application.properties"
Environment="JAR_FILE=/data/xxl-job/executor/xxl-job-executor-sample-
springboot-2.2.0-SNAPSHOT.jar"

User=xxl_job
Group=xxl_job
WorkingDirectory=/data/xxl-job/executor
ExecStart=/usr/bin/java -Dserver.port=\${PORT}
-Dspring.config.location=\${CONFIG_FILE} -jar \${JAR_FILE}
SuccessExitStatus=143
StandardOutput=null
StandardError=journal

[Install]
WantedBy=multi-user.target
EOF
```

6.1.4. 启动 XXL-JOB 服务端

```
systemctl enable xxl_job_admin
systemctl start xxl_job_admin
```

确认运行情况：

```
[root@geekcamp xxl-job]# systemctl status xxl_job_admin
● xxl_job.service - XXL-JOB Service
   Loaded: loaded (/usr/lib/systemd/system/xxl_job.service; enabled;
vendor preset: disabled)
   Active: active (running) since Thu 2020-01-16 13:56:05 CST; 1s ago
 Main PID: 3015 (java)
   CGroup: /system.slice/xxl_job.service
           └─3015 /usr/bin/java -Dserver.port=8080
-Dspring.config.location=/data/xxl-job/application.properties -jar
/data/xxl-job/xxl-job-admin-2.2.0-SNA...

Jan 16 13:56:05 geekcamp systemd[1]: Started XXL-JOB Service.

[root@geekcamp xxl-job]# ss -antpl|grep 8080
LISTEN     0      100       [::]:8080                  [::]:*
users:(("java",pid=2288,fd=21))
```

最后，浏览器访问 http://localhost:8080/xxl-job-admin

XXL-JOB默认帐号：admin
XXL-JOB默认密码：123456

6.1.5. 启动 XXL-JOB 客户端

```
systemctl enable xxl_job_executor
systemctl start xxl_job_executor
```

确认运行情况：

```
[root@localhost admin]# systemctl status xxl_job_executor
● xxl_job_client.service - XXL-JOB Client
   Loaded: loaded (/usr/lib/systemd/system/xxl_job_client.service;
enabled; vendor preset: disabled)
   Active: active (running) since Thu 2020-01-16 18:49:43 CST; 4s ago
 Main PID: 2876 (java)
   CGroup: /system.slice/xxl_job_client.service
           └─2876 /usr/bin/java -Dserver.port=8081
-Dspring.config.location=/data/xxl-job/executor/application.properties
-jar /data/xxl-job/executor/xxl-j...

Jan 16 18:49:43 localhost.localdomain systemd[1]: Started XXL-JOB Client.

[root@localhost admin]# ss -antpl|grep 8081
LISTEN     0      100       [::]:8081                  [::]:*
users:(("java",pid=4679,fd=11))
```

6.1.6. 工作状态确认

访问 http://192.168.2.11:8080/xxl-job-admin/jobgroup 并登录，确认结果与下图一致：

Figure 1. XXL-JOB 执行器界面

Chapter 7. Kvm

基于 CentOS 7

7.1. 安装配置

7.1.1. 安装kvm环境

```
yum install -y virt-manager libvirt virt-install qemu-kvm bridge-utils
```

```
# lsmod | grep kvm
kvm_intel       138567  0
kvm             441119  1 kvm_intel
```

```
systemctl start libvirtd

systemctl enable libvirtd

echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
sysctl -p
```

7.1.2. 设置网桥

设置内网网桥和 DHCP

```
编辑/etc/libvirt/qemu/networks/default.xml，修改IP段为 192.168.122.0/24
，stp='on'
```

```
<network>
  <name>default</name>
  <uuid>9719a6f8-7e16-4e7b-a592-909337ddaf73</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:dd:9e:e0'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

编辑

```
/var/lib/libvirt/dnsmasq/default.conf
```

```
dhcp-range=192.168.122.2,192.168.122.254
```

重启libvirtd

```
systemctl restart libvirtd
brctl stp virbr0 on
```

查看默认网桥virbr0

```
# ip a
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
state DOWN
    link/ether 52:54:00:dd:9e:e0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master
virbr0 state DOWN qlen 500
    link/ether 52:54:00:dd:9e:e0 brd ff:ff:ff:ff:ff:ff

# brctl show
bridge name bridge id       STP enabled interfaces
virbr0      8000.525400dd9ee0   off     virbr0-nic
```

7.1.3. 配置公网网桥，使虚拟机可以绑定公网IP

nmcli依赖于NetworkManager服务

```
systemctl unmask NetworkManager
systemctl start NetworkManager
systemctl status NetworkManager
```

添加物理网卡

```
nmcli c add type Ethernet autoconnect yes con-name eth2 ifname eth2
nmcli c add type bridge autoconnect yes con-name br1 ifname br1
nmcli c modify br1 bridge.stp no
nmcli c modify br1 ipv6.method ignore
nmcli c modify br1 ipv4.addresses 182.131.21.23/28 ipv4.method manual
nmcli c modify br1 ipv4.gateway 182.131.21.1
nmcli c modify br1 ipv4.dns "223.5.5.5 223.6.6.6"
```

删除物理网卡em1，将网卡加入网桥。em1的IP地址也自动加入到了网桥。 以下3条命令必须同时执行，最好放到脚本里面执行。因为，delete网卡后，可能会断网。重启网络服务或启动网桥后才能恢复

将em1换成自己的网卡

```
cat << EOF > /tmp/set_br1.sh
nmcli c delete em1
nmcli c add type bridge-slave autoconnect yes con-name em1 ifname em1
master br1
nmcli c up br1
EOF

chmod 755 /tmp/set_br1.sh
sh /tmp/set_br1.sh &
rm -f /tmp/set_br1.sh
```

重启网络

```
systemctl restart NetworkManager

systemctl status NetworkManager

reboot
```

CentOS 7 上意外关闭了 NetworkManager 服务，重启后导致无法访问网络。注意，保持NetworkManager开机运行。

7.2. KVM虚拟机

7.2.1. 挂载硬盘

查看硬盘

```
fdisk -l
```

硬盘分区

```
fdisk /dev/sdb
```

格式化

```
mkfs.xfs -n ftype=1 -f /dev/sdb1
```

查看uuid

```
blkid
```

添加到 /etc/fstab

```
UUID=0836d695-2bb4-48db-a440-faccc6131761 /mnt                       xfs
defaults        0 0
```

开机启动

```
mount -a
```

7.2.2. 虚拟机安装

增加虚拟机硬盘

```
qemu-img create -f qcow2 /var/lib/libvirt/images/vm192168122253.img 100G
```

安装 vir

```
virt-install \
--name centos_kvm \
--ram 4096 \
--disk path=/var/lib/libvirt/images/vm192168122253.img,size=100 \
--vcpus 2 \
--os-type linux \
--os-variant centos7.0 \
--network bridge=br1 \
--console pty,target_type=serial \
--cdrom=/data/CentOS-7-x86_64-Minimal-1908.iso \
--graphics vnc,password=geek,port=-1,listen=0.0.0.0
```

已有虚拟机镜像安装

```
virt-install \
--name gitlab \
--ram 2048 \
--disk path=/data/package/gitlab_centos7.img \
--vcpus 4 \
--os-type linux \
--os-variant centos7.0 \
--network bridge=br1 \
--console pty,target_type=serial \
--import \
--graphics vnc,password=geek,port=5901,listen=0.0.0.0
```

参数解释

```
  --name centos_kvm
```

虚拟机名称

```
--ram 4096
```

虚拟机内存

```
--disk path=/var/lib/libvirt/images/vm192168122253.img,size=100
```

硬盘位置大小

```
--vcpus 2
```

cpu数量

```
--os-type linux
```

系统类型

```
--os-variant centos7.0
```

选择对应系统

```
--network bridge=br1
```

网桥名称

```
--console pty,target_type=serial
```

默认参数

```
--cdrom=/data/CentOS-7-x86_64-Minimal-1908.iso
```

ios位置

```
--graphics vnc,password=geek,port=-1,listen=0.0.0.0
```

vnc服务，密码，端口，-1不指定端口

查系统类型

```
osinfo-query os|grep -i centos
```

开启虚拟机

```
virsh start vir-name
```

查看虚拟机

```
virsh list --all
```

删除虚拟机

```
virsh undefine  centos_kvm
```

设置开机启动

```
virsh autostart vir_name
```

关闭开机启动

```
virsh autostart --disable vir_name
```

Chapter 8. Service

基于 CentOS 7

8.1. systemd

8.1.1. .service 文件

在 /usr/lib/systemd/system 目录下创建服务配置 backdump.service 文件

编辑 backdump.service 文件

```
[Unit]
Description=mysql dump

[Service]
Type=oneshot
ExecStart=/usr/lib/locale/mysqldump.sh
IOSchedulingClass=idle

[Install]
WantedBy=multi-user.target
```

参数解释

```
[Unit] 通常是配置文件的第一个区块，用来定义 Unit 的元数据

Description 关于服务的简短描述

[Service] service 的配置区块，编写配置

Type 定义进程启动时的行为，oneshot 一次性进程，参数如下

Type=simple：默认值，执行ExecStart指定的命令，启动主进程
Type=forking：以 fork 方式从父进程创建子进程，创建后父进程会立即退出
Type=oneshot：一次性进程，Systemd 会等当前服务退出，再继续往下执行
Type=dbus：当前服务通过D-Bus启动
Type=notify：当前服务启动完毕，会通知Systemd，再继续往下执行
Type=idle：若有其他任务执行完毕，当前服务才会运行

ExecStart 启动当前服务的命令，加命令的绝对路径

[Install] 配置文件的最后一个区块，用来定义如何启动，以及是否开机启动

WantedBy 表示该服务所在的 Target，WantedBy=multi-user.target 指的是，sshd 所在的
Target 是multi-user.target
```

8.1.2. .timer 定时器文件

在 /usr/lib/systemd/system 目录下创建服务配置 .timer 文件

```
mkdir /usr/lib/systemd/system/backdump.timer
```

编辑 backdump.timer 文件

```
[Unit]
Description=mysql dump

[Timer]
OnCalendar=hourly

[Install]
WantedBy=timers.target
```

参数解释

```
[Unit] 通常是配置文件的第一个区块，用来定义 Unit 的元数据

Description  关于服务的简短描述

[Service] service 的配置区块，编写配置

OnCalendar 定义基于挂钟时间
(wallclock)的日历定时器，值是一个日历事件表达式，指定时间触发
```

时间

```
    minutely → *-*-* *:*:00
      hourly → *-*-* *:00:00
       daily → *-*-* 00:00:00
     monthly → *-*-01 00:00:00
      weekly → Mon *-*-* 00:00:00
      yearly → *-01-01 00:00:00
   quarterly → *-01,04,07,10-01 00:00:00
semiannually → *-01,07-01 00:00:00

相关的服务，参数如下
[Timer]部分定制定时器。Systemd 提供以下一些字段。

OnActiveSec：定时器生效后，多少时间开始执行任务
OnBootSec：系统启动后，多少时间开始执行任务
OnStartupSec：Systemd 进程启动后，多少时间开始执行任务
OnUnitActiveSec：该单元上次执行后，等多少时间再次执行
OnUnitInactiveSec： 定时器上次关闭后多少时间，再次执行
OnCalendar：基于绝对时间，而不是相对时间执行
AccuracySec：如果因为各种原因，任务必须推迟执行，推迟的最大秒数，默认是60秒
Unit：真正要执行的任务，默认是同名的带有.service后缀的单元
Persistent：如果设置了该字段，即使定时器到时没有启动，也会自动执行相应的单元
WakeSystem：如果系统休眠，是否自动唤醒系统

WantedBy 表示该服务所在的 Target，WantedBy=timers.target 指的是，sshd 所在的
Target 是timers.target
```

8.1.3. 启动 service

启动 backdump.service

```
systemctl start backdump
```

查看服务状态

```
systemctl status backdump
```

增加开机自启

```
systemctl enable backdump
```

修改服务文件后重载服务

```
systemctl daemon-reload
```

8.1.4. 启动 timer 服务

启动 backdump.timer

```
systemctl start backdump.timer
```

查看所有已启用的定时器

```
systemctl list-timers
```

增加开机自启

```
systemctl enable backdump.timer
```

重启 backdump.timer

```
systemctl restart backdump.timer
```

Chapter 9. Python38

9.1. CentOS7 Install Python38

9.1.1. 二进制压缩包安装

```
wget http://dl.cdgeekcamp.com/centos/7/python/3/python-3.8.1-
1.el7.x86_64.tar.gz -O python-3.8.1-1.el7.x86_64.tar.gz

tar xf python-3.8.1-1.el7.x86_64.tar.gz

rm -rf /usr/local/python-3.8.1
mv python-3.8.1 /usr/local/python-3.8.1

ln -s /usr/local/python-3.8.1 /usr/local/python3
ln -s /usr/local/python3/bin/pip3 /usr/local/bin/pip3
ln -s /usr/local/python3/bin/python3 /usr/local/bin/python3

pip3 install --upgrade -i https://pypi.tuna.tsinghua.edu.cn/simple pip
pip3 config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

9.1.2. 编译安装Python383

```
# Centos7编译安装 python38.3

mkdir ~/downloads
cd  ~/downloads
wget https://www.python.org/ftp/python/3.8.3/Python-3.8.3.tar.xz -O
Python-3.8.3.tar.xz
tar xf Python-3.8.3.tar.xz
cd Python-3.8.3

# libffi-devel 解决错误: "ImportError: No module named '_ctypes'"
yum install -y gcc make openssl-devel libffi-devel

./configure --prefix=/usr/local/python-3.8.3 \
    --with-openssl=/usr \
    --with-ssl-default-suites=openssl \
    --with-ensurepip \
    --enable-loadable-sqlite-extensions
make -j32
make install

ln -s /usr/local/python-3.8.3 /usr/local/python3

ln -s /usr/local/python3/bin/pip3 /usr/local/bin/pip38
ln -s /usr/local/python3/bin/python3 /usr/local/bin/python38

pip38 install --upgrade -i https://pypi.tuna.tsinghua.edu.cn/simple pip
pip38 config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

9.2. CentOS8 Install Python38

9.2.1. 二进制压缩包安装

```
wget http://dl.cdgeekcamp.com/centos/8/python/3/python-3.8.1-
1.el8.x86_64.tar.gz -O python-3.8.1-1.el8.x86_64.tar.gz

tar xf python-3.8.1-1.el8.x86_64.tar.gz

rm -rf /usr/local/python-3.8.1
mv python-3.8.1 /usr/local/python-3.8.1

ln -s /usr/local/python-3.8.1 /usr/local/python3
ln -s /usr/local/python3/bin/pip3 /usr/local/bin/pip3
ln -s /usr/local/python3/bin/python3 /usr/local/bin/python3

pip3 install --upgrade -i https://pypi.tuna.tsinghua.edu.cn/simple pip
pip3 config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
```

9.2.2. Python创建虚拟环境

python3.3以后 .创建虚拟环境

```
python38 -m venv env
```

进入虚拟环境

```
source env/bin/activate
```

退出

```
deactivate
```

虚拟环境中安装包使用绝对路径 (虚拟环境之外无法使用)

```
/data/env/bin/pip3 install pymysql
```

Chapter 10. Java

10.1. maven 安装

安装maven (配置环境变量)

```
wget http://ftp.meisei-u.ac.jp/mirror/apache/dist/maven/maven-
3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
tar -xf apache-maven-3.6.3-bin.tar.gz
mv apache-maven-3.6.3-bin.tar.gz /usr/local/

#添加环境变量
ln -s /usr/local/apache-maven-3.6.3/bin/mvn /usr/local/bin/

#查看版本
mvn -v
```

全局源

```
cat << EOF > ~/.m2/settings.xml
<settings>
  <mirrors>
    <mirror>
      <id>aliyun</id>
      <name>Aliyun Central</name>
        <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
</settings>
EOF
```

10.2. nodejs 安装

nodejs版本 12

```
wget https://nodejs.org/dist/v12.16.1/node-v12.16.1-linux-x64.tar.xz
tar -xf node-v12.16.1-linux-x64.tar.xz
mv node-v12.16.1-linux-x64 nodejs
mv nodejs /usr/local/

#环境变量
ln -s /usr/local/nodejs/bin/npm /usr/local/bin/
ln -s /usr/local/nodejs/bin/node /usr/local/bin/
#设置淘宝源
npm config set registry https://registry.npm.taobao.org
```

Chapter 11. DNS

11.1. DNS 服务搭建

安装 bind

```
yum install bind-chroot bind-utils
systemctl enable named-chroot
```

配置 bind

```
options {
        listen-on port 53 { any; };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
#       recursing-file  "/var/named/data/named.recursing";
#       secroots-file   "/var/named/data/named.secroots";
        allow-query     { any; };
        recursion yes;

        dnssec-enable no;
        dnssec-validation no;

        /* Path to ISC DLV key */
        bindkeys-file "/etc/named.root.key";

        managed-keys-directory "/var/named/dynamic";

        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "." IN {
        type hint;
        file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

添加正向解析域

```
vim /etc/named.rfc1912.zones

zone "xyt.com" IN {
        type master;
        file "xyt.com.zone";
        allow-update { none; };
};


cat /var/named/xyt.com.zone
$TTL 86400
@       IN      SOA      xytdt.com.     admin.xytdt.com. (
                                        102310111
                                        1D
                                        1H
                                        1W
                                        3H )

@               IN      NS      xytdt.com.
@               IN      A       10.10.10.11
oa              IN      A       10.10.10.13
cloud           IN      A       10.10.10.18
```

启动`bind

```
systemctl start named-chroot
```

检查配置

```
[root@localhost ~]# named-checkzone "ooxx.com" /var/named/ooxx.com.zone
zone ooxx.com/IN: loaded serial 0
OK
```

Chapter 12. Docker

12.1. Docker

12.1.1. 安装

安装

```
sudo yum install -y yum-utils
sudo yum-config-manager \
    --add-repo \
    https://mirrors.ustc.edu.cn/docker-ce/linux/centos/docker-ce.repo

sudo sed -i 's/download.docker.com/mirrors.ustc.edu.cn\/docker-ce/g'
/etc/yum.repos.d/docker-ce.repo

# 官方源
# $ sudo yum-config-manager \
#     --add-repo \
#     https://download.docker.com/linux/centos/docker-ce.repo


sudo yum install -y docker-ce docker-ce-cli containerd.io



sudo systemctl enable docker
sudo systemctl start docker
```

自动化脚本安装

```
# $ curl -fsSL test.docker.com -o get-docker.sh
curl -fsSL get.docker.com -o get-docker.sh
sudo sh get-docker.sh --mirror Aliyun
# $ sudo sh get-docker.sh --mirror AzureChinaCloud
```

12.1.2. 常用命令

拉取镜像

```
docker pull [选项] [Docker Registry 地址[:端口号]/]仓库名[:标签]
```

运行

```
docker run -itd --name nginx1 test/ubuntu:v1.0 /bin/bash
```

参数解释

-i

交互式操作

-t

终端

-d

后台运行

--name

容器名称

终止

```
docker stop ID
```

重启

```
docker restart ID
```

进入容器, attach 退出会导致容器终止，

```
docker attach ID

docker exec -it ID /bin/bash
```

列出镜像

```
docker images

docker image ls -a
```

列出所有容器

```
docker container ls -a
```

获取容器的输出信息

```
docker container logs ID
```

镜像体积

```
docker system df
```

删除镜像

```
docker image rm [选项] <镜像1> [<镜像2> ...]
```

导出容器

```
docker export 7691a814370e > ubuntu.tar
```

导入容器快照

```
cat ubuntu.tar | docker import - test/ubuntu:v1.0
```

12.1.3. Dockerfile 定制镜像

文件 Dockerfile

```
$ mkdir mynginx
$ cd mynginx
$ touch Dockerfile

dockerfile

FROM debian:stretch

RUN apt-get update
COPY package.json /usr/src/app/  #复制文件，源路径 目标路径
ADD ubuntu-xenial-core-cloudimg-amd64-root.tar.gz /
#高级文件复制，压缩文件自动解压，处理权限
CMD echo $HOME  #容器启动命令
USER <用户名>[:<用户组>]  #指定当前用户
```

构建镜像

在Dockerfile文件所在目录执行，-t指定镜像名称

```
docker build -t nginx:v3 .
```

12.1.4. Docker仓库

查找镜像

```
docker search centos

docker pull centos
```

推送镜像 (登录才能使用)

```
$ docker tag ubuntu:18.04 username/ubuntu:18.04

$ docker image ls

REPOSITORY                                                  TAG
IMAGE ID            CREATED             SIZE
ubuntu                                                      18.04
275d79972a86        6 days ago          94.6MB
username/ubuntu                                             18.04
275d79972a86        6 days ago          94.6MB

$ docker push username/ubuntu:18.04

$ docker search username

NAME                      DESCRIPTION
STARS               OFFICIAL            AUTOMATED
username/ubuntu
```

12.1.5. 数据管理

创建一个数据卷

```
docker volume create my-vol
```

查看所有的 数据卷

```
docker volume ls
```

查看数据卷的信息

```
docker volume inspect first
```

启动一个挂载数据卷的容器

```
docker run -d -P \
    --name web \
    # -v my-vol:/usr/share/nginx/html \
    --mount source=my-vol,target=/usr/share/nginx/html \
    nginx:alpine
```

删除数据卷

```
docker volume rm my-vol
```

清理数据卷

无主的数据卷可能会占据很多空间

```
docker volume prune
```

12.1.6. 外部访问容器

Chapter 13. Ansible

13.1. Ansible

13.1.1. 安装

添加源

```
yum -y install epel-release
```

安装

```
yum -y install ansible
```

ansible.cfg 是 Ansible 工具的配置文件；
hosts 用来配置被管理的机器；
roles 是一个目录，playbook 将使用它

13.2. SSH秘钥认证

生成密钥

```
ssh-keygen -N "" -f ~/.ssh/mk
```

上传SSH公钥文件

```
ssh-copy-id -i ~/.ssh/mk.pub root@118.24.186.166
```

创建配置文件

```
cat << EOF > ~/.ssh/config
IdentityFile ~/.ssh/mk
EOF
```

13.3. 添加被管理主机

添加主机

```
vim /etc/ansible/hosts

[test]
10.23.101.x
10.23.101.x

[ss]
192.168.102.12 ansible_ssh_user=tt ansible_ssh_port=28529
[ss:vars]
ansible_ssh_user=tt
ansible_ssh_pass=thcl@123      #tt用户
ansible_become_pass=thcl@123   #root用户
```

测试ansible

```
[root@localhost ~]# ansible "test" -m ping
10.23.101.x | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

13.4. ansible常用模块

配置文件 /etc/ansible/hosts

```
    #  连接目标主机的地址
    ansible_ssh_host
    #  连接目标主机的端口，默认 22 时无需指定
    ansible_ssh_port
    #  连接目标主机默认用户
    ansible_ssh_user
    #  连接目标主机默认用户密码
    ansible_ssh_pass
    #  目标主机连接类型，可以是 local 、ssh 或 paramiko
    ansible_ssh_connection
    #  连接目标主机的 ssh 私钥
    ansible_ssh_private_key_file
    #  指定采用非 Python 的其他脚本语言，如 Ruby 、Perl 或其他类似
ansible_python_interpreter 解释器
    ansible_*_interpreter
```

命令使用 command

```
ansible hostname -m command -a "free -m"
```

远程执行本地脚本script

script的功能是在远程主机执行主控端存储的 shell 脚本文件，相当于 scp + shell 组合。

```
ansible hostname -m script -a "/home/test.sh 12 34"
```

shell模块基本和command相同，但是shell支持管道符

```
ansible hostname -m shell -a "/home/test.sh"
```

copy模块 类似于SCP

向 Client 组中主机拷贝 test.sh 到 /tmp 下，属主、组为 root ，权限为 0755

```
ansible hostname -m copy -a "src=/home/test.sh dest=/tmp/ owner=root
group=root mode=0755"
```

远程主机系统服务管理

```
ansible Client -m service -a "name=nginx state=stoped"
ansible Client -m service -a "name=nginx state=restarted"
ansible Client -m service -a "name=nginx state=reloaded"
```

13.5. ansible-playbook

YAML语法

YAML文件扩展名通常为.yaml或者.yml，一定要对齐，只能使用空格

```
[root@localhost ~]# cat /etc/ansible/hellow.yml
---
- hosts: ss
  remote_user: tt   #tt 用户登陆
  become: yes       #使用 root权限
  become_method: su  #使用 root权限

  tasks:
  - name: touch aa.txt
    shell: touch /root/aa.txt
```

运行

```
[root@localhost ~]# ansible-playbook /etc/ansible/hellow.yml

PLAY [say 'hello world'] ***************************

TASK [Gathering Facts] *****************************
ok: [10.23.101.x]

TASK [echo 'hello world'] **************************
changed: [10.23.101.x]

TASK [print stdout] ********************************
ok: [10.23.101.x] => {
    "msg": ""
}

PLAY RECAP *****************************************************
10.23.101.x                : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Chapter 14. Elasticsearch

14.1. Elasticsearch

14.1.1. 基础使用

ELK安装包下载

```
https://artifacts.elastic.co/downloads/kibana/kibana-7.8.0-x86_64.rpm

https://artifacts.elastic.co/downloads/elasticsearch/kibana-7.8.0-
x86_64.rpm

https://artifacts.elastic.co/downloads/logstash/logstash-7.8.0-x86_64.rpm

https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.8.0-
x86_64.rpm
```

测试

```
curl 'http://localhost:9200/?pretty'
```

健康检查

```
[root@elk1 ~]# curl 'localhost:9200/_cluster/health?pretty'
{
  "cluster_name" : "master-node",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 2,
  "active_primary_shards" : 26,
  "active_shards" : 52,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

查看详细信息

```
curl 'localhost:9200/_cluster/state?pretty'
[source,bash]
```

检索文档是否存在

```
curl -i -XHEAD http://localhost:9200/test/test1/123
[source,bash]
```

空查询

```
[root@elk1 ~]# curl GET 'localhost:9200/_search?pretty'
curl: (6) Could not resolve host: GET; Unknown error
{
  "took" : 9,           #请求花费的毫秒数
  "timed_out" : false,
  "_shards" : {         #参与查询的分片数（ total 字段），多少成功，多少失败
    "total" : 26,
    "successful" : 26,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 58145,    # 文档总数
    "max_score" : 1.0,
    "hits" : [
        ...
```

在所有索引的所有类型中搜索

```
curl GET 'localhost:9200/_search
```

在索引 gb 的所有类型中搜索

```
curl GET 'localhost:9200/gb/_search
```

在索引 gb 和 us 的所有类型中搜索

```
curl GET 'localhost:9200/gb,us/_search
```

在以 g 或 u 开头的索引的所有类型中搜索

```
curl GET 'localhost:9200/g*,u*/_search
```

在索引 gb 的类型 user 中搜索

```
curl GET 'localhost:9200/gb/user/_search
```

在索引 gb 和 us 的类型为 user 和 tweet 中搜索

```
curl GET 'localhost:9200/gb,us/user,tweet/_search
```

分页，每页显示5个结果，页码从1到3

```
curl GET 'localhost:9200/_search?size=5'
curl GET 'localhost:9200/_search?size=5&from=5'
curl GET 'localhost:9200/_search?size=5&from=10'
```

14.1.2. 请求体查询

字符串查询

```
curl GET 'localhost:9200/index_2014*/type1,type2/_search
{}'
```

Post请求

```
curl POST 'localhost:9200/_search
{
"from": 30,
"size": 10
}'
```

14.1.3. 结构化查询

传递 query 参数进行查询

```
GET /_search
{
"query": YOUR_QUERY_HERE
}
```

空查询

```
GET /_search
{
"query": {
"match_all": {}
}
}
```

查询子句

```
GET /_search
{
"query": {
"match": {
"tweet": "elasticsearch"
}
}
}
```

14.1.4. 查询与过滤

term 过滤,主要用于精确匹配哪些值

```
{ "term": { "age": 26 }}
{ "term": { "date": "2014-09-01" }}
{ "term": { "public": true }}
{ "term": { "tag": "full_text" }}

#terms 允许指定多个匹配条件
{
"terms": {
"tag": [ "search", "full_text", "nosql" ]
}
}
```

range 按照指定范围查找数据

```
{
"range": {
"age": {
"gte": 20,
"lt": 30
}
}
}
```

范围操作符包含：

gt

大于

gte

大于等于

lt

小于

lte

小于等于

exists 和 missing

查找文档中是否包含指定字段或没有某个字段

```
{
"exists": {
"field": "title"
}
}
```

bool 过滤

bool 过滤可以用来合并多个过滤条件查询结果的布尔逻辑，它包含以下操作符：

must

多个查询条件的完全匹配,相当于 and 。

must_not

多个查询条件的相反匹配，相当于 not 。

should

至少有一个查询条件匹配, 相当于 or 。 这些参数可以分别继承一个过滤条件或者一个过滤条件的数组：

14.1.5. es集群搭建

IP准备

192.168.102.1
192.168.102.2
192.168.102.3

ES下载地址：https://elasticsearch.cn/download/

安装JAVA11设置JAVA版本

```
dnf install -y java-11-openjdk java-11-openjdk-devel java-11-openjdk-
headless
alternatives --config java
```

ESelasticserch7.8.0 下载 安装

```
#三台服务器都执行
mkdir -p ~/download/
cd download
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-
7.8.0-x86_64.rpm
yum localinstall -y elasticsearch-7.8.0-x86_64.rpm
```

修改配置

```
# ======================== Elasticsearch Configuration
=========================
# ---------------------------------- Cluster
-----------------------------------
cluster.name: es-master
# ------------------------------------ Node
------------------------------------
node.name: node-1
node.data: true
node.master: true
#node.attr.rack: r1
# ---------------------------------- Paths
------------------------------------
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
# ---------------------------------- Memory
-----------------------------------
#bootstrap.memory_lock: true
# ---------------------------------- Network
-----------------------------------
network.host: 192.168.102.1
#http.port: 9200
# --------------------------------- Discovery
----------------------------------
discovery.seed_hosts: ["192.168.102.1", "192.168.102.2", "192.168.102.3"]
cluster.initial_master_nodes: ["node-1", "node-2", "node-3"]
# ---------------------------------- Gateway
-----------------------------------
#gateway.recover_after_nodes: 3
xpack.security.enabled: true
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: certificate
xpack.security.transport.ssl.keystore.path: elastic-certificates.p12
xpack.security.transport.ssl.truststore.path: elastic-certificates.p12
```

另外两个节点同样的配置，只需要将network.host:和node.name:修改为自己的，暂不启动

ES集群xpack配置

官方文档

为ES集群创建节点认证中心

```
/usr/share/elasticsearch/bin/elasticsearch-certutil ca
#可以设置一个密码，也可以直接回车。
#默认文件会在 ES 根目录产生，名为 elastic-stack-ca.p12。
#然后可以将文件 elastic-stack-ca.p12 复制到每个 ES 节点的根目录下
```

为集群中的节点创建证书和私钥（每个节点执行）

```
/usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca ./elastic-
stack-ca.p12
#可以设置一个密码，也可以直接回车。
#默认会生成文件 elastic-certificates.p12
```

elastic-certificates.p12 复制到每个节点的配置文件目录下 /etc/elasticsearch/

```
cp /usr/share/elasticsearch/elastic-certificates.p12  /etc/elasticsearch/
#更改文件权限
chmod 777 /etc/elasticsearch/elastic-certificates.p12
```

如果之前配置节点证书时设置了密码，将密码添加到keystore

```
./bin/elasticsearch-keystore add
xpack.security.transport.ssl.keystore.secure_password
./bin/elasticsearch-keystore add
xpack.security.transport.ssl.truststore.secure_password
```

重启ES集群

```
systemctl restart elasticsearch.service
```

设置内置用户密码(第一个节点设置，其他节点不用)

```
/usr/share/elasticsearch/bin/elasticsearch-setup-passwords interactive
```

创建超级用户赋予权限

```
curl -XPOST -H 'Content-type: application/json' -u elastic:thcl12
'http://192.168.102.11:9200/_xpack/security/user/thcl?pretty' -d '{
  "password" : "superthcl",
  "full_name" : "thcl",
  "roles" : ["admin"],
  "email" : "thcl@thcl.com"
}'

curl -XPOST -H 'Content-type: application/json' -u elastic:thcl12
'http://192.168.102.11:9200/_xpack/security/role/admin?pretty' -d '{
   "run_as": [ "thcl" ],
   "cluster": [ "all" ],
   "indices": [
     {
       "names": [ "*" ],
       "privileges": [ "all" ]
     }
   ]
 }'
```

14.1.6. 创建索引时设置分片数和副本数(默认为5，1)

```
curl -u elastic:password -H 'Content-Type: application/json' -XPUT
'http://localhost:9200/movesss' -d'
{
  "settings" : {
    "index" : {
      "number_of_shards" : 5,
      "number_of_replicas" : 1
    }
  }
}
'
```

查看所有索引

```
curl -u elastic:password 'http://127.0.0.1:9200/_cat/indices?v'
```

14.2. ES快照备份

14.2.1. 备份

配置文件添加配置

```
echo "path.repo: /home/esuser/esbackup" >>
/etc/elasticsearch/elasticsearch.yml
```

重启

```
systemctl restart elasticsearch.service
```

查看快照仓库列表

```
curl -X GET "node1:9200/_cat/repositories?v"
```

创建快照仓库 backup

```
curl -u elastic:thcl12 -X PUT
"192.168.102.81:9200/_snapshot/backup?pretty" -H 'Content-Type:
application/json' -d'
{
  "type": "fs",
  "settings": {
    "location": "/data/elasticsearch_back",
    "max_snapshot_bytes_per_sec": "50mb",
    "max_restore_bytes_per_sec": "50mb",
    "compress": "true"
  }
}
'
```

type: 仓库的类型名称，请求里都是fs，表示file system。
location: 仓库的地址，要与elasticsearch.yml配置文件相同，否则会报错
max_snapshot_bytes_per_sec: 指定数据从Elasticsearch到仓库（数据备份）的写入速度上限，默认是20mb/s
max_restore_bytes_per_sec: 指定数据从仓库到Elasticsearch（数据恢复）的写入速度上限，默认也是20mb/s

创建快照 test

```
curl -u elastic:thcl12 -X PUT "192.168.102.81:9200/_snapshot/backup/test"

curl -u elastic:thcl12 -X PUT
"192.168.102.81:9200/_snapshot/backup/test?wait_for_completion=true"

#wait_for_completion 决定请求是在快照初始化后立即返回（默认）
```

恢复

```
curl -u elastic:thcl12 -X POST
"192.168.102.81:9200/_snapshot/backup/test/_restore"
```

恢复指定索引

```
curl -u elastic:thcl12 -X POST
"192.168.102.81:9200/_snapshot/backup/test/_restore" -H 'Content-Type:
application/json' -d'
{
  "indices": "tocol-otis-gecb",
  "ignore_unavailable": true,
  "include_global_state": true,
  "rename_pattern": "index_(.+)",
  "rename_replacement": "restored_index_$1"
}'
```

删除快照

```
curl -u elastic:thcl12 -X POST
"192.168.102.81:9200/_snapshot/es_backup/test"
```

查看快照信息

```
curl -u elastic:thcl12 -X GET "192.168.102.81:9200/_snapshot/backup/test"
```

根据时间清理数据

```
curl -u elastic:passwd -XPOST
'http://192.168.1.1:9200/index/_doc/_delete_by_query?refresh&slices=5&pret
ty' -H 'Content-Type: application/json' -d'{"query": {"bool": {"must":
[{"range": {"DropOffDateTime": {"lt": "2020-11-30 00:00:00"}}}]}}}'
```

清理120天前的数据

```
curl -u elastic:passwd -XPOST
'http://192.168.1.1:9200/index/_doc/_delete_by_query?refresh&slices=5&pret
ty' -H 'Content-Type: application/json'
-d'{"query":{"bool":{"must":[{"range":{"DateTime":{"lt":"now-120d"}}}]}}}'
```

查看所有index

```
curl -u elastic:passwd 'http://localhost:9200/_cat/indices?v'
```

删除文档

```
curl -u elastic:RKSVd3aLHzWOYvwrqwFk -XDELETE
http://192.168.102.12:9200/device_binding/_doc/1
```

清理数据后释放磁盘空间

```
curl -XPOST
'http://192.168.1.1:9200/index/_forcemerge?only_expunge_deletes=true&max_n
um_segments=1'
```

max_num_segments : 期望merge到多少个segments，1的意思是强行merge到1个segment
only_expunge_deletes : 只做清理有deleted的segments，即瘦身
flush : 清理完执行一下flush，默认是true

14.3. Kibana

14.3.1. 安装

```
dnf --disablerepo="*" --enablerepo="elasticsearch" install -y kibana
```

14.3.2. 配置

```
sed -i '7s/#server.host: "localhost"/server.host: "172.24.45.170"/'
/etc/kibana/kibana.yml
sed -i '28s/#elasticsearch.hosts: .*/elasticsearch.hosts:
["http:\/\/172.24.45.170:9200"]/' /etc/kibana/kibana.yml
sed -i '115s/#i18n.locale: "en"/i18n.locale: "en"/' /etc/kibana/kibana.yml
sed -i 's/#elasticsearch.username: "kibana_system"/elasticsearch.username:
"kibana_system"/' /etc/kibana/kibana.yml
sed -i 's/#elasticsearch.password: "pass"/elasticsearch.password:
"elasticsearch_password"/' /etc/kibana/kibana.yml
```

如果Elasticsearch的 `"network.host"` 参数值为具体的IP地址，比如 `"172.24.45.170"`。

那么，Kibana中的 `"elasticsearch.hosts"` 同样需要设置为 `"172.24.45.170"`，而不能使用 `"0.0.0.0"`。

14.3.3. 开机启动

```
systemctl enable kibana
```

14.3.4. 启动服务

```
systemctl start kibana
```

14.3.5. 最后

如果系统IP为 172.24.45.170，则访问 http://172.24.45.170:5601/ <<<

Chapter 15. MongoDB

15.1. MongoDB

15.1.1. 安装

15.1.2. 基本操作

创建数据库

```
use database_name
```

删除数据库

```
db.dropDatabase()
```

创建集合

```
db.createCollection("users")
```

删除集合

```
db.user.drop()
```

插入文档

```
db.users.insert({name:'zhangsan',age:18})` +
db.users.save({name:'lisisi',age:16})
```

insert和save方法都可以插入数据，当默认的“_id”值已存在时，调用insert方法插入会报错；而save方法不会,会更新相同的_id所在行数据的信息。

删除文档

```
db.users.remove({'name':'zhangsan'})` +
db.collection.drop()
```

修改文档

```
db.users.update({'name':'zhangsan'},{$set:{'name':'wanger'}})

#修改多条相同的文档，需要设置 multi 参数为 true
db.collection_name.update({"字段名":"字段值"},{$set:{"要修改的字段名":"修改后的字段值"}},{multi:true})
```

查询文档

```
db.collection.find(query, projection)
```

易读方式

```
db.collection.find(query, projection).pretty()
```

15.1.3. 条件查询

等于

db.col.find({"by":"root"}).pretty()

lt 小于

db.col.find({"likes":{$lt:50}}).pretty()

lte 小于或等于

db.col.find({"likes":{$lte:50}}).pretty()

gt 大于

db.col.find({"likes":{$gt:50}}).pretty()

gte 大于或等于

db.col.find({"likes":{$gte:50}}).pretty()

ne 不等于

db.col.find({"likes":{$ne:50}}).pretty()

and

db.col.find({key1:value1, key2:value2}).pretty()

or

db.col.find({$or: [{key1: value1}, {key2:value2}]}).pretty()

Limit

```
db.col.find().limit(NUMBER)
```

skip 跳过数据读取

```
db.col.find({},{"title":1,_id:0}).limit(1).skip(1)
```

排序

1：升序，-1：降序

```
db.col.find().sort({KEY:1})
```

索引

```
db.col.createIndex({"title":1})


查看集合索引 :: db.col.getIndexes()
查看集合索引大小 :: db.col.totalIndexSize()
删除集合所有索引 :: db.col.dropIndexes()
删除集合指定索引 :: db.col.dropIndex("索引名称")
```

聚合

```
db.col.aggregate([{$group : {_id : "$by_user", num : {$sum : 1}}}])


$sum :: 计算总和，{$sum: 1}表示返回总和×1的值(即总和的数量),使用{$sum: '$制定字段
'}也能直接获取制定字段的值的总和

$avg :: 求平均值

$min :: 求min值

$max :: 求max值

$push :: 将结果文档中插入值到一个数组中

$first :: 根据文档的排序获取第一个文档数据

$last ::  获取最后一个数据
```

15.2. 主从

15.2.1. 配置

修改配置文件

```
cat << EOF >/etc/mongo.conf
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

wiredTiger:
    engineConfig:
      configString : cache_size=512M
# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# Where and how to store data.
storage:
  dbPath: /data/mongo
  journal:
    enabled: true
  directoryPerDB: true
  wiredTiger:
     engineConfig:
        cacheSizeGB: 8
#如果一台机器启动一个实例这个可以注释选择默认，如果一台机器启动多个实例，需要设置内存大小
，避免互相抢占内存
        directoryForIndexes: true
#  engine:
#  wiredTiger:

# how the process runs
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid  # location of pidfile
  timeZoneInfo: /usr/share/zoneinfo

# network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0  # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6
addresses or, alternatively, use the net.bindIpAll setting.
  maxIncomingConnections: 5000
```

```
#security:

#operationProfiling:

#replication:
replication:
  replSetName: rs0

#sharding:

## Enterprise-Only Options

#auditLog:

#snmp:
security:
  authorization: enabled  #开启访问控制
  keyFile: /data/mongo/mongo.key #集群实现安全通信。作为所有
mongod后台进程允许加入集群的凭证，所有集群中的节点共用一个keyFile，避免其他
mongod非法加入集群。
EOF
```

在主节点生成keyfile文件,配置添加路径，移动到mongo目录，修改文件用户组，权限为600

```
openssl rand -base64 741 > mongo.key
chown mongod:mongod mongo.key
chmod 600 mongo.key
mv mongo.key /data/mongo
```

复制到主从机器

```
scp -P port mongo.key root@host:/src
```

配置文件修改完成，重启mongo

```
systemctl restart mongod
```

登陆mongo,输入配置变量

```
config_myset={_id:'rs0',members:[{_id:0,host:'118.116.4.131:27017'},{_id:1
,host:'118.123.245.177:27017'}]};
rs.initiate(config_myset)
rs.state()
rs.isMaster()
```

创建用户

```
use admin
db.createUser(
  {
    user: "root",
    pwd: "4MIJ6Axurmc9ns0Pt0Pr",
    roles: [ "root"]
  }
)
```

从库添加查看权限,从库执行

```
use admin
db.auth("name","passwd")      #主库的用户密码
rs.secondaryOk();
rs.slaveOK();
```

新增slave节点,上传access.key,并赋予权限，属主，600 在master节点操作

```
mongo
use admin
db.auth("root","123456")
rs.add("ip:port")
```

从库执行

```
use admin
db.auth("name","passwd")
rs.secondaryOk();
rs.slaveOK();
```

Chapter 16. Nginx

16.1. Nginx

16.1.1. 安装

添加源

```
cat << EOF > /etc/yum.repos.d/nginx.repo
[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
EOF
```

install

```
yum install -y nginx
```

页面字符串替换 压缩

```
    #压缩
    gzip on;
    gzip_comp_level 2;
    gzip_types text/plain application/octet-stream application/javascript
application/x-javascript text/css application/xml text/javascript
image/jpeg image/gif image/png font/ttf font/x-woff;
    gzip_vary on;

    location / {
    sub_filter 'xxxxx1111'  'ooooo2222';
    sub_filter_once off;
    sub_filter_last_modified on;
    sub_filter_types *;
    proxy_set_header Accept-Encoding '';
    proxy_pass http://127.0.0.1:8000;
    }
```

Ninx-TCP连接转发

```
upstream tcp{
    hash $remote_addr consistent;
    server      ip:9000 max_fails=3 fail_timeout=10s;
}
server{
    listen 8000;
    proxy_connect_timeout 20s;
    proxy_timeout 5m;
    proxy_pass tcp;
}
```

Chapter 17. Iptables

17.1. `Iptables

17.1.1. 安装

```
yum install -y iptables-services
```

停止firewalld

```
systemctl  stop  firewalld
```

禁用firewalld

```
systemctl mask firewalld.service
```

iptables 开机自起

```
systemctl enable iptables.service
```

删除所有的链条和规则

```
iptables -F
```

启用仓库源

```
yum-config-manager --enable repository…
```

禁用仓库源

```
yum-config-manager --disable repository…
```

显示已安装的仓库源

```
yum repolist all
```

17.1.2. 保存规则

保存规则到文件

```
iptables-save > /etc/sysconfig/iptables
```

保存规则

```
service iptables save
```

17.1.3. 使用

开放4545端口

```
iptables -I  INPUT -p tcp --dport 4545 -j ACCEPT
```

开放指定IP端口

```
iptables -I  INPUT -p tcp -s 47.108.220.208 --dport 8080 -j ACCEPT
```

开放50000-60000端口

```
iptables -I INPUT -p tcp --dport 50000:60000 -j ACCEPT
```

参数解释
-I 插入一条新的规则
-A 追加
-D 删除规则
-L 列出所有的规则链条
-F 删除所有的规则链条
-N 创建新的链条
-p 指定协议（tcp,upd）
--dport 目标端口
-j 指定是 ACCEPT 接收 或者 DROP 不接收

开放ping

```
iptables -I INPUT -p icmp -j ACCEPT
```

允许本地回环

```
iptables -I INPUT -i lo -j ACCEPT
```

列出规则和相对应的编号

```
iptables -L -n --line-number
```

删除对应链的规则

```
iptables -D INPUT 1
```

配置默认的不让进

```
tables -P INPUT DROP
```

默认的不允许转发

```
iptables -P FORWARD DROP
```

默认的可以出去

```
iptables -P INPUT ACCEPT
```

阻止Windows蠕虫的攻击

```
iptables -I INPUT -j DROP -p tcp -s 0.0.0.0/0 -m string --algo kmp
--string "cmd.exe"
```

防止SYN洪水攻击

```
iptables -A INPUT -p tcp --syn -m limit --limit 5/second -j ACCEPT
```

保存规则到配置文件中

```
cp /etc/sysconfig/iptables /etc/sysconfig/iptables.bak
.任何改动之前先备份，请保持这一优秀的习惯
iptables-save > /etc/sysconfig/iptables
cat /etc/sysconfig/iptables
```

Iptables详解

Chapter 18. Git

18.1. Git

18.1.1. 新建分支推送到远程服务器

查看远程分支

```
git branch -a
```

将远程主机的更新，全部取回本地

```
git fetch
```

再次查看远程分支：

```
git branch -a
```

然后拉取远程分支到本地

```
git checkout -b 远程分支名 origin/远程分支名
```

新建一个本地分支

```
git checkout -b name
```

把新建的本地分支push到远程服务器

```
git push origin name:name
```

Chapter 19. Linux

19.1. Linux系统相关

Linux显示线程限制

```
ulimit -n
```

列出系统打开的文件

```
lsof
```

筛选

```
lsof -c java| awk '{print $4}' |grep "^[0-9]" |wc -l
```

-c

根据程序筛选

-p

根据线程筛选

lsof详解

生成UUID

```
uuidgen enp1s0
```

排序去重

```
cat device_yu_name.txt |sort |uniq -c
```

更改系统日志级别

```
echo 'LogLevel=debug' >> /etc/systemd/system.conf
```

最大可以打开的文件

```
cat /etc/security/limits.conf
```

禁用SELINUX，必须重启才能生效

```
echo SELINUX=disabled>/etc/selinux/config

echo SELINUXTYPE=targeted>>/etc/selinux/config
```

bz2文件解压

```
yum install bzip2

bzip2 -d name.bz2
```

更新动态链接库

```
ldconfig
```

service服务增加文件限制

```
LimitNOFILE=65535
```

清理僵尸进程

```
ps -e -o stat,ppid,pid,cmd|egrep '^[Zz]'

kill -9 pid
```

'[Zz]'：这是正则表达式，表示第一个字符的位置，[Zz]，表示z或者大写的Z字母，即表示第一个字符为Z或者z开头的进程数据，这样是因为僵尸进程的状态信息以Z或者z字母开头。

监听文件变化

```
yum install inotify-tools

inotifywait -rme modify,attrib,move,create,delete,delete_self /root/test/
--format "%Xe %w%f"
```

19.2. shell

19.2.1. 变量

```
your_name="qinjx"
echo $your_name
echo ${your_name}

#只读变量
myUrl="http://www.google.com"
readonly myUrl
myUrl="http://www.runoob.com"


#删除变量
#变量被删除后不能再次使用。unset 命令不能删除只读变量。
unset myUrl



#多行注释
#EOF可换
:<<EOF
注释内容...
注释内容...
注释内容...
EOF
```

19.2.2. 传参

参数解释

$#

传递到脚本的参数个数

$*

以一个单字符串显示所有向脚本传递的参数。如"$*"用「"」括起来的情况、以"$1 $2 … $n"的形式输出所有参数。

$$

脚本运行的当前进程ID号

$!

后台运行的最后一个进程的ID号

$@

与$*相同，但是使用时加引号，并在引号中返回每个参数。如"$@"用「"」括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。

$-

显示Shell使用的当前选项，与set命令功能相同。

$?

显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。

$* 与 $@ 区别
相同点：都是引用所有参数
不同点：只有在双引号中体现出来。假设在脚本运行时写了三个参数 1、2、3，，则 " * " 等价于 "1 2 3"（传递了一个参数），而 "@" 等价于 "1" "2" "3"（传递了三个参数）。

19.2.3. 运算符

```
-eq 检测两个数是否相等，相等返回 true。                        [ $a -eq $b ] 返回 false。
-ne 检测两个数是否不相等，不相等返回 true。                    [ $a -ne $b ] 返回 true。
-gt 检测左边的数是否大于右边的，如果是，则返回 true。 [ $a -gt $b ] 返回 false。
-lt 检测左边的数是否小于右边的，如果是，则返回 true。 [ $a -lt $b ] 返回 true。
-ge 检测左边的数是否大于等于右边的，如果是，则返回 true。   [ $a -ge $b ] 返回 false。
-le 检测左边的数是否小于等于右边的，如果是，则返回 true。   [ $a -le $b ] 返回 true。


!   非运算，表达式为 true 则返回 false，否则返回 true。  [ ! false ] 返回 true。
-o  或运算，有一个表达式为 true 则返回 true。          [ $a -lt 20 -o $b -gt 100 ] 返回 true。
-a  与运算，两个表达式都为 true 才返回 true。          [ $a -lt 20 -a $b -gt 100 ] 返回 false。
```

条件表达式要放在方括号之间，并且要有空格，例如: [$a==$b] 是错误的，必须写成 [$a == $b]。

19.2.4. vim

0 或功能键[Home]

这是数字“ 0 ”:移动到这一列的最前面字符处 (常用)

$ 或功能键[End]

移动到这一列的最后面字符处(常用

G

移动到这个文件的最后一列(常用)

gg

移动到这个文件的第一列,相当于 1G 啊! (常用)

n<Enter>

为数字。光标向下移动 n 列(常用)

x, X

在一列字当中,x 为向后删除一个字符 [backspace] 亦即是倒退键) (常用)相当于[del]按键), X为向前删除一个字符

nx

n 为数字,连续向后删除 n 个字符。举例来说,我要连续删除 10 个字符, “10x” 。

dd

删除光标所在的那一整列(常用)

u

复原前一个动作。(常用)

ndd

n 为数字。删除光标所在的向下 n 列,例如 20dd 则是删除 20 列 (常用)

H

光标移动到这个屏幕的最上方那一列的第一个字符

M

光标移动到这个屏幕的中央那一列的第一个字符

L

光标移动到这个屏幕的最下方那一列的第一个字符

19.2.5. Systemctl

19.2.6. `systemctl` 参数

list-units

列出 systemd 当前已加载到内存中的单元

list-timers

列出当前已加载到内存中的定时器(timer)单元，并按照下次执行的时间点排序

is-active

检查指定的单元中， 是否有处于活动(active)状态的单元

cat

显示指定单元的单元文件内容

list-dependencies

显示单元的依赖关系

enable

设为"开机时自动启动"或"插入某个硬件时自动启动"

disable

撤销这些单元的"开机时自动启动"以及"插入某个硬件时自动启动"

mask

屏蔽指定的单元或单元实例(禁用)

unmask

解除对指定单元或单元实例的屏蔽，这是 mask 命令的反动作

edit

调用文本编辑器(参见下面的"环境变量"小节) 修改 指定的单元或单元实例。

list-jobs

列出正在运行中的任务。

daemon-reload

重新加载 systemd 守护进程的配置

19.2.7. 配置 `.service`

Description=

有利于人类阅读的、对单元进行简单描述的字符串

Documentation=

一组用空格分隔的文档URI列表， 这些文档是对此单元的详细说明

Before=a

此服务在a服务之前启动

After=a

此服务在a服务之后启动

19.2.8. 配置 `.timer`

service.timer

```
[Unit] # 定义元数据
[Timer] #定义定时器
OnActiveSec：定时器生效后，多少时间开始执行任务
OnBootSec：系统启动后，多少时间开始执行任务
OnStartupSec：Systemd 进程启动后，多少时间开始执行任务
OnUnitActiveSec：该单元上次执行后，等多少时间再次执行
OnUnitInactiveSec： 定时器上次关闭后多少时间，再次执行
OnCalendar：基于绝对时间，而不是相对时间执行,用于和 crond 类似的定时任务
,以实际时间执行。
AccuracySec：如果因为各种原因，任务必须推迟执行，推迟的最大秒数，默认是60秒
Unit：真正要执行的任务，默认是同名的带有.service后缀的单元
Persistent：如果设置了该字段，即使定时器到时没有启动，也会自动执行相应的单元
WakeSystem：如果系统休眠，是否自动唤醒系统
```

时间后缀

```
systemd 在解析时长字符串(用于赋值)的时候，
接受与时长的显示类似的语法，不同之处在于可以省略空格。 可以理解的时间单位如下：

usec, us, μs

(微秒)
msec, ms

(毫秒)
seconds, second, sec, s

(秒)
minutes, minute, min, m

(分钟)
hours, hour, hr, h

(小时)
days, day, d

(天)
weeks, week, w

(星期)
months, month, M

(月)[=30.44天]
years, year, y

(年)[=365.25天]
```

移除状态为faild的任务，只是状态移除，服务本身不会被删除

```
systemctl reset-faild
```

19.2.9. Ulimit

ulimit 用于限制启动进程所占用的资源

实例

```
[root@localhost ~]# ulimit -a
core file size          (blocks, -c) 0           #core文件的最大值为100
blocks。
data seg size           (kbytes, -d) unlimited   #进程的数据段可以任意大。
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited   #文件可以任意大。
pending signals                 (-i) 98304       #最多有98304个待处理的信号。
max locked memory       (kbytes, -l) 32
#一个任务锁住的物理内存的最大值为32KB。
max memory size         (kbytes, -m) unlimited
#一个任务的常驻物理内存的最大值。
open files                      (-n) 1024        #一个任务最多可以同时打开
1024的文件。
pipe size            (512 bytes, -p) 8           #管道的最大空间为4096字节。
POSIX message queues     (bytes, -q) 819200      #POSIX
的消息队列的最大值为819200字节。
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240       #进程的栈的最大值为
10240字节。
cpu time               (seconds, -t) unlimited   #进程使用的CPU时间。
max user processes              (-u) 98304
#当前用户同时打开的进程（包括线程）的最大个数为98304。
virtual memory          (kbytes, -v) unlimited   #没有限制进程的最大地址空间。
file locks                      (-x) unlimited
#所能锁住的文件的最大个数没有限制。
```

Chapter 20. Archlinux

20.1. Archlinux

20.1.1. 新装系统终端无法打开

```
localedef -f UTF-8 -i en_US en_US.UTF-8
```

tty终端无法打开

```
systemctl start getty@tty1
```

Chapter 21. Kafka

21.1. Kafka

21.1.1. kafka使用

删除topic

```
kafka-topics.sh --delete --zookeeper hadoop01:2181 hadoop02:2181
hadoop03:2181 --topic
```

修改topic的offset

```
kafka-consumer-groups.sh --bootstrap-server 192.168.102.12:9092 --group
rtd --reset-offsets --topic rtdReceive --to-offset 0 --execute
```

查询消费组

```
kafka-consumer-groups.sh --bootstrap-server {kafka连接地址} --list
```

删除消费组

```
kafka-consumer-groups.sh --bootstrap-server {kafka连接地址} --delete --group
{消费组}
```

查看topic详细信息

```
kafka-topics.sh -zookeeper hadoop01:2181 hadoop02:2181 hadoop03:2181
-describe -topic dping
```

列出所有topic

```
kafka-topics.sh --list --zookeeper hadoop01:2181 hadoop02:2181
hadoop03:2181
```

发送消息

```
kafka-console-producer.sh --broker-list hadoop01:9092 hadoop02:9092
hadoop03:9092 --topic dping
```

查看消费组的消费情况

```
kafka-consumer-groups.sh --bootstrap-server hadoop01:9092 --describe
--group thcl-consumer-group
```

创建topic

```
kafka-topics.sh --create --zookeeper hadoop01:2181 hadoop02:2181
hadoop03:2181 --replication-factor 4 --partitions 6 --topic test4replic
```

kafka单节点启动

```
kafka-server-start.sh -daemon /data/software/kafka_2.13-
2.7.0/config/server.properties
```

Broker：消息中间件处理结点，一个Kafka节点就是一个broker，多个broker可以组成一个Kafka集群。
Topic：一类消息，例如page view日志、click日志等都可以以topic的形式存在，Kafka集群能够同时负责多个topic的分发。
Partition：topic物理上的分组，一个topic可以分为多个partition，每个partition是一个有序的队列。
Segment：partition物理上由多个segment组成，下面2.2和2.3有详细说明。
offset：每个partition都由一系列有序的、不可变的消息组成，这些消息被连续的追加到partition中。 partition中的每个消息都有一个连续的序列号叫做offset,用于partition唯一标识一条消息.

创建topic

```
kafka-topics.sh –zookeeper localhost:2181 –create –topic my-topic
–partitions 1 –replication-factor 1 –config max.message.bytes=64000
–config flush.messages=1
```

创建topic后修改topic参数

```
kafka-configs.sh –zookeeper localhost:2181 –entity-type topics –entity-
name my-topic –alter –add-config max.message.bytes=128000
```

查看topic的offset情况

```
kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list hadoop01:9092
--topic realTimeData
```

查看consumer group列表

```
kafka-consumer-groups.sh  --bootstrap-server hadoop01:9092 --list
```

查看分区情况，节点信息，复制情况

```
kafka-topics.sh  --topic realTimeData --describe --zookeeper hadoop01:2181
hadoop02:2181 hadoop03:2181
```

kafka修改topic的副本数

需要新建json文件increase-replication-factor.json

```
{
    "version": 1,
    "partitions": [{
            "topic": "aaad",
            "partition": 0,
            "replicas": [1, 3]
        },
        {
            "topic": "aaad",
            "partition": 1,
            "replicas": [1, 2]
        },
        {
            "topic": "aaad",
            "partition": 2,
            "replicas": [2, 3]
        },
        {
            "topic": "aaad",
            "partition": 3,
            "replicas": [2, 3]
        },
        {
            "topic": "aaad",
            "partition": 4,
            "replicas": [1, 3]
        },
        {
            "topic": "aaad",
            "partition": 5,
            "replicas": [1, 2]
        }
    ]
}
```

然后执行命令

```
kafka-reassign-partitions.sh --zookeeper node01:2181 node02:2181
node03:2181  --reassignment-json-file increase-replication-factor.json
--execute --additional
```

配置

Chapter 22. Ceph

22.1. Ceph

22.1.1. 安装

配置基础yum源

```
yum clean all
rm -rf /etc/yum.repos.d/*.repo
wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-
7.repo
sed -i '/aliyuncs/d' /etc/yum.repos.d/CentOS-Base.repo
sed -i '/aliyuncs/d' /etc/yum.repos.d/epel.repo
```

配置ceph源

```
cat >> /etc/hosts << EOF
[ceph]
name=ceph
baseurl=http://mirrors.aliyun.com/ceph/rpm-jewel/el7/x86_64/
gpgcheck=0
priority =1
[ceph-noarch]
name=cephnoarch
baseurl=http://mirrors.aliyun.com/ceph/rpm-jewel/el7/noarch/
gpgcheck=0
priority =1
[ceph-source]
name=Ceph source packages
baseurl=http://mirrors.aliyun.com/ceph/rpm-jewel/el7/SRPMS
gpgcheck=0
priority=1
EOF
```

更新源

```
yum clean all && yum makecache
```

安装Ceph

```
yum install ceph -y
```