用友网络

分析云安装部署手册

适用 7.7.0（2211）版

2022-11-06

目录

1 软硬件要求

以下应用服务器是单台应用，如果是应用服务器建立高可用，应用服务器则至少需要两台以上。

1.1 关系数据库部署

	应用服务器	数据仓库服务器 最低要求
CPU	16C	16C
内存	32G	32G
存储	200G HDD	1T HDD（或 SSD）
数量	1 台	1 台
操作系统	CentOS 7.3 及更高版本 Ubuntu Server18.0 LTS 及更高 Windows2008 R2 及更高版本 基于 Debian 10 以上版本 Linux 银河麒麟高级服务器系统 V10 统信 UOS 服务器操作系统 V20	CentOS 7.3+ RHEL 7.3+ Windows2008 R2 及更高版本 银河麒麟高级服务器系统 V10 统信 UOS 服务器操作系统 V20
网络	1Gb	1Gb
软件		Oracle 11g，12c 或更高版本 SQL Server 2012 R2 或更高版本 MySQL 8.0.18 或更高版本 达梦数据库 8.x 以上版本 GBase8s 8.3 以上版本 Kingbase 8.6 以上版本 神州通用数据库 7.0 以上版本

1.2 大数据平台部署

	分析云应用服务器	大数据存储计算服务器 最低要求
CPU	16C	16C
内存	32G	64G
存储	200G HDD	1T HDD 或更大
数量	1 台	4 台
操作系统	CentOS 7.3 及更高版本 Ubuntu Server18.0 LTS 及更高 Windows2008 R2 及更高版本 基于 Debian 10 以上版本 Linux 银河麒麟高级服务器系统 V10 统信 UOS 服务器操作系统 V20	CentOS 7.3—7.9 RHEL 7.3+ Ubuntu Server18.0 LTS

网络	1Gb	不小于 1Gb

注： 1.大数据平台的 CPU 必须是 X86_64 架构，并且支持 SSE4.2 指令；
2.分析云服务器的 CPU 推荐使用 X86_64 架构，否则需要自行安装 JDK 和 PostgreSQL(或 MySQL)数据库。

1.3 ClickHouse 部署

	分析云应用服务器	**ClickHouse 数仓服务器**
CPU	16C	16C
内存	32G	32G
存储	200G HDD	1T HDD 或更大
数量	1 台	1 台
操作系统	CentOS 7.3 及更高版本 Ubuntu Server18.0 LTS 及更高 Windows2008 R2 及更高版本 基于 Debian 10 以上版本 Linux 银河麒麟高级服务器系统 V10 统信 UOS 服务器操作系统 V20	CentOS 7.3 及更高 RHEL 7.3 及更高 Ubuntu Server18.0 LTS 及更高 SLES 12 SP3 及更高
网络	1Gb	1Gb
		建议使用最新的 LTS 版本

注： 1.ClickHouse 数仓服务器的 CPU 如果是 X86_64 架构，则必须要支持 SSE4.2 指令；
2.分析云服务器的 CPU 推荐使用 X86_64 架构,否则需要自行安装 JDK 和 PostgreSQL(或 MySQL)数据库。
3.ClickHouse 如果要搭建集群，推荐使用三台做集群，每台 64G 内存；

1.4 客户端要求

	最低要求
CPU	2C
内存	4G
存储	40G
操作系统	Windows 7，Windows 8.1，Windows 10 ，Windows 11 Mac OS 10.10 或更高版本 Ubuntu16.04 LTS，18.04 LTS,20.04 LTS Deepin 15.2 以上版本 银河麒麟桌面系统 V10 统信 UOS 桌面操作系统 V20

网络	100Mb 以上
浏览器	Chrome 80 及更高版本(包括使用 Chromeium 引擎的浏览器，例如 360 浏览器，搜狗浏览器等) Firefox 70 以及更高版本 IE 11（仅支持浏览态）

2 单机安装及配置

2.1 Windows 平台安装

步骤 0：校验

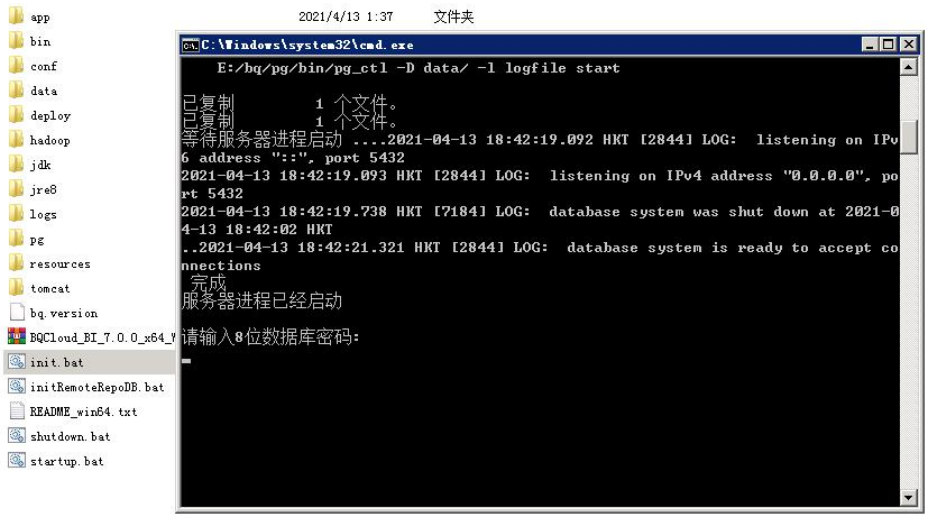
在 CMD 中，使用 certutil -hashfile BQCloud-BI-7_7_0-x64-Windows.zip SHA1 命令进行文件校验，得到 md 值，和 BQCloud-BI-7_7_0-x64-Windows.zip.SHA1 的值进行比对，一致就表示安装介质没有损坏或篡改。

步骤 1：解压

解压缩 BQCloud_BI_7_7_0_x64_Windows.zip 安装包到任一目录（例如，D:\bq），目录路径不要有中文、空格及特殊字符。**注意**：目录解压后，不要更改目录名称，否则后续的初始化会失败。

步骤 2：初始化

双击 init.bat 初始化，这里会弹出命令行窗口，首先会解压 PostgreSQL 和 JDK 文件，然后部署应用 war 包，最后初始化数据库 PostgreSQL。初始化数据库时，用户需要依据提示输入 8 位数据库密码（密码输入后需要回车；密码设置成功后，会提示“密码设置成功”）。密码设置成功后，命令行窗口会继续打印信息，执行完毕后，命令行窗口会自动关闭。

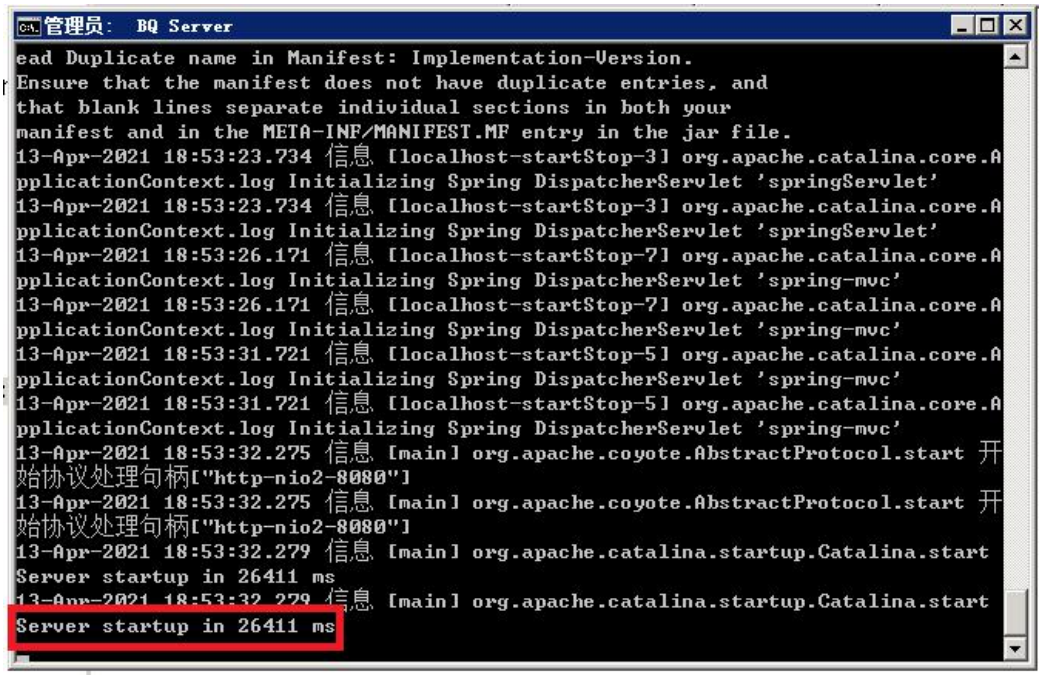


注意：某些 Windows 系统安装了杀毒软件，杀毒软件阻止初始化操作时，需要设置允许。

注意：某些 Windows(windows 2008 2012 等低版本版本)系统,双击 init.bat 后，可能会出现下图提示，这时可以进入 resources 文件夹下，分别双击执行 vcredist_x64_2013.exe 和 vcredist.x64_2015.exe， 用来安装 Visual C++ Redistributable。

步骤 3：启停服务

双击 startup.bat 启动服务，当看见 Server startup in *****ms，说明服务启动成功。



双击 shutdown.bat 停止服务。

2.2 Linux 平台安装

步骤 0：校验

在 shell 中使用 sha1sum　BQCloud-BI-7_7_0-x64-Linux.tar.gz 命令进行文件校验，得到 md 值，和 BQCloud-BI-7_7_0-x64-Linux.tar.gz.SHA1 的值进行比对，一致就表示安装介质没有损坏或篡改（如果在 Debian 等衍生系统中，如果使用的是非超级用户，执行的命令通常要前缀 sudo）。

步骤 1：解压

新建目录/home/bq，上传 BQCloud-BI-7_7_0-x64-Linux.tar.gz 到该目录。这里建议放在/home/bq，其它目录因为权限，可能会导致数据库初始化失败。

```
# mkdir /home/bq
```

进入目录，解压安装包到当前目录。

cd /home/bq

tar -zxvf BQCloud-BI-7_7_0-x64-Linux.tar.gz

步骤 2：初始化

初始化类似前述 Windows 平台，同样，会提示输入 8 位数据库密码，密码设置成功后，命令行窗口会继续打印执行信息，直至提示“init fxy success”完毕。

./init.sh

如果是 Debian 等衍生系统，包括 Ubuntu 等，使用 sudo bash init.sh 执行。

```
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    ./pg/bin/pg_ctl -D data/ -l logfile start


Please enter 8-bit database password:

Please enter the database password again:

Password set successfully!
```

步骤 3：启停服务

nohup ./startup.sh &

如果是 Debian 等衍生系统，包括 Ubuntu 等，使用 sudo nohup bash startup.sh & 执行。

```
root@mysql bq]# nohup ./startup.sh &
1] 5468
root@mysql bq]# nohup: ignoring input and appending output to ' nohup.out'
```

查看 nohup.out 文件，看到 Server startup in ***** ms 说明启动成功。

tail -f nohup.out

查看全部启动日志。

cat nohup.out

若要停止服务，直接运行如下命令。

./shutdown.sh

如果是 Debian 等衍生系统，包括 Ubuntu 等，使用 sudo bash shutdown.sh 执行。

2.3 报表配置

打开分析云安装目录，进入 D:\bq\deploy\bq，修改 app.esc 文件。其中的 127.0.0.1 需要修改为服务器的 IP 地址。分析云的端口默认为 8080，如果分析云未改端口，这里不需要修改；如果分析云端口发生改变，这里同样需要修改。


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<app codebase="http://127.0.0.1:8080/bq">
    <name>BQReport</name>
    <id>BQReport</id>
    <version>1.0.0</version>
    <productVersion>1.0</productVersion>
    <category>BQReport</category>
    <icon>client/logo.png</icon>
    <description>BQReport</description>
    <resources>
        <j2se version="1.8.0" href="client/jre-8u91-windows-i586.zip" family="windows" sharable="true"/>
        <j2se version="1.8.0" href="client/jre-8u91-macosx-x64.zip" family="mac" sharable="true"/>
        <jar file="lib/bqReportDesigner.jar" href="client/bqReportDesigner.jar"/>
        <zip href="client/appletjar/client.zip" location="lib" optional="false"/>
    </resources>
    <start mode="1">
        <profile>
            <java main="com.yonyou.bq.home.start.BQStarter">
                <classpath resource="lib/bqReportDesigner.jar"/>
                <args jvm=" -Dbq.bqstart.server=127.0.0.1 -Dbq.bqstart.protocol=http -Dbq.bqstart.port=8080 -Dae_home=./" />
            </java>
        </profile>
    </start>
</app>
```

2.4 访问服务

启动成功后，就可以访问分析云服务：http://IP:8080/（注意，这里的 IP 是指服务器的 IP 地址） 。分析云预置了三个用户，分别是超级管理员 admin、业务管理员 bq 和设计者 design，密码默认 000000 。

注意：新建用户，默认密码是 000000；如果觉得默认密码过于简单，也可以自定

义，在安装目录 D:\bq\conf 下，修改 serverConfig 文件，其中有配置项：user.secret.default.mima，修改后需要重启环境。

如果需要使用报表服务，还需要下载 uclient，下载地址为：http://uclient.yonyou.com/

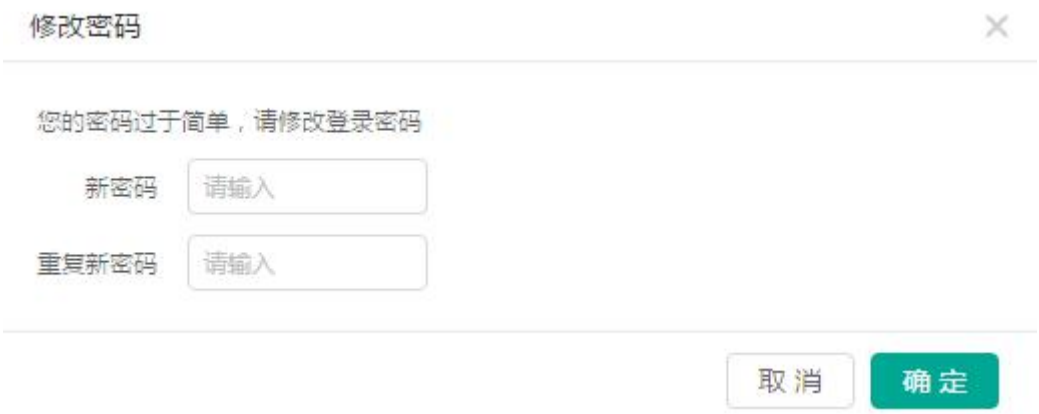
下载完成后，打开 uclient，添加应用，输入 http://IP:8080/bq，点击搜索，再点击添加，等待安装完成。

2.5 导入许可

启动服务后，必须要申请测试许可或正式许可，才能进一步操作。

在浏览器输入 http://IP:8080/打开网页，使用超级管理员 admin 登录，密码默认

是 000000。首次登陆需要修改密码。



登录后，选择系统设置-许可-生成硬件锁。选择对对应的网卡并输入产品号（例如可以输入：test），点击生成 hardkey.req 文件。

发送 hardkey.req 文件到邮箱 xiaah@yonyou.com，获取测试许可。**如果需要正式许可**，还需要发送**订单号、企业名称、激活码。**

点击导入授权，就可以导入许可。

2.6 文件服务配置

许可导入完成后，需要继续配置文件服务。该步骤是必须进行的步骤，否则后续图片上传等功能无法正常使用。

分析云支持两种存储文件的方式，分别为磁盘存储和 HBase 存储。通常使用磁盘存储即可，HBase 存储主要应用场景是需要安装使用大数据平台。如果选用磁盘存储，在基本设置中配置存储路径即可，这里路径支持 Windows 的远程共享磁盘以及 Linux 系统的挂载磁盘。

1.磁盘存储

1)基本设置

配置文件服务地址，此地址是分析云产品中文件上传后的保存位置。例如，分析云安装在目录 D:\bqhome，那么文件服务目录可以为 D:\bqhome\file。其中，外网 IP、外网端口、内网 IP、内网端口这 4 个配置项无特殊需要，不用配置，保持默认即可。

注意：服务器目录不要有中文、空格、下划线及其它特殊字符。

2)模块管理

文件服务对文件的管理采用模块化管理。不同模块的文件可单独进行压缩、加密存储。系统预置了一个 DEFAULT 模块，如无特殊需求，使用 DEFAULT 模块即可。

3)导出

导出功能对文件服务中的文件进行导出。导出的内容包含磁盘文件和数据库中文件元数据。导出可按文件上传时间、文件所属模块进行导出。导出目录为服务器目录。

4)导入

将导出的文件，在导入功能中进行导入。导入后，磁盘文件和数据库的文件元数据都会恢复到导出时的状态。导入导出功能用于文件数据备份、迁移场景。

5)诊断

为确保文件服务配置正确，诊断功能可以对文件服务进行上传、下载、删除测试。点击诊断，界面显示各项诊断结果。

2. HBase 存储

如果选用 HBase 存储文件，则需要进行两步操作：

第一步，修改分析云 home 下的配置文件，路径为 conf/serverConfig.properties，配置项为：

```
fs.hbase.zookeeper.quorum=host1,host2,host3
fs.hbase.rpc.timeout=2000
fs.hbase.client.retries.number=5
fs.hbase.client.pause=50
fs.hbase.client.operation.timeout=3000
```

其中各项参数根据实际情况进行配置。注意，fs.hbase.zookeeper.quorum 为 HBase 集群的 zookeeper 主机名，这里必须使用集群全部节点的映射主机名（在分析云服务器系统中的 Host 文件进行配置）。

第二步，更新资源库配置参数，在分析云资源库中执行以下 SQL：

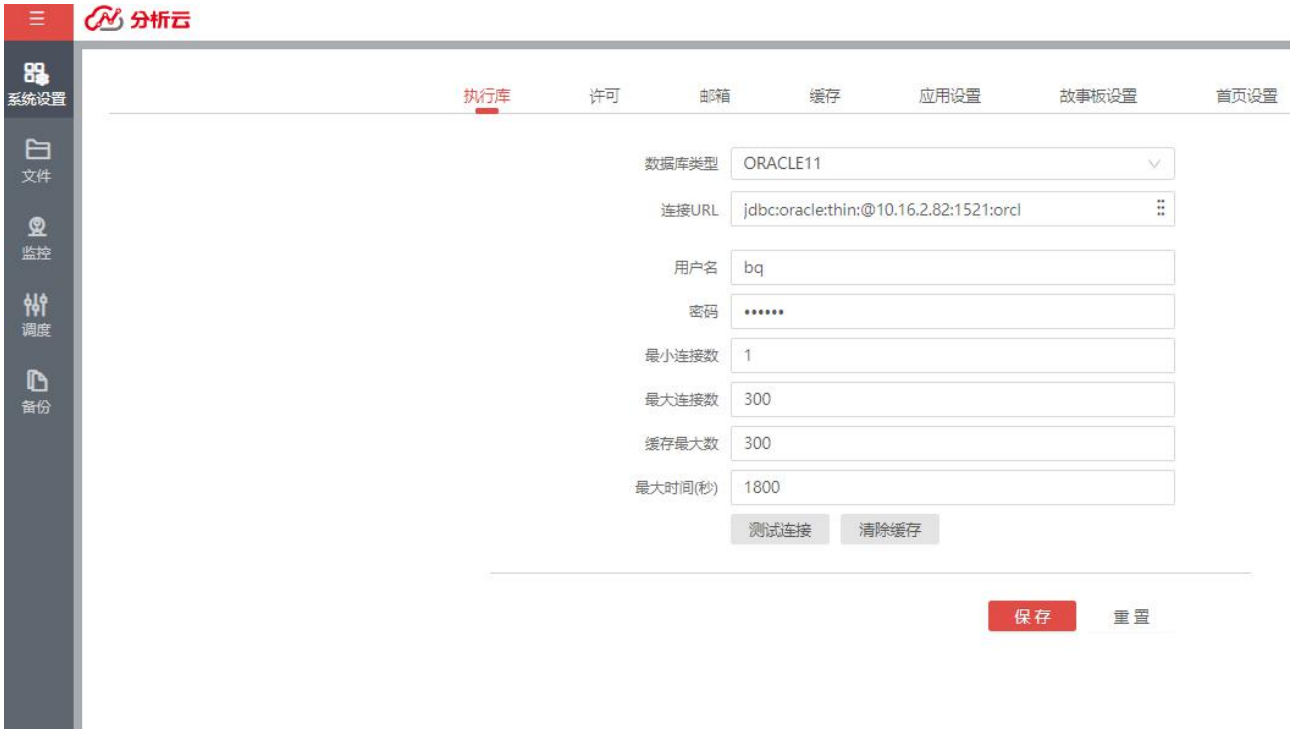
update console_fs_bucket set storetype = 1

这样就实现将文件存储在 HBase 中。

2.7 执行库配置

文件服务配置完成后，需要继续配置执行库。该步骤是必须进行的步骤，否则后续文件上传等功能无法正常使用。

选择系统设置-执行库，就可以配置。目前，执行库支持 Oracle、SQLServer、MySQL、达梦和 impala，注意，执行库至少需要以下权限：创建表、删除表、查询表、修改表、创建索引、删除索引、创建函数、删除函数、创建序列、删除序列、插入数据、删除数据、修改数据。



如果使用 Impala 作执行库时，应用服务器需要添加执行库的 hostname

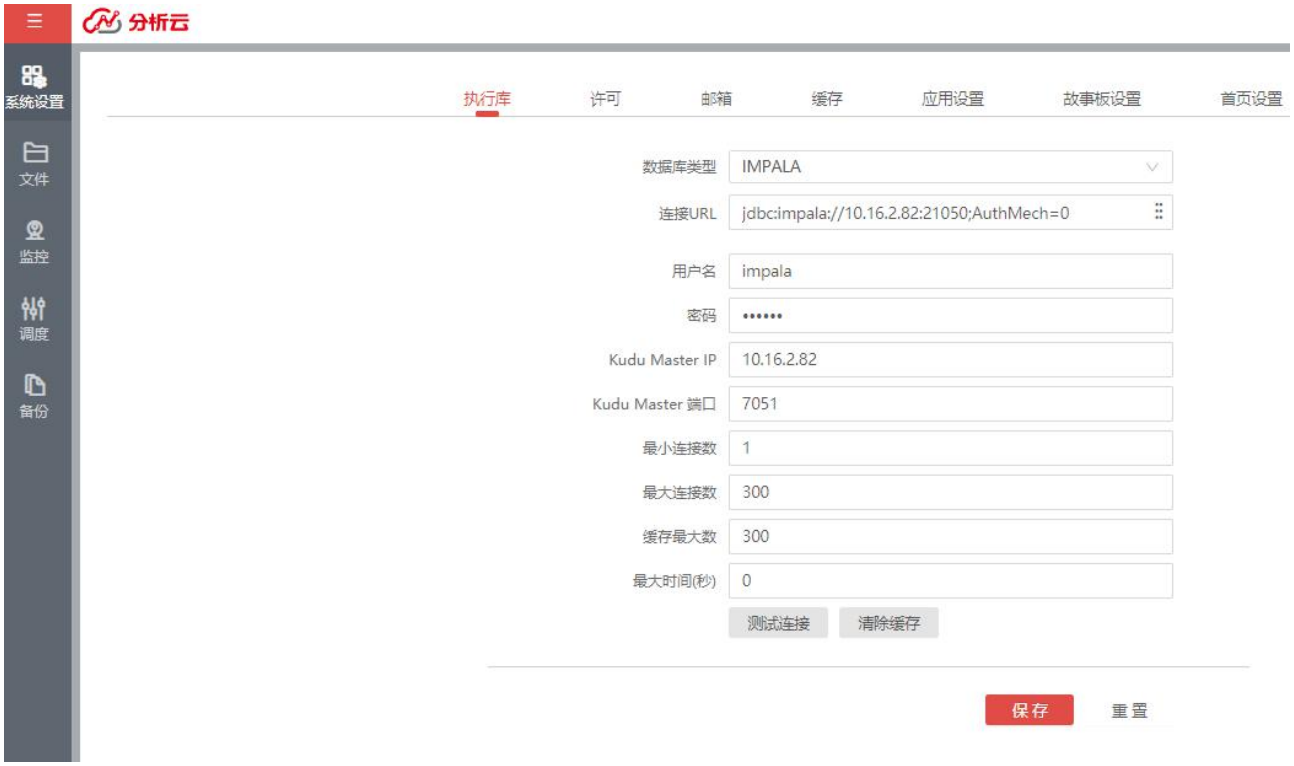
打开 C:\Windows\System32\drivers\etc（Linux 系统为 vi /etc/hosts），修改 hosts，添加执行库的 IP 和 hostname

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#127.0.0.1       localhost
#   ::1             localhost

[redacted] kudu1
[redacted] kudu2
[redacted] kudu3
```

impala 配置截图如下，安装文档会另行提供。



2.8 邮箱配置

邮箱服务配置是非必须步骤，用户可以等需要邮箱服务时，再配置。

选择系统设置-邮箱，输入用户邮箱服务器的相关信息，完成后可以发送测试邮件，测试通过保存即可。

2.9 Redis 设置

Redis 在分析云中主要用作缓存，集群必须使用 Redis 否则缓存将不一致，单机如果不配置 Redis 则使用的是内存作为缓存。

分析云支持 Redis 的版本有 3.0 以上版本，推荐使用 6.x 版本，支持的安装模式有单机，主从模式，Cluster 模式和哨兵模式,推荐使用哨兵模式。

在分析云的部署目录下的 conf/serverConfig.properties 配置文件中，找到如下配置：

```
#redis conf
redis.host =
redis.port = 6379
redis.password =
redis.user =
redis.maxTotal = 50
redis.maxIdle = 10
redis.minIdle = 5
```

```
redis.maxWaitMillis = 100
redis.sentinelsMode=
redis.masterName=
redis.databaseId=0
```

redis.sentinelsMode：如果是哨兵模式设置为 true，否则不设置或设置我 false。

redis.host：如果是单机直接在配置主机地址；主从或 cluster 模式设置 master 的地址，多个要用逗号隔开，如果不是默认端口，要加上端口；如果是哨兵模式则是哨兵的地址；地址格式如下：<host1:port1>,<host2:port2>,<host3:port3>.

redis.port：redis 端口，单机必须配置，集群或集群模式通常和地址写在一起，也可以不写使用该默认地址。

redis.user 和 redis.password 是分析云连接 redis 的用户名和密码。

redis.materName 是 Redis 哨兵模式的 maserName。

redis.databaseId 是指使用的 database 号，仅对单机和哨兵模式有效，主从和 Cluster 使用的是 database 0。

redis.maxTotal，redis.maxIdle，redis.minIdle，redis.maxWaitMillis 是 Redis 连接池的参数，分别是最大连接数，最大空闲数，最小连接数，最大等待毫秒。

3 信创环境安装

3.1 芯片和操作系统

服务器系统

银河麒麟高级服务器操作系统（飞腾版）V10

银河麒麟高级服务器操作系统（鲲鹏版）V10

银河麒麟高级服务器操作系统（龙芯版）V10

银河麒麟高级服务器操作系统（兆芯版）V10

银河麒麟高级服务器操作系统（海光版）V10

银河麒麟高级服务器操作系统（AMD64版）V10

统信服务器操作系统（海光版）V20

统信服务器操作系统（兆芯版）V20

统信服务器操作系统（龙芯版）V20

统信服务器操作系统（飞腾版）V20

统信服务器操作系统（鲲鹏版）V20

统信服务器操作系统（腾云版）V20

桌面系统

银河麒麟桌面操作系统（飞腾版）V10

银河麒麟桌面操作系统（鲲鹏版）V10

银河麒麟桌面操作系统（龙芯版）V10

银河麒麟桌面操作系统（兆芯版）V10

银河麒麟桌面操作系统（海光版）V10

银河麒麟桌面操作系统（AMD64版）V10

统信桌面操作系统（海光版）V20

统信桌面操作系统（兆芯版）V20

统信桌面操作系统（龙芯版）V20

统信桌面操作系统（飞腾版）V20

统信桌面操作系统（鲲鹏版）V20

统信桌面操作系统（腾锐版）V20

3.2 资源库配置和初始化

步骤 1：解压分析云

新建目录/home/bq，上传 BQCloud_BI_7.7.0_noarch_Linux.tar.gz 到该目录。

mkdir /home/bq

进入目录，解压安装包到当前目录。

cd /home/bq

tar -zxvf BQCloud_BI_7.7.0_noarch_Linux.tar.gz

步骤 2：资源库配置

进入分析云解压目录/home/bq/conf下，编辑修改serverConfig.properties文件。该文件中资源库，默认是PostgreSQL，如下截图。这里需要使用其它数据库，那么需要注释掉PostgreSQL，然后去掉需要使用的数据库的注释，并修改IP、端口及用户密码等信息。

注意：需要提前查看操作系统是否安装了 jdk，如果没有，需要安装 jdk 并设置 java 环境变量。可以参看章节 4.1，也可以自行去网络寻找相关安装资料。

```
#ServerConfig
#Tue, 13 Apr 2021 01:42:10 +0800
#repo postgreSql database setting
jdbc.dialect=com.yonyou.bq.framework.repository.BQPostgreSQL95Dialect
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc\:postgresql\://localhost\:5432/model_design
jdbc.username=pg
jdbc.mima=Encrypt:2be98afc86aa7f2e4fa4bfd248bc4f882
jdbc.validationQuery=select 1

#jdbc.dialect=org.hibernate.dialect.DmDialect
#jdbc.driverClassName=dm.jdbc.driver.DmDriver
#jdbc.url=jdbc:dm://172.16.75.6:5236
#jdbc.username=fxy_repo
#jdbc.password=yonyou@123
#jdbc.validationQuery=select 1

#jdbc.dialect=org.hibernate.dialect.Oracle10gDialect
#jdbc.driverClassName=oracle.jdbc.OracleDriver
#jdbc.url=jdbc:oracle:thin:@172.16.75.6:1521:aedb
#jdbc.username=CDC_TEST
#jdbc.password=1
#jdbc.validationQuery=select 1 from dual

#jdbc.dialect=org.hibernate.dialect.MySQL5Dialect
#jdbc.driverClassName=com.mysql.jdbc.Driver
#jdbc.url=jdbc:mysql://localhost:3306/model_design?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
#jdbc.username=root
#jdbc.password=123456
#jdbc.validationQuery=select 1

#jdbc.dialect=org.hibernate.dialect.MySQL8Dialect
#jdbc.driverClassName=com.mysql.cj.jdbc.Driver
#jdbc.url=jdbc:mysql://172.20.59.75:3306/bqdatasource?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
#jdbc.username=root
#jdbc.password=fenxiyun@%123
#jdbc.validationQuery=select 1
```

步骤 3：资源库初始化

执行分析云解压目录下的 initRemoteRepoDB.sh 文件，进行资源库的初始化，当程序执行完成会提示退出。（注意：当资源库中已经有要初始化的数据库表时，

脚本程序会提示是否要覆盖资源库中已经存在的表，如果选择覆盖，会删除已经存在的表，重新创建表并初始化）。

3.3 默认中间件的部署

默认中间件是 tomcat，执行分析云解压目录（/home/bq）下的 init.sh 文件，就可以部署分析云。

至此，分析云信创环境部署完成。

如果项目不能使用 tomcat，分析云还支持东方通中间件，具体参看下一章节。

3.4 东方通中间件的部署

步骤 1：安装 TongWeb7

参照《TongWeb7 用户使用手册.pdf》安装东方通中间件。例如，东方通安装在目录/home/tong/twns。

步骤 2：拷贝基础 jar 包

将/home/bq/app/common-lib/lib 目录下全部 jar 包，复制到东方通安装目录下的 lib/common 目录中。

cp /home/bq/app/common-lib/lib/* /home/tong/twns/lib/common

步骤 3：启动服务并登录控制台

参照《TongWeb7 用户使用手册.pdf》启动 TongWeb7，并登录管理控制台。

浏览器中输入管理控制台网址:

http://IP:端口/console

这里的 IP 为服务器 IP，端口为安装东方通时设置的控制台服务端口。默认登录用户名密码为：thanos/thanos123.com

步骤 4：配置启动参数

进入管理控制台，选择左侧导航树的“启动参数配置”选项，如下图。

右侧启动参数配置页面，选择“服务器参数”，如下图。

点击添加按钮，添加如下服务器参数：

-Dae_home=/home/bq/
-Dae_home_config=/home/bq/conf
-Dspring.profiles.active=single
-Dfile.encoding=UTF-8
-Dlog4j.configurationFile=file:///home/bq/conf/log4j2.xml
-DMODULE_PACKAGE_CODE=31

其中，/home/bq/为前述3.2中的分析云解压目录。

其中，MODULE_PACKAGE_CODE值，需要和分析云解压目录中/home/bq/tomcat/bin/catalina.bat文件中的MODULE_PACKAGE_CODE的数值一样。

添加完服务器参数，点击保存，重启 TongWeb 服务器。

如何重启 TongWeb 服务，请参照《TongWeb7 用户使用手册.pdf》。

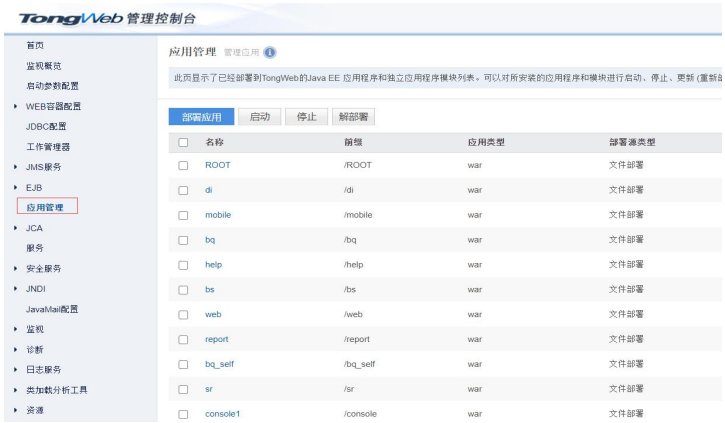
步骤 5：部署 war 包

再次进入管理控制台，依次部署如下 war 包（这些 war 包在分析云解压目录的 app 下）：

web.war、help.war、mobile.war、report.war、console.war、ROOT.war、sr.war、bq.war、bq_self.war、bs.war、di.war、spark_web.war

war 包的部署顺序：需要按照此顺序依次部署。以 web.war 举例。

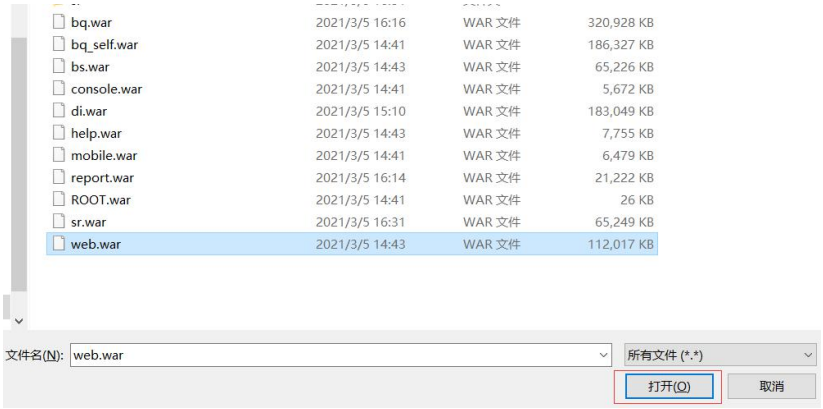
选择左侧导航树“应用管理”选项，如下图。



在右侧应用管理操作页面，点击“部署应用”按钮。

进入如下页面。

文件位置项，选择“本机”，然后点击“选择文件”按钮，进入分析云解压目录下的 app 文件夹，选择要部署的 war。



选择 web.war 文件后，点击打开按钮，进入如下界面。

点击“开始部署”按钮，进入如下界面。

上图界面的所有输入项，保持默认即可。

注意 1：当部署 console.war 时,由于东方通管理控制台的应用名就叫 console，所以“应用名称”栏不能用默认值，要改为 console1

注意 2：当部署 ROOT.war 时，“应用前缀“栏需要改为/（即把默认的/ROOT 改为/）

点击下一步，进入如下界面。

虚拟主机选择默认的 server 即可，直接点击完成，或者点击下一步然后在点击完

成。

按照上述部署 war 包的描述，依次部署其他 war 包。

按上述步骤部署完成，在浏览器输入分析云访问地址（IP 是服务器的 IP 地址，端口号是安装 TongWeb 时配置的应用服务器的端口号），即可访问到分析云登录页面。

4 其他环境自定义安装

分析云是使用 java8 开发的应用，理论上只要是能够有 jdk8 支持的操作系统就能够运行分析云。数据库能否安装主要看是否支持的数据库能否支持该操作系统，通常开源的数据库可以编译安装。

4.1 操作系统和 JDK

步骤 1： 下载适合 CPU 架构和操作系统的 JDK，要下载 JAVA 8 版本：

OracleJDK 下载地址（Oracle JDK8-201 以后不是免费的了）：

https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

IBM JDK 下载

https://developer.ibm.com/javasdk/downloads/sdk8/

步骤 2：安装 JDK

chmod a+x ibm-java-ppc64-sdk-8.0-6.11.bin

执行命令安装。

./ibm-java-ppc64-sdk-8.0-6.11.bin

安装过程是交互式的，需要回答一些问题，并且可以自主选择安装的路径。

安装成功后会显示文件安装路径：

******** successfully

installed to:

/opt/ibm/java-ppc64-80

步骤 3： 设置环境变量

用 vi 编辑/etc/profile，添加如下：

JAVA_HOME=/home/ibm/java-ppc64-80

JRE_HOME=/home/ibm/java-ppc64-80/jre

PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib

:

export JAVA_HOME JRE_HOME PATH CLASSPATH

重新加载 profile 文件：

source /etc/profile

4.2 资源库安装

注：资源库使用 MySQL5.7 和 8.0，PostgreSQL9.5 和 10 都可以，这里举例使用了 PostgreSQL。

需要用到的软件：

gcc

readline

1) 下载源码包

https://www.postgresql.org/ftp/source/v9.5.1/

2) 解压

tar -zxvf postgresql-9.5.1.tar.gz

3) 添加用户及用户组

增加组：

mkgroup postgres

增加用户名：

useradd -d /usr/local/pgsql -g postgres -m postgres

passwd postgres

4) 安装数据库

a. 进入 postgresql-9.5.1 目录，安装 PostgreSQL 数据库

配置：

./configure --prefix=/usr/local/pgsql/

编译：

gmake

安装：

gmake install

b. 进入 postgresql-9.5.1/contrib/dblink 目录下,安装 dblink

编译：

gmake

安装:

gmake install

到此，库安装结束

注意：建议数据库与分析云安装在同一台服务器，如果为同一台服务器，接下来的步骤不用操作，直接跳转到 4.3 power 架构下的分析云安装。

5) 初始化数据库

/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data --locale=C --encoding=UTF8

a. 将/usr/local/pgsql/data 下的 pg_hba.conf 替换为分析云安装包中 db 下的 pg_hba.conf

b. 将/usr/local/pgsql/data 下的 postgresql.conf 替换为分析云安装包中 db 下的 postgresql.conf

6) 修改环境变量

vi /etc/profile

主要添加内容：

PG_HOME=/usr/local/pgsql

PGDATA=$PG_HOME/data

PATH=$PG_HOME/bin:$PATH

PGLIB=$PG_HOME/lib

export PG_HOME PGDATA PATH PGLIB

source /etc/profile（使环境变量生效）

7) 初始化数据库

a. 切换到 bin 目录下

cd /usr/local/pgsql/bin/

b. 启动数据库：

pg_ctl -w -D ../data start

c. 创建用户和数据库

#./createuser -d pg -p 5432

#./createuser -d -s replication

#./createdb -O pg model_design -p 5432

d. 创建表

将 resources 文件夹复制到/usr/local 下，并将下列代码编写成一个可执行的脚本，脚本名称自定义，要求脚本要与 resources 文件夹在一个层级，脚本执行成功后，查看数据库表是否创建成功。

```
currentdir=`pwd "$0"`
postgrescripts="$currentdir"/resources/scripts
filelist=`ls $postgrescripts`
for file in $filelist
do
#echo $postgrescripts/$file
        templist=`ls $postgrescripts/$file/postgre`
        for f in $templist
        do
   #echo $f
            echo $postgrescripts/$file/postgre/$f
            su -c "db/pgsql/bin/psql -U pg -d model_design -f $postgrescripts/$file/postgre/$f -h localhost -p $dbport" postgre
        done
done
```

4.3 分析云安装

安装包上传成功解压后先进行以下操作：

1) 修改 startup.sh

export JRE_HOME=”$currentdir“/jre8

替换为

export JRE_HOME=/home/ibm/java-ppc64-80/jre

2) 修改 shutdown.sh

export JRE_HOME=”$currentdir”/jre8

export JRE_BIN_HOME=”$currentdir”/jre8/bin/

替换为

export JRE_HOME=/home/ibm/java-ppc64-80/jre

export JRE_BIN_HOME=/home/ibm/java-ppc64-80/jre/bin/

同时在在 catalina.sh 的 156 行后，157 行添加

JAVA_OPTS="$JAVA_OPTS

-Dcom.ibm.crypto.provider.DoRSATypeChecking=false"

3) 修改 serverConfig.properties 配置文件

如果 postgre 数据库与分析云不在同一台服务器上，修改 conf/serverConfig.properties 文件：jdbc.url 中的 localhost 改为数据库服务器的 ip 地址。

修改完成后按照 2.2 Linux 平台安装进行安装部署。

5 高可用集群安装

5.1 Nginx 安装配置

5.1.1 编译安装

1、从 http://nginx.org/en/download.html 下载最新稳定版安装包，目前为 nginx-1.16.1.tar.gz
2、为了让 nginx 能够对 Session 进行支持，我们还需要安装一个补丁 nginx-sticky-module，该补丁可以从 https://bitbucket.org/nginx-goodies/nginx-sticky-module-ng/downloads/下载
3、将 nginx-1.12.1.tar.gz，和 nginx-sticky-module 补丁 nginx-goodies-nginx-sticky-module-ng-08a395c66e42.zip 上传至服务器，这里上传到了/home 目录下
4、cd /home 进入到/home 目录下，使用 tar -zxvf nginx-1.12.1.tar.gz 解压安装包，
使用 unzip nginx-goodies-nginx-sticky-module-ng-08a395c66e42.zip 解压补丁
5、执行 mv nginx-goodies-nginx-sticky-module-ng-08a395c66 nginx-1.12.1/nginx-sticky-module
6、 执行如下命令进行安装

```
   cd nginx-1.16.1

   ./configure --user=www --group=www --prefix=/home/nginx
--with-http_stub_status_module --with-http_ssl_module
--with-http_gzip_static_module --add-module=nginx-sticky-module

   make

   make install
```

7、 Nginx 配置建议适用下面的配置文件，在 133-134 修改为实际分析云部署的服务器 ip。

nginx.conf

8、 执行 sbin/nginx 命令即可启动服务

5.1.2 Nginx 启动关闭重新加载配置

使用 sbin/nginx 命令启动 nginx
使用 ps -ef | grep nginx 可以查看 nginx 的进程
使用 sbin/nginx -s stop 或 pkillnginx 命令停止 nginx
使用 sbin/nginx -s reload 命令重新加载配置文件，该操作不停止 nginx 服务器，而将配置文件的修改读入并使之生效

5.2 Redis 安装配置

准备的主机如下：

hostname	ip
h1	192.168.0.181
h2	192.168.0.182

h3	192.168.0.183

5.2.1 Redis 的单机安装

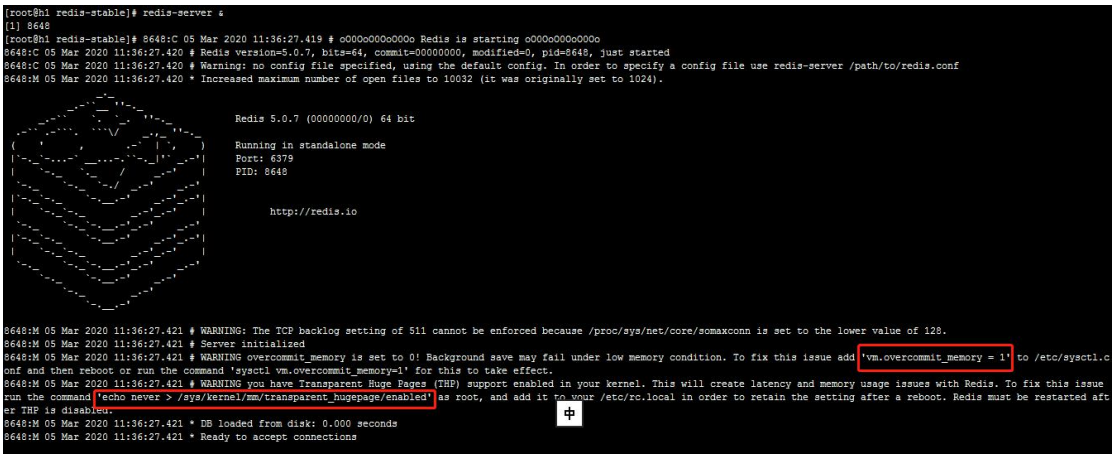
我们准备在 h1,h2,h3 三台主机上分别安装 redis。每台主机上的安装步骤如下。

将安装介质 redis-stable.tar.gz 上传到服务器上/home 目录下，依次执行以下命令进行安装：

```
cd /home
tar -zxvf redis-stable.tar.gz
cd /home/redis-stable
make
make install
```

安装完成！

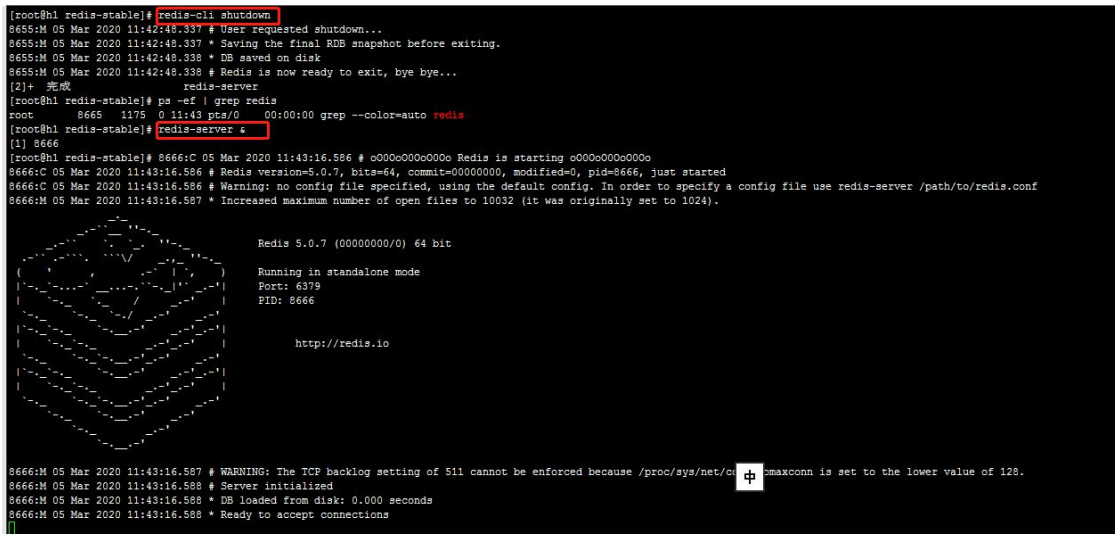
用 `redis-server &` 命令测试启动 redis：



会有警告信息，按提示去除警告信息，需要依次执行：

```
sysctl vm.overcommit_memory=1
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

先用 `redis-cli shutdown` 命令停止 redis，再用 `redis-server &` 命令启动 redis，你会发现警告信息没有了，如下图。



5.2.2 主从复制

首先，为使 redis 对外可用，要配置一下各主机上的 redis.conf 文件（我的在/home/redis-stable/redis.conf），首先搜索 bind 127.0.0.1，并将改行注释掉，然后搜索 protected-mode，并将该值设置为 no，再搜索 daemonize，将其值设置为 yes。

```
#bind 127.0.0.1
protected-mode no
daemonize yes
```

后续要启动 redis 时都需要指定 redis.conf，如：

```
redis-server /home/redis-stable/redis.conf
```

接下来，我们准备搭建 1 个一主两从的 redis 主从复制系统，角色分配如下：

主	从	从
h1:6379	h2:6379	h3:6379

然后再在 3 台主机上各设立一个哨兵，每个哨兵都同时监控着这个主从系统。

分别在 3 台主机上执行如下相应命令来启动集群（**注意先启动 master，再启动 slave**）。

h1	redis-server /home/redis-stable/redis.conf --port 6379
h2	redis-server /home/redis-stable/redis.conf --port 6379 --slaveof 192.168.0.181 6379
h3	redis-server /home/redis-stable/redis.conf --port 6379 --slaveof 192.168.0.181 6379

执行完后可以依次使用 redis-cli 连接各个 redis 节点并用 info replication 以查看节点状态，如下 3 个图所示。

```
[root@h1 redis-stable]# redis-cli -h 192.168.0.181 -p 6379
192.168.0.181:6379> info replication
# Replication
role:master
connected_slaves:2
slave0:ip=192.168.0.182,port=6379,state=online,offset=504,lag=1
slave1:ip=192.168.0.183,port=6379,state=online,offset=504,lag=1
```

```
[root@h1 redis-stable]# redis-cli -h 192.168.0.182 -p 6379
192.168.0.182:6379> info replication
# Replication
role:slave
master_host:192.168.0.181
master_port:6379
```

```
[root@h1 redis-stable]# redis-cli -h 192.168.0.183 -p 6379
192.168.0.183:6379> info replication
# Replication
role:slave
master_host:192.168.0.181
master_port:6379
```

接下来测试主从复制效果：

先用 redis-cli 连接 master，并设置键 hello 的值为 world，然后再用 redis-cli 连接其中一个 slave，获取键 hello 的值，发现其值 world，说明我们成功的搭建的 redis 主从复制。

```
[root@h1 redis-stable]# redis-cli -h 192.168.0.181 -p 6379
192.168.0.181:6379> set hello world
OK
192.168.0.181:6379> exit
[root@h1 redis-stable]# redis-cli -h 192.168.0.182 -p 6379
192.168.0.182:6379> get hello
"world"
192.168.0.182:6379>
```

5.2.3 哨兵

在主从复制基础上，接下来我们分别在 h1,h2,h3 上个设置一个哨兵，3 个哨兵均配置为监视这个主从复制系统。

说明：为什么需要分别在 3 个主机上各设置一个哨兵呢？

这是为了避免出现哨兵的单点问题。设立 3 个哨兵的话，就算其中一个哨兵 down 了，其它的哨兵仍然能够负责自动主从切换的工作。

1，首先在 h1 上设置一个哨兵。

编辑/home/redis-stable/sentinel.conf 文件。

```
#搜索 daemonize，将其值设置为 yes
daemonize  yes

#搜索 logfile 冰将其值设置为"/home/redis-stable/sentinel.log"
logfile  "/home/redis-stable/sentinel.log"

#搜索 sentinel  monitor 并将默认的一行配置进行修改
sentinel  monitor  mymaster  192.168.0.181  6379  2
```

运行哨兵：redis-sentinel /home/redis-stable/sentinel.conf

并查看日志：cat /home/redis-stable/sentinel.log

```
[root@h3 redis-stable]# redis-sentinel /home/redis-stable/sentinel.conf
[root@h3 redis-stable]# cat /home/redis-stable/sentinel.log
9181:X 05 Mar 2020 14:27:57.681 # o000o000o000o Redis is starting o000o000o000o
9181:X 05 Mar 2020 14:27:57.682 # Redis version=5.0.7, bits=64, commit=00000000, modified=0, pid=9181, just started
9181:X 05 Mar 2020 14:27:57.682 # Configuration loaded
9182:X 05 Mar 2020 14:27:57.686 * Increased maximum number of open files to 10032 (it was originally set to 1024).
9182:X 05 Mar 2020 14:27:57.687 * Running mode=sentinel, port=26379.
9182:X 05 Mar 2020 14:27:57.687 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/c  omaxconn is set to the lower value of 128.
9182:X 05 Mar 2020 14:27:57.688 # Sentinel ID is 3824b7a2ff288965e4cc38cc5677c42d83ceef6a
9182:X 05 Mar 2020 14:27:57.688 # +monitor master mymaster 192.168.0.181 6379 quorum 2
9182:X 05 Mar 2020 14:27:57.691 * +slave slave 192.168.0.182:6379 192.168.0.182 6379 @ mymaster 192.168.0.181 6379
9182:X 05 Mar 2020 14:27:57.691 * +slave slave 192.168.0.183:6379 192.168.0.183 6379 @ mymaster 192.168.0.181 6379
```

通过日志我们知道，哨兵监控了 1 个主从系统，并且发现了这个主从系统的 master 及这个 master 的两个 slave。

至此，h1 上的哨兵就设置成功了。

2，接下来再在 h2 和 h3 上用同样的方法各设置一个哨兵。

3，测试哨兵：

我们在 h1 上执行如下命令停止掉这个主从系统的 master（即 h1 上的那个 redis 服务）：

```
redis-cli -p 6379 shutdown
```

/home/redis-stable/sentinel.conf 中会增加如下日志：

```
8818:X 05 Mar 2020 14:33:09.934 # +sdown master mymaster 192.168.0.181 6379
8818:X 05 Mar 2020 14:33:10.004 # +new-epoch 1
8818:X 05 Mar 2020 14:33:10.005 # +vote-for-leader f400ed9960c4a7332c50845efb8f63d6e6d9a275 1
8818:X 05 Mar 2020 14:33:10.011 # +odown master mymaster 192.168.0.181 6379 #quorum 2/2
8818:X 05 Mar 2020 14:33:10.011 # Next failover delay: I will not start a failover before Thu Mar  5 14:39:10 2020
8818:X 05 Mar 2020 14:33:11.188 # +config-update-from sentinel f400ed9960c4a7332c50845efb8f63d6e6d9a275 192.168.0.182 @ mymaster 192.168.0.181 6379
8818:X 05 Mar 2020 14:33:11.188 # +switch-master mymaster 192.168.0.181 6379 192.168.0.182 6379
8818:X 05 Mar 2020 14:33:11.189 * +slave slave 192.168.0.183:6379 192.168.0.183 6379 @ mymaster 192.168.0.182 6379
8818:X 05 Mar 2020 14:33:11.189 * +slave slave 192.168.0.181:6379 192.168.0.181 6379 @ mymaster 192.168.0.182 6379
8818:X 05 Mar 2020 14:33:41.207 # +sdown slave 192.168.0.181:6379 192.168.0.181 6379 @ mymaster 192.168.0.182 6379
```

该日志说明，哨兵发现了 master 已经断开，然后将 h2 上的 slave 提升为了 master，并将 h1 和 h3 设为了 slave。

我们可以通过查看节点状态验证一下，确实如以上日志所说：

```
[root@h1 redis-stable]# redis-cli -p 6379 -h 192.168.0.181
Could not connect to Redis at 192.168.0.181:6379: Connection refused
not connected> exit
[root@h1 redis-stable]#
```

```
[root@h1 redis-stable]# redis-cli -p 6379 -h 192.168.0.182
192.168.0.182:6379> info replication
# Replication
role:master
connected_slaves:1
slave0:ip=192.168.0.183,port=6379,state=online,offset=721545,lag=1
```

```
[root@h1 redis-stable]# redis-cli -p 6379 -h 192.168.0.183
192.168.0.183:6379> info replication
# Replication
role:slave
master_host:192.168.0.182
master_port:6379
```

接着我们再启动被我们停了的 h1 上的 redis 节点：

```
redis-server /home/redis-stable/redis.conf --port 6379
```

其中一个哨兵节点的/home/redis-stable/sentinel.conf 中会增加如下日志：

```
9182:X 05 Mar 2020 14:46:01.845 # -sdown slave 192.168.0.181:6379 192.168.0.181 6379 @ mymaster 192.168.0.182 6379
9182:X 05 Mar 2020 14:46:11.836 * +convert-to-slave slave 192.168.0.181:6379 192.168.0.181 6379 @ mymaster 192.168.0.182 6379
```

该日志说明，h1 上的 redis 节点又被重新启动了，并被作为新 master 的 slave 加入到集群中了。我们可以通过查看节点状态验证一下，确实如以上日志所说：

```
[root@h1 redis-stable]# redis-cli -p 6379 -h 192.168.0.181
192.168.0.181:6379> info replication
# Replication
role:slave
master_host:192.168.0.182
master_port:6379
```

```
[root@h1 redis-stable]# redis-cli -p 6379 -h 192.168.0.182
192.168.0.182:6379> info replication
# Replication
role:master
connected_slaves:2
slave0:ip=192.168.0.183,port=6379,state=online,offset=886784,lag=0
slave1:ip=192.168.0.181,port=6379,state=online,offset=886784,lag=0
```

以上说明，我们的哨兵工作良好。

5.2.4 Redis-Cluster

哨兵与集群是两个独立的功能，但从特性来看哨兵可以视为集群的子集，当不需要数据分片或者已经在客户端进行分片的场景下哨兵就足够使用了。
但如果需要进行水平扩容，则集群是一个非常好的选择。

为了使用集群，我们需要将每个 redis 节点的 cluster-enabled 配置选项打开。所以我们应编辑三台主机上的/home/redis-stable/redis.conf，搜索 cluster-enabled 并将该行配置打开。

每个集群中至少需要 3 个主数据库才能正常运行。为了演示集群的应用场景以及故障恢复等操作，这里以配置一个 3 主 3 从的集群系统为例。

1，创建 redis 节点工作目录

```
cd /home/redis-stable
mkdir cluster
mkdir cluster/7000
mkdir cluster/7001
```

2，在 h1，h2，h3 上各执行如下命令以启动两个 redis 节点：

```
cd /home/redis-stable/cluster/7000
redis-server /home/redis-stable/redis.conf --pidfile /var/run/redis_7000.pid --port 7000 --cluster-config-file nodes_7000.conf
cd /home/redis-stable/cluster/7001
redis-server /home/redis-stable/redis.conf --pidfile /var/run/redis_7001.pid --port 7001 --cluster-config-file nodes_7001.conf
```

在每台主机上执行 `ps -ef | grep redis` 以确定 3 台主机都成功启动了两个 redis 节点：

```
[root@h2 redis-stable]# ps -ef | grep redis
root       8802      1  0 15:40 ?        00:00:00 redis-server *:7000 [cluster]
root       8804      1  0 15:40 ?        00:00:00 redis-server *:7001 [cluster]
root       8812   1180  0 15:41 pts/0    00:00:00 grep --color=auto redis
```

3，将上面启动的 6 个 redis 节点加到一个集群中

```
redis-cli --cluster create 192.168.0.181:7000 192.168.0.181:7001 192.168.0.182:7000 192.168.0.182:7001 192.168.0.183:7000 192.168.0.183:7001 --cluster-replicas 1
```

注：redis-trib.rb 已过时了，用 redis-cli --cluster

```
[root@h1 7001]# redis-cli --cluster create 192.168.0.181:7000 192
>>> Performing hash slots allocation on 6 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 192.168.0.182:7001 to 192.168.0.181:7000
Adding replica 192.168.0.183:7001 to 192.168.0.182:7000
Adding replica 192.168.0.181:7001 to 192.168.0.183:7000
M: a9a3a4abc8dbfd1277e305c4c6edff1ad0cf59ec 192.168.0.181:7000
   slots:[0-5460] (5461 slots) master
S: cff708624442458e1d1ac69a7cf732d4ee856631 192.168.0.181:7001
   replicates e70ce95588371f030bcb2971f92d9b1df8990d5d
M: 5f99ca0c5581fc042f8eb71135545f4df81fbe66 192.168.0.182:7000
   slots:[5461-10922] (5462 slots) master
S: e520f63830bf6e6225aaaff3378380a2dd9835e7 192.168.0.182:7001
   replicates a9a3a4abc8dbfd1277e305c4c6edff1ad0cf59ec
M: e70ce95588371f030bcb2971f92d9b1df8990d5d 192.168.0.183:7000
   slots:[10923-16383] (5461 slots) master
S: 8e65a67714c0afd4937db540027ae448ffbe4926 192.168.0.183:7001
   replicates 5f99ca0c5581fc042f8eb71135545f4df81fbe66
Can I set the above configuration? (type 'yes' to accept): yes
```

输入 yes

```
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
>>> Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join
.......
>>> Performing Cluster Check (using node 192.168.0.181:7000)
M: a9a3a4abc8dbfd1277e305c4c6edff1ad0cf59ec 192.168.0.181:7000
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: cff708624442458e1d1ac69a7cf732d4ee856631 192.168.0.181:7001
   slots: (0 slots) slave
   replicates e70ce95588371f030bcb2971f92d9b1df8990d5d
M: 5f99ca0c5581fc042f8eb71135545f4df81fbe66 192.168.0.182:7000
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
S: e520f63830bf6e6225aaaff3378380a2dd9835e7 192.168.0.182:7001
   slots: (0 slots) slave
   replicates a9a3a4abc8dbfd1277e305c4c6edff1ad0cf59ec
M: e70ce95588371f030bcb2971f92d9b1df8990d5d 192.168.0.183:7000
   slots:[10923-16383] (5461 slots) master
   1 additional replica(s)
S: 8e65a67714c0afd4937db540027ae448ffbe4926 192.168.0.183:7001
   slots: (0 slots) slave
   replicates 5f99ca0c5581fc042f8eb71135545f4df81fbe66
[OK] All nodes agree about slots configuration.
```

4，验证

先用 redis-cli 连接 h1:7000（加参数-c 可连接到集群），并设置键 hello 的值为 world，然后再用 redis-cli 连接 h2:7001，获取键 hello 的值，发现其值 world，说明我们成功的搭建的 redis 主从复制。

```
[root@h1 7001]# redis-cli -h 192.168.0.181 -p 7000 -c
192.168.0.181:7000> set hello world
OK
192.168.0.181:7000> exit
[root@h1 7001]# redis-cli -h 192.168.0.182 -p 7001 -c
192.168.0.182:7001> get hello
-> Redirected to slot [866] located at 192.168.0.181:7000
"world"
192.168.0.181:7000> exit
```

5.3 PostgreSQL 资源库集群安装配置

5.3.1 Centos7 安装 PostgreSQL

准备好安装介质，去 pg 官网下载好要安装的 pg 包（postgresql-9.6.0.tar.gz），并上传到 centos 服务器上（比如/opt 目录下）。

下载 uuid ossp 库（uuid-1.6.2.tar.gz）并上传到服务器上。

1，安装必要基本软件，执行以下命令。

```
yum install -y gcc.x86_64 glibc.x86_64 glibc-devel.x86_64 vim-enhanced.x86_64 gcc-java apr apr-devel openssl openssl-devel libgcc.x86_64 java-1.8.0-openjdk.x86_64 java-1.8.0-openjdk-devel.x86_64 perl-Module-Install.noarch
yum install -y readline-devel.x86_64
```

2，创建 postgres 用户并设置密码

```
useradd postgres
passwd postgres
```

3，解压编译安装 uuid 库

```
cd /opt
tar -zxvf uuid-1.6.2.tar.gz
cd uuid-1.6.2
./configure
make
make install
```

4，解压编译安装 pg

```
cd /opt
tar -zxvf postgresql-9.6.0.tar.gz
cd postgresql-9.6.0
#配置 prefix 是程序放哪里
./configure --prefix=/home/postgres --enable-thread-safety --with-uuid=ossp --with-libs=/usr/local/lib --with-includes=/usr/local/include
make
make install
# 安装 contrib 工具包
```

```
cd contrib
make
make install
# 配置 uuid 的软连接，先查找 libuuid.so.16 的路径，再为找到的 libuuid.so.16 路径建立软连接到 postgres 的 lib 目录
find / -name libuuid.so.16
ln -s /usr/local/lib/libuuid.so.16 /home/postgres/lib
```

5，把 postgres 目录全部赋权给 postgres 用户

```
chown -R postgres.postgres /home/postgres/
```

6，配置环境变量

```
# 切换到 postgres 账户
su - postgres
# 编辑用户下配置文件
vim /home/postgres/.bashrc
```

编辑内容如下：

```
PGHOME=/home/postgres
export PGHOME
PGDATA=$PGHOME/data
export PGDATA
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$PGHOME/bin
export PATH
```

编辑完成，按 esc，输入 wq!保存退出，重新启用下配置文件

```
source /home/postgres/.bashrc
```

7，初始化数据库

```
#在 postgres 账户下执行
initdb -D $PGDATA
```

8，启动数据库

```
#在 postgres 账户下执行
pg_ctl start -D $PGDATA
```

9，设置数据库用户密码

```
#在 postgres 账户下执行
#使用 postgres 账户进入控制台(现在密码应该是空)
[postgres@pg1 ]# psql -U postgres
postgres=# \password
Enter new password: <123456>
Enter it again: <123456>
#把密码设置成 123456 可以使用\q 命令退出控制台
```

10，设置监听

修改 postgres/data 目录下的 pg_hba.conf

```
vim $PGDATA/pg_hba.conf
```

修改 IPv4 一行内容如下：

```
# IPv4 local connections:
host    all             all             0.0.0.0/0               trust
```

修改 postgresql.conf：

```
vim $PGDATA/postgresql.conf
```

修改监听一节如下：

```
# - Connection Settings -
listen_addresses = '*'
port = 5432
```

重启 pg 服务使配置生效：

```
pg_ctl restart -D $PGDATA
```

5.3.2 PostgreSQL 流复制热备

1，主备机规划

主机名	Ip	角色	端口
h1	192.168.0.181	master	5432
h2	192.168.0.182	slave	5432

首先在两台主机上按照前面介绍的《Centos7 安装 PostgreSQL》的 1~6 安装好 pg 数据库。

2，设置 host，master 和 slave 上都要操作。

```
su root
vim /etc/hosts
```

#编辑内容如下：

```
192.168.0.181 master
192.168.0.182 slave
```

3，初始化 master 库，并为其创建流复制账户

```
su - postgres
initdb -D $PGDATA
pg_ctl start -D $PGDATA
psql
#在 pg 的命令行中执行以下脚本创建流复制用户
CREATE USER repuser replication LOGIN CONNECTION LIMIT 3 ENCRYPTED PASSWORD 'repuser';
```

4，配置 master 的 pg_hba.conf，使得其内容如下：

```
local    all             all                                     md5
host     all             all             0.0.0.0/0               md5
host     all             all             0/0                     md5
host     replication     repuser         slave                   md5
```

5，配置 master 的 postgresql.conf 如下：

```
listen_addresses = '*'
port = 5432

max_wal_senders = 10
wal_level = replica
archive_mode = on
archive_command = 'cd ./'
hot_standby = on
wal_keep_segments = 64
full_page_writes = on
wal_log_hints = on
```

6，重启 master 数据库使配置生效

```
pg_ctl restart -D $PGDATA
```

7，在 salve 端的 postgres 账户下通过 pg_basebackup 创建备库

```
su - postgres
#从主库备份创建备库，此操作要输入 repuser 的密码，就输入前面设置的密码 repuser
pg_basebackup -D $PGDATA -Fp -Xs -v -P -h master -p 5432 -U repuser
```

8，修改 slave 中 data 目录下的 pg_hba.conf 最后一行修改如下：

```
host  replication        repuser        master                md5
```

9，配置 master 端的 recovery.done

```
cp $PGHOME/share/recovery.conf.sample $PGDATA/recovery.done
vim $PGDATA/recovery.done
```

编辑内容如下

```
recovery_target_timeline = 'latest'
standby_mode = on
primary_conninfo = 'host=slave port=5432 user=repuser password=repuser'
trigger_file = '/home/postgres/data/trigger_file'
```

10，配置 slave 端的 recovery.conf

```
cp $PGHOME/share/recovery.conf.sample $PGDATA/recovery.conf
vim $PGDATA/recovery.conf
```

编辑内容如下

```
recovery_target_timeline = 'latest'
standby_mode = on
primary_conninfo = 'host=master port=5432 user=repuser password=repuser'
trigger_file = '/home/postgres/data/trigger_file'
```

11，配置.pgpass

master 上配置访问 slave 参数：

```
vim $PGHOME/.pgpass
#内容如下：
slave:5432:postgres:repuser:repuser
```

slave 上配置访问 master 参数：

```
vim $PGHOME/.pgpass
#内容如下：
master:5432:postgres:repuser:repuser
```

12，流复制同步测试

分别启动 master，slave 数据库。

在 master 上修改数据库密码，然后创建一个数据库和一张表，

```
#使用 postgres 账户进入控制台(现在密码应该是空)
psql -U postgres
```

```
#设置密码
\password
```

```
#创建测试数据库和测试表
create database test;
\c test
create table tt(id serial not null,name text);
insert into tt(name) values ('china');
```

然后在 slave 上查询刚才创建的表和数据，判定是否有数据同步

```
psql -U postgres
\c test
select * from tt;
```

你会发现从库已经同步了主库的数据，到此可以说 PG 流复制热备已经创建结束了。

13，查询主备信息

主数据库是读写的，备数据库是只读的。

pg_controldata 命令可以用于查询当前 pg 的主备信息，查询结果中 Database cluster state 属性可以体现出来，值为 in production 表示当前的 pg 为 master，值为 in archive recovery 表示当前的 pg 为 slave。

14，主备切换

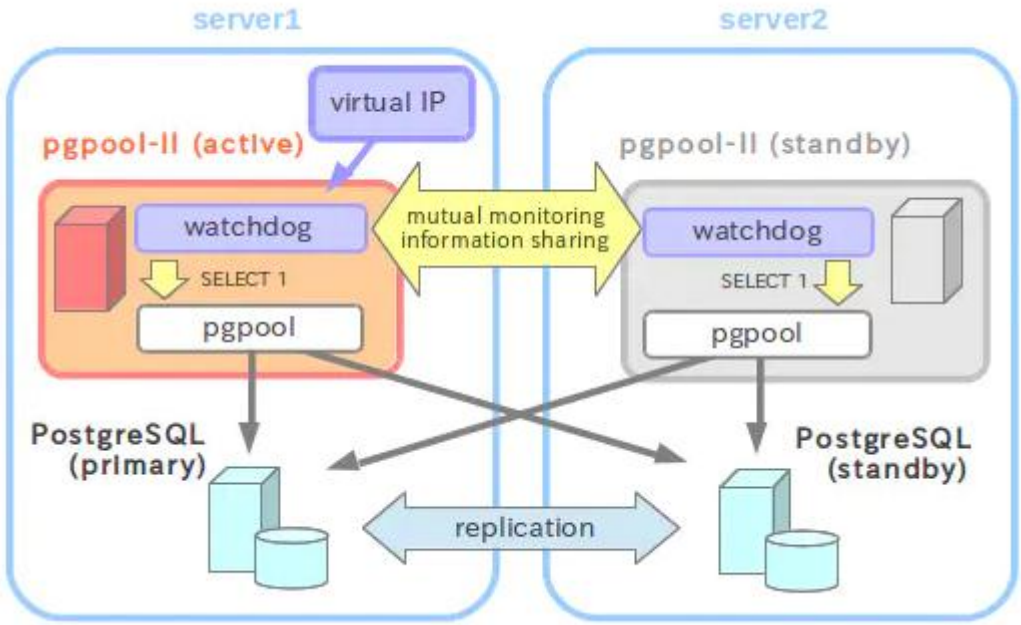
当主数据库宕机了，可以通过建立触发文件，将备数据库提升为主数据库，实现一些基本的 HA 应用。但是需要手动操作，比较麻烦，不推荐。后续会介绍 pgpool 来解决这个问题。

5.3.3 PGPool-II+PG 流复制实现 HA 主备切换

基于 PG 的流复制能实现热备切换，但是是要手动建立触发文件实现，对于一些 HA 场景来说，需要当主机 down 了后，备机自动切换， pgpool-II 可以实现这种功能。

在配置 pgpool 之前需分别在两台规划机上安装好 pg 数据库，且配置好了流复制环境，这个我们在《Centos7 安装 PostgreSQL》和《PostgreSQL 流复制热备》中已经完成了。

基于 PGPool 的双机集群如上图所示：pg 主节点和备节点实现流复制热备，pgpool1，pgpool2 作为中间件，将主备 pg 节点加入集群，实现读写分离，负载均衡和 HA 故障自动切换。两 pgpool 节点可以委托一个虚拟 ip 节点作为应用程序访问的地址，两节点之间通过 watchdog 进行监控，当 pgpool1 宕机时，pgpool2 会自动接管虚拟 ip 继续对外提供不间断服务。



1，主备机规划

主机名	Ip	角色	端口
h1	192.168.0.181	pg master	5432
h2	192.168.0.182	pgpool1	9999
h1	192.168.0.181	pg slave	5432
h2	192.168.0.182	pgpool2	9999
vip	192.168.0.150	虚拟 ip	9999

在 h1 和 h2 上设置 host

```
su root
vim /etc/hosts
```

#编辑内容如下：

```
192.168.0.181 master
```

```
192.168.0.182 slave
192.168.0.150 vip
```

修改完后，别忘了执行： source /etc/hosts

2，配置 ssh 密钥

在 master,slave 机器上都生成 ssh 如下：

```
su - postgres
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

分别将 master 的公钥复制到 slave，slave 的公钥复制到 master。：

```
#master 端
scp ~/.ssh/authorized_keys postgres@slave:~/.ssh/
#slave 端
scp ~/.ssh/authorized_keys postgres@master:~/.ssh/
```

验证下 ssh 配置是否成功：

```
#master 端
[postgres@h1 ~]$ ssh postgres@slave
Last login: ......
#slave 端
[postgres@h2 ~]$ ssh postgres@master
Last login: ......
```

exit 命令用于登出 ssh。

3，在 master 和 slave 上都安装上 pgpool

先准备好安装介质 pgpool-II-3.6.0.tar.gz，并上传到主机上/opt 路径下。
接下来安装 pg。

```
su - root
cd /opt
tar -zxvf pgpool-II-3.6.0.tar.gz
# 文件权限设置为 postgres（其实并非一定装在 postgres 账户，只不过之前 ssh 设置都在 postgres 下，为了方便）
chown -R postgres.postgres /opt/pgpool-II-3.6.0
mkdir /opt/pgpool
chown -R postgres.postgres /opt/pgpool
```

```
su - postgres
cd /opt/pgpool-II-3.6.0
./configure --prefix=/opt/pgpool --with-pgsql=path --with-pgsql=/home/postgres
make
make install
```

再安装必须的函数

```
#安装 pgpool_regclass
cd /opt/pgpool-II-3.6.0/src/sql/pgpool-regclass
make
make install
psql template1
=# CREATE EXTENSION pgpool_regclass;
```

```
#建立 insert_lock 表
cd /opt/pgpool-II-3.6.0/src/sql
psql -f insert_lock.sql template1
```

```
#安装 pgpool_recovery
cd /opt/pgpool-II-3.6.0/src/sql/pgpool-recovery
make
make install
psql template1
=# CREATE EXTENSION pgpool_recovery;
```

安装结束。

4，配置 pgpool

4.1，配置 pgpool 环境变量

pgpool 装在了 postgres 账户下，在该账户中添加环境变量，master,slave 节点都执行。

```
su - postgres
vim /home/postgres/.bashrc
```

编辑内容如下：

```
PGPOOLHOME=/opt/pgpool
export PGPOOLHOME
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$PGHOME/bin:$PGPOOLHOME/bin
export PATH
```

编辑完成后，按 esc 输入:wq 保存退出，然后执行：

```
source /home/postgres/.bashrc
```

4.2 配置 pool_hba.conf

pool_hba.conf 是对登录用户进行验证的，要和 pg 的 pg_hba.conf 保持一致，要么都是 trust，要么都是 md5 验证方式，这里采用了 md5 验证方式如下设置：

```
cd /opt/pgpool/etc
cp pool_hba.conf.sample pool_hba.conf
vim pool_hba.conf
```

```
#编辑内容如下
```

```
local    all          all                                   md5
host     all          all          0.0.0.0/0                md5
host     all          all          0/0                      md5
```

4.3 配置 pcp.conf

pcp.conf 配置用于 pgpool 自己登陆管理使用的，一些操作 pgpool 的工具会要求提供密码等，配置如下：

```
cd /opt/pgpool/etc
cp pcp.conf.sample pcp.conf
# 使用 pg_md5 生成配置的用户名密码，这里假设了你设置的 pg 密码为 123456
pg_md5 nariadmin    #会输出 e10adc3949ba59abbe56e057f20f883e
#pcp.conf 是 pgpool 管理器自己的用户名和密码，用于管理集群。
vim pcp.conf
```

```
#编辑内容如下
postgres:6b07583ba8af8e03043a1163147faf6a
#保存退出!
```

4.4 配置系统命令权限

配置 ifconfig，arping 执行权限 ，执行 failover_stream.sh 需要用到，可以让其他普通用户执行。

```
su - root
chmod u+s /sbin/ifconfig
chmod u+s /usr/sbin
```

4.5 配置 master 和 slave 上的 pgpool.conf

```
su - postgres
cd /opt/pgpool/etc
cp pgpool.conf.sample pgpool.conf
vim pgpool.conf
```

master 上的 pgpool.conf 需编辑内容如下：

```
# CONNECTIONS
listen_addresses = '*'
port = 9999
pcp_listen_addresses = '*'
pcp_port = 9898

# - Backend Connection Settings -

backend_hostname0 = 'master'
backend_port0 = 5432
backend_weight0 = 1
```

```
backend_data_directory0 = '/home/postgres/data'
backend_flag0 = 'ALLOW_TO_FAILOVER'

backend_hostname1 = 'slave'
backend_port1 = 5432
backend_weight1 = 1
backend_data_directory1 = '/home/postgres/data'
backend_flag1 = 'ALLOW_TO_FAILOVER'

# - Authentication -
enable_pool_hba = on
pool_passwd = 'pool_passwd'

# FILE LOCATIONS
pid_file_name = '/opt/pgpool/pgpool.pid'

replication_mode = off
load_balance_mode = on
master_slave_mode = on
master_slave_sub_mode = 'stream'

sr_check_period = 5
sr_check_user = 'repuser'
sr_check_password = 'repuser'
sr_check_database = 'postgres'

#------------------------------------------------------------------------
------------------------
# HEALTH CHECK 健康检查
#------------------------------------------------------------------------
------------------------

health_check_period = 10 # Health check period
                                        # Disabled (0) by default
health_check_timeout = 20
                                        # Health check timeout
                                        # 0 means no timeout
health_check_user = 'postgres'
                                        # Health check user
health_check_password = 'nariadmin' #数据库密码
                                        # Password for health check user
health_check_database = 'postgres'
#必须设置，否则 primary 数据库 down 了，pgpool 不知道，不能及时切换。从库流复制还在连接数据，报连接失败。
```

```
#只有下次使用 pgpool 登录时，发现连接不上，然后报错，这时候，才知道挂了，pgpool 进行切换。


#主备切换的命令行配置
#------------------------------------------------------------------------
# FAILOVER AND FAILBACK
#------------------------------------------------------------------------

failover_command = '/opt/pgpool/failover_stream.sh %H '

#------------------------------------------------------------------------
# WATCHDOG
#------------------------------------------------------------------------

# - Enabling -
use_watchdog = on
# - Watchdog communication Settings -

wd_hostname = 'master'
                                    # Host name or IP address of this watchdog
                                    # (change requires restart)
wd_port = 9000
                                    # port number for watchdog service
                                    # (change requires restart)
# - Virtual IP control Setting -

delegate_IP = 'vip'
                                    # delegate IP address
                                    # If this is empty, virtual IP never bring up.
                                    # (change requires restart)
if_cmd_path = '/sbin'
                                    # path to the directory where if_up/down_cmd exists
                                    # (change requires restart)
if_up_cmd = 'ifconfig eth1:0 inet $_IP_$ netmask 255.255.255.0'
                                    # startup delegate IP command
                                    # (change requires restart)
                                    # eth1 根据现场机器改掉
if_down_cmd = 'ifconfig eth1:0 down'
                                    # shutdown delegate IP command
```

```
                                    # (change requires restart)
                                    # eth1 根据现场机器改掉
# -- heartbeat mode --

wd_heartbeat_port = 9694
                                    # Port number for receiving heartbeat signal
                                    # (change requires restart)
wd_heartbeat_keepalive = 2
                                    # Interval time of sending heartbeat signal (sec)
                                    # (change requires restart)
wd_heartbeat_deadtime = 30
                                    # Deadtime interval for heartbeat signal (sec)
                                    # (change requires restart)
heartbeat_destination0 = 'slave'
                                    # Host name or IP address of destination 0
                                    # for sending heartbeat signal.
                                    # (change requires restart)
heartbeat_destination_port0 = 9694
                                    # Port number of destination 0 for sending
                                    # heartbeat signal. Usually this is the
                                    # same as wd_heartbeat_port.
                                    # (change requires restart)
heartbeat_device0 = 'eth1'
                                    # Name of NIC device (such like 'eth0')
                                    # used for sending/receiving heartbeat
                                    # signal to/from destination 0.
                                    # This works only when this is not empty
                                    # and pgpool has root privilege.
                                    # (change requires restart)
                                    # eth1 根据现场机器改掉
# - Other pgpool Connection Settings -

other_pgpool_hostname0 = 'slave' #对端
                                    # Host name or IP address to connect to for other pgpool
0
                                    # (change requires restart)
other_pgpool_port0 = 9999
                                    # Port number for othet pgpool 0
                                    # (change requires restart)
other_wd_port0 = 9000
                                    # Port number for othet watchdog 0
                                    # (change requires restart)
```

slave 上的 pgpool.conf 配置，将 master 中 pgpool.conf 内容复制过来，改动以下几个属性：

```
wd_hostname = 'slave'  #本端
heartbeat_destination0 = 'master' #对端
other_pgpool_hostname0 = 'master' #对端
```

配置文件里，故障处理配置的是 failover_command = '/opt/pgpool/failover_stream.sh %H '，因此，需要在/opt/pgpool 目录中写个 failover_stream.sh 脚本：

```
cd /opt/pgpool
touch failover_stream.sh
vim failover_stream.sh
```

注意这里使用了 promote 而不是触发文件，触发文件来回切换有问题，编辑内容如下：

```
#! /bin/sh
# Failover command for streaming replication.
# Arguments: $1: new master hostname.

new_master=$1
trigger_command="$PGHOME/bin/pg_ctl promote -D $PGDATA"

# Prompte standby database.
/usr/bin/ssh -T $new_master $trigger_command

exit 0;
```

4.6，在 pgpool 中添加 pg 数据库的用户名和密码

```
#数据库登录用户是 postgres,这里输入登录密码，不能出错，输入密码后，在 pgpool/etc 目录下会生成一个 pool_passwd 文件
pg_md5 -p -m -u postgres pool_passwd
```

4.7，在 master,slave 节点创建两个日志文件

```
su - root
mkdir /var/log/pgpool
chown -R postgres.postgres /var/log/pgpool
mkdir /var/run/pgpool
chown -R postgres.postgres /var/run/pgpool
```

5，PGPool 集群管理

5.1，启动集群

分别启动 primary,standby 的 pg 库

```
#master 上操作
su - postgres
```

```
pg_ctl start -D $PGDATA
#slave 上操作
su - postgres
pg_ctl start -D $PGDATA
```

分别启动 pgpool 命令：

```
#master 上操作
# -D 会重新加载 pg_nodes 的状态如 down 或 up
pgpool -n -d -D > /var/log/pgpool/pgpool.log 2>&1 &
#slave 上操作
pgpool -n -d -D > /var/log/pgpool/pgpool.log 2>&1 &
```

启动 pgpool 后，可在 master 或 slave 节点上查看集群节点状态：

```
psql -h vip -p 9999
postgres=# show pool_nodes;
```

注意快速终止 pgpool 命令为：

```
pgpool -m fast stop
```

5.2 Pgpool 的 HA

5.2.1 模拟 master 端 pgpool 宕机

```
#在 master 节点上停止 pgpool 服务
pgpool -m fast stop
#稍等片刻后，访问集群
psql -h vip -p 9999
postgres=# show pool_nodes;
#访问成功，在 master 节点上的 pgpool 宕机后，由 slave 节点的 pgpool 接管 vip 和集群服务，并未中断应用访问。
#在 master 上重新启动 pgpool 后，停止 slave 上的 pgpool 服务，结果也一样。
```

5.2.2 模拟 master 端 pg primary 宕机

```
#在 master 节点上停止 pg
pg_ctl stop
#稍等片刻后，访问集群
psql -h vip -p 9999
#slave 已经被切换成 primary，且 master 节点状态是 down
```

5.2.3 修复 master 节点重新加入集群

master 节点 down 机后，slave 节点已经被切换成了 primary，修复好 master 后应重新加入节点，作为 primary 的 standby。

修复 master 端并启动操作：

```
cd $PGDATA
mv recovery.done recovery.conf   #一定要把.done 改成.conf
pg_ctl start
```

在 pgpool 集群中加入节点状态：

```
#注意 master 的 node_id 是 0，所以-n 0
pcp_attach_node -d -U postgres -h vip -p 9898 -n 0   #提示输入密码，输入 pcp 管理密码。
postgres=# show pool_nodes；#查看当前状态
#可以看见，master 和 slave 都是 up 状态，master 为 standby 角色，slave 为 primary 角色
```

5.2.4 主机直接 down 机

当前 slave 节点是 primay，我们直接将 slave 服务器直接关机后，发现实现了主备切换，slave 已经 down 了，而 master 已经被切换成了 primary。

5.3，数据线同步

在主备切换时，在我们修复节点并重启后，需要按照流复制模式将修复的节点重新加入集群，这时可能会由于要修复的节点数据与 primary 数据不一致（primary 数据发生变化，或修复的节点数据发生变化）而报时间线不同步错误。产生这种情况，需要根据 pg_rewind 工具同步数据时间线，具体分 5 步走。

5.3.1，首先停掉需要做同步的节点 pg 服务

```
pg_ctl stop
```

5.3.2，用 pg_rewind 命令同步 primary 节点上的时间线

```
#这里假设了 primary 角色是 master 主机，pg 的 postgres 用户密码为 123456。
pg_rewind  --target-pgdata=/home/postgres/data --source-server='host=master port=5432 user=postgres dbname=postgres password=123456'
#输出：servers diverged at WAL location 0/2B0001B0 on timeline 10
rewinding from last common checkpoint at 0/2B000108 on timeline 10
Done!
```

5.3.3 修改 pg_hba.conf 与 recovery.done 文件

```
#pg_hba.conf 与 recovery.done 都是从 primary 同步过来的，要改成 standby 自己的
cd $PGDATA
```

```
vi pg_hba.conf
#下面红字部分改成流复制对端的主机名
host    replication    repuser    master    md5
```

```
mv recovery.done recovery.conf
vi recovery.conf
#下面红字部分改成流复制对端的主机名
primary_conninfo = 'host=master port=5432 user=repuser password=repuser'
```

5.3.4 启动 pg 服务

```
pg_ctl start
```

5.3.5 重新加入集群

```
#注意 slave 的 node_id 是 1，所以-n 1
pcp_attach_node -d -U postgres -h vip -p 9898 -n 1  #提示输入密码，输入 pcp 管理密码。
```

5.3.6 查看集群节点状态

```
psql -h vip -p 9999
postgres=# show pool_nodes;
#确保恢复后，状态无误
```

5.4 MySQL 集群安装配置

5.4.1 使用 Galera 搭建 MySQL 集群

1，Galera 集群简介

Galera replication 是 codership 提供的 MySQL 数据同步方案，声称全世界最先进的免费开源集群技术，目前为止使用案例较多，技术比较成熟，稳定性好。

1.1，Galera 集群优点

1. 同步复制，只有所有结点复制完成，才认为事务执行成功
2. Active-active 的多主拓扑结构，集群任意节点可以读和写
3. 自动身份控制，失败节点自动脱离集群，自动节点接入
4. 真正的基于"行"级别和 ID 检查的并行复制
5. 无单点故障，易扩展

1.2，注意事项

1、使用 Galera 必须要给 MySQL—Server 打 wsrep 补丁。可以直接使用官方提供的已经打好补丁的 MySQL 安装包，如果服务器上已经安装了标准版 MYSQL，需要先卸载再重新安装。卸载前注意备份数据。

2、MySQL/Galera 集群只支持 InnoDB 存储引擎。如果你的数据表使用的 MyISAM，需要转换为 InnoDB，否则记录不会在多台复制。可以在备份老数据时，为 mysqldump 命令添加 –skip—create—options 参数，这样会去掉表结构的声明信息，再导入集群时自动使用 InnoDB 引擎。不过这样会将 AUTO_INCREMENT 一并去掉，已有 AUTO_INCREMENT 列的表，必须在导入后重新定义。

3、MySQL 5.5 及以下的 InnoDB 引擎不支持全文索引(FULLTEXT indexes)，如果之前使用 MyISAM 并建了全文索引字段的话，只能安装 MySQL 5.6 with wsrep patch。

4、所有数据表必须要有主键（PRIMARY），如果没有主键可以建一条 AUTO_INCREMENT 列。

5、MySQL/Galera 集群不支持下面的查询：LOCK/UNLOCK TABLES，不支持下面的系统变量：character_set_server、utf16、utf32 及 ucs2。

6、数据库日志不支持保存到表，只能输出到文件（log_output = FILE），不能设置 binlog—do—db、binlog—ignore—db。

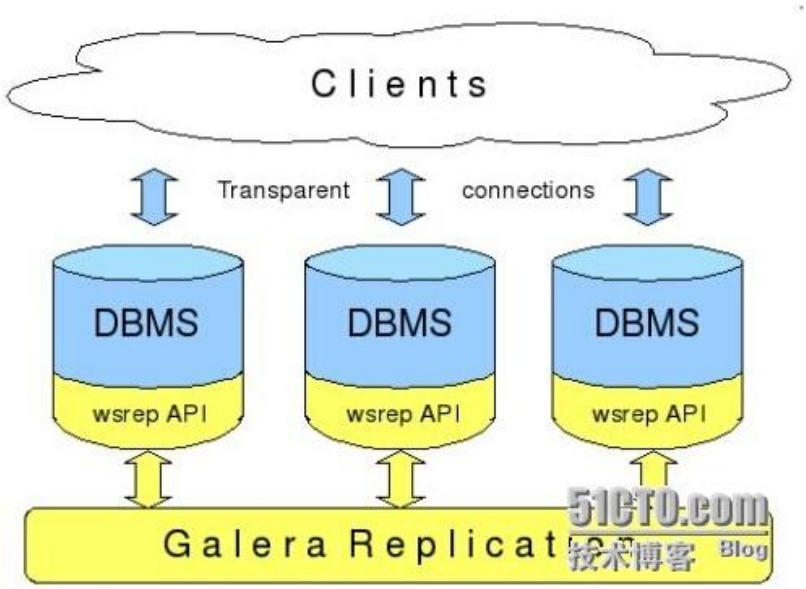
7、跟其他集群一样，为了避免节点出现脑裂而破坏数据，建议 Galera 集群最低添加 3 个节点。

8、在高并发的情况下，多主同时写入时可能会发生事务冲突，此时只有一个事务请求会成功，其他的全部失败。可以在写入/更新失败时，自动重试一次，再返回结果。

9、节点中每个节点的地位是平等的，没有主次，向任何一个节点读写效果都是一样的。实际可以配合 VIP/LVS 或 HA 使用，实现高可用性。

10、如果集群中的机器全部重启，如机房断电，第一台启动的服务器必须以空地址启动：mysqld_safe – wsrep_cluster_address=gcomm:// >/dev/null &

1.3，Galera 复制架构图



2，Galera 集群安装前准备工作

1，Centos7.0 以上版本操作系统服务器系统服务器一台或多台

说明：

Centos6 以下或其他版本 Linux 流程大致相同，细节略有差别

可以在一台机器上安装多个 MySQL，通过不同的端口构建一个集群，也可以使用多台服务器搭建一个集群，生产环境推荐使用多台服务器

2，root 用户，或其他具备相应权限的用户

3，可以使用网络 Yum 源，以方便安装一些依赖库，这里选用阿里巴巴的 yum 源，yum 源配置请自行百度一下

4，不存在 MariaDB 数据库，或其他版本的 MySQL 数据库，会对安装操作造成干扰，如果存在，将之卸载

查看是否安装了 mariadb 数据库使用如下命令：

```
rpm -qa | grep mariadb
```

如果存在 mariadb 软件包会列举出来，卸载软件包使用如下命令：

```
rpm -e --nodeps 软件包名
```

卸载成功后无输出，卸载成功后继续通过 rpm -qa | grep mariadb 查看，可以发现刚执行卸载的软件包已不存在，效果如下：

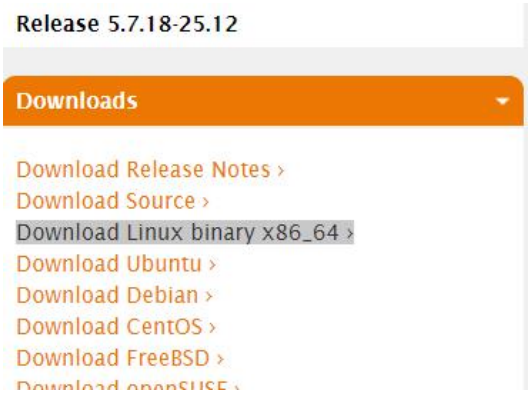
```
[root@h1 ~]# rpm -qa | grep mariadb
mariadb-libs-5.5.64-1.el7.x86_64
[root@h1 ~]# rpm -e --nodeps mariadb-libs-5.5.64-1.el7.x86_64
[root@h1 ~]# rpm -qa | grep mariadb
[root@h1 ~]#
```

3，安装源获取

Galera for MySQL 集群需要安装复制组件 Galera 3 replication library 和 MySQL 数据库。

Galera 3 replication library 是用于集群复制的核心组件，完成数据复制操作，下载地址：http://releases.galeracluster.com/galera-3/centos/7/x86_64/。

galera-3-25.3.21-2.el7.x86_64.rpm 即是 Galera 使用的 MySQL 数据库，不是普通官方原版的 MySQL，而是打了 wsrep 补丁的 MySQL，通过如下地址下载：http://galeracluster.com/downloads/。该链接相应时间较长，容易失败，多尝试几次，即可进入，进入该页面后点击 Download Linux binary x86_64 下载二进制安装包，如下图所示：



说明：

这里选择二进制进行安装，原因是该安装过程简介，方便，解压即可，另外可以轻松实现绿化，

即可以再一台机器上安装多个 MySQL

4，安装复制组件

1、首先将我们准备好的复制组件软件包 galera-3-25.3.21-2.el7.x86_64.rpm 上传至服务器，位置随意，这里我放到 home 目录下了
2、使用命令 cd /home 进入到 home 目录下。
3、使用命令 rpm -ivh galera-3-25.3.21-2.el7.x86_64.rpm 执行安装
4、如果提示如下所示依赖检测失败错误，请先安 libboost_program_options.so.1.53.0()(64bit)，

```
[root@h1 home]# rpm -ivh galera-3-25.3.29-1.el7.x86_64.rpm
警告：galera-3-25.3.29-1.el7.x86_64.rpm: 头V4 RSA/SHA512 Signature, 密钥 ID bc19ddba: NOKEY
错误：依赖检测失败：
        libboost_program_options.so.1.53.0()(64bit) 被 galera-3-25.3.29-1.el7.x86_64 需要
[root@h1 home]# yum -y install libboost_program_options.so.1.53.0\(\)\(64bit\)
```

使用命令 yum -y install libboost_program_options.so.1.53.0\(\)\(64bit\) 执行安装即可，由于 yum 命令不能使用"("和")"，需要对其转义，安装成功后，执行第 3 步安装即可
5、linux 常用依赖库列表

有时候在 linux 上安装软件，总是缺少一些依赖库，每台机器依赖库安装情况会随着 linux 的安装级别的不同略有差异，比如 linux 最简安装就比较容易缺少依赖库，这里把常用的依赖库列举如下，如果缺失，可以再下面查找，通过 yum -y install 依赖库名进行安装。当然可以通过如下命令安装全部依赖库，但是不推荐这样做，推荐在下面列表中找到缺失的依赖库进行安装。

yum install -y gcc gdb strace gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2 glib2-devel bzip2 bzip2-devel ncurses ncurses-devel curl curl-devel e2fsprogs patch e2fsprogs-devel krb5-devel libidn libidn-devel openldap-devel nss_ldap openldap-clients openldap-servers libevent-devel libevent uuid-devel uuid mysql-devel openssl-devel libnfnetlink-devel

6、复制组件安装成功后效果如下

```
[root@h1 home]# rpm -ivh galera-3-25.3.29-1.el7.x86_64.rpm
警告：galera-3-25.3.29-1.el7.x86_64.rpm: 头V4 RSA/SHA512 Signature, 密钥 ID bc19ddba: NOKEY
准备中...                          ################################# [100%]
正在升级/安装...
   1:galera-3-25.3.29-1.el7         ################################# [100%]
```

安装成功后可以看到 cd /usr/lib64/galera-3 目录下存在 libgalera_smm.so 文件，该文件即配置集群时需要指定的复制组件

```
[root@h1 home]# cd /usr/lib64/galera-3/
[root@h1 galera-3]# ls
libgalera_smm.so
```

注：如果在多台服务器上安装集群，那么需要在每一台机器上都需要执行上述过程，安装复制组件

5，安装 MySQL 数据库

1、将事先准备好的带有 wsrep 补丁的安装包 mysql-wsrep-5.7.18-25.12-linux-x86_64.tar.gz 上传

至服务器，这里放到了/home 目录下。
2、执行 cd /home 进入到/home 目录下
3、执行 tar -zxvf mysql-wsrep-5.7.18-25.12-linux-x86_64.tar.gz 命令进行解压，解压成功后，可以看到/home 目录下存在 mysql-wsrep-5.7.18-25.12-linux-x86_64 文件夹
4、执行 mv mysql-wsrep-5.7.18-25.12-linux-x86_64 mysql-1 将文件夹重命名成 mysql-1，简化一下目录，该目录作为我们的第一个 mysql 数据库。
此时发现 mysql-wsrep-5.7.18-25.12-linux-x86_64 文件夹名称已经被修改成 mysql-1
5、执行 cp -rf mysql-1 mysql-2 和 cp -rf mysql-1 mysql-3，将 mysql-1 复制两份，作为我们的另外两个 mysql 结点，这里用三个演示集群，如果需要更多，继续复制即可
6、该过程演示一台服务器上安装多个 MySQL，来创建集群，如果有多台服务器，不必执行上述操作，只需要把上面的全部过程在每一台服务器上执行一遍即可

7、执行如下命令创建 mysql 用户和组，并设置目录所属 mysql 组 mysql 用户

```
groupadd mysql
useradd -r -g mysql mysql
cd /home
chgrp -R mysql mysql-1  (chown -R mysql.mysql /home/mysql-1)
```

8、执行 cd mysql-1，进入到 mysql-1，执行以下命令，安装 MySQL

```
mkdir run           #创建执行线程文件目录
mkdir data          #创建数据目录
mkdir binlog        #创建二进制日志目录
chmod -R 775 .      #执行授权
chmod -R 777 run
chmod -R 777 binlog
```

执 行 bin/mysqld --initialize-insecure --user=mysql --basedir=`pwd` --datadir=`pwd`/data/#安装 MySQL，并指定数据目录为当前目录下的 data 目录，安装成功后效果如下：

```
[root@localhost mysql-1]# mkdir run    #创建执行线程文件目录
[root@localhost mysql-1]# mkdir data   #创建数据目录
[root@localhost mysql-1]# chmod -R 775 .    #执行授权
[root@localhost mysql-1]# bin/mysqld --initialize-insecure --user=mysql --basedir=`pwd` --d
atadir=`pwd`/data/   #安 装MySQL，并指定数据目录为当前目录下的data目录
2017-08-28T08:25:23.566150Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated.
 Please use --explicit_defaults_for_timestamp server option (see documentation for more deta
ils).
2017-08-28T08:25:24.925772Z 0 [Warning] InnoDB: New log files created, LSN=45790
2017-08-28T08:25:25.075037Z 0 [Warning] InnoDB: Creating foreign key constraint system table
s.
2017-08-28T08:25:25.136382Z 0 [Warning] No existing UUID has been found, so we assume that t
his is the first time that this server has been started. Generating a new UUID: 70de5758-8bc
a-11e7-a309-000c29b713cf.
2017-08-28T08:25:25.138065Z 0 [Warning] Gtid table is not ready to be used. Table 'mysql.gti
d_executed' cannot be opened.
2017-08-28T08:25:25.138954Z 1 [Warning] root@localhost is created with an empty password ! P
lease consider switching off the --initialize-insecure option.
[root@localhost mysql-1]#
```

9、 在 mysql-1 目录下，使用 vi 编辑器命令 vi my.cnf 创建 MySQL 核心配置文件 my.cnf 内容如下，其中需要调整的部分已标红色，根据自己环境适当调整


my.cnf

IP 配置说明：

上述 wsrep_node_address='127.0.0.1:4567'的 IP 为本机 IP，为简单起见，使用时 127.0.0.1，需要替换成具体 IP 即可，如果多个 MySQL 部署在一台机器上，那么这些 MySQL 的 IP 是相同的，此时必须为每一个 MySQL 指定不同的端口号，如果 MySQL 部署在不同的服务器上，那么此时 IP 已经不同，端口号是否相同无关紧要。此行 wsrep_node_address='127.0.0.1:4567'指定的端口为数据复制端口号，必须与 MySQL 端口号（如上述的 3306）不同，wsrep_cluster_address 时集群各个节点的 ip。

10、在 mysql-1 下执行如下命令，以创建 mysqld 服务，mysqld 即为 MySQL 启动停止等操作使用的脚本

```
cp support-files/mysql.server mysqld
chmod -R 775 mysqld
```

11、执行 vi mysqld 编辑 mysqld 文件，修改如下两行内容如下：

```
basedir=`pwd`
datadir=$basedir/data
```

12、至此，MySQL 安装集群安装流程已完成，如果多台机器上安装 MySQL 集群，需要在每一个机器上执行 **1.3 安装复制组件+1.4 安装 MySQL 数据库**全部内容。如果多个 MySQL 安装到一台机器上，那么只需在每一个结点上执行 **1.4 安装 MySQL 数据库**中的 8~11 步即可，即在之前创建好的 mysql-2 和 mysql-3 中分别执行上述的 8~11 步，注意需要调整配置文件 my.cnf 的 IP 端口号复制端口号安装路径等

5.4.2 MySQL 集群的维护

1，MySQL 集群的启动

Galera for MySQL 启动方式如下：

Galera 集群启动时，**需要指定已经启动的全部其他结点的 IP 和端口**，第一个结点以空地址启动，即之前没有任何结点加入，第二个结点启动时要把第一个结点加入其中，第三个结点启动时要把第一个和第二个结点加入其中，以此类推…

例如：有如下三个 MySQL，127.0.0.1:4567，127.0.0.1:4568，127.0.0.1:4569（**注：此 IP 和端口为复制 IP 和端口号**）。

MySQL 启动时候，需要先进入到 MySQL 安装目录下，如上述的 mysql-1，该目录包含上述我们复制创建的 mysqld 脚本，使用该脚本执行启动和关闭，该脚本不要再其他目录下执行，**注意：一定要在 MySQL 的安装目录下执行，即先执行 cd MySQL 的安装目录（上述的 mysql-1，mysql-2，mysql-3 等），然后再执行启动和关闭命令。进入到哪个目录，操作的就是哪个 mysql。**

第一个 MySQL 启动时使用如下命令（逗号结尾）：

```
./mysqld start --wsrep-new-cluster
```

其他 MySQL 启动时需要把第一个加入其中，命令如下：

```
./mysqld start
```

2，MySQL 集群的关闭

使用如下命令即可关闭 MySQL，同样需要在 MySQL 安装目录下执行，在哪个安装目录下执行，操作的就是哪个 MySQL

```
./mysqld stop
```

MySQL 停止效果图：

```
[root@localhost mysql-1]# ./mysqld stop
Shutting down MySQL.....                                   [  确定  ]
[root@localhost mysql-1]#
```

3，MySQL 客户端的使用

MySQL 安装目录下的 bin 目录下有一个 mysql 命令，该命令用于进入 MySQL 客户端。在 MySQL 安装目录下执行如下命令即可进入 MySQL 客户端。**注意启动时需要指定配置文件 my.cnf**

```
bin/mysql --defaults-file=`pwd`/my.cnf -hlocalhost -uroot -p
```

首次安装 root 用户密码为空，在需要输入密码的提示处，直接回车即可，MySQL 客户端效果如下：

```
[root@localhost mysql-1]# bin/mysql --defaults-file=`pwd`/my.cnf -hlocalhost
 -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 5.7.18-log MySQL Community Server (GPL), wsrep_25.12

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statemen
t.

mysql>
```

4，MySQL 集群的维护

Galera for MySQL 集群维护需要通过查询 wsrep 参数进行维护。通过在 MySQL 客户端执行如下命令查看 wsrep 参数：

```
show status like '%wsrep%';
```

以下对监控状态参数进行分类说明。

集群完整性检查：

wsrep_cluster_state_uuid：在集群所有节点的值应该是相同的，有不同值的节点，说明其没有连接入集群。

wsrep_cluster_conf_id：正常情况下所有节点上该值是一样的。如果值不同，说明该节点被临时"分区"了。当节点之间网络连接恢复的时候应该会恢复一样的值。

wsrep_cluster_size：如果这个值跟预期的节点数一致，则所有的集群节点已经连接。

wsrep_cluster_status：集群组成的状态。如果不为"Primary"，说明出现"分区"或是"split-brain"状况。

节点状态检查：

wsrep_ready：该值为 ON，则说明可以接受 SQL 负载。如果为 Off，则需要检查 wsrep_connected。

wsrep_connected：如果该值为 Off，且 wsrep_ready 的值也为 Off，则说明该节点没有连接到集群。(可能是 wsrep_cluster_address 或 wsrep_cluster_name 等配置错造成的。具体错误需要查看错误日志)

wsrep_local_state_comment：如果 wsrep_connected 为 On，但 wsrep_ready 为 OFF，则可以从该项查看原因。

复制健康检查：

wsrep_flow_control_paused：表示复制停止了多长时间。即表明集群因为 Slave 延迟而慢的程度。值为 0~1，越靠近 0 越好，值为 1 表示复制完全停止。可优化 wsrep_slave_threads 的值来改善。

wsrep_cert_deps_distance：有多少事务可以并行应用处理。wsrep_slave_threads 设置的值不应该高出该值太多。

wsrep_flow_control_sent：表示该节点已经停止复制了多少次。

wsrep_local_recv_queue_avg：表示 slave 事务队列的平均长度。slave 瓶颈的预兆。

最慢的节点的 wsrep_flow_control_sent 和 wsrep_local_recv_queue_avg 这两个值最高。这两个值较低的话，相对更好。

检测慢网络问题：

wsrep_local_send_queue_avg：网络瓶颈的预兆。如果这个值比较高的话，可能存在网络瓶

冲突或死锁的数目：

wsrep_last_committed：最后提交的事务数目

wsrep_local_cert_failures 和 wsrep_local_bf_aborts：回滚，检测到的冲突数目

5.5 分析云集群节点安装配置

同单机部署，只是把启动和 shutdown.sh 中把 47 行注释掉，在 startup.sh 中把 61-69 行注释掉，即注释 postgresql 的启动和停止的脚本。

如果资源库时自行安装的，则需要手动执行 resources/script 下的对应数据库的建库脚本，执行方法可参见 4.2.

5.6 文件服务器的配置

集群时文件服务可以设置 windows 共享目录或通过远程挂载的方式共享文件夹。

也可以把文件存放到一个节点的目录下，在其他节点 conf/serverConfig.properties 文件中配置 fs.http.master 配置存放文件服务器的地址：例如：

fs.http.master=http://192.168.1.78:8080

6 升级安装

本版本的分析云支持从 20190827 版本（包括后续补丁版）直接升级；如果从之前老版本升级，为了稳妥起见（中间有元数据模型的升级，可能需要手动调整部分模型设置），需要先升级到 20190827 版，再升级到本版本。

6.1 单机安装升级

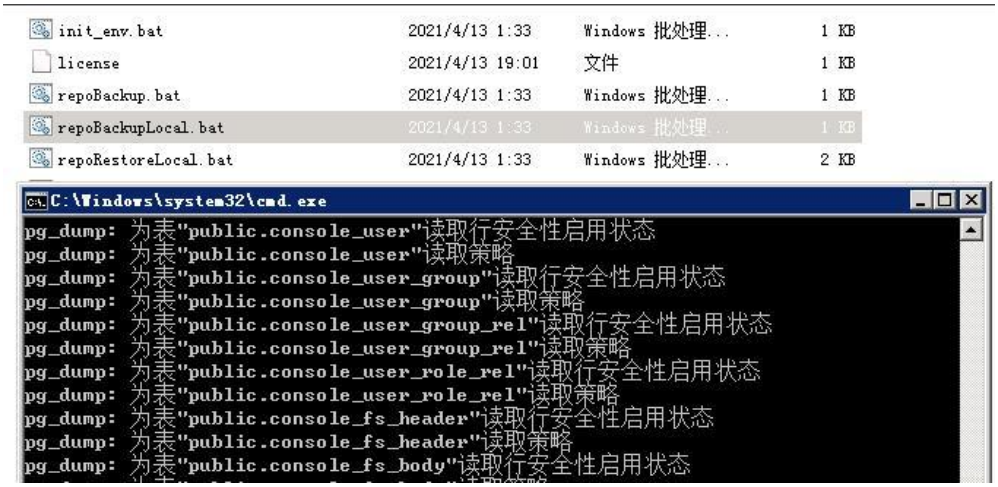
分析云安装目录 bin 下，有 repoBackupLocal 用于资源库的备份，有 repoRestoreLocal 用于资源库的恢复，如下图。

步骤 1：备份旧版分析云资源库

在升级前的 Windows 环境，进入安装目录 D:\bq\bin 下，双击 repoBackupLocal 就可以备份资源库，备份文件在 D:\bq\backup 下（分析云 7.0.0 之前版本是在

db\repobak 下），如下图。

注意：备份时，分析云必须是启动状态！



Linux 环境备份（假如安装目录为/home/bq）命令为：

cd /home/bq/bin

#./repoBackupLocal.sh

步骤 2：停止旧版分析云并备份

停止旧版分析云，并将目录改名，例如加上旧版的发版日期，为 bq20190827。

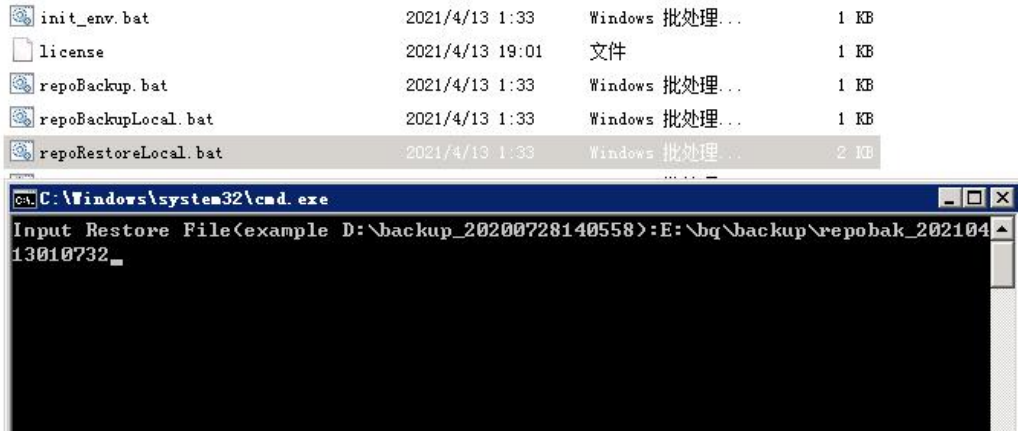
注意：升级前的旧环境，千万不能删除！

步骤 3：安装新版分析云

按照前述章节 2 单机安装及配置安装分析云，注意目录名要和旧版一致，这样许可可以继续使用。安装完后，**注意：不需要启动环境！**

步骤 4：恢复资源库

进入 bin 目录下，双击执行 repoRestoreLocal，提示【Input Restore File(example D:\backup_20200728140558)】，在这里，输入备份文件的路径和名称，然后回车，如下图所示。



如果旧版分析云，在 conf/serverConfig.properties 中有修改过配置，在升级后的新环境中也要同步修改。

Linux 环境恢复（假如安装目录为/home/bq）命令为：

cd /home/bq/bin

./repoRestoreLocal.sh

6.2 从自定义安装资源库升级

备份原来的资源库，再在资源库执行 resources\upgrade\下相对应库的 SQL 脚本。脚本中最后注释掉一部分 SQL 语句，检查此部分 SQL 对应表字段、索引等是否存在，不存在则执行此部分 SQL；存在则无需执行，使用新的分析云连接该资源库即可。

7 性能调优

7.1 Java 内存调优

根据服务器的实际内存情况，修改安装盘的 bin 目录下的 catalina 文件，根据内存大小，将文件中对应代码放开。例如：服务器内存为 8G，将此段代码放开，删掉代码前的注释即可。

rem set "JAVA_OPTS=-server -Xms6G -Xmx6G -XX:PermSize=128m -XX:+UseG1GC -XX:MaxTenuringThreshold=15 -XX:MaxGCPauseMillis=200 -XX:InitiatingHeapOccupancyPercent=70"

7.2 数据仓库调优

在分析云的部署目录的 logs/druid/sql.log 文件，打印了执行时间超过 5 秒的 SQL 日志；对行存储的输出库，可根据数据情况在维度字段和关联字段增加适度的索引，用以提高速度。

1. Oracle

Oracle 数据库推荐使用 OLAP 模式，如果数据量大，或为后续扩展，数据仓库建议适用大数据平台。

调高数据库的内存占用，优化数据库的参数设置。

2. SQLServer

2016 及以上版本在数据连接的 url 上增加以下参数会提高批量插入速度 30%-50%，并且当作业定时执行时间比上次慢时，加入下列参数也可解决此问题。参数如下：

statementPoolingCacheSize=10;disableStatementPooling=false;enablePrepareOnFirstPreparedStatementCall=true;

7.3 缓存设置

在设置执行库和数据连接时，有缓存设置，如果最大时间大于 0，则会根据 sql 把数据放到缓存，下次执行就不用执行该 SQL，直接从缓存中获取，如下图：

如果数据进行了更新，而缓存还在有效时间的话，读的数据可能是旧数据，此时，推荐适用 DI 的作业项“清除缓存”，执行“刷新”策略，会重新查询数据并放入缓存，执行“清除”策略，会清除缓存但不重新查询。

也可以在数据连接的“已导入的表”弹出的对话框中，点击清除缓存，此时缓存会清除掉，下次查询从数据库中查询。

7.4 安装 tomcat—native

Tomcat Native 是一个利用 APR 来提升 Tomcat 性能的本地 API。对于 Windows，分析云自带了 tomcat-native ddl，无需安装，对于 CentOS，RHEL、Fedora，OEL 操作系统，安装 resouces 下 rpm 包，在分析云部署目录下，安装命令如下：

```
yum localinstall resources/*.rpm -y
```

增加环境变量，修改/etc/profile 文件

```
vim /etc/profile

#增加以下内容

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib64/tomcat8/lib

#保存之后，重新加载文件

source /etc/profile
```

7.5 使用 Redis 加速

如果 SQL 的计算量非常大，但返回结果数据不大的情况下，可以利用 DI 的表输入和 Reids 输出把数据写入到 redis 中。在与其他系统交互时把交互的数据放入到 redis 中；在分析云中建立 Redis 连接，并建立 Redis 的表，这样可以快速的查询数据。详细参见 Redis 连接的使用。

8 对外服务

8.1 对外取数服务

使用场景：

a. 所有用户都获取某数据集中部分相同数据
b. 根据不同用户获取某数据集中不同数据
c. 如何通过开放接口传递参数获取数据

注意：获取数据的前提该用户对该数据集有权限

操作概述：

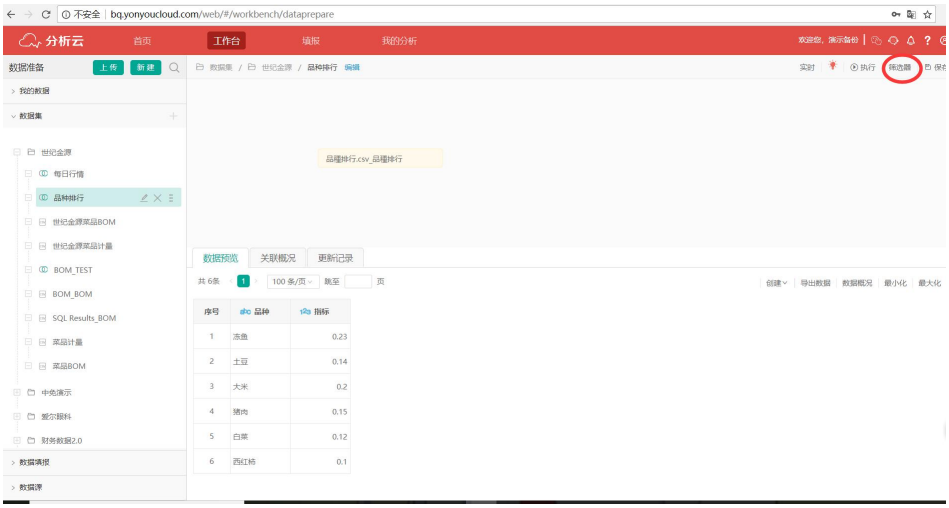
a 场景解决办法：对数据集本身做筛选

假设现在已存在名称为"品种排名"的数据集，不对其做筛选时执行结果如下：

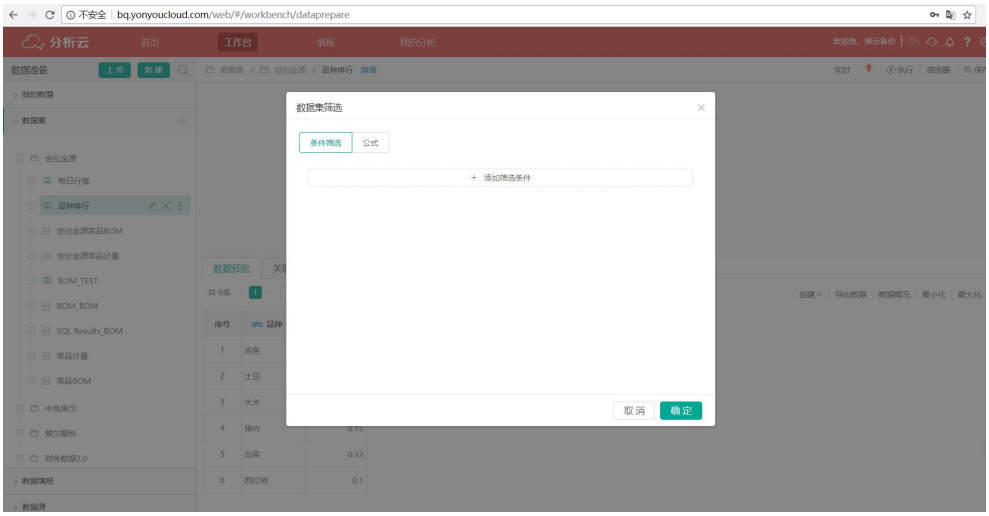
序号	abc 品种	123 指标
1	冻鱼	0.23
2	土豆	0.14
3	大米	0.2
4	猪肉	0.15
5	白菜	0.12
6	西红柿	0.1

如：只想让所有看到该数据集的用户只能看到品种名为“冻鱼”这一条数据，操作方法如下：

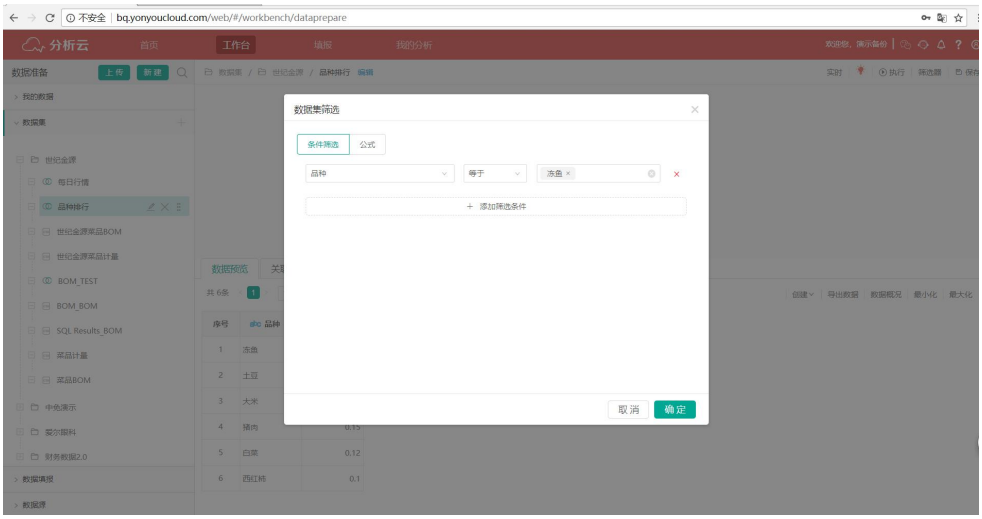
（1） 选中该数据集并进入编辑页面



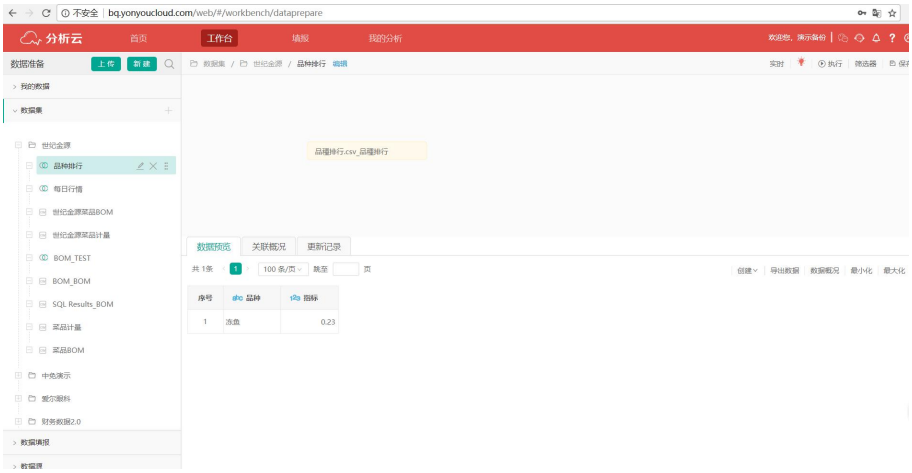
（2） 点击右上角“筛选器”按钮，弹出如下界面：



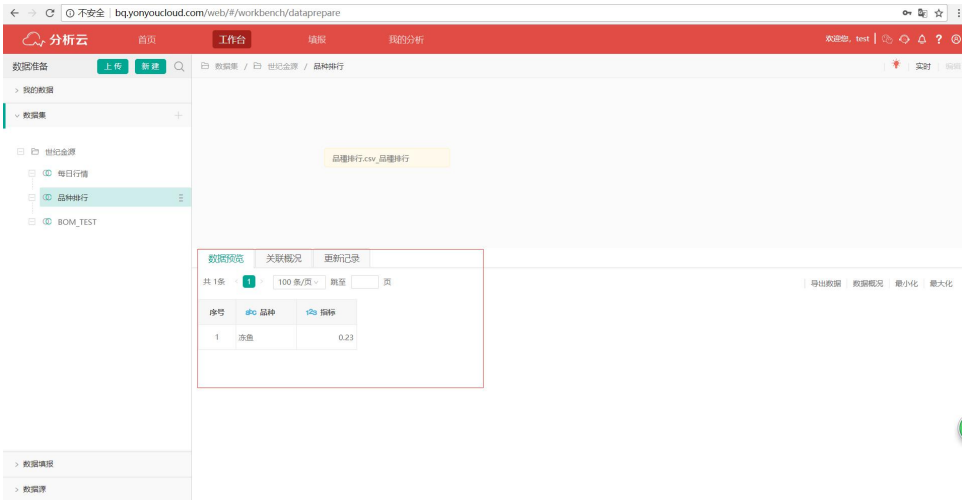
（3） 点击“添加筛选条件”，输入相关筛选条件



（4） 点击“确定”按钮，点击“ 保存”，点击“执行”，可看到筛选后结果



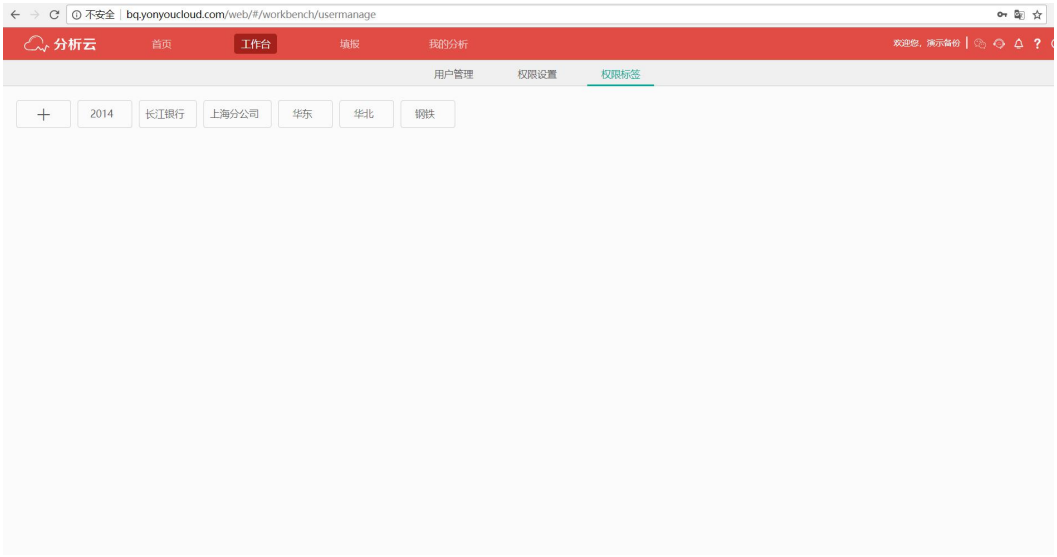
（5） 现用另一用户(例如：test 用户)登录，查看该数据集：



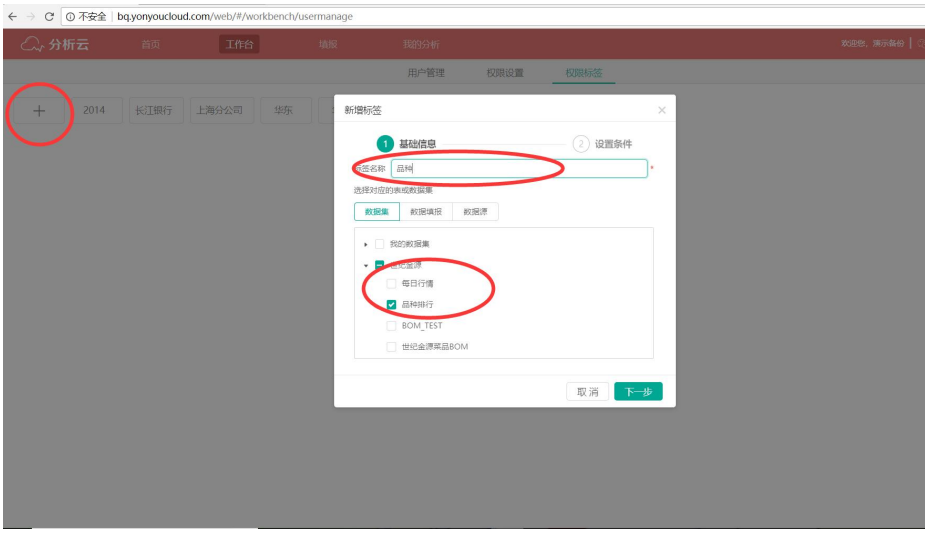
注意：若 test 用户对该数据集也有修改权限并对其设置了筛选条件，则会替换掉该数据集之前设置的筛选条件。

b 场景解决办法：为特定用户设置权限标签如：只允许 test 用户看到该数据集品种为“冻鱼”的这一条数据，而其他用户可以看到该数据集的全部数据，操作方法如下：

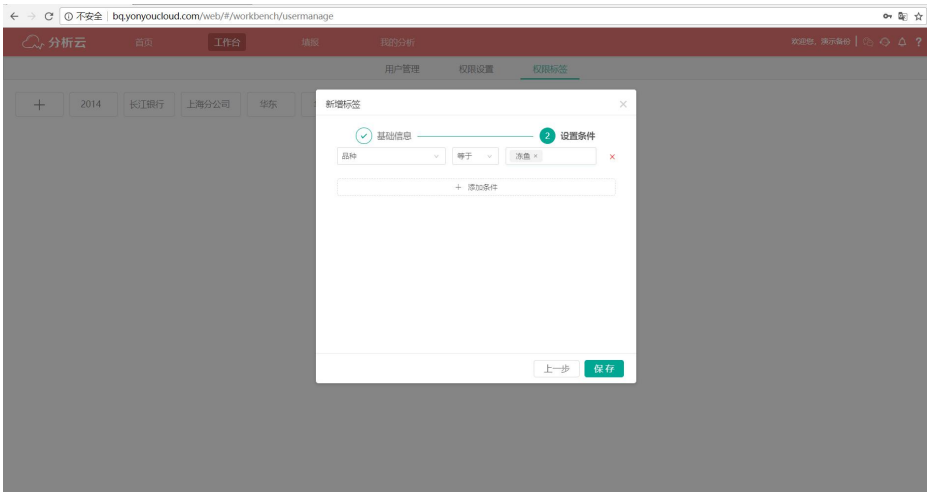
（1） 为该数据集设置权限标签：点击“工作台”标签下的“用户管理”，选择“权限标签”页签：



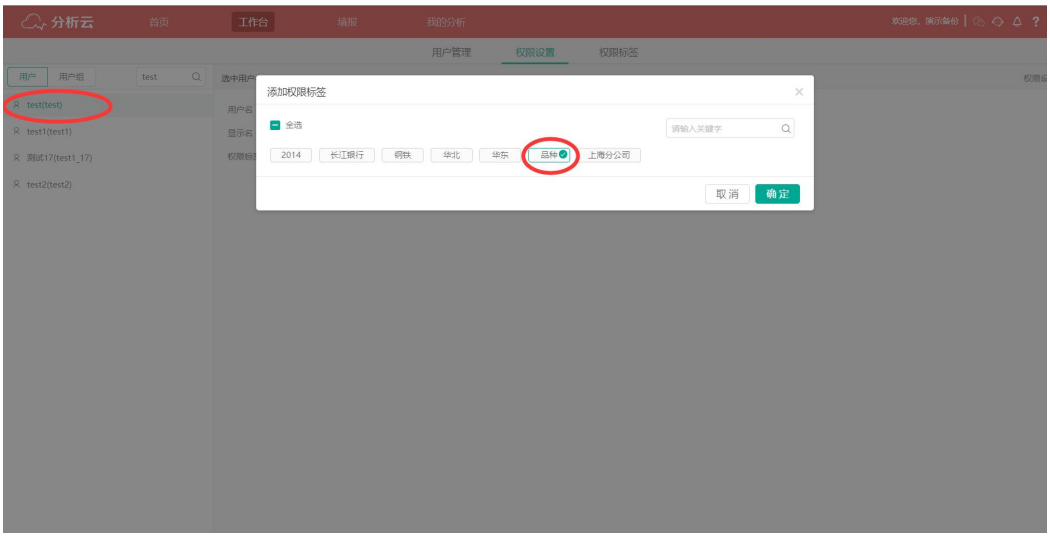
（2） 点击添加标签图标，弹出新增标签对话框，填写标签名称并选中该数据集



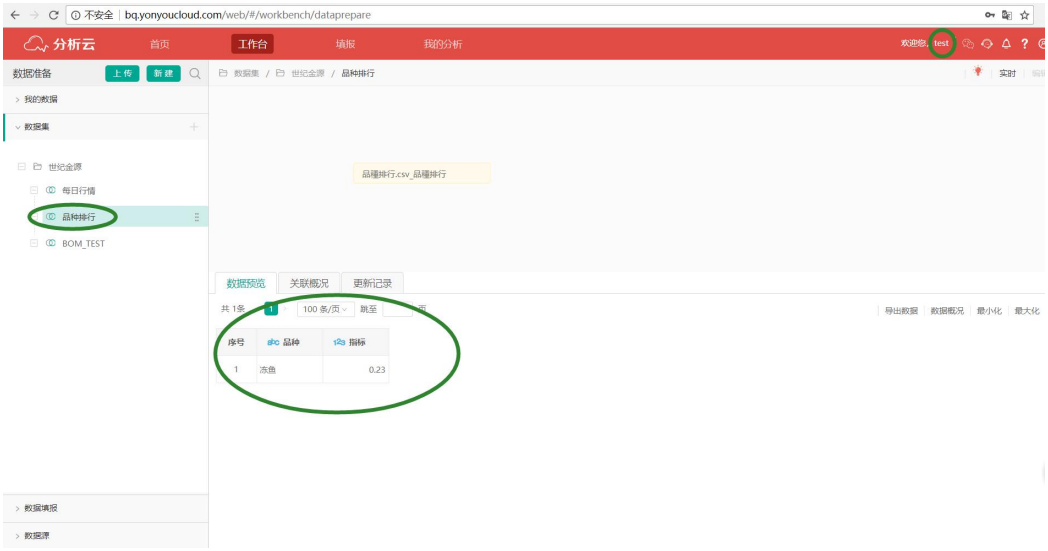
（3） 点击“下一步”，填写相关筛选条件，点击“保存”



（4） 将保存好的权限标签赋给某用户(例如 test 用户)：选中“权限设置”标签，选中左侧用户，点击“添加权限”图标，勾选刚创建的标签，点击确定按钮。

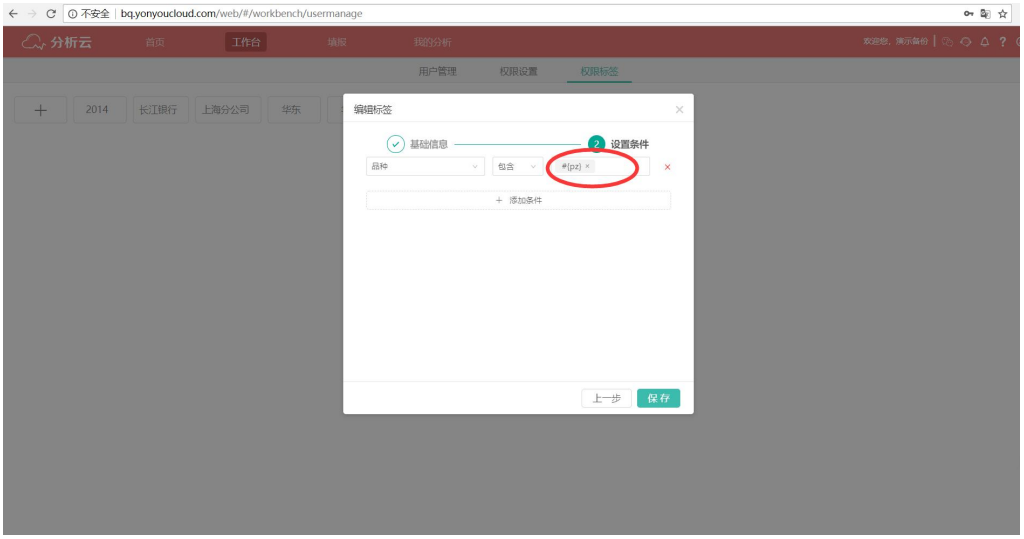


(5) 登录，查看结果：



c 如何通过开放接口传递参数获取数据：调用分析云提供的对外开放接口，以根据不同品种获取相关数据为例，具体操作如下：

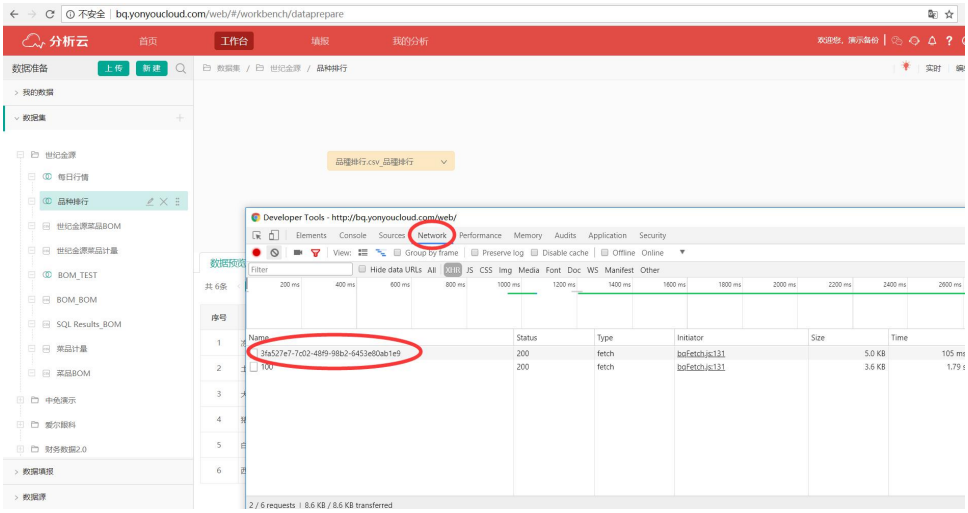
(1) 通过自定义宏变量传递参数，同样需要对用户设置权限标签：



注意：约定宏变量名称前后都需加#号和大括号，例如：#{pz},宏变量名称很重要，需要作为参数传递给对外开放接口；为某个用户的数据集添加带有自定义宏变量的标签后，直接点击该数据集是无法返回相关数据的，必须通过接口获取。

(2) 点击“保存”以相同的方法赋给用户。

(3) 获取数据集的 id,供开放接口使用：点击某一数据集，打开调试窗口(f12)，选中”Network”页签，获取数据集 id.



(4) 组织参数，调用接口获取数据,方法如下：

请求链接：url:'/bq_self/ds/web/data/getdatabyparams/{数据集 id}

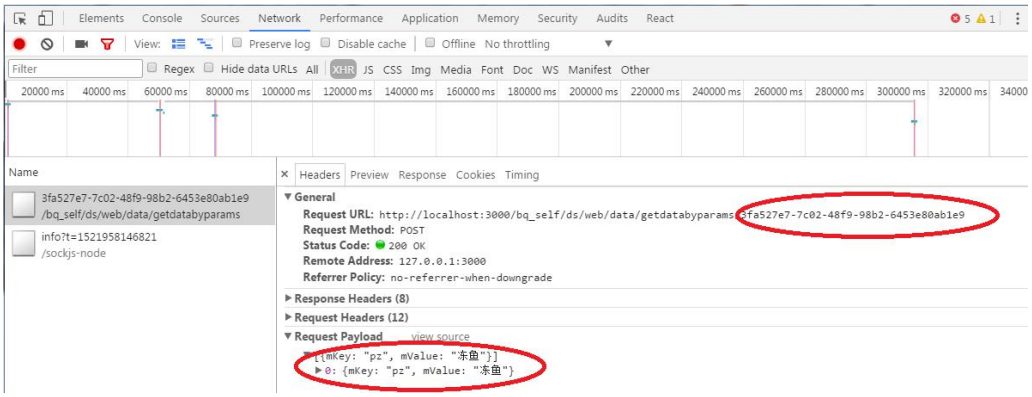
参数：数组形式

如下：

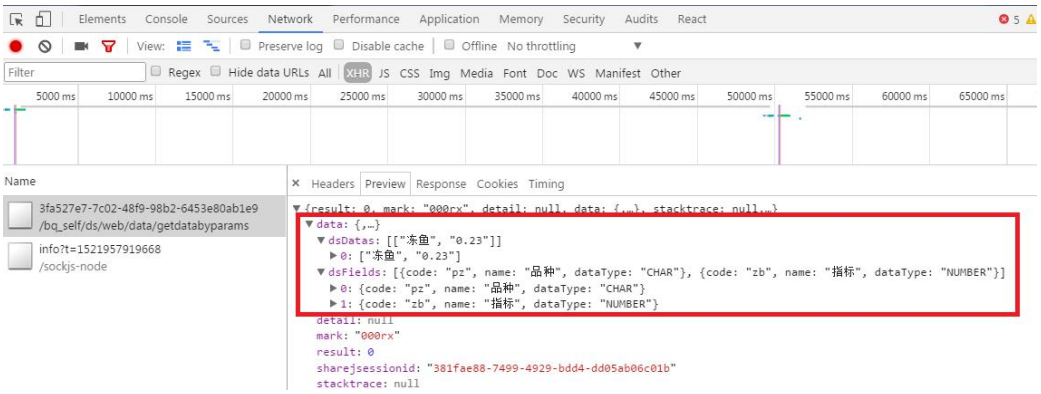
```
params=[
        {mKey:"pz", mValue :“冻鱼”}//mKey:对应自定义宏变量名 mValue:对应参数值
    ]
    $.ajax({
        url:'/bq_self/ds/web/data/getdatabyparams/
3fa527e7-7c02-48f9-98b2-6453e80ab1e9',
        type: "POST",
```

```
        dataType ： "json",
        cache ： false,
        contentType："application/json;charset=utf-8",
        data:JSON.stringify(params),
        success:function(response){
            //请求成功
}
});
```

请求如下：



结果如下：



注：如果没有参数需要传递一个空的参数，否则可能会出错，例如[{mKey:"", mValue ：“”}]

附：

如果是外部系统想获取数据，必须登录获取 sessionId，才能获取数据，登录的接口是发送 http 请求，描述如下：

项目	值	备注
url	/console/login	
方法	post	
Content-Type	multipart/form-data	注意不是 json，是表单发送方式
client	mobile	请求 header 中附加项
X-Requested-With	XMLHttpRequest	请求 header 中附加项

username	用户名	请求发送的参数
password	密码	请求发送的参数
domainCode	00000	请求发送的参数固定值
moduleCode	BQ	请求发送的参数固定值
language	zh-CN	请求发送的参数固定值

成功返回的数据格式如下：

{"success":true,"SHAREJSESSIONID":"8sv9vx5gmf6f42rlhnt34b92u341496dx7itc","message":"登录成功"}

请求数据的携带的 cookies 格式如下：

SHAREJSESSIONID=8sv9vx5gmf6f42rlhnt34b92u341496dx7itc; language=zh-CN

8.2 数据集成运行服务

项目	说明	备注
URL	/di/job/rest_execute?sign=<sign>	sign 的值是配置在分析云 conf/serverConfig.properties 配置文件的 di.job.rest.execute.secret 项的值拼接 body 的 json 字符串的 md5 值
Http Method	POST	
Content-Type	application/json; charset=utf-8	
Body	{ "jobPath": "/zuoym", "jobName": "test1", "logLevel": "Basic", "parameters": { "param1": "==========" }, "ts": "2022" }	jobPath 作业的路径，/<项目名>/[文件夹路径]； jobName 作业的名称； logLevel 日志级别，取值有：Error，Basic，Detailed； parameters 作业使用的变量参数，是个对象的形式； ts 时间戳。
返回值	{ "success": true, "message": "作业加入执行队列成功！" }	success 调用是否成功； message 提示消息

注：自分析云 7.5.0，DI4.5.0 拥有该功能

9 安全加强

9.1 分析云端口

分析云默认对外监听端口目前只有：8080，其他端口都为程序内部使用。端口被占用，修改方法如下：

配置文件为%BQ_HOME%/conf/serverConfig.properties

修改 IP：binding.local.ip=localhost

修改端口：tomcat.http.port=8080

其他端口被占用，修改 serverConfig.properties 相应的配置参数，即可。

9.2 大数据平台安全

一种办法是建立统一的硬件防火墙，把分析云和大数据建立到防火墙之后，只允许分析云的 http 请求通过分析云。

另外一种方式开启每个大数据服务器的软件防火墙，在防火墙增加对大数据其他服务器、分析云的 IP 信任，或指定网段的信任。

9.3 Tomcat 使用 https

防火墙开放 443 端口或者关闭防火墙

0) 文件服务器配置

在 admin 中文件服务器配置中启用 https 访问

1) 生成 jks

a. 有申请证书

由于 Tomcat 证书不支持直接使用 pem + 私钥的方式，因此，需要多一步使用 Openssl 将 full_chain.pem+private.key 转换为 jks 的步骤，首先将 full_chain.pem 和 private.key 放到一个任何目录，假设存放的目录是：/root/apache-tomcat-ssl，如下图：

```
[root@tank apache-tomcat-ssl]# pwd
/root/apache-tomcat-ssl
[root@tank apache-tomcat-ssl]# ll
total 8
-rw-r--r-- 1 root root 2866 May 21 08:20 full_chain.pem
-rw-r--r-- 1 root root  264 May 21 08:20 private.key
```

随后使用如下命令，在当前目录下生成一个名为 freeSSL.jks 的文件，如果使用不了如下命令，尝试考虑升级 Openssl 到最新版本：

```
openssl pkcs12 -export -out /root/apache-tomcat-ssl/freeSSL.jks -in ./full_chain.pem -inkey ./private.key
```

```
[root@tank apache-tomcat-ssl]# openssl pkcs12 -export -out /root/apache-tomcat-ssl/freeSSL.jks -in ./full_chain.pem -inkey ./private.
Enter Export Password:
Verifying - Enter Export Password:
[root@tank apache-tomcat-ssl]# ll
total 12
-rw-r--r-- 1 root root 2590 May 21 19:59 freeSSL.jks
-rw-r--r-- 1 root root 2866 May 21 08:20 full_chain.pem
-rw-r--r-- 1 root root  264 May 21 08:20 private.key
```

命令过程中会要求输入 keystore 密码，两次确保一致，并记住该密码。

b. 无申请证书，自己生成 jks

使用 jdk 的 keytool 工具生成.keystore 文件。

打开命令行窗口，然后进入到 jdk 安装目录下的 bin 目录中，输入以下命令（需要按需修改）：

keytool -genkey -alias "10.2.108.37" -keypass "yonyou@1" -keyalg RSA -keysize 1024 -validity 365 -keystore "c:\10.2.108.37.keystore"

- -alias "10.2.108.37"：指定别名，这里指定的是分析云服务的 ip(10.2.108.37)。
- -keypass "yonyou@1"：自定义一个密钥密码，这里指定的是 yonyou@1。
- -keyalg RSA：指定签名算法为 RSA。

- -keysize 1024：指定密钥长度 1024。
- -validity 365：指定有效期为 365 天。
- -keystore "c:\10.2.108.37.keystore"：指定生成的 keystore 文件的位置。

输入完成之后按回车，命令行会需要你指定密钥库密码，这个与上面命令中指定的密钥密码保持一致比较好，即 yonyou@1 。完成之后会在 C 盘下生成 10.2.108.37.keystore 文件。

2) 在分析云服务器上配置

首先将 keystore 文件上传到分析云服务器。然后在分析云服务器上，分析云初始换后，找到分析云”tomcat/conf/server.xml”并，在已有的<Connector>元素后再加如下一个<Connector>元素：

```
<Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="200" SSLEnabled="true" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
           keystoreFile="c:\10.2.108.37.keystore"
           keystorePass="yonyou@1" />
```

注意：端口指定为 443，keystoreFile 指定该 keystore 文件路径（上传到的路径），keystorePass 指定密钥密码（第 1 步设置的）。

另外此文件中其它几个<Connector>元素使用了 8443 端口的地方，都要将 8443 改为 443。

如果希望 http 自动转为 https：编辑分析云下的/conf/web.xml 文件，在其中添加如下代码：

```
<!-- 增加所有网址自动跳转 https -->
 <security-constraint>
  <web-resource-collection>
    <web-resource-name >SSL</web-resource-name>
     <url-pattern>/*</url-pattern>
    </web-resource-collection>
   <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
   </user-data-constraint>
     </security-constraint>
```

最后重启分析云服务。

9.4 nginx 启用 https

9.4.1 nginx 的 ssl 模块安装

约定$NGINX_HOME 为 NGINX 安装目录。

查看 nginx 是否安装 http_ssl_module 模块。

```
$NGINX_HOME/sbin/nginx -V
```

如果出现 configure arguments 中包含"--with-http_ssl_module"，则已安装。
如果未安装，则执行下列操作进行安装

```
#1，先停掉 nginx
#2，备份
cp $NGINX_HOME/sbin/nginx $NGINX_HOME/sbin/nginx.bak
#3，进入 nginx 源码包目录，运行：
#注意红色部分是“$NGINX_HOME/sbin/nginx -V”查出的结果，蓝色的部分是新增的模块
./configure --prefix=$NGINX_HOME --with-http_stub_status_module
--with-http_ssl_module
#4，此处不能用 make install ，否则就是覆盖安装（时当前目录会出现 objs 文件夹）
make
#5，用新的 nginx 文件覆盖当前的 nginx 文件
$ cp ./objs/nginx $NGINX_HOME/sbin/
#6，然后启动 nginx，仍可以通过命令查看是否已经加入成功
```

```
$NGINX_HOME /sbin/nginx -V
```

9.4.2 ssl 证书部署

准备好证书文件（这里是用的 pem 与 key 文件）。
在 nginx 目录下新建 cert 文件夹用于存放证书文件，将这两个文件上传至服务器的 cert 目录里。

9.4.3 nginx.conf 配置

```
server {
        listen          80;
        server_name     你的域名 www.你的域名;
        root            /data/www(你 nginx 配置的静态资源目录);

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

# 将 http 强制转 https
        location / {
            rewrite (.*) https://www.你的域名$1 permanent;
        }

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

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

```
server {
        listen          443 ssl http2 default_server;
        listen          [::]:443 ssl http2 default_server;
        server_name     你的域名 www.你的域名;
        root            /data/www(你 nginx 配置的静态资源目录);
         ssl_certificate "你刚才传到服务器上的证书 crt 文件路径"
         ssl_certificate_key "你刚才传到服务器上的证书 key 文件路径";
         ssl_session_cache shared:SSL:1m; ssl_session_timeout 10m;
         ssl_ciphers HIGH:!aNULL:!MD5; ssl_prefer_server_ciphers on;
         # 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 {

        }
    }
```

9.5 nginx 支持 websocket

分析云 v7.2.0 开始，系统页面的实时数据刷功能，基于 stomp 消息传输框架实现。而 stomp 派生于 websocket。因此当系统使用 nginx 作为反向代理时，需要配置 nginx 支持 websocket。下面是 nginx 配置支持 websocket 的步骤：

在 location 匹配配置中添加如下三行内容

```
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection"Upgrade";
```

示例如下：

```
http {
    server {
        location / {
            root   html;
            index  index.html index.htm;
            proxy_pass http://webscoket;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection"Upgrade";
        }
    }
}
```

9.6 嵌入分析云跨域问题

Chrome 81 开始，浏览器的 Cookie 新增加了一个 SameSite 属性，用来防止 CSRF 攻击和用户追踪。并且，chrome 在 80 版本之后，更新了 cookies 的携带机制，把原来 Cookie 的 SameSite 属性值，由 None 改成了 Lax，这就会导致一些需要第三方 cookie 的应用产生了异常。

在跨站请求的时候，你点击非顶级导航，如 iframe 页面中的链接跳转到设置了 SameSite 值为 Lax 的网站， Cookie 是不会发送的。

解决方案： nginx 做网站映射

通过 ngnix 反向代理做网站映射，使分析云系统和要做嵌入的系统站点相同（同站，域名/IP 要保持一致，端口可以不一样。因此 Nginx 做映射，使系统域名或 IP 相同）。可以有两种配置方式，配置分析云和第三方系统 ip 相同，端口不同，根据端口号来区分不同系统。也可以使分析云和第三

方系统 ip 相同，端口也相同，根据 url 应用上下文路径区分不同系统。

一、分析云和第三方系统 ip 相同，端口不同：为第三方系统和分析云系统配置两个 server，使两个 server 服务 ip 或域名相同，端口不同。

```
server {
        listen          端口 1;
        server_name   服务 ip 或域名;
        location / {
            proxy_pass 第三方系统 ip 端口;
            proxy_set_header Host $proxy_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Via "nginx";
        }
    }
    server {
        listen          端口 2;
        server_name   服务 ip 或域名;
        location / {
            proxy_pass 分析云系统 ip:端口;
            proxy_set_header Host $proxy_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Via "nginx";
        }
    }
```

二、分析云和第三方系统 ip 相同，端口相同：第三方系统和分析云系统配置为一个 server，共用一个 ip/域名和端口。根据 location 设置 url 上下文路径，映射到不同的系统路径。

```
server {
        listen          端口;
        server_name   服务 ip 或域名;
        location / {
            proxy_pass 第三方系统 ip:端口;
            proxy_set_header Host $proxy_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Via "nginx";
        }
        location
~ ^/(console|web|sr|bq|bs|di|bq|report|spark_web|bq_self|help|portal|mobile) {
            proxy_pass 分析云系统 ip:端口;
            proxy_set_header Host $proxy_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Via "nginx";
```

```
        }
    }
```

9.7 SQL server2008 和 2012 数据连接问题

主要原因是 sqlserver 驱动对 jdk 的安全要求增强，通过这两种方法可以连接这两种数据库：

1．使用 jTDS 驱动

建立连接使用 jdbc:jtds:sqlserver://<IP>:<Port>;databaseName=<DBName>这种格式的 url。

2．修改 JRE 的安全策略

在分析云安装 HOME 下，编辑/jre8/jre/lib/security/java.security 文件：

```
# 这个是修改后的配置，需要注意的是 TLSv1，TLSv1.1 在这里被排除掉
jdk.tls.disabledAlgorithms=SSLv3，RC4，DES，MD5withRSA，\
    DH keySize < 1024，EC keySize < 224，DES40_CBC，RC4_40，\
    include jdk.disabled.namedCurves

# 这个是修改后的配置，这里将 3DES_EDE_CBC 删除了
jdk.tls.legacyAlgorithms= \
        K_NULL，C_NULL，M_NULL，\
        DH_anon，ECDH_anon，\
        RC4_128，RC4_40，DES_CBC，DES40_CBC
```