

pdp-cloud 部署说明

目录

一、 环境软件要求	2
二、 系统最低要求	2
三、 部署包结构	2
四、 中间件安装	2
1. JDK 安装	2
2. Maven 安装	4
3. Mysql 安装	5
4. Redis 安装	8
5. Minio 安装	11
6. RocketMQ 安装	12
7. Xxl-job 安装	16
8. Linux 开机启动设置	18
9. 配置防火墙	20
五、 应用服务端中间件安装	21
1. 安装 JDK 【同上】	21
2. 安装 Maven 【同上】	21
3.kkfileView 安装	21
六、 后端部署	22
1. 导入 sql	22
2.安装 Nacos	22
4.部署 jar	25
七、 前端部署	26
1. 上传前端代码包	26
2. 安装 Nginx	27
八、 系统授权	31

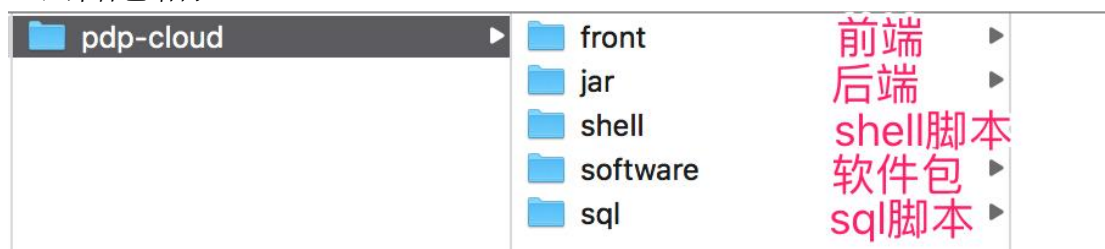
一、环境软件要求

服务器系统	Linux	
JDK	1.8	pdp-cloud/software/jdk-8u221-linux-x64.tar.gz
Mysql	5.7 以上	pdp-cloud/software/mysql-5.7.25-linux-glibc2.12-x86_64.tar.gz
Maven	3.6	pdp-cloud/software/apache-maven-3.6.2-bin.tar.gz
Redis	5	pdp-cloud/software/redis-5.0.5.tar.gz
RocketMQ	4.9	pdp-cloud/software/rocketmq-all-4.9.3-bin-release.zip pdp-cloud/software/rocketmq-console-ng-1.0.1.jar
Minio	最新	pdp-cloud/software/minio
Nginx	1.17.9	pdp-cloud/software/nginx-1.17.9.tar.gz
kkfileView	4.0	pdp-cloud/software/Apache_OpenOffice_4.1.12_Linux_x86-64_install-rpm_zh-CN.tar.gz pdp-cloud/software/kkFileView-4.0.0.tar.gz
Xxl-job	2.3.0	pdp-cloud/software/xxl-job-2.3.0.tar.gz pdp-cloud/sql/pdp_job.sql pdp-cloud/shell/中间件服务脚本/xxljob-depoly.sh

二、系统最低要求

服务器类型	操作系统	数量	处理器	内存	存储	应用描述
X86 server	CentOS 7.5(64bit)	1	8 core*2.4G	16G	1T	中间件
X86 server	CentOS 7.5(64bit)	1	8 core*2.4G	64G	1T	应用服务

三、部署包结构



四、中间件安装

1. JDK 安装

第一步：删除存在的 openJDK

```
java -version

[root@localhost opt]# java -version
openjdk version "1.8.0_322"
OpenJDK Runtime Environment (build 1.8.0_322-b06)
OpenJDK 64-Bit Server VM (build 25.322-b06, mixed mode)
```

2、检测 jdk 安装包

```
rpm -qa | grep java
```

3、卸载 openjdk

```
rpm -qa | grep java | xargs rpm -e --nodeps
```

查看卸载情况:

```
rpm -qa | grep java
```

第二步：安装新的 JDK

4、安装新的 jdk

创建文件夹

```
mkdir -p /usr/lib/jvm
```

上传并解压

```
cd /opt
```

```
tar -zxvf jdk-8u221-linux-x64.tar.gz -C /usr/lib/jvm
```

5、设置环境变量

```
vim /etc/profile
```

添加到最后位置

注意 jdk 路径

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_221
```

```
export JRE_HOME=${JAVA_HOME}/jre
```

```
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
```

```
export PATH=${JAVA_HOME}/bin:$PATH
```

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_221
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib

export MAVEN_HOME=/opt/maven
export PATH=${JAVA_HOME}/bin:$PATH:$MAVEN_HOME/bin
```

6、执行 profile 文件

```
source /etc/profile
```

7、检查新安装的 jdk

```
java -version
```

返回:

```
java version "1.8.0_221"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

2. Maven 安装

1.创建目录

```
mkdir -p /opt/maven
```

上传并解压:

```
cd opt
```

```
tar -xzf apache-maven-3.6.2-bin.tar.gz
```

移动文件:

```
mv apache-maven-3.6.2/* /opt/maven
```

2.配置环境变量

```
vim /etc/profile
```

添加: 注意 Maven 路径

```
export MAVEN_HOME=/opt/maven
```

```
export PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin
```

```
#JDK
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_221
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
#export PATH=${JAVA_HOME}/bin:$PATH

#Maven
export MAVEN_HOME=/opt/maven
export PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin
```

配置生效:

```
source /etc/profile
```

查找 Maven 版本:

```
mvn -v
```

3.maven 配置

创建本地仓库目录

```
mkdir -p /opt/repo
```

修改 Maven 的 settings.xmls

```
vim /opt/maven/conf/settings.xml
```

添加仓库:

```
<localRepository>/opt/repo</localRepository>
```

```

|<!-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- localRepository
    | The path to the local repository maven will use to store artifacts.
    |
    | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->

  <localRepository>/opt/repo</localRepository>

  <!-- interactiveMode
    | This will determine whether maven prompts you when it needs input. If set to false,
    | maven will use a sensible default value, perhaps based on some other setting, for
  -->

```

假如有私服可以添加私服:

示例: 添加阿里云私服地址:

```

<mirror>
  <id>alimaven</id>
  <mirrorOf>central</mirrorOf>
  <name>aliyun maven</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
</mirror>

```

3. Mysql 安装

1. 安装

cd /opt

上传并解压

tar -xzf mysql-5.7.25-linux-glibc2.12-x86_64.tar.gz

重命名

mv mysql-5.7.25-linux-glibc2.12-x86_64 mysql

创建 data 目录

cd /opt/mysql

mkdir data

创建用户组以及用户和密码

groupadd mysql

useradd -g mysql mysql

授权用户

chown -R mysql:mysql /opt/mysql

切换到 bin 目录下

cd /opt/mysql/bin

初始化数据库

```

./mysqld --user=mysql --basedir=/opt/mysql --datadir=/opt/mysql/data/ --initialize
--lower-case-table-names=1

```

初始化命令执行后, 在最后会显示初始密码, 记得一定要先保存下来, 第一次登录需要使用。

初始化参数说明:

【--user=mysql】: 安装 mysql, 用户身份是 mysql 用户组。

【--basedir=/usr/local/mysql】: 指定了安装 MySQL 的安装路径。

【--datadir=/usr/local/mysql/data/】: 指定了 MySQL 的数据库文件放在什么路径下。

【--initialize】: 初始化。

【--lower-case-table-names=1】: 是否数据目录所在的文件系统对文件名的大小写敏感。

0-大小写敏感; 1-大小写不敏感; 2-大小写不敏感

2. 编辑 my.cnf 文件

vim /etc/my.cnf

内容全部替换

配置: 参考部署包: [pdp-cloud/shell/中间件服务脚本/mysql/my.cnf](#)

```
[mysql]
default-character-set = utf8mb4

[mysqld]
port = 19001

basedir=/opt/mysql
datadir=/opt/mysql/data
socket=/opt/mysql/mysql.sock
character-set-server=UTF8MB4

default-storage-engine=INNODB
lower_case_table_names = 1
max_connections=2000
symbolic-links=0
sync_binlog=1000
innodb_buffer_pool_size=1G
innodb_flush_log_at_trx_commit=2

log-error=/opt/mysql/data/mysqld-error.log
pid-file=/opt/mysql/data/mysqld.pid
```

```
[mysql]
default-character-set = utf8mb4

[mysqld]
port = 19001

basedir=/opt/mysql
datadir=/opt/mysql/data
socket=/opt/mysql/mysql.sock
character-set-server=UTF8MB4
default-storage-engine=INNODB
lower_case_table_names = 1
max_connections=2000

symbolic-links=0

sync_binlog=1000
innodb_buffer_pool_size=1G
innodb_flush_log_at_trx_commit=2

log-error=/opt/mysql/data/mysqld-error.log
pid-file=/opt/mysql/data/mysqld.pid
```

3.添加 mysql 服务到系统服务中

```
cd /opt/mysql
```

```
cp -a ./support-files/mysql.server /etc/init.d/mysql
```

授权以及添加服务

```
chmod +x /etc/init.d/mysql
```

```
chkconfig --add mysql
```

4.启动 mysql

```
service mysql start
```

#停止

```
#service mysql stop
```

查看启动状态

```
service mysql status
```

将 mysql 命令添加到服务

```
ln -s /opt/mysql/bin/mysql /usr/bin
```

5.设置密码

登录, mysql 密码, 使用之前初始密码

```
mysql -uroot -p
```

回车 输入初始密码

[报错: Cannot connect to local MYSQL server through socket '/tmp/mysql.sock'(2)]

[解决:设置软连接: ln -s /opt/mysql/mysql.sock /tmp/mysql.sock]

重新登陆连接

修改【root】用户的密码

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

执行【flush privileges;】使密码立即生效

```
flush privileges;
```

6.设置远程访问

选择 mysql 数据库

```
use mysql;
```

报错:

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

修改远程连接并立即生效

```
update user set host='%' where user='root';
```

```
flush privileges;
```

选择 mysql 数据库

```
use mysql;
```

成功列出数据库

退出连接

```
exit
```

4. Redis 安装

1.安装

```
cd /opt
```

上传并解压

```
tar -zxvf redis-5.0.5.tar.gz
```

更改文件夹名称为 redis

```
mv redis-5.0.5 redis
```

编译

```
cd /opt/redis
```

```
make
```

安装

```
make install PREFIX=/opt/redis
```


prefix 这个关键字的作用是编译的时候用于指定程序存放的路径。

2.启动 redis

此时未修改 redis 任何配置文件，默认连接 redis 密码为空。

```
cd /opt/redis
```

```
./bin/redis-server& ./redis.conf
```

查看 Redis 是否正在运行

ctrl+c 退出日志

(1)采取查看进程方式

```
ps -aux|grep redis
```

(2)采取端口监听查看方式

```
netstat -lanp|grep 6379
```

3.redis 配置

修改配置文件

```
cd /opt/redis
```

```
vim redis.conf
```

配置: 参考部署包: [pdp-cloud/shell/中间件服务脚本/redis/redis.conf](#)

操作:

1).注释掉 bind 127.0.0.1

```
# Examples:
#
# bind 192.168.1.100 10.0.0.1
# bind 127.0.0.1 ::1
#
# ~~~ WARNING ~~~ If the computer running Redis is directly exposed to the
# internet, binding to all the interfaces is dangerous and will expose the
# instance to everybody on the internet. So by default we uncomment the
# following bind directive, that will force Redis to listen only into
# the IPv4 loopback interface address (this means Redis will be able to
# accept connections only from clients running into the same computer it
# is running).
#
# IF YOU ARE SURE YOU WANT YOUR INSTANCE TO LISTEN TO ALL THE INTERFACES
# JUST COMMENT THE FOLLOWING LINE.
# ~~~~~
#bind 127.0.0.1
#
# Protected mode is a layer of security protection, in order to avoid that
# Redis instances left open on the internet are accessed and exploited.
```

2).保护模式修改: protected-mode 的值默认为 yes, 将其修改为 no

```
# "bind" directive.
# 2) No password is configured.
#
# The server only accepts connections from clients connecting from the
# IPv4 and IPv6 loopback addresses 127.0.0.1 and ::1, and from Unix domain
# sockets.
#
# By default protected mode is enabled. You should disable it only if
# you are sure you want clients from other hosts to connect to Redis
# even if no authentication is configured, nor a specific set of interfaces
# are explicitly listed using the "bind" directive.
protected-mode no
```

3).守护进程修改: daemonize 的值默认为 no, 将其修改为 yes
#yes 表示启用守护进程, 默认是 no 即不以守护进程方式运行

```
##### GENERAL #####
# By default Redis does not run as a daemon. Use 'yes' if you need it.
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
daemonize yes
```

4).redis 密码修改:requirepass 放开注释
#设置 redis 连接密码:witroot

```
##### SECURITY #####
# Require clients to issue AUTH <PASSWORD> before processing any other
# commands. This might be useful in environments in which you do not trust
# others with access to the host running redis-server.
#
# This should stay commented out for backward compatibility and because most
# people do not need auth (e.g. they run their own servers).
#
# Warning: since Redis is pretty fast an outside user can try up to
# 150k passwords per second against a good box. This means that you should
# use a very strong password otherwise it will be very easy to break.
#
requirepass witroot
# Command renaming.
```

5).修改访问端口:port=19003

```
# Accept connections on the specified port, default is 6379 (IANA #815)
# If port 0 is specified Redis will not listen on a TCP socket.
port 19003
```

4.重启

ps -ef|grep redis

kill -9 进程号

cd /opt/redis

./bin/redis-server ./redis.conf

因设置了守护进程, 此操作同采取后台进程方式

5.验证

cd /opt/redis/bin

#连接

./redis-cli -h ip 地址 -p 端口 -a 密码

```
[root@localhost bin]#  
[root@localhost bin]# ./redis-cli -h 192.168.196.190 -p 19003 -a witroot  
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.  
192.168.196.190:19003>  
192.168.196.190:19003>  
192.168.196.190:19003>
```

[redis 默认 16 个库,单节点需要指定数据库, 数据库之间隔离不共享, 切换: select 数字]
退出

exit

停止

ps aux | grep redis | grep -v grep | awk '{print \$2}' | xargs kill -9

5. Minio 安装

1.安装

cd /opt

mkdir minio

cd minio

将 minio 文件上传进文件夹

#创建配置文件夹

mkdir config

创建 data 文件夹

mkdir data

#赋予 minio 文件执行权限

chmod +x minio

2.脚本设置账户密码

#脚本

#创建脚本文件

vim /opt/minio/minio.sh

配置: 参考部署包: [pdp-cloud/shell/中间件服务脚本/minio/minio.sh](#)

添加:

```
#!/bin/bash
```

```
export MINIO_ROOT_USER=root
```

```
export MINIO_ROOT_PASSWORD=witrootroot
```

```
nohup ./minio server --address ":9000" --console-address ":19002"
```

```
--config-dir ./config ./data > ./minio.log 2>&1 &
```

属性解释:

[MINIO_ROOT_USER:账号]

[MINIO_ROOT_PASSWORD:密码]

#给启动文件授予权限

chmod 777 minio.sh

#启动

sh minio.sh

#查看

ps -ef | grep minio

#关闭进程可以使用:

使用 kill -9 进程 id 号, 即可关闭进程

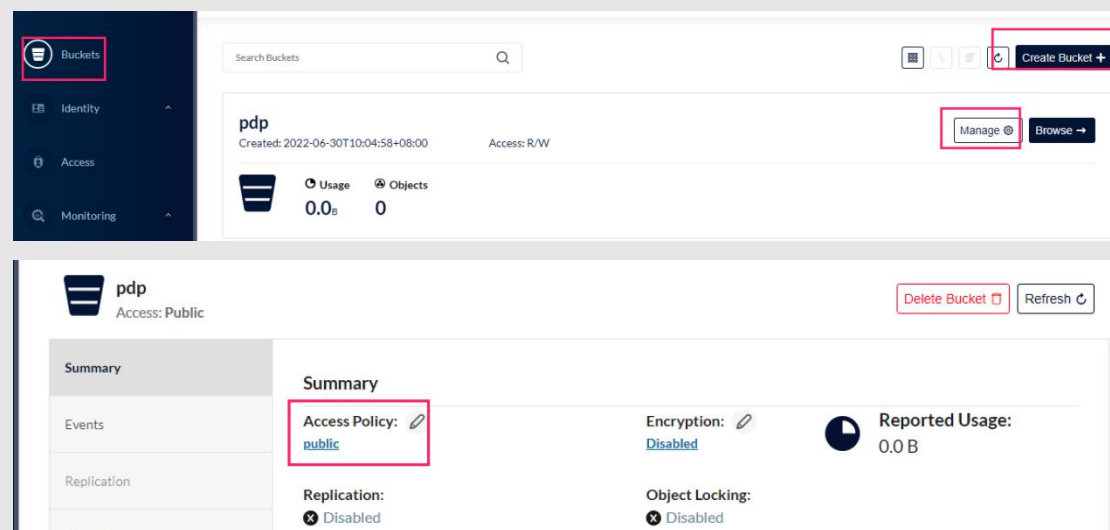
3.测试访问:

http://ip:19002

4.使用浏览器访问并创建桶:

Create bucket -> pdp

设置隐私: Manage->Summary->Access Policy->public



5.前端直接根据 fileKey 预览图片

前提: 将桶的权限[Summary -> Access Policy]设置成: public

使用: 公网 IP+内网端口[9000]/fileKey

6. RocketMQ 安装

1.配置环境

推荐 64 位操作系统, Linux/Unix/Mac;

64 位 JDK 1.8+;

Maven 3.2.x;

2. 安装

cd /opt

上传并解压

unzip rocketmq-all-4.9.3-bin-release.zip

#重命名

mv rocketmq-4.9.3 rocketmq

3.配置 JVM 参数

cd /opt/rocketmq/bin

vim runbroker.sh

```
choose_gc_options()
{
    JAVA_MAJOR_VERSION=$(("$JAVA" -version 2>&1 | head -1 | cut -d' ' -f2 | sed 's/"1\."/ /' | cut -d'.' -f1)
    if [ -z "$JAVA_MAJOR_VERSION" ] || [ "$JAVA_MAJOR_VERSION" -lt "8" ]; then
        JAVA_OPT="$({JAVA_OPT} -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelOldSpaceEnabled -XX:SurvivorRatio=8 -XX:-UseParNewGC"
    else
        JAVA_OPT="$({JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRe
    fi

    if [ -z "$JAVA_MAJOR_VERSION" ] || [ "$JAVA_MAJOR_VERSION" -lt "9" ]; then
        JAVA_OPT="$({JAVA_OPT} -verbose:gc -Xloggc:${GC_LOG_DIR}/mq_srv_gc_%p_%t.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGC
        JAVA_OPT="$({JAVA_OPT} -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m"
    else
        JAVA_OPT="$({JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRe
        JAVA_OPT="$({JAVA_OPT} -Xlog:gc*:file=${GC_LOG_DIR}/mq_srv_gc_%p_%t.log:time,tags:filecount=5,filesize=30M"
    fi
}

choose_gc_log_directory
JAVA_OPT="$({JAVA_OPT} -server -Xms8g -Xmx8g"
choose_gc_options
JAVA_OPT="$({JAVA_OPT} -XX:-OmitStackTraceInFastThrow"
JAVA_OPT="$({JAVA_OPT} -XX:+AlwaysPreTouch"
JAVA_OPT="$({JAVA_OPT} -XX:MaxDirectMemorySize=15g"
JAVA_OPT="$({JAVA_OPT} -XX:-UseLargePages -XX:-UseBiasedLocking"
#JAVA_OPT="$({JAVA_OPT} -Xdebug -Xrunjdwp:transport=dt_socket,address=9555,server=y,suspend=n"
JAVA_OPT="$({JAVA_OPT} ${JAVA_OPT_EXT}"
JAVA_OPT="$({JAVA_OPT} -cp ${CLASSPATH}"

numactl --interleave=all pwd > /dev/null 2>&1
if [ $? -eq 0 ]
then
    if [ -z "$RMQ_NUMA_NODE" ]; then
        numactl --interleave=all $JAVA ${JAVA_OPT} $@
```

改成:

-server -Xms256m -Xmx256m -Xmn256m


```

JAVA_MAJOR_VERSION=$( "$JAVA" -version 2>&1 | head -1 | cut -d " " -f2 | sed 's/1\./7/ | cut -d "." -f1 )
if [ -z "$JAVA_MAJOR_VERSION" ] || [ "$JAVA_MAJOR_VERSION" -lt "8" ] ; then
    JAVA_OPT="$JAVA_OPT" -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccupancyFraction=70 -XX:CMSParallelRemarkEnabled -XX:SurvivorRatio=8 -XX:-UseParNewGC"
else
    JAVA_OPT="$JAVA_OPT" -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRefLRUPolicyMSPerMB=50
fi

if [ -z "$JAVA_MAJOR_VERSION" ] || [ "$JAVA_MAJOR_VERSION" -lt "9" ] ; then
    JAVA_OPT="$JAVA_OPT" -verbose:gc -Xloggc:${GC_LOG_DIR}/rmq_srv_gc_%p_%t.log -XX:+PrintGCDateStamps
    JAVA_OPT="$JAVA_OPT" -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m
else
    JAVA_OPT="$JAVA_OPT" -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRefLRUPolicyMSPerMB=50
    JAVA_OPT="$JAVA_OPT" -Xlog:gc*:file=${GC_LOG_DIR}/rmq_srv_gc_%p_%t.log:time,tags:filecount=5,filesize=30M
fi
}

choose_gc_log_directory
choose_gc_options
JAVA_OPT="$JAVA_OPT" -server -Xms256m -Xmx256m -Xmn256m
choose_gc_options
JAVA_OPT="$JAVA_OPT" -XX:-OmitStackTraceInFastThrow"
JAVA_OPT="$JAVA_OPT" -XX:+AlwaysPreTouch"
JAVA_OPT="$JAVA_OPT" -XX:MaxDirectMemorySize=15g"
JAVA_OPT="$JAVA_OPT" -XX:-UseLargePages -XX:-UseBiasedLocking"
#JAVA_OPT="$JAVA_OPT" -Xdebug -Xrunjdwp:transport=dt_socket,address=9555,server=y,suspend=n"
JAVA_OPT="$JAVA_OPT" ${JAVA_OPT_EXT}"

```

vim runserver.sh

```

# Example of JAVA_MAJOR_VERSION value : '1', '9', '10', '11', ...
# '1' means releases before Java 9
JAVA_MAJOR_VERSION=$( "$JAVA" -version 2>&1 | sed -r -n 's/.* version "([0-9]*).*/\1/p' )
if [ -z "$JAVA_MAJOR_VERSION" ] || [ "$JAVA_MAJOR_VERSION" -lt "9" ] ; then
    JAVA_OPT="$JAVA_OPT" -server -Xms4g -Xmx4g -Xmn2g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m
    JAVA_OPT="$JAVA_OPT" -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccupancyFraction=70 -XX:CMSParallelRemarkEnabled -XX:SurvivorRatio=8 -XX:-UseParNewGC"
    JAVA_OPT="$JAVA_OPT" -verbose:gc -Xloggc:${GC_LOG_DIR}/rmq_srv_gc_%p_%t.log -XX:+PrintGCDateStamps
    JAVA_OPT="$JAVA_OPT" -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m
else
    JAVA_OPT="$JAVA_OPT" -server -Xms4g -Xmx4g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m
    JAVA_OPT="$JAVA_OPT" -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRefLRUPolicyMSPerMB=50
    JAVA_OPT="$JAVA_OPT" -Xlog:gc*:file=${GC_LOG_DIR}/rmq_srv_gc_%p_%t.log:time,tags:filecount=5,filesize=30M
fi
}

```

#都改成

-server -Xms256m -Xmx256m -Xmn256m

```

}

choose_gc_options()
{
    # Example of JAVA_MAJOR_VERSION value : '1', '9', '10', '11', ...
    # '1' means releases before Java 9
    JAVA_MAJOR_VERSION=$( "$JAVA" -version 2>&1 | sed -r -n 's/.* version "([0-9]*).*/\1/p' )
    if [ -z "$JAVA_MAJOR_VERSION" ] || [ "$JAVA_MAJOR_VERSION" -lt "9" ] ; then
        JAVA_OPT="$JAVA_OPT" -server -Xms256m -Xmx256m -Xmn256m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m
        JAVA_OPT="$JAVA_OPT" -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccupancyFraction=70 -XX:CMSParallelRemarkEnabled -XX:SurvivorRatio=8 -XX:-UseParNewGC"
        JAVA_OPT="$JAVA_OPT" -verbose:gc -Xloggc:${GC_LOG_DIR}/rmq_srv_gc_%p_%t.log -XX:+PrintGCDateStamps -XX:+PrintGCDetails
        JAVA_OPT="$JAVA_OPT" -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m"
    else
        JAVA_OPT="$JAVA_OPT" -server -Xms256m -Xmx256m -Xmn256m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m
        JAVA_OPT="$JAVA_OPT" -XX:+UseG1GC -XX:G1HeapRegionSize=16m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=30 -XX:SoftRefLRUPolicyMSPerMB=50
        JAVA_OPT="$JAVA_OPT" -Xlog:gc*:file=${GC_LOG_DIR}/rmq_srv_gc_%p_%t.log:time,tags:filecount=5,filesize=30M"
    fi
}

choose_gc_log_directory
choose_gc_options
JAVA_OPT="$JAVA_OPT" -XX:-OmitStackTraceInFastThrow"
JAVA_OPT="$JAVA_OPT" -XX:-UseLargePages"
}

```

4.设置环境变量

vim /etc/profile

#添加

export NAMESRV_ADDR=服务器IP:9876

```

#JDK
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_221
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
#export PATH=${JAVA_HOME}/bin:$PATH

#Maven
export MAVEN_HOME=/opt/maven
export PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin

#MQ
export NAMESRV_ADDR=192.168.196.190:9876

```

#重新加载环境变量

source /etc/profile

5.启动

cd /opt/rocketmq

#创建日志目录

mkdir log

#启动

nohup bin/mqnamesrv > log/mqname.log 2>&1 &

#开启自动创建 topic

nohup bin/mqbroker -n 服务器地址:9876 -c conf/broker.conf
autoCreateTopicEnable=true > log/borker.log 2>&1 &

```

#启动MQ
cd /opt/rocketmq
sh bin/mqshutd broker
sh bin/mqshutd broker

nohup bin/mqnamesrv > log/mqname.log 2>&1 &
nohup bin/mqbroker -n 192.168.196.175:9876 -c conf/broker.conf autoCreateTopicEnable=true > log/borker.log 2>&1 &

```

#若启动失败，日志中出现:

#nohup: failed to run command 'bin/mqbroker' : Permission denied

#则对当前总目录进行授权:

sudo chmod -R xxx #某一目录

6.查看进程

jps

#出现两个

17239 BrokerStartup

17150 NameservStartup

#关闭 Broker

sh bin/mqshutdown broker

```
#关闭 NameServer
sh bin/mqshutdown namesrv
```

7.模拟发送消息 和 消费消息

目前 我们 NameServer 和 Broker 都已经启动成功，就可以进行发送和消费消息，RocketMQ 安装包默认提供了一个模拟发送消息和消费消息的测试类，我们来验证一下：

#创建生产者发送消息

```
sh bin/tools.sh org.apache.rocketmq.example.quickstart.Producer
```

#消费消息

```
sh bin/tools.sh org.apache.rocketmq.example.quickstart.Consumer
```

退出日志： **ctrl+c**

8.安装可视化界面

```
mkdir -p /opt/rocketmq-console
mkdir -p /opt/rocketmq-console/data
mkdir -p /opt/rocketmq-console/log
```

#上传并运行 jar 包

```
cd /opt/rocketmq-console
```

```
[root@localhost rocketmq-console]# ll
总用量 32496
drwxr-xr-x 2 root root      6 10月 13 11:27 data
drwxr-xr-x 2 root root      6 10月 13 11:27 log
-rw-r--r-- 1 root root 33275633 6月 30 14:21 rocketmq-console-ng-1.0.1.jar
[root@localhost rocketmq-console]#
```

#添加执行权限

```
chmod +777 rocketmq-console-ng-1.0.1.jar
```

#启动并设置端口

```
nohup java -jar rocketmq-console-ng-1.0.1.jar --server.port=19004 >>
/opt/rocketmq-console/log/out.log 2>&1 &
```

```
#启动MQ页面
cd /opt/rocketmq-console
nohup java -jar rocketmq-console-ng-1.0.1.jar --server.port=19004 >> /opt/rocketmq-console/log/out.log 2>&1 &
```

运行成功在浏览器输入 <http://ip:19004> 即可看到运行界面，账号密码: admin/admin

7. Xxl-job 安装

1.创建数据库并添加表

```
pdp-cloud/sql/pdp_job.sql
```

2.安装 xxljob

2.1.设置路径


```
mkdir -p /opt/xxljob/log
cd /opt
chmod +777 xxljob
```

2.2.上传并解压

```
cd /opt/xxljob
tar -zxvf xxl-job-2.3.0.tar.gz
```

也可以在 **idea** 上进行项目的配置修改并打包，打包好之后放到 **linux** 服务器上。

2.3.修改配置文件

```
vim /opt/xxljob/xxl-job-2.3.0/xxl-job-admin/src/main/resources/application.properties
```

#修改端口 以及 数据库连接

```
server.port=19008
spring.datasource.url=jdbc:mysql://ip:19001/pdp_job?serverTimezone=Asia/Shanghai&
characterEncoding=utf8&useUnicode=true&useSSL=false&autoReconnect=true&allowM
ultiQueries=true
spring.datasource.username=root
spring.datasource.password=witroot
```

2.4.修改日志地址:

```
vim /opt/xxljob/xxl-job-2.3.0/xxl-job-admin/src/main/resources/logback.xml
```

修改地址:

```
<property name="log.path" value="/opt/xxljob/log/xxl-job-admin.log"/>
```

踩坑: 连接数据库报错

解决: 连接中添加 **&useSSL=false** 因为线上环境一般是 https,需要 ssl 证书

2.5.打包

到根目录下

```
cd /opt/xxljob/xxl-job-2.3.0
mvn package
```

注意: 需要有网络, java 环境, maven 环境

打包完之后会生成可执行的 jar

/opt/xxljob/xxl-job-2.3.0/xxl-job-admin/target/xxl-job-admin-2.3.0.jar

将 **jar** 包和启动脚本放到 **opt/xxljob** 下并赋权:

```
chmod 777 xxl-job-admin-2.3.0.jar
```

```
[root@localhost xxljob]# ll
总用量 38088
drwxr-xr-x 2 root root      6 10月 13 13:07 log
-rwxrwxrwx 1 root root 38997819 10月 13 13:59 xxl-job-admin-2.3.0.jar
-rw-r--r-- 1 root root    291 10月 13 13:59 xxljob-depoly.sh
[root@localhost xxljob]#
```

2.6.创建启动脚本:

```
cd /opt/xxljob
```

```
vim xxljob-depoly.sh
```

脚本位置: [pdp-cloud/shell/中间件服务脚本/xxljob-depoly.sh](#)

```
#!/bin/sh
```

```
#刷新环境
```

```
source /etc/profile
```

```
#删除日志
```

```
rm -f /opt/xxljob/log/*
```

```
#干掉进程
```

```
ps aux | grep xxl-job-admin-2.3.0.jar | grep -v grep | awk '{print $2}' | xargs kill -9
```

```
#启动
```

```
nohup java -jar /opt/xxljob/xxl-job-admin-2.3.0.jar > /opt/xxljob/log/admin.log 2>&1 &
```

2.7.执行脚本

```
chmod 777 xxljob-depoly.sh
```

```
sh xxljob-depoly.sh
```

2.8.查看

```
ps -ef|grep xxljob
```

```
[root@localhost xxljob]# ps -ef|grep xxljob
root      18485      1  21  14:03 pts/0    00:00:12 java -jar /opt/xxljob/xxl-job-admin-2.3.0.jar
root      18567    4092   0  14:03 pts/0    00:00:00 grep --color=auto xxljob
```

3.访问:

ip:19008/xxl-job-admin

账号/密码 admin/123456

8. Linux 开机启动设置

1.编写 sh 文件

```
mkdir -p /opt/start
```

```
cd /opt/start
```

```
vim start.sh
```

编写: [必须以: `#!/bin/bash` 开头]

脚本: [pdp-cloud/shell/中间件服务脚本/start.sh](#)

```
#!/bin/bash
```

#停止防火墙

```
systemctl stop firewalld
```

#启动 Mysql

```
cd /opt/mysql
```

```
chkconfig --add mysql
```

```
service mysql stop
```

```
service mysql start
```

#启动 Minio

```
ps aux | grep minio | grep -v grep | awk '{print $2}' | xargs kill -9
```

```
cd /opt/minio
```

```
./minio.sh
```

#启动 Redis

```
ps aux | grep redis | grep -v grep | awk '{print $2}' | xargs kill -9
```

```
cd /opt/redis
```

```
./bin/redis-server ./redis.conf
```

#启动 MQ

```
cd /opt/rocketmq
```

```
sh bin/mqshutdown broker
```

```
sh bin/mqshutdown namesrv
```

```
nohup bin/mqnamesrv > log/mqname.log 2>&1 &
```

#注意修改地址

```
nohup bin/mqbroker -n 192.168.196.175:9876 -c conf/broker.conf  
autoCreateTopicEnable=true > log/borker.log 2>&1 &
```

#启动 MQ 页面

```
cd /opt/rocketmq-console
```

```
nohup java -jar rocketmq-console-ng-1.0.1.jar --server.port=19004 >>  
/opt/rocketmq-console/log/out.log 2>&1 &
```

#重启 xxljob

```
source /opt/xxljob/xxljob-depoly.sh
```

```

#!/bin/bash

#停止防火墙
systemctl stop firewalld

#启动Mysql
cd /opt/mysql
chkconfig --add mysql
service mysql stop
service mysql start

#启动Minio
ps aux | grep minio | grep -v grep | awk '{print $2}' | xargs kill -9
cd /opt/minio
./minio.sh

#启动Redis
ps aux | grep redis | grep -v grep | awk '{print $2}' | xargs kill -9
cd /opt/redis
./bin/redis-server ./redis.conf

#启动MQ
cd /opt/rocketmq
sh bin/mqshutdown broker
sh bin/mqshutdown namesrv

nohup bin/mqnamesrv > log/mqname.log 2>&1 &
nohup bin/mqbroker -n 192.168.196.175:9876 -c conf/broker.conf autoCreateTopicEnable=true > log/borker.log 2>&1 &

#启动MQ页面
cd /opt/rocketmq-console
nohup java -jar rocketmq-console-ng-1.0.1.jar --server.port=19004 >> /opt/rocketmq-console/log/out.log 2>&1 &

#重启xxljob
source /opt/xxljob/depoly.sh

```

2.对 sh 文件授权

chmod +x /opt/start/start.sh

或

chmod +777 /opt/start/start.sh

3.修改 rc.local

vim /etc/rc.d/rc.local

添加脚本:

source /opt/start/start.sh

3.对 rc.local 授权

cd /etc/rc.d

chmod +777 rc.local

9. 配置防火墙

查看:

systemctl status firewalld

关闭:

systemctl stop firewalld

systemctl disable firewalld

开放端口: 8088

firewall-cmd --zone=public --add-port=8088/tcp --permanent

关闭端口

```
firewall-cmd --zone=public --remove-port=8088/tcp --permanent
```

开放或关闭端口后，需要刷新

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

五、应用服务端中间件安装

关闭防火墙

```
systemctl stop firewalld
```

1. 安装 JDK 【同上】

2. 安装 Maven 【同上】

3.kkfileView 安装

1.环境要求

Java: 1.8+

OpenOffice 或 LiberOffice(Windows 下已内置，CentOS 或 Ubuntu 下会自动下载安装，MacOS 下需要自行安装)

2.安装 OpenOffice

#1.上传并解压压缩包，执行下面的命令，解压后的文件夹名称为 zh-CN

```
cd /opt
```

```
tar -zxvf Apache_OpenOffice_4.1.12_Linux_x86-64_install-rpm_zh-CN.tar.gz
```

#2.进入解压目录 zh-CN/RPMS/下，执行安装命令

```
cd zh-CN/RPMS/
```

```
rpm -ivh *.rpm
```

#安装完成,在/opt 下自动创建 openoffice4 文件夹

#3.进入 OpenOffice 安装目录,开启 openOffice 服务，端口为 8100

```
cd /opt/openoffice4/program/
```

```
./soffice -headless -accept="socket,host=127.0.0.1,port=8100:urp;" -nofirststartwizard &
```

```
#kkfile
cd /opt/openoffice4/program
./soffice -headless -accept="socket,host=127.0.0.1,port=8100:urp;" -nofirststartwizard &
```

#4.查看开启的服务

```
ps -ef | grep soffice
```

4.安装 kk

```
cd /opt
```

#上传并解压

```
tar -zxvf kkFileView-4.0.0.tar.gz
```

```
mv kkFileView-4.0.0 kk
```

#启动

```
cd /opt/kk/bin
```

```
./startup.sh
```

#查看

浏览器访问:<http://ip:8012>

上传文件点击预览

六、后端部署


1. 导入 sql


将部署包下的 sql 文件夹中的 sql 脚本导入到数据库。


每一个脚本文件对应一个数据库。[除了 **base_province.sql**]


注意: **pdp_base** 数据库需要将脚本内容查询执行


base_province.sql 在 **pdp_base** 库下导入


 base_province.sql


 pdp_base.sql


 pdp_basedata.sql


 pdp_calc.sql

 pdp_core_config.sql

 pdp_job.sql

 pdp_message.sql

 pdp_nacos.sql

 pdp_saas.sql

 pdp_wbs.sql

2. 安装 Nacos

1. 上传压缩包

创建路径

```
mkdir -p /opt/server/registry
```

```
cd /opt/server/registry
```

将部署包下 jar 文件夹中的 **nacos-server.tar.gz** 上传至 registry 中。

2. 解压

```
tar -zxvf nacos-server.tar.gz
```

```
[root@localhost nacos-server]# ll
总用量 32
drwxr-xr-x 3 root root   94 5月  1 17:36 bin
drwxr-xr-x 2 root root  205 5月 11 20:52 conf
drwxr-xr-x 7 root root   87 6月 20 18:32 data
-rw-r--r-- 1 root root 16899 4月 18 15:38 LICENSE
drwxr-xr-x 2 root root  4096 6月 23 10:25 logs
-rw-r--r-- 1 root root  1340 4月 18 15:38 NOTICE
drwxr-xr-x 2 root root   30 4月 18 15:38 target
[root@localhost nacos-server]#
```

3. 修改配置

cd /opt/server/registry/nacos-server/conf/

```
[root@localhost nacos-server]# cd conf/
[root@localhost conf]# ll
总用量 80
-rw-r--r-- 1 root root 1250 4月 18 15:38 1.4.0-ipv6_support-update.sql
-rw-r--r-- 1 root root 7746 5月 11 20:51 application.properties
-rw-r--r-- 1 root root 6692 4月 18 15:38 application.properties.example
-rw-r--r-- 1 root root  691 4月 18 15:38 cluster.conf.example
-rw-r--r-- 1 root root 31984 4月 18 15:38 nacos-logback.xml
-rw-r--r-- 1 root root 10878 4月 18 15:38 nacos-mysql.sql
-rw-r--r-- 1 root root  9022 4月 18 15:38 schema.sql
[root@localhost conf]#
```

修改配置文件: application.properties

vim application.properties

修改端口以及数据库连接

```
application.properties
16
17 ***** Spring Boot Related Configurations *****#
18 # Default web context path:
19 server.servlet.contextPath=/nacos
20 ## Default web server port:
21 server.port=30099
22
23 ***** Network Related Configurations *****#
24 ## If prefer hostname over ip for Nacos server addresses in cluster.conf:
25 # nacos.inetutils.prefer-hostname-over-ip=false
26
27 ## Specify local server's IP:
28 # nacos.inetutils.ip-address=
29
30
31 ***** Config Module Related Configurations *****#
32 ## If use MySQL as datasource:
33 spring.datasource.platform=mysql
34
35 ## Count of DB:
36 db.num=1
37
38 ## Connect URL of DB:
39 db.url.0=jdbc:mysql://192.168.199.163:19001/pdp_nacos?characterEncoding=utf8&connectTimeout=1000&socketT
40 db.user=root
41 db.password=witroot
42
43 ## Connection pool configuration: hikariCP
44 db.pool.config.connectionTimeout=30000
45 db.pool.config.validationTimeout=10000
46 db.pool.config.maximumPoolSize=20
47 db.pool.config.minimumIdle=2
48
```

4. 启动

cd /opt/server/registry/nacos-server/bin/

授权

chmod +777 startup.sh

chmod +777 shutdown.sh

```
[root@localhost nacos-server]# cd bin/
[root@localhost bin]# ll
总用量 20
-rw-r--r-- 1 root root 978 4月 18 15:38 shutdown.cmd
-rwxrwxrwx 1 root root 951 5月 1 17:36 shutdown.sh
-rw-r--r-- 1 root root 3438 4月 18 15:38 startup.cmd
-rwxrwxrwx 1 root root 4926 5月 1 17:35 startup.sh
drwxr-xr-x 3 root root 20 4月 18 20:14 work
[root@localhost bin]#
```

启动: ./startup.sh

关闭: ./shutdown.sh

访问: ip:端口/nacos

账号: nacos/nacos

3.修改 nacos 中相关配置

<input type="checkbox"/>	datasource.yaml	DEFAULT_GROUP	redis 数据库
<input type="checkbox"/>	resources.yaml	DEFAULT_GROUP	minio
<input type="checkbox"/>	router.yaml	DEFAULT_GROUP	
<input type="checkbox"/>	tenant.yaml	DEFAULT_GROUP	
<input type="checkbox"/>	system-config.yaml	DEFAULT_GROUP	
<input type="checkbox"/>	pdp-basedata-datasource.yaml	DEFAULT_GROUP	数据库
<input type="checkbox"/>	pdp-coreconfig-datasource.yaml	DEFAULT_GROUP	数据库
<input type="checkbox"/>	pdp-calc-datasource.yaml	DEFAULT_GROUP	数据库
<input type="checkbox"/>	pdp-wbs-datasource.yaml	DEFAULT_GROUP	数据库
<input type="checkbox"/>	pdp-saas-datasource.yaml	DEFAULT_GROUP	数据库
<input type="checkbox"/>	pdp-rocketmq.yaml	DEFAULT_GROUP	MQ
<input type="checkbox"/>	pdp-redis.yaml	DEFAULT_GROUP	Redis
<input type="checkbox"/>	pdp-openapi.yaml	DEFAULT_GROUP	PLM CREO
<input type="checkbox"/>	pdp-job.yaml	DEFAULT_GROUP	XXL-job
<input type="checkbox"/>	pdp-message-datasource.yaml	DEFAULT_GROUP	数据库

4.部署 jar

创建文件夹:

```
mkdir -p /opt/server/pdp-cloud
```

```
mkdir -p /opt/server/pdp-cloud-log
```

将部署包下 jar 文件夹中的服务 jar 包上传至 pdp-cloud 中。

将部署包下 **shell/应用服务脚本/jar_shell.sh** 上传至 pdp-cloud 中。

为脚本文件设置权限: `chmod +777 jar_shell.sh`

```

[root@localhost pdp-cloud]# ll
总用量 2871272
-rwxrwxrwx 1 root root    5024 10月 13 16:56 jar_shell.sh
-rw-r--r-- 1 root root 160698145 9月 30 11:24 jnpf-file-server.jar
-rw-r--r-- 1 root root 128247060 9月 30 11:22 jnpf-gateway.jar
-rw-r--r-- 1 root root 179725818 9月 30 11:25 jnpf-message-server.jar
-rw-r--r-- 1 root root 171591772 9月 30 11:23 jnpf-oauth-server.jar
-rw-r--r-- 1 root root 163028937 9月 30 11:23 jnpf-permission-server.jar
-rw-r--r-- 1 root root 168221964 9月 30 11:23 jnpf-system-server.jar
-rw-r--r-- 1 root root 153061920 9月 30 11:25 jnpf-visualdata-server.jar
-rw-r--r-- 1 root root 169072239 9月 30 11:24 jnpf-visualdev-server.jar
-rw-r--r-- 1 root root 169061448 9月 30 11:24 jnpf-workflow-server.jar
-rw-r--r-- 1 root root 176019102 9月 30 17:38 pdp-basedata-server.jar
-rw-r--r-- 1 root root 174406112 9月 30 15:37 pdp-calc-server.jar
-rw-r--r-- 1 root root 176557203 10月 12 18:12 pdp-coreconfig-server.jar
-rw-r--r-- 1 root root 173730164 9月 30 11:27 pdp-message-server.jar
-rw-r--r-- 1 root root 174864230 9月 30 11:27 pdp-openapi-server.jar
-rw-r--r-- 1 root root 176503644 9月 30 11:28 pdp-report-server.jar
-rw-r--r-- 1 root root 189709595 9月 30 17:38 pdp-saas-server.jar
-rw-r--r-- 1 root root 183432971 9月 30 17:35 pdp-wbs-server.jar
-rw-r--r-- 1 root root 52212674 9月 30 11:28 spring-boot-admin.jar

```

执行脚本:

启动并指定 nacos 的地址:

sh jar_shell.sh start all nacosip:端口

停止:

sh jar_shell.sh stop all

```

[root@localhost pdp-cloud]#
[root@localhost pdp-cloud]# sh jar_shell.sh start all 192.168.196.176:30099
jnpf-gateway---jnpf-gateway: 已经启动成功,PID=23790
jnpf-oauth-server---jnpf-oauth-server: 已经启动成功,PID=23818
jnpf-system-server---jnpf-system-server: 已经启动成功,PID=23855
jnpf-permission-server---jnpf-permission-server: 已经启动成功,PID=23891
jnpf-file-server---jnpf-file-server: 已经启动成功,PID=23927
jnpf-app-server---jnpf-app-server: 文件不存在无法运行
jnpf-extend-server---jnpf-extend-server: 已经启动成功,PID=23978
jnpf-message-server---jnpf-message-server: 已经启动成功,PID=24005
jnpf-visualdata-server---jnpf-visualdata-server: 文件不存在无法运行
jnpf-visualdev-server---jnpf-visualdev-server: 文件不存在无法运行
jnpf-workflow-server---jnpf-workflow-server: 文件不存在无法运行
jnpf-tenant-server---jnpf-tenant-server: 文件不存在无法运行
pdp-basedata-server---pdp-basedata-server: 已经启动成功,PID=24082
pdp-calc-server---pdp-calc-server: 已经启动成功,PID=24118
pdp-coreconfig-server---pdp-coreconfig-server: 已经启动成功,PID=24154
pdp-openapi-server---pdp-openapi-server: 已经启动成功,PID=24176
pdp-wbs-server---pdp-wbs-server: 已经启动成功,PID=24213
pdp-saas-server---pdp-saas-server: 已经启动成功,PID=24256
report-console---report-console: 文件不存在无法运行
spring-boot-admin---spring-boot-admin: 已经启动成功,PID=24308
xxl-job-admin---xxl-job-admin: 文件不存在无法运行
.....本次共启动:14个服务.....

```

七、前端部署

1. 上传前端代码包

创建目录: `mkdir -p /opt/front/pdp-cloud`

`cd /opt/front`

将目录设置权限:

`chmod +777 pdp-cloud`

将部署包下 front 文件夹中的文件上传至/opt/front/pdp-cloud 并解压。

`tar -zxvf pre-front-cloud-web.tar.gz`

```
[root@localhost nginx-conf]# cd /opt/front/
[root@localhost front]# ll
总用量 0
drwxrwxrwx 4 root root 177 6月 23 13:28 pdp-cloud
[root@localhost front]# cd pdp-cloud/
[root@localhost pdp-cloud]# ll
总用量 5296
drwxr-xr-x 4 root root 42 6月 23 13:28 cdn
-rw-r--r-- 1 root root 828451 6月 23 13:28 css.worker.js
-rw-r--r-- 1 root root 127444 6月 23 13:28 editor.worker.js
-rw-r--r-- 1 root root 9662 6月 23 13:28 favicon.ico
-rw-r--r-- 1 root root 544081 6月 23 13:28 html.worker.js
-rw-r--r-- 1 root root 10528 6月 23 13:28 index.html
-rw-r--r-- 1 root root 237632 6月 23 13:28 json.worker.js
drwxr-xr-x 6 root root 51 6月 23 13:28 static
-rw-r--r-- 1 root root 3646363 6月 23 13:28 ts.worker.js
[root@localhost pdp-cloud]#
```

2. 安装 Nginx

1、安装所需环境

#安装 gcc

`yum install gcc-c++`

#安装 PCRE pcre-devel

`yum install -y pcre pcre-devel`

#安装 zlib

`yum install -y zlib zlib-devel`

#安装 Open SSL

`yum install -y openssl openssl-devel`

注意: 第一步安装不了时直接从第二步开始试

2、安装

创建目录

`cd /opt`

#上传并解压 tar

`tar -xvf nginx-1.17.9.tar.gz`

`mv nginx-1.17.9 nginx`

`cd nginx/`

#执行命令 考虑到后续安装 ssl 证书 添加两个模块

```
./configure --with-http_stub_status_module --with-http_ssl_module
```

```
#执行 make 命令
```

```
make
```

```
#执行 make install 命令
```

```
make install
```

3、启动

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

```
#启动
```

```
./nginx
```

```
#停止
```

```
./nginx -s stop
```

4、配置:

```
创建文件夹: mkdir -p /opt/nginx-conf
```

```
将部署包下 shell/应用服务脚本/pdp-nginx.conf 上传至 nginx-conf 中并修改地址。
```

```
设置权限: chmod +777 pdp-nginx.conf
```

```
修改配置:
```

```
pdp-nginx.conf
server {
    listen 11000;
    server_name _;
    index index.html index.htm;
    root /opt/front/pdp-cloud;

    location / {
        #try_files $uri $uri/ @rewrites;
        try_files $uri $uri/ /index.html;
    }

    #location @rewrites {
    #    #    rewrite ^.*$ /index.html last;
    #}

    #添加头部信息
    proxy_set_header Cookie $http_cookie;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    location /api {
        proxy_pass http://192.168.199.162:30000;
    }

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

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

前端的端口和文件路径

后端的gateway

5、将自定义配置加进 nginx
cd /usr/local/nginx/conf
vim nginx.conf
#添加:
include /opt/nginx-conf/*.conf;


```

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid logs/nginx.pid;

events {
    worker_connections 1024;
}

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

    #加载配置文件
    include /opt/nginx-conf/*.conf;

    #log_format main '$remote_addr - $remote_user [$time_local] "$request
    #                  $status $body_bytes_sent "$http_referer" '
    #                  '$http_user_agent' "$http_x_forwarded_for"';

    #access_log logs/access.log main;

    sendfile on;
    #tcp_nopush on;

    #keepalive_timeout 0;
    keepalive_timeout 65;

    #gzip on;

    server {
        listen 80;

```

重启 nginx

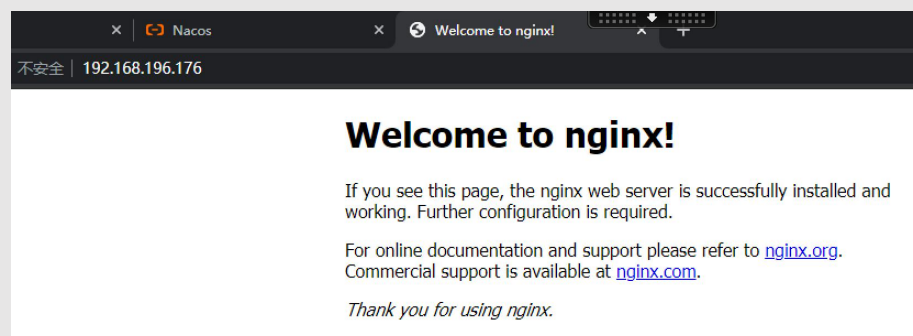
cd /usr/local/nginx/sbin

./nginx -s stop

./nginx

6、验证

Ip:80



3. 添加开机启动文件

```
mkdir -p /opt/start
```

vim start.sh

参考: [pdp-cloud/shell/应用服务脚本/start.sh](#)

```
#!/bin/bash

#关闭防火墙
systemctl stop firewalld

#nacos
cd /opt/server/registry/nacos-server/bin
sh shutdown.sh
sh startup.sh

#启动nginx
cd /usr/local/nginx/sbin
./nginx -s stop
./nginx

#kkfile
cd /opt/openoffice4/program
./soffice -headless -accept="socket,host=127.0.0.1,port=8100:urp;" -nofirststartwizard &

cd /opt/kk/bin
sh shutdown.sh
sh startup.sh
```

2.对 sh 文件授权

chmod +777 /opt/start/start.sh

3.修改 rc.local

vim /etc/rc.d/rc.local

添加脚本:

source /opt/start/start.sh

5.对 rc.local 授权

cd /etc/rc.d

chmod +777 rc.local

八、系统授权

初次启动 gateway 服务系统需要授权,所以后台服务会报权限异常:

```
cd /opt/server/pdp-cloud-log
tail -f jnpf-gateway.log
```

```
at org.springframework.boot.loader.PropertiesLauncher.main(PropertiesLauncher.java:466)
Caused by: org.springframework.beans.BeanInstantiationException: Failed to instantiate [java.security.KeyPair]: Factory method 'key' threw exception; nested exception is jnpf.exception.JnpfExc
当前机器缺少授权文件
at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiate(SimpleInstantiationStrategy.java:185)
at org.springframework.beans.factory.support.ConstructorResolver.instantiate(ConstructorResolver.java:650)
... 20 common frames omitted
Caused by: jnpf.exception.JnpfException: 当前机器缺少授权文件
at jnpf.encryption.KeyUtil.readPdpKeyFile(KeyUtil.java:35)
at jnpf.encryption.KeyUtil.checkMac(KeyUtil.java:55)
at jnpf.filter.KeyFilter.key(KeyFilter.java:37)
at jnpf.filter.KeyFilter$$EnhancerBySpringCGLIB$$8fd17a25.CGLIB$key$0(<generated>)
at jnpf.filter.KeyFilter$$EnhancerBySpringCGLIB$$8fd17a25$$FastClassBySpringCGLIB$$fc971b4c.invoke(<generated>)
```

解决:

1. 获取授权码

在启动日志上方找到:

```

2022-10-14 22:54:18.647 INFO 10958 --- [main] com.alibaba.nacos.client.naming : initializer namespace from System Property :null
2022-10-14 22:54:19.010 INFO 10958 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 21 endpoint(s) beneath base path '/actuator'
2022-10-14 22:54:19.228 INFO 10958 --- [main] jnpf.filter.KeyFilter : >>>开始授权验证
2022-10-14 22:54:19.517 INFO 10958 --- [main] jnpf.filter.KeyFilter : >>>开始授权验证 mac:52:54:00:c6:ac:08
2022-10-14 22:54:19.518 WARN 10958 --- [main] org.springframework.web.server.applicationcontext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'key' defined in class path resource [jnpf/filter/KeyFilter.class]: Bean instantiation via factory method failed; nested exception is java.lang.reflect.InvocationTargetException

```

2. 生成授权文件

将授权码发给管理员，管理员生成文件

3. 放置授权文件

将生成好的授权文件，放置在 gateway 服务所在机器的/opt/server/pdp-cloud 下:

```

[root@localhost pdp-cloud]# ll
总用量 3534920
-rwxrwxrwx 1 root root 1021 9月 5 17:43 depoly.sh
-rwxrwxrwx 1 root root 4866 6月 24 16:36 jar_shell.sh
-rw-r--r-- 1 root root 15932444 9月 5 17:44 jnpf-extend-server.jar
-rw-r--r-- 1 root root 160695653 9月 26 17:50 jnpf-file-server.jar
-rw-r--r-- 1 root root 128259603 10月 14 22:58 jnpf-gateway.jar
-rw-r--r-- 1 root root 179740232 10月 13 21:25 jnpf-message-server.jar
-rw-r--r-- 1 root root 171596474 10月 11 17:07 jnpf-oauth-server.jar
-rw-r--r-- 1 root root 163037270 10月 11 17:08 jnpf-permission-server.jar
-rw-r--r-- 1 root root 168228807 10月 11 17:08 jnpf-system-server.jar
-rw-r--r-- 1 root root 150719882 5月 28 00:46 jnpf-tenant-server.jar
-rw-r--r-- 1 root root 153057244 9月 1 20:03 jnpf-visualdata-server.jar
-rw-r--r-- 1 root root 169067575 9月 1 20:01 jnpf-visualdev-server.jar
-rw-r--r-- 1 root root 159977752 10月 11 17:09 jnpf-workflow-server.jar
drwxr-xr-x 7 root root 4096 8月 29 00:00 log
-rw-r--r-- 1 root root 176043481 10月 14 09:30 pdp-basedata-server.jar
-rw-r--r-- 1 root root 174406109 9月 30 14:38 pdp-calc-server.jar
-rw-r--r-- 1 root root 176565489 10月 13 15:18 pdp-coreconfi-server.jar
-rw-r--r-- 1 root root 256 10月 14 2022 pdp-key.txt
-rw-r--r-- 1 root root 173726821 9月 20 22:33 pdp-message-server.jar
-rw-r--r-- 1 root root 174859107 9月 21 10:08 pdp-openapi-server.jar
-rw-r--r-- 1 root root 176500668 9月 21 17:50 pdp-report-server.jar
-rw-r--r-- 1 root root 189727177 10月 14 09:31 pdp-saas-server.jar
-rw-r--r-- 1 root root 183445524 10月 13 10:59 pdp-wbs-server.jar
-rw-r--r-- 1 root root 215134350 8月 29 20:10 report-console.jar
-rw-r--r-- 1 root root 52212678 9月 22 15:39 spring-boot-admin.jar
-rw-r--r-- 1 root root 163362794 10月 14 17:07 triz-base-server.jar

```

4.重新启动 gateway 服务