1、CentOS7 系列 @CreateByShadow

1、系统常用工具

- 文本编辑工具

```
yum install -y  vim
```

- 网络工具

```
yum install -y  net-tools
```

- 上传下载组件

```
yum install -y  lrzsz
```

- yum 工具包

```
yum install -y  yum-utils
```

- 编译工具

```
yum install -y  gcc-c++
```

- perl 兼容

```
yum install -y  pcre pcre-devel
```

- SSL

```
yum install -y  openssl openssl-devel patch
```

- 下载工具

```
yum install -y  wget
```

- 自动补全

```
yum install -y  bash-completion
```

- lsof 查看打开文件工具

```
yum install -y  lsof
```

- zip unzip 解压缩工具

```
yum install -y  zip unzip
```

- 目录树

```
yum install -y tree
```

2、配置 yum 源

- 进入 /etc/yum.repos.d/

```
cd /etc/yum.repos.d/
```

- 备份 CentOS-Base.repo

```
mv CentOS-Base.repo CentOS-Base.repo.bak
```

- 下载新 yum 源并修改名称

```
wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/CentOS-7.repo
```

- 清理缓存生成新缓存

```
yum clear all
yum makecache
```

- 查看 yum 源

```
# 所有的 yum 源
yum repolist all
# 可以的 yum 源
yum repolist enabled
```

3、安装 java 环境

1. yum 安装方式

- 查看是否有 java 信息

```
yum list java-1.8*
```

- 下载 jdk

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

- 查看安装目录

```
ll /usr/lib/jvm/java-1.8.0-openjdk...
```

- 配置 java 环境

```
# 编辑 profile 文件
vim /etc/profile
# 输入内容
# jdk 安装目录
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk....
# CLASSPATH
export
CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/t
ools.jar
# PATH
export PATH=$PATH:$JAVA_HOME/bin
```

- 配置生效

```
source /etc/profile
```

- 查看 java 版本

```
java -version
```

2. rpm 安装方式

- 下载 rpm 安装包

```
jdk-8u202-linux-x64.rpm
```

- 本地安装

```
yum localinstall jdk-8u202-linux-x64.rpm
```

- 查看安装目录

```
find / -name java
# /usr/java/jdk1.8.9_202-amd64
```

- 配置 java 环境

```
# 编辑 profile 文件
vim /etc/profile
# 输入一下内容
# jdk 安装目录
export JAVA_HOME=/usr/java/jdk1.8.9_202-amd64
# CLASSPATH
export
CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/t
ools.jar
# PATH
export PATH=$PATH:$JAVA_HOME/bin
```

- 配置生效

```
source /ect/profile
```

- 查看 java 版本

```
java -version
```

3. tar 安装方式

- 下载 tar 压缩包

```
jdk-8u291-linux-x64.tar.gz
```

- 解压缩

```
tar -zxvf jdk-8u291-linux-x64.tar.gz -C /usr/local
```

- 查看安装目录

```
find / -name java
# /usr/local/jdk1.8.0_291
```

- 配置 java 环境

```
# java 安装目录
echo 'export JAVA_HOME=/usr/local/jdk1.8.0_291' >> /etc/profile
# CLASSPATH
echo 'export
CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/t
ools.jar' >> /etc/profile
# PATH
echo 'export PATH=$PATH:$JAVA_HOME/bin' >> /etc/profile
```

- 配置生效

```
source /etc/profile
```

- 查看 java 版本

```
java -version
```

4、安装 maven 环境

1. yum 安装方式

- 卸载旧版本

```
# 找到已安装的 maven
yum list installed | grep maven
# 删除
yum remove maven.xxxx
```

- 下载 maven repo

```
wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
# 替换内容
sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
```

- 安装

```
yum -y install apache-maven
# 安装过程中出现冲突移除对于的包再次尝试安装
# yum -y remove plexus-cipher.noarch plexus-sec-dispatcher.noarch
```

- 查看 maven 安装目录

```
# 查找包路径
rpm -qa | grep apache-maven
# 根据包路径查找安装目录
rpm -ql apache-maven
# /usr/share/apache-maven
# maven 配置文件位置
/etc/maven/setting.xml
```

- 配置 maven 镜像

```
# 编辑 setting.xml
vim /etc/maven/setting.xml
# 指定 maven 依赖包位置
<localRepository>/usr/maven/repository</localRepository>
# 配置阿里云镜像
<mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>阿里云公共仓库</name>
    <url>https://maven.aliyun.com/repository/public</url>
</mirror>
```

- 配置 maven 环境

```
# 编辑 profile 文件
vim /etc/profile
# 输入以下内容
# maven 安装目录
export M2_HOME=/usr/share/apache-maven
# PATH
export PATH=$PATH:$M2_HOME/bin
```

- 配置生效

```
source /etc/profile
```

- 查看 maven 版本

```
mvn -v
```

2. tar 安装方式

- 下载安装包

```
apache-maven-3.5.4-src.tar.gz
```

- 解压缩

```
tar -zxvf apache-maven-3.5.4-src.tar.gz -C /usr/local
```

- 安装目录

```
/usr/local/apache-maven-3.5.4
```

- 配置 maven 环境

```
# 编辑 profile 文件
vim /etc/profile
# 输入以下内容
# maven 安装目录
export M2_HOME=/usr/local/apache-maven-3.5.4
# PATH
export PATH=$PATH:$M2_HOME/bin
```

- 配置生效

```
source /etc/profile
```

- 查看 maven 版本

```
mvn -v
```

5、安装 git 环境

1. git 环境搭建

- yum 安装

```
yum -y install git
```

- 查看 git

```
git --version
```

- 配置 git 账户

```
# git 用户名称
git config --global user.name "shadow"
# git 邮箱
git config --global user.email "111@qq.com"
```

- 生成秘钥

```
ssh-keygen -t rsa -C "111@qq.com"
```

- 查看生成的秘钥

```
cat xxx/.ssh/id_rsa.pub
```

- 得到秘钥后可在网页上配置 SSH
- 检查是否可用

```
ssh -T git@github.com
```

2. git 常用命令

- 项目克隆(会自动与远程建立关联)

```
git clone https://github.com/wangchirl/concurrent
```

- 本地项目与 github 项目建立关联

```
# 初始化
git init
# 查看是否存在关联
git remote -v
# 建立关联
git remote add origin https://github.com/wangchirl/concurrent.git
# 再次查看关联
git remote -v
```

- 查看本地修改内容

```
git status
```

- 提交修改内容

```
# 提交修改的全部文件
git add .
# 提交修改的指定文件
git add xxx.java
# 提交说明
git commit -m "update"
# 提交到远程分支
git push origin dev-branch
# 拉取远程代码
git pull origin dev-branch
```

- 暂存修改内容

```
# 查看暂存的列表
git stash list
# 暂存当前修改的内容
git stash save local
# 弹出暂存的内容 n 可指定,一般为0
git stash pop stash@{n}
```

- 切换分支

```
git checkout prd-branch
```

- 合并分支内容

```
# 切换到要合并的分支
git checkout dev-branch
# 拉取当前分支最新代码
git pull origin dev-branch
# merge prd-branch 的代码到当前分支
git merge prd-branch
# 有冲突解决冲突，没有就走一遍提交流程
git status
git add .
git commit -m "merge"
git push origin dev-branch
```

- 回滚(谨慎使用，最好先创建一个分支做好备份)

```
# 回滚到某个提交节点
git reset --hard xxxSHA
# 强制回滚
git push origin HEAD --force
```

6、安装 docker 环境

1. docker 环境搭建

- 卸载原来的 docker 环境

```
yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
```

- 配置 docker

```
sudo yum install -y yum-utils

yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

- 安装 docker

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

- 启动 docker

```
systemctl start docker
```

- 测试 docker

```
docker -v # 检查版本
docker ps # 检查运行的容器
docker images # 检查已有镜像
```

- 开机启动 docker

```
systemctl enable docker
```

- 配置 docker 镜像

```
# 创建文件夹
mkdir -p /etc/docker
# 修改配置，设置镜像
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://vw9qapdy.mirror.aliyuncs.com"]
}
EOF
# 重启后台线程
systemctl daemon-reload
# 重启 docker
systemctl restart docker
```

2. docker 基本操作

1. 镜像操作

- 查看镜像

```
docker images
```

> - **REPOSITORY**：表示镜像的仓库源
> - **TAG**：镜像的标签
> - **IMAGE ID**：镜像ID
> - **CREATED**：镜像创建时间
> - **SIZE**：镜像大小

- 搜索镜像

```
# docker search <镜像名称>
docker search mysql
```

> **NAME:** 镜像仓库源的名称
>
> **DESCRIPTION:** 镜像的描述
>
> **STARS:** 类似 Github 里面的 star，表示点赞、喜欢的意思。
>
> **OFFICIAL:** 是否 docker 官方发布
>
> **AUTOMATED:** 自动构建。

- 拉取镜像

```
# 默认拉取最新版本，可指定 TAG
# docker pull <镜像名称>
# docker pull <镜像名称>:TAG
docker pull mysql:5.7
```

- 删除镜像

```
# 按名称删除
# docker rmi <镜像名称>
# 按镜像ID删除
# docker rmi <镜像ID>
# 删除所有镜像
# docker rmi `docker images -q`
docker rmi mysql
```

- 镜像构建

```
# docker build -t <镜像名称> <Dockerfile文件>
```

> **-t**： 指定要创建的目标镜像名
>
> Dockerfile 文件绝对路径

- 镜像标签

```
# docker tag <镜像ID> 用户名称/镜像源名称:tag名称
docker tag xxx shadow/centos:dev
```

2. 容器操作

- 查看容器

```
# 查看启动的容器
docker ps
# 查看全部的容器
docker ps -a
# 查看最后一次运行的容器
docker ps -l
# 查看停止的容器
docker ps -f status=exited
```

- 创建与启动容器

```
docker run [ -i -t -v -d --name ] <镜像名称>:TAG /bin/bash
```

> **-i**: 交互式操作
>
> **-t**: 终端
>
> **-v**: 目录挂载
>
> **-d:**让容器在后台运行
>
> **--name**: 容器名称

  - 交互方式创建容器

```
# docker run -it --name=<容器名称> <镜像名称>:TAG [/bin/bash|bash]
docker run -it --name=mysql mysql:5.7 /bin/bash
# 或者
docker run -it --name=mysql mysql:5.7 bash
# 退出交互，这时容器将会退出停止
exit
```

  - 守护方式创建容器

```
# docker run -id --name=<容器名称> <镜像名称>:TAG
docker run -id --name=mysql mysql:5.7
# exec 命令登录容器
# docker exec -it <容器名称/ID> /bin/bash
docker exec -it mysql /bin/bash
# 退出容器，exec 命令进入容器的情况下退出不会导致容器停止
exit
# attach 命令登录容器
# docker attach -it <容器名称/ID> /bin/bash
docker attach -it mysql /bin/bash
# attach命令进入容器情况下 exit 退出容器时，容器将停止
```

- 停止容器

```
# docker stop <容器名称/ID>
docker stop mysql
```

- 启动已有容器

```
# docker start <容器名称/ID>
docker start mysql
```

- 重启容器

```
# docker restart <容器名称/ID>
docker restart mysql
```

- 文件拷贝

```
# 将文件拷贝到容器内
docker cp <文件/目录> 容器名称:容器目录
# 将容器内文件拷贝到宿主机
docker cp 容器名称:容器目录 <宿主机路径>
```

- 目录挂载

```
# docker run -id --name=<容器名称> -v <宿主机目录>:<容器目录> <镜像名称>
docker run -id --name=mysql -v /usr/mysql/conf:/usr/local/conf mysql:5.7
```

- 端口映射

```
# docker run -id --name=<容器名称> -p <宿主机端口>:<容器端口> <镜像名称>
docker run -id --name=mysql -p 3306:3306 mysql:5.7
```

- 查看容器 ip

```
# 查看容器的各种数据
# docker inspect <容器名称/ID>
docker inspect mysql
# 直接输出容器IP地址
# docker inspect --format='{{.NetworkSettings.IPAddress}}' <容器名称/ID>
docker inspect --format='{{.NetworkSettings.IPAddress}}' mysql
```

- 删除容器

```
# 先要停止运行的容器才可删除
# docker rm <容器名称/ID>
docker stop mysql
docker rm mysql
# 强制删除
docker rm -f mysql
```

3. 仓库操作

- 下载仓库镜像

```
docker pull registry
```

- 启动仓库镜像

```
docker run -id --name=registry -p 5000:5000 registry
```

- daemon.json 添加对使用镜像的信任

```
# 编辑 daemon.json 文件
vim /etc/docker/daemon.json
{
 "registry-mirrors":["http://hub.mirror.c.163.com/"],
 "insecure-registries":["192.168.2.113:5000"]
}
# 重启 docker
systemctl restart docker
# 重启私有仓库
docker start registry
```

- 提交镜像到私有镜像仓库

```
# 模板命令
# docker tag <镜像名称> <自定义tag名称>
docker tag mysql 113:5000/mysql
# 推送到私有镜像仓库
# docker push <自定义tag名称>
docker push 113:5000/mysql
```

- 仓库私有仓库

> 打开网址: http://192.168.2.113:5000/_catalog

- 下载镜像

```
docker pull 113:5000/mysql
```

3. docker 系统命令

- 容器随着docker 启动

```
# docker update <容器名称/ID> --restart=always
docker update nginx --restart=always
```

- 查看容器启动日志

```
# docker logs <容器名称/ID>
docker logs nginx
```

4. docker-compose 安装

- 下载 docker-compose

```
curl -L "https://github.com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- 添加执行权限

```
chmod +x /usr/local/bin/docker-compose
```

- 查看版本

```
docker-compose --version
```

7、安装 Nginx 环境

1. docker 方式

- 搜索镜像

```
docker search nginx
```

- 拉取镜像

```
# 默认最新版本 latest
docker pull nginx
# 查看镜像
docker images
```

- 启动 nginx 容器

```
docker run --name nginx -p 8080:80 -d nginx
# 可指定外部配置文件
# 创建文件夹
# mkdir -p /usr/local/docker/nginx/conf
# 创建 nginx 配置文件
# touch nginx.conf
# docker run --name nginx -p 8080:80 -v
/usr/local/docker/nginx/conf:/etc/nginx -d nginx
```

- 查看 ngixn 服务

```
# 查看启动的容器
docker ps
# 进入容器内部
docker exec -it nginx /bin/bash
```

- 访问 http://ip:8080

2. 普通方式

- 下载 nginx 安装包

```
nginx-1.19.10.tar.gz
```

- 解压安装包

```
tar -zxvf nginx-1.19.10.tar.gz
```

- 进入解压文件夹

```
cd nginx-1.19.10
```

- 编译执行

```
./configure
make && make install
# 查看日志可知道安装路径
# /usr/local/nginx
```

- 启动 nginx

```
cd /usr/local/nginx
cd sbin
./nginx
```

- 查看 http://ip:80
- 常见命令

```
# 重新加载配置文件
./nginx -s reload
# 重启 nginx
./nginx -s reopen
# 停止 nginx
./nginx -s stop
```

8、安装 MySQL 环境

1. docker 方式

- 搜索镜像

```
docker search mysql
```

- 拉取镜像

```
# 指定版本 5.7
docker pull mysql:5.7
# 查看镜像
docker images
```

- 启动 mysql 容器

```
docker run -itd --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root
mysql:5.7
# 可指定配置文件
# docker run --name mysql -v /usr/local/docker/mysql:/etc/mysql/conf.d -e
MYSQL_ROOT_PASSWORD=root -d mysql:5.7
```

- 查看 mysql 服务

```
docker ps
```

- 进入容器操作数据库

```
# 进入容器内部
docker exec -it mysql /bin/bash
# 连接 mysql
mysql -uroot -proot
# 查看数据库
mysql>show databases;
```

- 客户端工具连接

```
# 连接信息
ip : ip
port : 3306
username : root
password : root
```

2. 普通方式

- 下载 rpm 安装包

```
wget -i -c http://dev.mysql.com/get/mysql57-community-release-el7-
10.noarch.rpm
```

- 安装 rpm 包

```
yum -y install mysql57-community-release-el7-10.noarch.rpm
```

- 安装 mysql 服务

```
yum -y install mysql-community-server
# 安装太慢可先下载文件上传到服务器进行安装
# https://cdn.mysql.com//Downloads/MySQL-5.7/mysql-community-server-5.7.26-1.el7.x86_64.rpm
# https://cdn.mysql.com//Downloads/MySQL-5.7/mysql-community-client-5.7.26-1.el7.x86_64.rpm
# https://cdn.mysql.com//Downloads/MySQL-5.7/mysql-community-common-5.7.26-1.el7.x86_64.rpm
# https://cdn.mysql.com//Downloads/MySQL-5.7/mysql-community-libs-5.7.26-1.el7.x86_64.rpm
# 安装顺序
# common-->libs-->client-->server
# 安装命令
# rpm -ivh mysql-community-****-5.7.26-1.el7.x86_64.rpm --force --nodeps
```

- 启动 mysql 服务

```
systemctl start mysqld.service
# 或者
service mysqld start
```

- 查看 mysql 启动状态

```
systemctl status mysqld.service
# 或者
service mysqld status
```

- 连接 mysql 服务

```
# 找到初始密码
cat /var/log/mysqld.log | grep password
# A temporary password is generated for root@localhost: xxxx
mysql -uroot -pxxxx
```

- 修改 root 密码

```
# 密码设置必须要大小写字母数字和特殊符号,或者做以下设置
# policy = 0 ，仅限制密码的长度
# 修改默认密码的长度
mysql>set global validate_password_policy=0;
mysql>set global validate_password_length=4;
mysql>alter user 'root'@'localhost' identified by 'root';
```

- 允许远程连接

```
mysql>GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;
mysql>FLUSH PRIVILEGES;
```

- 查看数据库

```
mysql>show databases;
# 编码是latin
mysql>status;
```

- 修改字符编码

```
# etc目录下的my.cnf文件
vim /etc/my.cnf
# 输入以下内容
[client]
default-character-set=utf8
[mysqld]
character-set-server=utf8
collation-server=utf8_general_ci
```

- 重新 mysql 服务

```
systemctl restart mysqld.service
# 或者
service mysqld restart
# 连接再次查看编码变 utf8
```

- 客户端工具连接

```
# 连接信息
ip : ip
port : 3306
username : root
password : root
```

- 停止 mysql 服务

```
systemctl stop mysqld.service
# 或者
service mysqld stop
```

9、安装 Redis 环境

1. docker 方式

- 搜索镜像

```
docker search redis
```

- 拉取镜像

```
docker pull redis:5.0.12
```

- 启动 redis 容器

```
docker run -itd --name redis -p 6379:6379 redis:5.0.12
# 映射配置
# mkdir -p /usr/local/docker/redis/data
# mkdir -p /usr/local/docker/redis/conf
# cd /usr/local/docker/redis/conf && vim redis.conf
# #bind 127.0.0.1 //允许远程连接
# protected-mode no
# appendonly yes //持久化
# requirepass root //密码
# redis-server /etc/redis/redis.conf:以配置文件启动redis，加载容器内的conf文件
# docker run -p 6379:6379 --name redis -v
/usr/local/docker/redis/conf/redis.conf:/etc/redis/redis.conf -v
/usr/local/docker/redis/data:/data -d redis:5.0.12 redis-server
/etc/redis/redis.conf
```

- 查看 redis 服务

```
docker ps
```

- 进入容器连接 redis 服务

```
# 进入容器内部
docker exec -it redis redis-cli
# redis 密码校验
>auth root
# redis 命令
>set k1 v1
```

- 客户端连接 redis 服务

```
# 连接信息
ip : ip
port : 6379
auth : root
```

2. 普通方式

- 下载安装包

```
redis-5.0.12.tar.gz
```

- 解压缩安装包

```
tar -zxvf redis-5.0.12.tar.gz
```

- 安装

```
cd redis-5.0.12
make
```

- 启动 redis 服务

```
./src/redis-server
# 后台启动
# nohup ./src/redis-server & amp;
```

- 连接 redis 服务

```
./src/redis-cli
# 指定iphe端口命令：./src/redis-cli -h localhost -p 6379
>set k1 v1
```

- 修改配置文件

```
vim redis.conf
# protected-mode no
# appendonly yes //持久化
# requirepass root //密码
```

- 指定配置文件启动

```
# 指定配置文件后台启动
nohup ./src/redis-server ./redis.conf & amp;
# 查看进程
ps -ef | grep redis
# 客户端连接
./src/redis-cli
>auth root
>set k1 v1
```

- 客户端工具连接

```
# 连接信息
ip : ip
port : 6379
auth : root
```

- 注册 redis 服务为系统服务

```
# 进入 redis 安装目录下 utils 文件夹
./install_server.sh
# 按四次回车后，输入 redis 服务路径
/usr/local/redis-5.0.12/src/redis-server
# 成功后在 /etc/init.d 目录下出现 redis_6379 文件，修改为 redisd
mv redis_6379 redisd
```

- redis 系统服务

```
service redisd stop
service redisd start
service redisd restart
service redisd status
```

10、安装 MongoDB 环境

1. docker 方式

- 搜索镜像

```
docker search mongo
```

- 拉取镜像

```
docker pull mongo
# 查看镜像
docker images
```

- 启动 mongo 容器

```
docker run --name mongod -p 27017:27017 -d mongo
# 带密码启动
# docker run --name mongod -p 27017:27017 -d mongo --auth
```

- 查看 mongodb 服务

```
docker ps
# 查看启动日志
docker logs mongod
```

- 进入容器连接 mongodb 服务

```
docker exec -it mongod /bin/bash
# 连接 mongodb
mongo
>show dbs
>use shadow
>db.user.insert({name:"shadow",age:20})
>db.user.find()
```

- 客户端工具连接

```
# 连接信息
ip : ip
port : 27017
auth : 无
```

2. 普通方式

- 下载安装包

```
mongodb-linux-x86_64-rhel70-4.0.24.tgz
```

- 解压缩安装包

```
tar -zxvf mongodb-linux-x86_64-rhel70-4.0.24.tgz
# 修改文件夹名称
mv mongodb-linux-x86_64-rhel70-4.0.24 mongodb4.0.24
```

- 配置 mongodb 配置文件

```
cd mongodb4.0.24
# 创建文件夹
mkdir data
mkdir logs/mongodb.log
# 创建配置文件
vim mongodb.conf
```

输入以下内容

```
# 数据文件存放目录
dbpath = /usr/local/mongodb4.0.24/data
# 日志文件存放目录
logpath = /usr/local/mongodb4.0.24/logs/mongodb.log
# 端口
port = 27017
# 以守护程序的方式启用，即在后台运行
fork = true
# nohttpinterface = true
#[建议练习条件下为false 认证字段]
auth=false
bind_ip=0.0.0.0
```

> --dbpath 数据库路径(数据文件)
>
> --logpath 日志文件路径
>
> --master 指定为主机器
>
> --slave 指定为从机器
>
> --source 指定主机器的IP地址
>
> --pologSize 指定日志文件大小不超过64M.因为resync是非常操作量大且耗时，最好通过设置一个足够大的oplogSize来避免resync(默认的 oplog大小是空闲磁盘大小的5%)。
>
> --logappend 日志文件末尾添加，即使用追加的方式写日志
>
> --journal 启用日志
>
> --port 启用端口号
>
> --fork 在后台运行
>
> --only 指定只复制哪一个数据库
>
> --slavedelay 指从复制检测的时间间隔
>
> --auth 是否需要验证权限登录(用户名和密码)
>
> --syncdelay 数据写入硬盘的时间（秒），0是不等待，直接写入
>
> --notablescan 不允许表扫描
>
> --maxConns 最大的并发连接数，默认2000
>
> --pidfilepath 指定进程文件，不指定则不产生进程文件 --bind_ip 绑定IP，绑定后只能绑定的IP访问服务

- 启动 mongodb 服务

```
./bin/mongod -f ./mongodb.conf
```

- 客户端连接

```
# 指定ip端口 ./bin/mongo --host localhost --port 27017
./bin/mongo
>show dbs
>use shadow
>db.user.insert({name:"shadow",age:20})
>db.user.find()
```

- 客户端工具连接

```
# 连接信息
ip : ip
port : 27017
auth : 无
```

11、安装 ElasticSearch 环境

1. docker 方式

- 搜索镜像

```
docker search elasticsearch
```

- 拉取镜像

```
docker pull elasticsearch:7.12.1
# 查看镜像
docker images
```

- 启动 es 容器

```
docker run -d --name elasticsearch -p 9200:9200 -p 9300:9300 -e
ES_JAVA_OPTS="-Xms512m -Xmx512m" -e "discovery.type=single-node"
elasticsearch:7.12.1
```

- 测试

```
curl http://10.0.0.250:9200
```

- 配置 es 配置文件

```
mkdir -p /usr/local/docker/es/conf
mkdir -p /usr/local/docker/es/data
mkdir -p /usr/local/docker/es/plugins
vim /usr/local/docker/es/conf/elasticsearch.yml
# 输入以下内容
cluster.name: "docker-cluster"
network.host: 0.0.0.0
http.cors.enabled: true
http.cors.allow-origin: "*"
# 目录权限设置
chmod 777 -R /usr/local/docker/es
```

- 重新启动容器

```
# 停止旧容器
docker stop elasticsearch
# 删除旧容器
docker rm ealsticsearch
# 重新创建并启动容器
docker run --name elasticsearch -p 9200:9200 -p 9300:9300 \
-e "discovery.type=single-node" \
-e ES_JAVA_OPTS="-Xms512m -Xmx512m" \
-v
/usr/local/docker/es/conf/elasticsearch.yml:/usr/share/elasticsearch/config/
elasticsearch.yml \
-v /usr/local/docker/es/data:/usr/share/elasticsearch/data \
-v /usr/local/docker/es/plugins:/usr/share/elasticsearch/plugins \
-d elasticsearch:7.12.1
```

- 安装 ik 分词器

```
# docker 方式，切换到 plugins 映射的目录
cd /usr/local/docker/es/plugins
wget https://github.com/medcl/elasticsearch-analysis-
ik/releases/download/v7.12.1/elasticsearch-analysis-ik-7.12.1.zip
# 解压缩
unzip elasticsearch-analysis-ik-7.12.1.zip -d ik
# 删除压缩包
rm -rf elasticsearch-analysis-ik-7.12.1.zip
# 修改ik目录权限
chmod -R 777 ik/
# 启动 elasticsearch 服务
docker start elasticsearch
# 查看启动日志
docker logs elasticsearch
```

2. 普通方式

- 下载 elasticsearch 压缩包
- 解压缩并修改目录名称

```
tar -zxvf elasticsearch-7.12.1-linux-x86_64.tar.gz
mv elasticsearch-7.12.1 elasticsearch
```

- 配置 elasticsearch

```
cd elasticsearch/config
# 编辑 elasticsearch 配置文件
vim elasticsearch.yml
# 输入以下内容
cluster.name: es
node.name: node-1
path.data: /usr/local/elasticsearch/data
path.logs: /usr/local/elasticsearch/log
network.host: 0.0.0.0
http.port: 9200
# 跨域配置
http.cors.enabled: true
http.cors.allow-origin: "*"
cluster.initial_master_nodes: ["node-1"]
```

- 配置 jvm 参数

```
vim jvm.options
# 修改内存配置
-Xms512m
-Xmx512m
```

- 修改宿主机配置

```
# 到宿主机上打开文件
vim /etc/sysctl.conf
# 增加这样一条配置，一个进程在VMAs(虚拟内存区域)创建内存映射最大数量
vm.max_map_count=655360
# 让配置生效
sysctl -p
```

- 启动 elasticsearch 服务

```
# 修改目录权限
chmod -R 777 /usr/local/elasticsearch
# 创建用户
adduser es
# 切换用户
su es
# 启动 elasticsearch 服务
./bin/elasticsearch
```

- 检查 elasticsearch 服务

```
curl localhost:9200
# 或者 浏览器输入地址
http://10.0.0.250:9200
```

> 可能出现的问题解决方案
>
> - X-Pack is not supported and Machine Learning is not available for [windows-x86]; you can use the other X-Pack features (unsupported)

```
# vim elasticsearch.yml
xpack.ml.enabled: false
```

- java.lang.RuntimeException: can not run elasticsearch as root

```
# 切换用户运行 es 服务
# 目录权限修改
chmod -R 777 /usr/local/elasticsearch
su es
```

- max file descriptors [4096] for elasticsearch process is too low

```
# 最大线程数设置的太低了，需要改成4096
vi /etc/security/limits.conf
# ElasticSearch添加如下内容:
    * soft nofile 65536
    * hard nofile 131072
    * soft nproc 2048
    * hard nproc 4096
```

- max number of threads [1024] for user [elsearch] is too low

```
# Centos6不支持SecComp，而ES5.2.0默认bootstrap.system_call_filter为true
# 解决：切换到root用户，进入limits.d目录下修改配置文件。
vi /etc/security/limits.d/90-nproc.conf
# 修改如下内容:
    * soft nproc 1024
# 修改为
    * soft nproc 4096
```

- system call filters failed to install; check the logs and fix your configuration

```
vim config/elasticsearch.yml
# 添加
bootstrap.system_call_filter: false
bootstrap.memory_lock: false
```

- 安装 ik 分词器

```
# 普通方式 ，切换到安装目录 plugins 目录下
cd /usr/local/elasticsearch/plugins
```

- 下载 ik 分词器解压缩

```
wget https://github.com/medcl/elasticsearch-analysis-
ik/releases/download/v7.12.1/elasticsearch-analysis-ik-7.12.1.zip
# 解压缩
unzip elasticsearch-analysis-ik-7.12.1.zip -d ik
# 删除压缩包
rm -rf elasticsearch-analysis-ik-7.12.1.zip
# 修改ik目录权限
chmod -R 777 ik/
```

- 重启 elasticsearch 服务

```
./bin/elasticsearch &
```

- 自定义分词器

```
# 进入到 ik 分词器
cd /usr/local/elasticsearch/plugins/ik/config
# 编写文件 customer.dic
echo "乔碧萝" > customer.dic
echo "社死" >> customer.dic
echo "乔碧萝殿下" >> customer.dic
# 编辑 IKAnalyzer.cfg.xml
vim IKAnalyzer.cfg.xml
# 配置以下内容 customer.dic
<entry key="ext_dict">customer.dic</entry>
```

- 重启 elasticsearch 服务
- Postman 测试效果

```
POST  10.25.175.108:9200/shopping/_analyze
{
 "analyzer":"ik_smart",
 "text":"乔碧罗殿下社死"
}
```

12、安装 Kibana 环境

1. docker 方式

- 搜索镜像

```
docker search kibana
```

- 拉取镜像

```
docker pull kibana:7.12.1
```

- 启动 kibana 容器

```
docker run --name kibana \
-e ELASTICSEARCH_HOSTS=http://10.0.0.250:9200 \
-p 5601:5601 \
-d kibana:7.12.1
```

- 访问 http://10.0.0.250:5601
- 配置 kibana

```
# 创建 kibana 映射文件夹
mkdir -p /usr/local/docker/kibana/conf
```

- 改变权限

```
chmod -R 777 /usr/local/docker/kibana
```

- 创建 kibana.yml 配置文件

```
 vim /usr/local/docker/kibana/conf/kibana.yml
```

输入以下内容

```
# kibana 服务名称
 server.name: kibana
# 对外暴露服务的地址 -宿主机IP
 server.host: "0.0.0.0"
# es 服务
# 有些版本是 elasticsearch.url
 elasticsearch.hosts: ["http://10.0.0.250:9200"]
```

- 重新创建容器

```
# 停止旧容器
docker stop kibana
# 删除旧容器
docker rm kibana
# 启动新容器
docker run --name kibana \
-p 5601:5601 \
-v /usr/local/docker/kibana/conf:/usr/share/kibana/config \
-d kibana:7.12.1
```

2. 普通方式

- 下载安装包

```
kibana-7.12.1-linux-x86_64.tar.gz
```

- 解压缩安装包

```
tar -zxvf kibana-7.12.1-linux-x86_64.tar.gz
# 修改文件夹名称
mv kibana-7.12.1-linux-x86_64 kibana
```

- 配置 kibana

```
cd kibana/config
# 编辑配置文件
vim kibana.yml
# 修改以下内容
# kibana 对外暴露服务地址 - 主机IP
server.host: "10.0.0.250"
# es 服务
elasticsearch.hosts: ["http://10.0.0.250:9200"]
```

- 启动 kibana 服务

```
# 改变目录权限
chmod -R 777 /usr/local/kibana
# 切换用户
# 没有用户就创建用户
su es
# 启动服务
./bin/kibana
```

- 访问 http://10.0.0.250:5601

13、安装 Logstash 环境

1. docker 方式

- 搜索镜像

```
docker search logstash
```

- 拉取镜像

```
docker pull logstash:7.12.1
```

- 启动 logstash 容器

```
docker run --name logstash -p 5044:5044 -p 9600:9600 -d logstash:7.12.1
```

- 配置 logstash

```
# 创建 logstash 映射目录
mkdir -p /usr/local/docker/logstash
# 拷贝容器内文件到宿主机
docker cp logstash:/usr/share/logstash/config/
/usr/local/docker/logstash/config /usr/local/docker/logstash/conf
# 重命名config
mv /usr/local/docker/logstash/config /usr/local/docker/logstash/conf
# 配置 logstash 配置文件
vim /usr/local/docker/logstash/conf/logstash.yml
# 输入以下内容
http.host: "0.0.0.0"
xpack.monitoring.elasticsearch.hosts: [ "http://10.0.0.250:9200" ]
# 修改 jvm 参数
vim /usr/local/docker/logstash/conf/jvm.options
# 修改参数
-Xms512m
-Xmx512m
```

- 重新创建容器启动

```
# 停止旧容器
docker stop logstash
# 删除旧容器
docker rm logtash
# 创建新容器启动
docker run --name logstash \
-p 5044:5044 \
-p 9600:9600 \
-v /usr/local/docker/logstash/conf:/usr/share/logstash/config \
-d logstash:7.12.1
```

2. 普通方式

- 下载安装包

```
logstash-7.12.1-linux-x86_64.tar.gz
```

- 解压缩安装包

```
tar -zxvf logstash-7.12.1-linux-x86_64.tar.gz
# 文件夹重命名
mv logstash-7.12.1 logstash
```

- 启动 logstash 服务

```
cd logstash
# 启动 logstash 服务
./bin/logstash -e 'input { stdin { } } output { stdout {} }'
# 启动完成输入任意字符出现如下类似效果
{
          "host" => "soft",
      "@version" => "1",
    "@timestamp" => 2021-05-04T03:41:36.421Z,
       "message" => "hello"
}
```

- logstash 的配置格式

```
input { #输入
    stdin { ... } #标准输入
}
filter { #过滤，对数据进行分割、截取等处理
    ...
}
output { #输出
    stdout { ... } #标准输出
}
```

- 测试

```
# 编写配置文件
vim shadow-pipeline.conf
```

```
# 输入以下内容
input {
    file {
        path => "/usr/local/app/logs/app.log"
        start_position => "beginning"
    }
}
filter {
        mutate {
            split => {"message"=>"|"}
        }
    }
# 输出到控制台
output {
    stdout { codec => rubydebug }
}
```

启动服务测试

```
./bin/logstash -f shadow-pipeline.conf
# 输入如下格式的日志
"2021-05-04 21:21:21|ERROR|读取数据出错|参数：id=1001"
# 日志结构
echo "2021-05-04 21:21:21|ERROR|读取数据出错|参数：id=1001" >>
/usr/local/app/logs/app.log
# 输出结果
{
          "path" => "/usr/local/app/logs/app.log",
      "@version" => "1",
       "message" => [
        [0] "-05-04 21:21:21",
        [1] "ERROR",
        [2] "读取数据出错",
        [3] "参数：id=1001"
    ],
    "@timestamp" => 2021-05-04T05:26:47.842Z,
          "host" => "soft"
}
```

```
# 输出到 es，修改shadow-pipeline.conf
output {
    elasticsearch {
        hosts => ["10.0.0.250:9200"]
    }
}
```

- 打开网址查看

```
http://10.0.0.250:9200/logstash-2021.05.04-000001/_search
```

14、安装 Beats 环境

1. docker 方式

- 搜索镜像
- 拉取镜像
- 启动 beats容器

2. 普通方式

15、安装 Zookeeper 环境

1. docker 方式

- 搜索镜像

```
docker search zookeeper
```

- 拉取镜像

```
docker pull zookeeper:3.7
# 查看镜像
docker images
```

- 启动 zk 容器

```
docker run -it  --name zk01 -d zookeeper:3.7
```

- 连接 zk

```
# 进入容器内部
docker exec -it zk01 bash
# 客户端连接
zkCli.sh
# zk 命令
>ls /
>create /shadow
>set /shadow k1
```

- docker-compose

```
vim stack.yml
```

输入以下内容

```
version: '3.7'

services:
  zoo1:
    image: zookeeper:3.7
    restart: always
    hostname: zoo1
    ports:
      - 2181:2181
    environment:
      ZOO_MY_ID: 1
```

```
      ZOO_SERVERS: server.1=0.0.0.0:2888:3888;2181
server.2=zoo2:2888:3888;2181 server.3=zoo3:2888:3888;2181

  zoo2:
    image: zookeeper:3.7
    restart: always
    hostname: zoo2
    ports:
      - 2182:2181
    environment:
      ZOO_MY_ID: 2
      ZOO_SERVERS: server.1=zoo1:2888:3888;2181
server.2=0.0.0.0:2888:3888;2181 server.3=zoo3:2888:3888;2181

  zoo3:
    image: zookeeper:3.7
    restart: always
    hostname: zoo3
    ports:
      - 2183:2181
    environment:
      ZOO_MY_ID: 3
      ZOO_SERVERS: server.1=zoo1:2888:3888;2181 server.2=zoo2:2888:3888;2181
server.3=0.0.0.0:2888:3888;2181
```

启动集群

```
docker-compose -f stack.yml up
```

查看容器启动情况

```
docker ps
```

2. 普通方式

- 下载安装包

```
apache-zookeeper-3.7.0-bin.tar.gz
```

- 解压缩安装包

```
#进入到指定命令
cd /usr/local/zk
# 解压
tar -zxvf apache-zookeeper-3.7.0-bin.tar.gz
# 重命名
mv apache-zookeeper-3.7.0-bin zk
```

- 配置 zk 服务

```
cd zk
# 创建数据和日志命令
mkdir data && mkdir logs
# 进入 conf 文件夹
cd conf
# 复制 zoo.cfg 配置文件
cp zoo_sample.cfg zoo.cfg
# 配置 zk
vim zoo.cfg
# 配置以下内容
dataDir=/usr/local/zk/zk/data
dataLogDir=/usr/local/zk/zk/logs
```

- 启动 zk 服务

```
cd ..
./bin/zkServer.sh start
```

- 连接 zk 服务

```
./bin/zkCli.sh
# zk 命令
>ls /
>create /shadow
>set /shadow k1
>get /shadow
>quit
```

- zk 集群

```
# 复制 zk 三份到 zk01 zk02 zk03
cp -r zk zk01 && cp -r zk zk02 && cp -r zk zk03
# 修改三个副本的配置文件
# zk01配置，data和logs目录有记录先删除目录下的文件
cd zk01
# 修改配置文件
vim conf/zoo.cfg
# 配置以下内容
dataDir=/usr/local/zk/zk01/data
dataLogDir=/usr/local/zk/zk01/logs
clientPort=2187
server.0=10.0.0.250:2887:3887
server.1=10.0.0.250:2888:3888
server.2=10.0.0.250:2889:3889
# data 目录下创建 myid
echo "0" > data/myid
# zk02 配置
cd ../zk02
vim conf/zoo.cfg
dataDir=/usr/local/zk/zk02/data
dataLogDir=/usr/local/zk/zk02/logs
clientPort=2188
server.0=10.0.0.250:2887:3887
server.1=10.0.0.250:2888:3888
server.2=10.0.0.250:2889:3889
```

```
echo "1" > data/myid
# zk03
cd ../zk03
vim conf/zoo.cfg
dataDir=/usr/local/zk/zk03/data
dataLogDir=/usr/local/zk/zk03/logs
clientPort=2189
server.0=10.0.0.250:2887:3887
server.1=10.0.0.250:2888:3888
server.2=10.0.0.250:2889:3889
echo "2" > data/myid
```

- 启动集群

```
cd /usr/local/zk
./zk01/bin/zkServer.sh start
./zk02/bin/zkServer.sh start
./zk03/bin/zkServer.sh start
```

- 查看集群启动状态

```
./zk01/bin/zkServer.sh status
./zk02/bin/zkServer.sh status
./zk03/bin/zkServer.sh status
```

- 客户端连接

```
./zk01/bin/zkCli.sh -server localhost:2188
```

16、安装 Etcd 环境

1. docker 方式

2. 普通方式

17、安装 gitLab 环境

1. docker 方式

- 搜索镜像

```
docker search gitlab-ce
```

- 拉取镜像

```
docker pull gitlab/gitlab-ce
# 查看镜像
docker images
```

- 启动容器

```
# 创建需要的文件夹
mkdir -p /usr/local/docker/gitlab/log
mkdir -p /usr/local/docker/gitlab/opt
mkdir -p /usr/local/docker/gitlab/etc
# 启动容器
docker run \
 -itd  \
 -p 10080:80 \
 -p 10022:22 \
 -v /usr/local/docker/gitlab/etc:/etc/gitlab  \
 -v /usr/local/docker/gitlab/log:/var/log/gitlab \
 -v /usr/local/docker/gitlab/opt:/var/opt/gitlab \
 --restart always \
 --privileged=true \
 --name gitlab \
 gitlab/gitlab-ce
```

- 访问 http://10.0.0.250:10080

2. 普通方式

- 下载 rpm 包

```
gitlab-ce-13.0.9-ce.0.el7.x86_64.rpm
```

- 安装

```
# 依赖安装
yum install -y curl openssh-server postfix cronie
yum -y install policycoreutils-python
yum install -y openssh-server openssh-clients postfix
# 开启ssh服务&设置开机自启
systemctk enable sshd && sudu systemctl start sshd
# 设置postfix开机自启，postfix支持gitlab发信功能
systemcl enable postfix && systemctl start postfix
# 开启ssh以及http服务，然后重新加载防火墙列表
firewall-cmd --add-service=ssh --permanent
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
# 安装 gitlab-ce
rpm -ivh gitlab-ce-13.0.9-ce.0.el7.x86_64.rpm
```

- 修改 gitlab 的配置

```
vim /etc/gitlab/gitlab.rb
# 修改内容
external_url 'http://10.0.0.190:8888'
nginx['listen_port'] = 8888
# 配置生效
gitlab-ctl reconfigure
gitlab-ctl restart
```

- 把端口添加到防火墙

```
firewall-cmd --zone=public --add-port=8888/tcp --permanent
firewall-cmd --reload
```

- 访问地址 http://10.0.0.190:8888
- 忘记 root 密码找回

```
# 进入 gitlab 控制台
su - git
gitlab-rails console
# 修改密码
>user = User.where(id:1).first
>user.password='12345678'
>user.password_confirmation='12345678'
>user.save!
```

- 查看gitlab的数据库

```
# 数据库配置文件
cat /var/opt/gitlab/gitlab-rails/etc/database.yml
# 登录
su - gitlab-psql
# 连接
psql -h /var/opt/gitlab/postgresql -d gitlabhq_production
# 测试
# 帮助命令
\h
# 查看数据库
\l
# 查看多表
\dt
# 查看表单
\d abuse_reports
# 查看索引
\di
# 查看表空间
SELECT spcname FROM pg_tablespace;
# 退出
\q
```

18、安装 Jenkins 环境

1. docker 方式

- 搜索镜像

```
docker search jenkins
```

- 拉取镜像

```
docker pull jenkins
# 查看镜像
docker images
```

- 启动

```
docker run --name jenkins -p 8080:8080 -p 50000:50000 -d jenkins
```

- 访问 http://10.0.0.250:8080

```
# 获取密码
docker exec -it jenkins bash
cat /var/jenkins_home/secrets/initialAdminPassword
```

2. 普通方式

- 下载 rmp 安装包

```
jenkins-2.279-1.1.noarch.rpm
```

- 安装

```
# 需要 JDK 环境
yum install -y java-1.8.0-openjdk*
# 安装 Jenkins
rpm -ivh jenkins-2.279-1.1.noarch.rpm
```

- 启动

```
service jenkins start
```

- 访问 http://10.0.0.200:8080
- 获取 jenkins 账号的密码

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

- 配置文件

```
vim /etc/sysconfig/jenkins
# 修改用户
JENKINS_USER="root"
JENKINS_PORT="8888"
```

- 重启 Jenkins 服务

```
systemctl restart jenkins
```

- 插件安装优化

```
# default.json 文件位置
/var/lib/jenkins/updates/default.json
# 切换路径
cd /var/lib/jenkins/updates
# google替换baidu
sed -i
's/http:\/\/updates.jenkinsci.org\/download/https:\/\/mirrors.tuna.tsinghua.
edu.cn\/jenkins/g' default.json && sed -i
's/http:\/\/www.google.com/https:\/\/www.baidu.com/g' default.json
# jenkins 网站打开配置镜像
https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json
# 重启服务
http://10.0.0.200:8888/restart
```

- 重要插件

```
# 中文汉化
localization-zh-cn.hpi
# 用户权限管理
Role-based Authorization Strategy
# 凭证管理功能
Credentials Binding
# git 插件
Git
# Deploy to container - tomcat 容器部署
Deploy to container
# 项目构建方式，默认只支持 freestyle方式，Maven Integration支持 maven方式构建项目
Maven Integration
# 支持流水线构建方式
Pipeline
# 构建触发器 Git hook
# 默认四种：
# 1. 触发远程构建
# 2. 其他工程构建触发 Build after other projects are built
# 3. 定时触发 Build periodically
# 4. 定时扫描仓库代码情况有修改触发 Poll SCM
Gitlab Hook、GitLab
# 邮件服务
Email Extension Template
# 代码审计
SonarQube Scanner
# 远程调用 Publish Over SSH
Publish Over SSH
# NodeJS环境 支持前端发布
NodeJS
# 参数复选 - 多项目复选框
Extended Choice Parameter
# k8s
Kubernetes
Kubernetes Continuous Deploy
```

- war包位置

```
# /usr/lib/jenkins
cd /usr/lib/jenkins
```

- 忘记账号密码

```
# 普通用户存放
cd /var/lib/jenkins/users
# 进入用户目录
cd shadow_114062620393xxx
# 备份 config.xml 文件
cp config.xml config.xml.bak
# 修改内容
vim config.xml
# 修改内容如下,对于明文密码 123456
<passwordHash>#jbcrypt:$2a$10$MiIVROrr/UhQBqT.bBqOQehTiQVqgNpUGyWW2nJObaVAM/
2xSQdSq</passwordHash>
```

```
# 重启 Jenkins 服务
systemctl restart jenkins
```

- 卸载 jenkins

```
systemctl stop jenkins.service
rpm -e jenkins
rpm -qa | grep jenkins      # 查看是否还有jenkins依赖，有就删除
rm -rf /etc/sysconfig/jenkins.rpmsave
rm -rf /var/cache/jenkins/
rm -rf /var/lib/jenkins/
rm -rf /var/log/jenkins
rm -rf /usr/lib/jenkins
```

- ssh 认证

```
>ssh-keygen -t rsa
>cd .ssh
>ls -a
```

19、安装 环境

1. docker 方式

2. 普通方式